INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2023, VOL. 69, NO. 3, PP. 571-577
Manuscript received May 21, 2023; revised July, 2023. DOI: 10.24425/ijet.2023.146509

Overcoming Overfitting Challenges with HOG Feature Extraction and XGBoost-Based Classification for Concrete Crack Monitoring

Ida Barkiah, and Yuslena Sari

***Abstract*—This study proposes a method that combines Histogram of Oriented Gradients (HOG) feature extraction and Extreme Gradient Boosting (XGBoost) classification to resolve the challenges of concrete crack monitoring. The purpose of the study is to address the common issue of overfitting in machine learning models. The research uses a dataset of 40,000 images of concrete cracks and HOG feature extraction to identify relevant patterns. Classification is performed using the ensemble method XGBoost, with a focus on optimizing its hyperparameters. This study evaluates the efficacy of XGBoost in comparison to other ensemble methods, such as Random Forest and AdaBoost. XGBoost outperforms the other algorithms in terms of accuracy, precision, recall, and F1-score, as demonstrated by the results. The proposed method obtains an accuracy of 96.95% with optimized hyperparameters, a recall of 96.10%, a precision of 97.90%, and an F1-score of 97%. By optimizing the number of trees hyperparameter, 1200 trees yield the greatest performance. The results demonstrate the efficacy of HOG-based feature extraction and XGBoost for accurate and dependable classification of concrete fractures, overcoming the overfitting issues that are typically encountered in such tasks.**

***Keywords*—HOG; XGBoost; classification; feature extraction; concrete crack monitoring**

I. Introduction

EXCESSIVE loading, extreme temperature fluctuations, and soil movement can all contribute to the formation of concrete cracks. These fractures can weaken a building's structure and reduce the durability of concrete [1]–[6]. Therefore, monitoring surface fissures is essential for infrastructure safety and quality. Although traditional monitoring by inspectors can record crack data such as location, length, and breadth, manual visual inspection is regarded as less effective in terms of safety, accuracy, cost, and dependability [7]–[11].

Numerous researchers and engineers have studied concrete crack monitoring to devise safer, more cost-effective, and more effective methods[12]–[14]. However, a significant challenge is accurately identifying fractures from images containing actual concrete cracks and distinguishing cracks from non-cracks. In recent decades, owing to the development of image processing techniques and machine learning, numerous fissure identification methods have been proposed [15]–[18] . These methods have demonstrated effective performance, but a significant disadvantage is that overfitting frequently occurs in training and testing results. In addition, low-quality images make it difficult to attain accurate performance in image processing [19]–[25]. Therefore, it is difficult for infrastructure managers to employ these research findings directly to real-world scenarios.

Ensemble methods have recently gained popularity for addressing overfitting issues. Extreme gradient boosting (XGBoost) is a reliable ensemble method with several adjustable hyperparameters to optimize the model's performance [26]–[31]. The following are some advantages of XGBoost hyperparameters: (1) Managing overfitting: Hyperparameters such as max_depth, min_child_weight, and gamma can help prevent overfitting, which occurs when the model becomes overly complex and learns noise from the training data, rendering it incapable of accurately predicting new data. XGBoost can take advantage of multiple CPU cores to optimize training performance. Adjusting hyperparameters such as nthread can optimize CPU utilization. (3) handling unbalanced data: XGBoost's hyperparameters scale_pos_weight and max_delta_step can aid in handling unbalanced data in which the number of samples in one class is substantially larger than the others. (4) improved accuracy XGBoost has several adjustable hyperparameters, such as learning_rate, subsample, and colsample_bytree, to optimize the model's accuracy. By optimizing these hyperparameters, XGBoost is able to produce more accurate models than other machine learning techniques [32]–[36].

HOG (Histogram of Oriented Gradients) is utilized to derive features. The structure of this paper is as follows: In Section 1, the history of concrete fractures and a brief overview of XGBoost are presented. The second section discusses the components and fundamentals of the concrete fracture classification framework. Section 3 describes the research methodology, beginning with data collection and ending with performance evaluation. The fourth section contains experimental results and discussions. In Section 5, conclusions and perspectives are presented. These are the contributions of

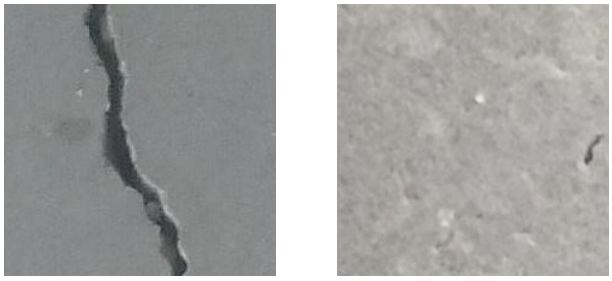
Ida Barkiah is with Department of Civil Engineering, Universitas Lambung Mangkurat, Indonesia (email: idabarkiah@ulm.ac.id).
Yuslena Sari is with Department of Information Technology, Universitas Lambung Mangkurat, Indonesia (email: yuzlena@ulm.ac.id).

this research:

1) HOG-based feature extraction for concrete fissure classification.
2) Comparison of classification models for concrete cracks using XGBoost, Random Forest, and...
3) Improve the efficacy of the XGBoost algorithm by analyzing its hyperparameters.

II. MATERIAL

This research utilized a computer system with the following specifications: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz (8 CPUs), ~2.8GHz, 16384MB RAM, Intel(R) Iris(R) Xe Graphics as the VGA. The coding was done in Python 3.9.13. The research data was obtained from Kaggle [Özgenel, Çağlar Fırat (2019), "Concrete Crack Images for Classification", Mendeley Data, V2, https://doi.org/10.17632/5y9wdsg2zt.2]. The research data consists of 40,000 (forty thousand) image data with a size of 227 x 227 pixels and RGB channels. Each class comprises 20,000 data, with 20,000 positive-labeled images and 20,000 negative-labeled images.



a b

Fig. 1. Positive (a) and Negative (b) Concrete Image

Data division technique using hold-out technique. This method is utilized due to the availability of sufficient data and the reduced efficiency of hold-out estimates in comparison to resampling validation. The dataset is arbitrarily partitioned into training data and test data, with a proportion of 70% training data and 30% test data. Therefore, we acquire 28,000 examples for the training data and 12,000 examples for the test data. Each dataset's performance will be evaluated using a confusion matrix to calculate precision, recall, F1-score, and accuracy values.

III. PROPOSE METHOD

This study will apply algorithms to classify concrete cracks. Several popular classification algorithms such as Random Forest (RF), AdaBoost, and XGBoost will be utilized. The study employs the train_test_split() function to divide the dataset. The classification model will be trained with the training data and subsequently used to classify concrete cracks. The research workflow can be seen in Figure 2.

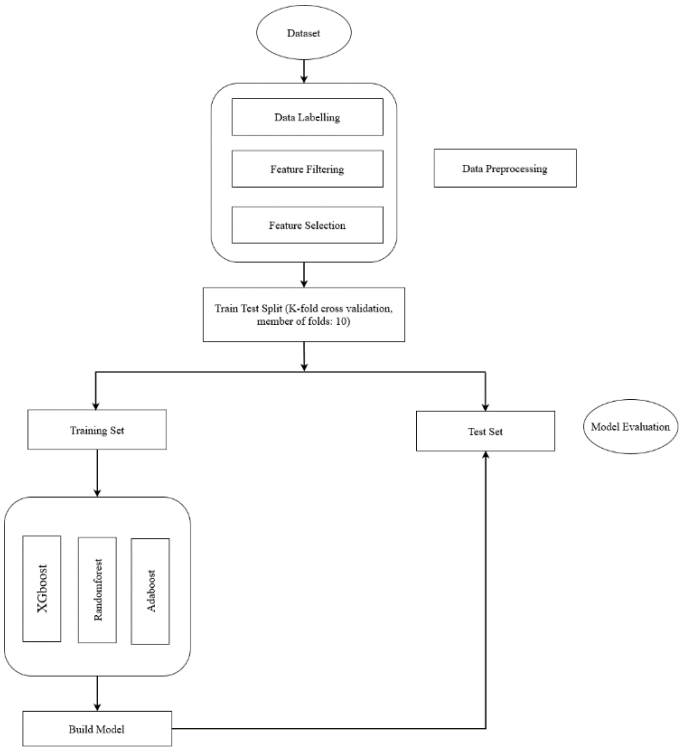
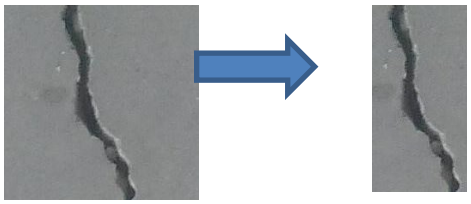


Fig. 2. The research workflow

A. HOG Feature Extraction

Histogram of Oriented Gradient (HOG) offers superior performance in comparison to other extant features. The fundamental hypothesis is that the distribution of local gradient intensities or edge directions can frequently adequately characterize the appearance and local objects. HOG aims to represent an image with a histogram of local gradient orientations [37]–[42]. This study employs images with a resolution of 227 by 227 pixels, with clarity-enhancing patches extracted from the original images. HOG requires a specific dimension to compute feature values; therefore, to obtain feature values, the researchers will resize the images to 126 by 64 pixels.



a b

Fig. 3. 227 x 277 (a) Concrete Image and 126 x 6 (b) Concrete Image

The Histogram of Oriented Gradients (HOG) method is used to identify objects based on gradient patterns suitable for concrete fissure images during the image's feature extraction phase. The

HOG method is implemented in the Python programming language using the scikit-image library. There is a module within the scikit-image library that provides computations for the HOG method. In the HOG implementation for feature extraction, the following parameters are utilized: the number of orientations is set to 8, the number of pixels per cell is set to 8, and the number of cells per block is set to 2. Image feature extraction is implemented as follows in the Python programming language:

```
fd, hog_image = hog(image, orientations=8,
pixels_per_cell=(8, 8),
cells_per_block=(2, 2), visualize=True)
```

The output of feature extraction is 23,328 features, resulting in a data size of 40,000 rows by 23,329 columns. The results are presented in table I.

TABLE I
HOG FEATURE EXTRACTION

0	1	2	…	23325	23326	23327	Label
0.17 4694	0.14 2588	0.04 6688	…	0.2399 43	0.2100 49	0.1259 05	*Positive*
0.21 8352	0.04 9377	0.27 8014	…	0.0463 93	0.0824 43	0.1044 40	*Postive*
0.28 0815	0.00 0000	0.12 2703	…	0.1058 28	0.0880 20	0.1671 59	*Postive*
0.15 3354	0.13 1769	0.11 6218	…	0.2479 18	0.2589 77	0.0986 37	*Postive*

B. XGBoost

XGBoost is a gradient-enhanced ensemble decision tree designed for scalability. XGBoost constructs an additive extension of the function in an effort to minimize the loss function [4] using (1) and (2).

$$L_{xgb} = \sum_{i=1}^{N} L(y_i, F(x_i)) + \sum_{m=1}^{M} \Omega(h_m) \quad (1)$$

$$\Omega(h) = \gamma T + \frac{1}{2}\lambda||w||^2 \quad (2)$$

This loss function can be incorporated into the individual criteria of the decision tree, leading to a strategy for pre-pruning. The higher the x value, the simpler the tree or derivative. The minimal loss reduction required to split an internal node is controlled by the y value. Shrinkage is an additional regularization parameter in XGBoost that reduces the bulk of additive expansion steps. Other strategies, such as derived depth, can also be used to reduce derivative complexity. XGBoost enhances model performance by concentrating primarily on model speed and performance. This algorithm has several features that promote parallelization by generating decision trees in parallel in order to accomplish speed and efficiency. It employs distributed computing techniques to evaluate large or intricate models. The algorithm also employs out-of-core computation to ensure that large and diverse data sets are analyzed and the cache is optimized to make use of the most efficient hardware and resources. All of these distinctive characteristics make XGBoost the ideal tool for this study

The XGBoost algorithm is explained in the following five steps:

- Step 1: The objective function must be defined. As demonstrated by (3) and (4), the objective function consists of two parts.

$$L(\boldsymbol{\phi}) = \sum_i \boldsymbol{l}(\boldsymbol{y_i}, \boldsymbol{\hat{y}_i} + \sum_k \Omega \boldsymbol{f}(\boldsymbol{k}) \quad (3)$$

$$\sum_k \Omega \boldsymbol{f}(\boldsymbol{k}) = \boldsymbol{\gamma T} + \frac{\boldsymbol{1}}{\boldsymbol{2}}\boldsymbol{\lambda}||\boldsymbol{w}||^2 \quad (4)$$

In equations $\sum_i \boldsymbol{l}(\boldsymbol{y_i}, \boldsymbol{\hat{y}_i})$ is a loss term, measuring how good the model is. $\sum_k \Omega \boldsymbol{f}(\boldsymbol{k})$ is a regularization term that measures the complexity of a tree. $\boldsymbol{y_i}$ is a real category. $\boldsymbol{\hat{y}_i}$ is the classification value. $\boldsymbol{l}$ used to find the difference between $\boldsymbol{y_i}$ and $\boldsymbol{\hat{y}_i}$, this is usually called the loss function.

T: indicates a leaf node.
W: represents the mass of the k leaf node
γ : useful for regulating leaf node
λ : Utilizable for regulating the weight of leaf nodes.
Regularization is utilized to control overfitting issues.

- Step 2: Following the formation of t trees, the newly created tree is used to match the remainder of the previously made predictions. Consequently, (5) and (6) convey the classification of values.

$$\hat{y}_i = \hat{y}_i^{(t-1)} + f_t(x_i) \quad (5)$$

Then rewrite the objective function as below:

$$obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_i(x_i)) + \Omega(f_t) \quad (6)$$

- Step 3: Expansion of the Tylor series for the loss function, as shown in (7) and (8).

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \quad (7)$$

$$g_i = \frac{\delta l(y_i, \hat{y}_i^{(t-1)})}{\delta \hat{y}_i^{(t-1)}} \; and h_i = \frac{\delta^2 l(y_i, \hat{y}_i^{(t-1)})}{\delta (\hat{y}_i^{(t-1)})^2} \quad (8)$$

Converting the original objective function to a dominant Euclidean function using conventional optimization techniques.

$$\sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} \quad (9)$$

- In Step 4, maintain the constant expression (9). The previous loss function has no bearing on the required time and effort to construct a decision tree. After removing the constant term, must satisfy the given equation (10)

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \quad (10)$$

- Step 5 presupposes that the instance group of leaves j is defined as $I_j = \{i|q(x_i) = j\}$ then obtain the equation (11) and (12) by expanding the term Ω.

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \quad (11)$$

$$\mathcal{L}^{(t)} \simeq \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)\, w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)\, w_j^2] + \gamma T \quad (12)$$

For a given structure of ($q(x_i)$), the value of the optimization objective function and the weight optimization (w_i) of a leaf node j are expressed as (13) and (14).

$$w_i^* = \frac{-\sum g_i}{\sum_{i\in I_j} h_i + \lambda} \quad (13)$$

$$\mathcal{L}^{t}(q) = -\frac{1}{2}\sum_{j=1}^{T}[\frac{-(\sum_{i\in I_j} g_i)^2}{(\sum_{i\in I_j} h_i + \lambda)}] + \gamma T \quad (14)$$

After assessing the tree's structure and dividing the tree's nodes, the resulting in equation (15).

$$\mathcal{L}_{split} = \frac{1}{2}[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} + \frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda}] - \gamma \quad (15)$$

In addition, randomization strategies have been used into XGBoost in order to slow down the process of overfitting and speed up the training process. XGBoost additionally includes the implementation of numerous strategies for increasing the pace of training decision trees. These approaches are not directly related to the accuracy of the ensemble. XGBoost, in particular, places an emphasis on simplifying the computational work necessary to locate the optimal separation [4]. The maximum depth, the learning rate (eta), and the number of estimators are the three input parameters that XGBoost requires. The following is an illustration of the implementation:

```
import xgboost as xgb

xgb_classifier =
xgb.XGBClassifier(n_estimators=1000,eta=0.1,max_depth=
3)
```

C. Modeling

During the modeling phase, the three algorithms are utilized to generate the most accurate classification for predicting concrete cracking. Cross-validation, a method for measuring or validating the accuracy of a model derived from a training dataset, is the only method for determining the optimal model for each algorithm. The model is then evaluated using training data and measurements of the confusion matrix.

- Modeling with cross-validation using XGBoost
  In XGBoost, an attractive type of hypermeter must be set using cross validation techniques. The list of hypermeters is displayed in table II [1].
  These are the stages involved in the XGBoost modeling phase:
  - Defines a set of hyperparameter configurations for XGBoost
  - Using the features chosen from the training dataset, perform XGBoost modeling 100 times with arbitrarily selected hyperparameter combinations and 10 times cross-validation. The hyperparameter with the greatest ROC score is saved as the optimal model configuration.
  - Evaluate the best model by displaying all relevant metrics and model evaluation scores derived from the training dataset.
  - Using the best model to forecast the positive and negative labels of concrete crack images in the test dataset, and then evaluating the results of the prediction using metrics and model evaluation scores.

TABLE II
THE LIST HYPERMETERS

Hyperparamter	Value
max_dept	[3, 5, 7, 9]
min_child_weight	[1, 3, 5]
gamma	[0.0, 0.33333, 0.25, 0.5, 0.66667, 0.75]
reg_alpha	[1e-5, 1e-2, 0.1, 1, 100]

Figure 4 depicts the XGBoost modeling flowchart that follows.

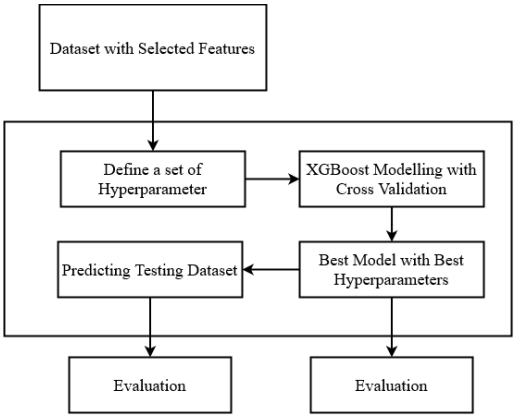


Fig. 4. XGBoost Modeling Flowchart

D. Evalution Models

By calculating the Confusion Matrix, the classification model is assessed. According to Bisri and Wahono (2015), the classification model is evaluated based on its accuracy. The precision of a model can be measured using a Confusion Matrix. The results of the confusion matrix are used to calculate the accuracy, sensitivity, and specificity of the classification model's predictions using the formulation shown in equation (16).

TABLE III
CONFUSION MATRIX

Actual Class	Predicted Class	
	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \quad (16)$$

IV. Results and Discussion

A. Performance Evaluations of Method

Using a confusion matrix, this research evaluates the precision of feature extraction and classification. The efficacy of the method is evaluated using the default parameters that have been determined by the XGBoost library. Utilizing the Python programming language's scikit-learn library, testing was conducted. The library is searched for the values for accuracy, f1 score, precision, and recall.

In the confusion matrix, accuracy corresponds to the correct prediction value for the entire data set. Precision refers to the genuine positive predictive value in comparison to the overall positive predicted outcome. Compared to all positive data, recall is the genuine positive prediction value. The F1 score is a comparison of the average precision and recall with weighting applied.

The writing of program code in Python is as follows:

```
accuracy_score(label, pred)

f1_score(label, pred, average=macro)

precision_score(label, pred, average= macro))

recall_score(label, pred, average= macro)
```

In the table of HOG and XGBoost performance evaluation results, the values of precision, recall, f1-score, and accuracy can be viewed alongside the corresponding results.

TABLE IV
PERFORMANCE EVALUATION RESULT FOR HOG AND XGBOOST

Accuracy	*F1 Score*	*Precision*	*Recall*
0.92216666666 6667	0.9220168328 41968	0.925163053989 5845	0.92206940904 45604

From these results, the best precision, recall, f1-score, and accuracy values remain in the XGBoost method.

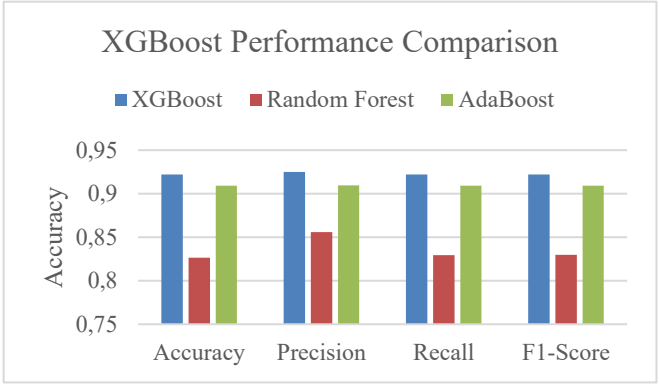


Fig. 5. Performance Comparison

In XGBoost, one of the hyper-parameters is the number of trees (n_estimators). The number of trees represents the number of processes XGBoost will attempt to learn. Number of trees is a prevalent hyperparameter in boosting methods [Liquefaction prediction with robust machine learning algorithms (SVM, RF, and XGBoost) supported by genetic algorithm-based feature selection and parameter optimization from the viewpoint of data processing | SpringerLink].

TABLE V
EVALUTION PARAMETERS NUMBER OF TREES FOR PERFORMANCE HOG AND XGBOOST

No.	*Number*	*Accuracy*	*f1-score*	*Precision*	*Recall*
1.	100	0.92216	0.92201	0.92516	0.92207
2.	200	0.94475	0.94471	0.94582	0.94469
3.	250	0.94842	0.94838	0.94929	0.94837
4.	300	0.95208	0.95206	0.95278	0.95204
5.	350	0.95534	0.95532	0.95591	0.95529
6.	500	0.96116	0.96115	0.96151	0.96113
7.	1000	0.96850	0.96849	0.96870	0.96849
8.	1200	0.96958	0.96956	0.96976	0.96956

Based on the Table V, a value of 96.95% was obtained for 1200 trees, representing the number of trees with the utmost accuracy. The greater the value of the number of trees, the greater the obtained precision.

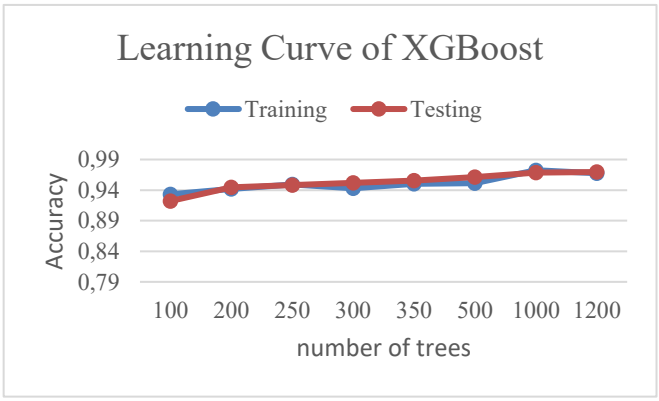


Fig. 6. Learning Curve XGBoost

Figure 6 provides the results of training and testing trials that do not occur over fitting or under fitting. The manual calculation for the confusion matrix with a number of trees of 1200 from XGBoost obtained from the HOG feature extraction can be seen in Table VI

TABLE IV
CONFUSION MATRIX

	Positive	*Negative*
Positive	5888	126
Negative	239	5747

From these data it can be calculated:

- $Accuracy = \frac{TP+TN}{(TP+FP+TN+FN)} = \frac{5888+5747}{(5888+126+5747+239)} = \frac{11622}{12000} = 0.969583333 = 96.95\%$
- $Recall = \frac{TP}{(TP+FN)} = \frac{5888}{(5888+239)} = \frac{5888}{6127} = 0.960992329 = 96.10\%$
- $Precision = \frac{TP}{(TP+FP)} = \frac{5888}{(5888+126)} = \frac{5888}{6014} = 0.979048886 = 97.90\%$
- $F1-score = \frac{2\,(Recall\ x\ Precision)}{(Recall+Precision)} = \frac{2\,(0.960992329x\ 0.979048886)}{(0.960992329+\ 0.979048886)} = 0.969936579 = 97\%$

Based on the calculation above, the accuracy, recall, precision, and f1-score values for HOG and XGBoost with a number of trees of 1000 are obtained, respectively 96.95%, 96.10%, 97.90%, and 97%. With the performance results of the method obtained, it took about ± 5 hours for each running of experimental data measuring 40,000 rows x 23,329 columns which were carried out to implement HOG as feature extraction and XGBoost as a classification method for machine learning.

B. Discussion

XGBoost employs a shrinkage method to avoid overfitting. This method reduces the contribution of each tree in the model, preventing the model from focusing excessively on the training data. In addition, XGBoost employs regularization and early halting techniques to prevent overfitting of the model's performance. The hyper parameters with 1200 trees yield the greatest accuracy, f1-score, precision, and recall. The ensemble method is a machine learning technique that combines the results of multiple learning models to enhance predictive performance and reduce the risk of overfitting. This method utilizes the assets of multiple models to generate more accurate predictions than using a single model alone. Several methods, including bagging, boosting, and layering, can be used for ensemble learning. Comparing the performance of XGBoost with other ensemble methods demonstrates that it is a reliable method for detecting concrete fractures.

CONCLUSION

In this paper, we address the problem of overfitting that often occurs in the results of training and testing of machine learning methods. XGBoost is a reliable method for overcoming over-fitting with improving hyper parameters. The results of improving the number of tree hyper parameter obtained a value of 1200 is the highest result of the overall performance tested. These results also show that the higher the number of trees, the better the performance value. XGBoost can overcome over fitting with shrinkage, regularization, and early stopping techniques. Comparison of the ensemble method between XGBoost, Random Forest, and ADBoost produces the most reliable XGBoost method for classifying cracked concrete images.

ACKNOWLEDGEMENTS

We are very grateful to Big Data Laboratory Faculty of Engineering Universitas Lambung Mangkurat.

REFERENCES

[1] K. Gao, H. Xie, Z. Li, J. Zhang, and J. Tu, "Study on eccentric behavior and serviceability performance of slender rectangular concrete columns reinforced with GFRP bars," *Compos. Struct.*, vol. 263, no. February, p. 113680, 2021, https://doi.org/10.1016/j.compstruct.2021.113680

[2] P. Guo, W. Meng, and Y. Bao, "Automatic identification and quantification of dense microcracks in high-performance fiber-reinforced cementitious composites through deep learning-based computer vision," *Cem. Concr. Res.*, vol. 148, no. July, p. 106532, 2021, https://doi.org/10.1016/j.cemconres.2021.106532

[3] K. Harsh, P. P. V., P. J. B., and K. Nikhil, "Implementation of Computer Vision Technique for Crack Monitoring in Concrete Structure," *J. Inst. Eng. Ser. A*, vol. 104, no. 1, 2023, https://doi.org/10.1007/s40030-022-00695-5

[4] X. Xie, L. Zhang, and Z. Qu, "A Critical Review of Methods for Determining the Damage States for the In-plane Fragility of Masonry Infill Walls," *J. Earthq. Eng.*, vol. 26, no. 9, pp. 4523–4544, 2022, https://doi.org/10.1080/13632469.2020.1835749

[5] C. Yuan, B. Xiong, X. Li, X. Sang, and Q. Kong, "A novel intelligent inspection robot with deep stereo vision for three-dimensional concrete damage detection and quantification," *Struct. Heal. Monit.*, vol. 21, no. 3, pp. 788–802, 2022, https://doi.org/10.1177/14759217211010238

[6] C. Camille, D. Kahagala Hewage, O. Mirza, and T. Clarke, "Full-scale static and single impact testing of prestressed concrete sleepers reinforced with macro synthetic fibres," *Transp. Eng.*, vol. 7, p. 100104, 2022, https://doi.org/10.1016/j.treng.2022.100104

[7] J. Deng, Y. Lu, and V. C. S. Lee, "Concrete crack detection with handwriting script interferences using faster region-based convolutional neural network," *Comput. Civ. Infrastruct. Eng.*, vol. 35, no. 4, pp. 373–388, 2020, https://doi.org/10.1111/mice.12497

[8] V. P. Golding, Z. Gharineiat, H. S. Munawar, and F. Ullah, "Crack Detection in Concrete Structures Using Deep Learning," *Sustain.*, vol. 14, no. 13, 2022, https://doi.org/10.3390/su14138117

[9] H. Kim, S. Lee, E. Ahn, M. Shin, and S. H. Sim, "Crack identification method for concrete structures considering angle of view using RGB-D camera-based sensor fusion," *Struct. Heal. Monit.*, vol. 20, no. 2, pp. 500–512, 2021, https://doi.org/10.1177/1475921720934758

[10] and R. C. L. Deng, T. Sun, L. Yang, "Binocular video-based 3D reconstruction and length quantification of cracks in concrete structures," *Autom. Constr*, vol. 148, 2023, https://doi.org/10.1016/j.autcon.2023.104743

[11] H. Kim, E. Ahn, M. Shin, and S. H. Sim, "Crack and Noncrack Classification from Concrete Surface Images Using Machine Learning," *Struct. Heal. Monit.*, vol. 18, no. 3, pp. 725–738, 2019, https://doi.org/10.1177/1475921718768747

[12] N. Kheradmandi and V. Mehranfar, "A critical review and comparative study on image segmentation-based techniques for pavement crack detection," *Constr. Build. Mater.*, vol. 321, 2022, https://doi.org/10.1016/j.conbuildmat.2021.126162

[13] S. S. N., K. S., and R. G, "Review and Analysis of Crack Detection and Classification Techniques based on Crack Types," *Int. J. Appl. Eng. Res.*, vol. 13, no. 8, p. 6056, 2021, https://doi.org/10.37622/ijaer/13.8.2018.6056-6062

[14] R. G. Sheerin Sitara N., K. S., "Review and Analysis of Crack Detection and Classification Techniques based on Crack Types," *Int. J. Appl. Eng. Res.*, vol. 13, no. 8, 2021, https://doi.org/10.37622/ijaer/13.8.2018.6056-6062

[15] L. Li, K. Ota, and M. Dong, "Deep Learning for Smart Industry: Efficient Manufacture Inspection System with Fog Computing," *IEEE Trans. Ind. Informatics*, vol. 14, no. 10, pp. 4665–4673, 2018, https://doi.org/10.1109/TII.2018.2842821

[16] B. Kim and S. Cho, "Automated vision-based detection of cracks on concrete surfaces using a deep learning technique," *Sensors (Switzerland)*, vol. 18, no. 10, 2018, https://doi.org/10.3390/s18103452.

[17] Y. Sari, P. B. Prakoso, and A. R. Baskara, "Road Crack Detection using Support Vector Machine (SVM) and OTSU Algorithm," *2019 6th Int. Conf. Electr. Veh. Technol.*, pp. 349–354, 2019, https://doi.org/10.1109/ICEVT48285.2019.8993969

[18] Y. Sari, P. B. Prakoso, and A. R. Baskara, "Application of neural network method for road crack detection," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 18, no. 4, pp. 1962–1967, 2020, https://doi.org/10.12928/TELKOMNIKA.V18I4.14825

[19] H. Hofbauer, F. Autrusseau, and A. Uhl, "Low Quality and Recognition of Image Content," *IEEE Trans. Multimed.*, vol. 24, pp. 3595–3610, 2022, https://doi.org/10.1109/TMM.2021.3103394

[20] T. He and X. Li, "Image quality recognition technology based on deep learning," *J. Vis. Commun. Image Represent.*, vol. 65, p. 102654, 2019, https://doi.org/10.1016/j.jvcir.2019.102654

[21] Y. Gao, L. Gao, and X. Li, "A Generative Adversarial Network Based Deep Learning Method for Low-Quality Defect Image Reconstruction and Recognition," *IEEE Trans. Ind. Informatics*, vol. 17, no. 5, pp. 3231–3240, 2021, https://doi.org/10.1109/TII.2020.3008703

[22] R. A. Pramunendar, D. P. Prabowo, D. Pergiwati, Y. Sari, P. N. Andono, and M. A. Soeleman, "New workflow for marine fish classification based on combination features and CLAHE enhancement technique," *Int. J. Intell. Eng. Syst.*, vol. 13, no. 4, pp. 293–304, 2020, https://doi.org/10.22266/IJIES2020.0831.26

[23] Y. Sari, M. Alkaff, and R. A. Pramunendar, "Classification of coastal and Inland Batik using GLCM and Canberra Distance," *AIP Conf. Proc.*, vol. 1977, no. June 2022, 2018, https://doi.org/10.1063/1.5042901

[24] Y. Sari, A. R. Baskara, and R. Wahyuni, "Classification of Chili Leaf Disease Using the Gray Level Co-occurrence Matrix (GLCM) and the Support Vector Machine (SVM) Methods," *2021 6th Int. Conf. Informatics Comput. ICIC 2021*, 2021, https://doi.org/10.1109/ICIC54025.2021.9632920

[25] Y. Sari, M. Alkaff, and M. Maulida, "Classification of Rice Leaf using Fuzzy Logic and Hue Saturation Value (HSV) to Determine Fertilizer Dosage," 2020, https://doi.org/10.1109/ICIC50835.2020.9288585

[26] A. Alazba and H. Aljamaan, "Software Defect Prediction Using Stacking Generalization of Optimized Tree-Based Ensembles," *Appl. Sci.*, vol. 12, no. 9, 2022, https://doi.org/10.3390/app12094577

[27] S. Deng, Y. Zhu, S. Duan, Z. Fu, and Z. Liu, "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, https://doi.org/10.3390/systems10040108

[28] V. Rathakrsihman, S. B. Beddu, and A. N. Ahmed, "Comparison Studies Between Machine Learning Optimisation Technique on Predicting Concrete Compressive Strength," *Eur. PMC*, pp. 1–5, 2021, https://doi.org/10.21203/rs.3.rs-381936/v1

[29] J. J. Liu and J. C. Liu, "Permeability Predictions for Tight Sandstone Reservoir Using Explainable Machine Learning and Particle Swarm Optimization," *Geofluids*, vol. 2022, no. 2, 2022, https://doi.org/10.1155/2022/2263329

[30] D. Xiaoming, C. Ying, Z. Xiaofang, and G. Yu, "Study on Feature Engineering and Ensemble Learning for Student Academic Performance Prediction," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 5, pp. 495–502, 2022, https://doi.org/10.14569/IJACSA.2022.0130558

[31] W. Chen, H. Zhang, M. K. Mehlawat, and L. Jia, "Mean–variance portfolio optimization using machine learning-based stock price prediction," *Appl. Soft Comput.*, vol. 100, p. 106943, 2021, https://doi.org/10.1016/j.asoc.2020.106943.

[32] Y. Qiu, J. Zhou, M. Khandelwal, H. Yang, P. Yang, and C. Li, "Performance evaluation of hybrid WOA-XGBoost, GWO-XGBoost and BO-XGBoost models to predict blast-induced ground vibration," *Eng. Comput.*, vol. 38, no. 0123456789, pp. 4145–4162, 2022, https://doi.org/10.1007/s00366-021-01393-9

[33] G. Zhou, Z. Ni, Y. Zhao, and J. Luan, "Identification of Bamboo Species Based on Extreme Gradient Boosting (XGBoost) Using Zhuhai-1 Orbita Hyperspectral Remote Sensing Imagery," *Sensors*, vol. 22, no. 14, 2022, https://doi.org/10.3390/s22145434

[34] A. Ibrahem Ahmed Osman, A. Najah Ahmed, M. F. Chow, Y. Feng Huang, and A. El-Shafie, "Extreme gradient boosting (Xgboost) model to predict the groundwater levels in Selangor Malaysia," *Ain Shams Eng. J.*, vol. 12, no. 2, pp. 1545–1556, 2021, https://doi.org/10.1016/j.asej.2020.11.011

[35] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, *A comparative analysis of gradient boosting algorithms*, vol. 54, no. 3. Springer Netherlands, 2021.

[36] C. Bouchayer, J. M. Aiken, K. Thogersen, F. Renard, and T. V. Schuler, "A Machine Learning Framework to Automate the Classification of Surge-Type Glaciers in Svalbard," *JGR Earth Surf.*, 2022, https://doi.org/10.1029/2022JF006597

[37] J. P. Tanjung and M. Muhathir, "Classification of facial expressions using SVM and HOG," *J. Informatics Telecommun. Eng.*, vol. 3, no. 2, pp. 210–215, 2020, https://doi.org/10.31289/jite.v3i2.3182

[38] T. Tri Saputra Sibarani and C. Author, "Analysis K-Nearest Neighbors (KNN) in Identifying Tuberculosis Disease (Tb) By Utilizing Hog Feature Extraction," *Int. Comput. Sci. Inf. Technol. JournalISSN*, vol. 1, no. 1, pp. 33–38, 2020.

[39] S. T. Narasimhaiah and L. Rangarajan, "Recognition of compound characters in Kannada language," *Int. J. Electr. Comput. Eng.*, vol. 12, no. 6, pp. 6103–6113, 2022, https://doi.org/10.11591/ijece.v12i6.pp6103-6113

[40] K. V. Greeshma and K. Sreekumar, "Fashion-MNIST classification based on HOG feature descriptor using SVM," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 5, pp. 960–962, 2019.

[41] S. Bakheet and A. Al-Hamadi, "A framework for instantaneous driver drowsiness detection based on improved HOG features and naïve bayesian classification," *Brain Sci.*, vol. 11, no. 2, pp. 1–15, 2021, https://doi.org/10.3390/brainsci11020240

[42] T. Zhang *et al.*, "HOG-ShipCLSNet: A Novel Deep Learning Network with HOG Feature Fusion for SAR Ship Classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, no. June, 2022, https://doi.org/10.1109/TGRS.2021.3082759