Computer Science and Information Technologies
Vol. 6, No. 2, July 2025, pp. 112~121
ISSN: 2722-3221, DOI: 10.11591/csit.v6i2.pp112-121

Optimizing EfficientNet for imbalanced medical image classification using grey wolf optimization

Khusnul Khotimah, Sugiyarto Surono, Aris Thobirin
Department of Mathematics, Faculty of Applied Science and Technology, Universitas Ahmad Dahlan, Yogyakarta, Indonesia

Article Info

Article history:

Received Jan 19, 2025
Revised Apr 19, 2025
Accepted May 23, 2025

Keywords:

Augmentation
Deep learning
Hyperparameter optimization
Image classification
Imbalanced dataset

ABSTRACT

The advancement of deep learning in computer vision has result in substantial progress, particularly in image classification tasks. However, challenges arise when the model is applied to small and unbalanced datasets, such as X-ray data in medical applications. This study aims to improve the classification performance of fracture X-ray images using the EfficientNet architecture optimized with grey wolf optimization (GWO). EfficientNet was chosen for its efficiency in handling small datasets, while GWO was applied to optimize hyperparameters, including learning rate, weight decay, and dropout to improve model accuracy. Random cropping, rotation, flipping, color jittering, and random erasing, were used to expand the diversity of the dataset, and class weighting is applied to overcome class imbalance. The evaluation uses accuracy, precision, recall, and F1-score metrics. The combination of EfficientNetB0 and GWO resulted in an average 4.5% improvement in model performance over baseline methods. This approach provides benefits in developing deep learning methods for medical image classification, especially in dealing with small and imbalanced datasets.

Corresponding Author:

Khusnul Khotimah
Department of Mathematics, Faculty of Applied Science and Technology, Universitas Ahmad Dahlan
Yogyakarta 55191, Indonesia
Email: khusnul2100015009@webmail.uad.ac.id

1. INTRODUCTION

Technological developments in the current digital era have encouraged rapid innovation in various areas of life, with deep learning in computer vision [1], [2] being one of the most prominent advancements. This technology focuses on image data processing [3]. It utilizes mathematical principles such as statistics, linear algebra, calculus, and optimization [4], to build and train algorithms that allow computers to recognize patterns, interpret images [5], and optimize models to obtain accurate results. These capabilities have led to extraordinary achievements in various computer vision applications [6], including image classification tasks. In this domain, convolutional neural networks (CNN) have become a mainstay method due to their superior capacity to extract features from image data automatically, produce high accuracy, and manage data efficiently [7], [8]. Even though it has been proven effective in analyzing and classifying various types of images, CNN performance is also influenced by multiple factors, namely, the characteristics of the dataset used, such as data size and distribution [9], the architectural design of the CNN model applied [10], and the hyperparameters chosen, such as learning rate [11].

Earlier research has shown the effectiveness of CNNs in image classification tasks, especially when using large datasets such as CIFAR-100. Stanford Dogs, and montréal institute for learning algorithms-traffic camera dataset (MIO-TCD). The choice of Xception model architecture in developing the CNN model also shows high accuracy in image classification. This study also explored hyperparameter optimization methods

like grid search, random search, Bayesian optimization [12], and asynchronous successive halving algorithm (ASHA), which are proven to increase accuracy by several percent [13]. Although CNN is effective in image classification, another challenge arises when the model is applied to small and imbalanced datasets. In real-world applications such as in the healthcare field, medical image data often has limitations in terms of the number and uneven distribution of classes, for example, in X-ray data [14]. This imbalance can cause the CNN model to be more likely to recognize the majority class, which reduces the accuracy of the minority class and negatively impacts the diagnostic performance. One solution to overcome this is to choose an efficient model architecture and hyperparameter optimization algorithm that can significantly improve model performance. The EfficientNet architecture outperforms traditional CNN architectures, particularly in medical image classification tasks involving small datasets [15]. Conversely, the grey wolf optimization (GWO) algorithm offers an effective metaheuristic method for optimizing the model's hyperparameters. GWO is more efficient compared to conventional optimization methods such as random search and grid search, particularly in enhancing the performance of deep learning models on medical datasets [16].

Various methods have been applied in X-ray image classification. The CNN-based approach was conducted to classify fracture images with a total of 1,521 images distributed into 12 classes. Visual geometry group 16-layer network (VGG16) was used for feature extraction, and principal component analysis (PCA) was used for dimension reduction. This study showed an accuracy of 94% [17]. Meanwhile, object detection and deep learning ensemble-based methods such as a combination of YOLOv5 and EfficientNetB3 have also been used in the classification of distal radius fractures with a total of 400 images composed of 2 classes. This study showed an accuracy of 81% [18]. Differences in method selection, model architecture, optimization strategies, and dataset characteristics are the main factors that affect the performance of the medical image classification system.

This study contributes in several aspects. First, this study explores the application of hyperparameter optimization on a small and imbalanced X-ray dataset, which is different from the CIFAR-100, MIO-TCD, and Stanford Dogs datasets used in previous studies. Second, this study integrates the EfficientNet architecture for classification tasks and the GWO metaheuristic optimization algorithm to improve model performance. This is different from previous studies that used the Xception architecture with its hyperparameter optimization using classical optimization. With this approach, this study is expected to provide new insights into the development of deep learning methods for X-ray data processing, especially in dealing with the challenges of class imbalance and small dataset sizes.

2. METHOD

This research aims to develop an image classification model through an experimental approach. It combines EfficientNet as a CNN model for X-ray image classification and GWO for hyperparameter optimization. This approach also involves class augmentation and weighting techniques to deal with small and unbalanced datasets. All experiments in this approach were implemented using the Python programming language and the PyTorch framework. The study stages are shown in Figure 1.

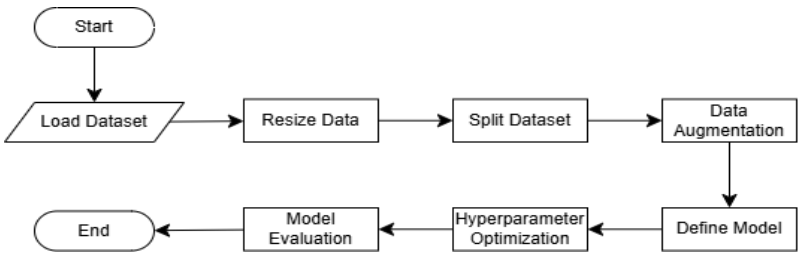


Figure 1. The research stages

2.1. Datasets

In this study, the data used were medical X-ray images of various types of bone fractures. The data were obtained from the Kaggle platform. From this data, only 660 images were taken, which were divided into four classes: avulsion fracture, comminuted fracture, fracture dislocation, and greenstick fracture. These images have various resolutions and are formatted in JPG files. The data is small and unbalanced because the number of samples for each class is different. There are 147 images in the avulsion fracture class, 178 in the comminuted fracture class, 188 in the fracture dislocation class, and 147 in the greenstick fracture class. Figure 2 shows some sample data used, where Figure 2(a) shows an image from the avulsion fracture class,

Figure 2(b) shows a sample from the comminuted fracture class. Figures 2(c) and 2(d) depict sample images from the fracture dislocation and greenstick fracture classes.

(a) (b) (c) (d)

Figure 2. Sample datasets for (a) avulsion fracture, (b) comminuted fracture, (c) fracture dislocation, and (d) greenstick fracture

2.2. Data preprocessing

During this phase, various techniques are applied to address issues in the data. This prepares the data for use, allowing the model to perform better and deliver optimal results. The techniques used in this phase are:

- Resize data: bone fracture X-ray image data has various dimensions. Resizing is needed so that the input image has uniform and consistent dimensions so that the CNN architecture can process data more efficiently [19]. This procedure also reduces computational demands and increases model training speed. This study standardized all data with dimensions of 224 pixels (width) and 224 pixels (height). This measure was chosen because it is commonly used in the EfficientNet architecture model [20].
- Data split: in the data pre-processing stage, the data will be divided into three categories: training, validation, and test data. There are 2 data sharing scenarios in this approach. In the first scenario, of all the data, 80% of the data is allocated to the training set used to train the model, 10% of the data is allocated to the validation set used to monitor the development of the model, and the remaining 10% is used to test the performance of the model that has been trained on images that have not been used. The second scenario is also the same, with comparisons of 70%, 15%, and 15%, respectively.
- Data augmentation: various augmentation techniques were used to make the training data more varied. Because it uses PyTorch, the augmentation is done via transform. The transformation starts with ToPILImage to convert the data to an image format, followed by RandomResizedCrop, which randomly crops the image to 224×224 to add scale and area focus variations. Next, ColorJitter modifies brightness, contrast, saturation, and hue to make the model more resilient to lighting variations. Random rotation is applied using RandomRotation, while RandomHorizontalFlip and RandomVerticalFlip add variations in image orientation. The RandomAffine transformation introduces geometric distortion by random rotation, scaling, and shear. The data is then converted into a tensor using ToTensor, and RandomErasing is added to remove a small portion of the image randomly, which functions as regularization to prevent overfitting. Finally, normalization is carried out by using the mean and standard deviation of the ImageNet dataset. Normalization can remove bias from pixel values that are too large or too small, allowing the model to adapt more quickly to existing data patterns [21].
- Class weighting: this study uses class weighting to handle class imbalance in the dataset. This approach gives greater weight to classes with fewer images, thereby strengthening the contribution of minority classes in the model training process. The weight of each class can be calculated using (1).

$$w_k = \frac{n}{c \cdot n_k} \tag{1}$$

where w_k is the weight for each class, n is the number of all images in the training set, c is the number of classes, and n_k is the number of images for each class.

2.3. Building models

In this study, the model used is EfficientNetB0. The EfficientNetB0 is a CNN model proposed by Tan and Le [22]. The model is designed with efficiency as a core principle, maximizing accuracy using fewer parameters and less computing power. The model uses a method known as compound scaling, which optimally adjusts the network's depth, width, and resolution simultaneously. This allows the model to

provide high accuracy on image recognition tasks while utilizing computational resources more effectively than traditional CNN models. In the base model, adjustments are made only to the classification layer, namely changing the number of classes according to the data currently used and adding dropouts with the desired values to prevent overfitting during the training process. This model uses the EfficientNetB0 architecture, which has been pre-trained using weights from the ImageNet1K dataset to utilize the initial features learned.

2.4. Grey wolf optimization

GWO is a metaheuristic algorithm that mimics the social behavior and hunting mechanisms of grey wolves in the wild [23]. This algorithm uses a leadership hierarchy consisting of alpha (paramount leader), beta (supporter), delta (helper), and omega (follower) to direct the search for optimal solutions [24]. GWO works by simulating three main stages in hunting, namely tracking, encirclement, and attack on prey, which are adapted to explore and exploit the solution space.

The social structure of grey wolves involves taking care of injured and weaker members, with omega wolves being the lowest-ranking individuals who follow the commands of the others. The hunting success of wolves is largely dependent on this social hierarchy. The social behavior of grey wolves can be mathematically modeled by considering the prey's location as the optimal solution, while the wolf's position in the search space represents a potential solution. Alpha wolves are considered the best solution as they are closest to the prey, with beta and delta wolves representing the next best solutions based on their social rank. In the search space, omega wolves adjust their position according to the positions of the alpha, beta, and delta wolves. The positions of the alpha, beta, delta, and omega wolves are denoted as $X_\alpha, X_\beta, X_\delta, X_\omega$, respectively. The main steps of GWO include prey encircling, hunting, attacking, and searching. Prey encircling is the process where wolves surround the prey during hunting, which is mathematically expressed through a set of equations.

The process of prey encirclement is the next stage after the prey is tracked. This process is mathematically [25] represented by (2) to (5):

$$\vec{D} = \left|\vec{C} \cdot \overrightarrow{X_p}(t) - \vec{X}(t)\right| \quad (2)$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \vec{A}\vec{D} \quad (3)$$

$$\vec{C} = 2 \cdot \vec{a} \cdot \overrightarrow{rand_1} - \vec{a} \quad (4)$$

$$\vec{A} = 2 \cdot \overrightarrow{rand_2} \quad (5)$$

where $\vec{X}(t)$, $\overrightarrow{X_p}(t)$, is the current position vector of the grey wolf and prey. The coefficients $\vec{A}$ and $\vec{C}$ are adaptive parameter vectors. The $rand_1$ dan $rand_2$ vectors are vectors with random values in the range [0,1]. Meanwhile, $\vec{a}$ is a vector whose value decreases gradually from 2 to 0 during iteration.

The alpha wolf, with the help of beta and delta wolves, guides the prey-hunting process. It is presumed that these three wolves have the best knowledge about the location of the prey, so they become the best search agents that help update the position of other wolves, mathematically [26] shown in (6) to (12).

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1} \cdot \overrightarrow{X_\alpha} - \vec{X}\right| \quad (6)$$

$$\overrightarrow{D_\beta} = \left|\overrightarrow{C_2} \cdot \overrightarrow{X_\beta} - \vec{X}\right| \quad (7)$$

$$\overrightarrow{D_\delta} = \left|\overrightarrow{C_3} \cdot \overrightarrow{X_\delta} - \vec{X}\right| \quad (8)$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \vec{A}_1 \cdot \overrightarrow{D_\alpha} \quad (9)$$

$$\overrightarrow{X_2} = \overrightarrow{X_\beta} - \vec{A}_2 \cdot \overrightarrow{D_\beta} \quad (10)$$

$$\overrightarrow{X_3} = \overrightarrow{X_\delta} - \vec{A}_3 \cdot \overrightarrow{D_\delta} \quad (11)$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \quad (12)$$

The attack phase on the prey is governed by the parameter $\vec{a}$, which regulates exploitation. When the prey halts, the moving wolf attacks the prey. The value $\vec{A}$ is a random value within the range $[-2r, 2r]$, where r is a random value between $[-1,1]$. The new search position is random between the wolf's current and the prey's positions. The attack condition is adequate if $|\vec{A}| < 1$.

The search or exploration process for an optimal solution is modeled based on the search behavior of wolves, where wolves disperse to search for prey and regroup when prey is found. Wolves disperse for better prey if $|\vec{A}| > 1$, and converge towards the prey when $|\vec{A}| < 1$. The random parameter $\vec{C}$ prevents the algorithm from getting stuck in local optima and supports exploration. This parameter provides random values not only in the initial stages of the algorithm but also in the final stages, thereby increasing the exploration capabilities without bias.

In this study, hyperparameter optimization aims to minimize prediction errors by calculating the inverse value of the accuracy of the validation data. The model is built using a combination of hyperparameters learning rate, weight decay, and dropout rate and then trained using training data. The objective function in this study can be seen in (13):

$$\min(\theta) = 1 - accuracy(\theta) \tag{13}$$

where θ is the optimized hyperparameter vector, and accuracy can be calculated by (14). The steps for carrying out GWO optimization are as follows [27]: i) initialize the grey wolf population as well as controlling parameters such as a, A, C, and the maximum number of iterations, ii) compute the fitness value for the initial population using the objective function, iii) set the best wolf as X_α, second best wolf X_β and the third best wolf as X_δ, iv) as long as the number of iterations is still insufficient, v) update all agents' current positions and a, A, C values. Recalculate the fitness values and update the values $X_\alpha, X_\beta, X_\delta$, vi) end the loop, and vii) return the value of X_α as the optimal solution.

2.5. Evaluation metrics

The confusion matrix aims to provide an overview of the performance of the classification model by displaying model predictions compared with the actual labels [28]. It serves as a fundamental tool for evaluating classification accuracy in machine learning. This matrix contains four elements: true positive (TP), which indicates when the model accurately predicts the positive class; true negative (TN), which indicates when the model accurately predicts the negative class; false positive (FP), which occurs when the model incorrectly predicts the positive class; and false negative (FN), which occurs when the model incorrectly predicts the negative class.

In evaluating the model, the metrics used are accuracy, precision, recall, and F1-score [29]–[31]. Accuracy refers to the ratio of correct predictions to the total number of predictions. As in (14) can be used to calculate accuracy. Next is precision, which measures the proportion of correct optimistic predictions out of all optimistic predictions. This metric can be calculated with (15). Next is recall or sensitivity, which calculates the proportion of positive data the model detects. Recall can be determined through calculations in (16). The last one is the F1-score, the harmonic mean between precision and recall, used to assess overall model performance, especially on imbalanced datasets. F1-score can be calculated via (17). The following are the equations for evaluating the model.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{14}$$

$$Precision = \frac{TP}{TP+FP} \tag{15}$$

$$Recall = \frac{TP}{TP+FN} \tag{16}$$

$$F1_{score} = \frac{2 \times Precision \times Recall}{Precision+Recall} \tag{17}$$

3. RESULTS AND DISCUSSION

In this research, the data was tested in two scenarios: a comparison of 80% training data, 10% validation, 10% test, and a comparison of 70% training data, 15% validation, and 15% test. Because the data is unbalanced, after dividing the data with the mentioned comparisons, class weighting is carried out on the training set. By using (1), the weighting values for each class are obtained, as shown in Table 1.

Table 1. Weighting for each class

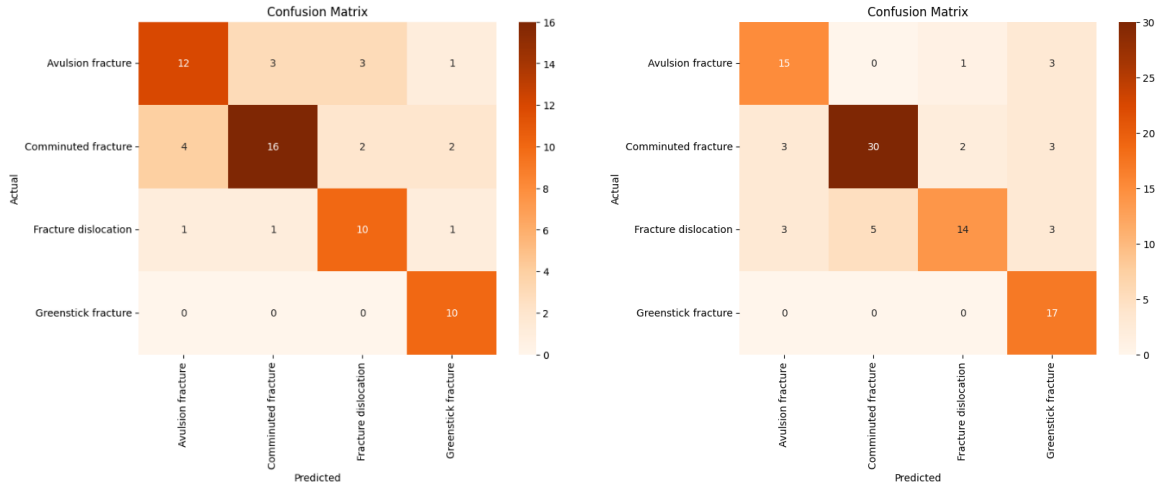
Ratio	Avulsion fracture	Comminuted fracture	Fracture dislocation	Greenstick fracture
80:10:10	1.1186	0.9167	0.8407	1.2110
70:15:15	1.1106	0.9094	0.8370	1.2419

In the first scenario (data comparison 80:10:10), the greenstick fracture class has the highest weight of 1.2110, which shows that it has fewer samples than the other classes. In contrast, the comminuted fracture class has the lowest weight of 0.9167, indicating that it is more dominant in the training data distribution. While the second scenario (data comparison 70:15:15) shows a slight change in the class weight values. The greenstick fracture class still has the highest weight of 1.2419, while the comminuted fracture class has the lowest weight of 0.9094. This reflects that despite the data comparison changes, the minority classes maintain significant weighting to increase the model's sensitivity to minority data. This class weighting is then applied to the model's objective function through weighted cross-entropy loss. Next, both comparison scenarios were tested on the EfficientNetB0 model, where the hyperparameters were determined manually and without optimization. Table 2 shows the specified hyperparameters.

Table 2. Defined hyperparameters

Hyperparameter	Value
Learning rate	1×10^{-4}
Weigh decay	1×10^{-4}
Dropout rate	5×10^{-1}

The combination of hyperparameters was chosen manually to get an initial picture of model performance before further optimization. The learning rate is determined to maintain the stability of the training process, while weight decay and dropout are utilized to reduce the risk of overfitting. Figure 3 is the confusion matrix that shows the distribution of model predictions on actual labels using the hyperparameters in Table 2. Figure 3(a) shows the classification results with a dataset split of 80:10:10, while Figure 3(b) shows the results with a dataset split of 70:15:15.



(a) (b)

Figure 3. Confusion matrix of model classification results based on hyperparameter configuration in Table 2 (a) results with data comparison 80:10:10 and (b) results with data comparison 70:15:15

Based on Figure 3, the confusion matrix shows that the model can recognize classes quite well even though not all predictions are correct. The class that was most correctly predicted, both on the 80:10:10 and 70:15:15 training sets, was the communicated fracture class. An evaluation using precision, recall, and F1-score metrics, shown in Tables 3 and 4, was carried out to provide a more detailed picture of the model's performance in each class for both comparisons.

The evaluation results show that the model performance before optimization still has room for improvement. For this reason, hyperparameter optimization was carried out using the GWO algorithm to improve model performance. The optimized hyperparameters are first defined within a specific range. These hyperparameters are shown in Table 5.

Table 3. Evaluation metrics in the 80:10:10 ratio

Class	Precision	Recall	F1-score
Avulsion fracture	0.71	0.63	0.67
Comminuted fracture	0.80	0.67	0.73
Fracture dislocation	0.67	0.77	0.71
Greenstick fracture	0.71	1.00	0.83

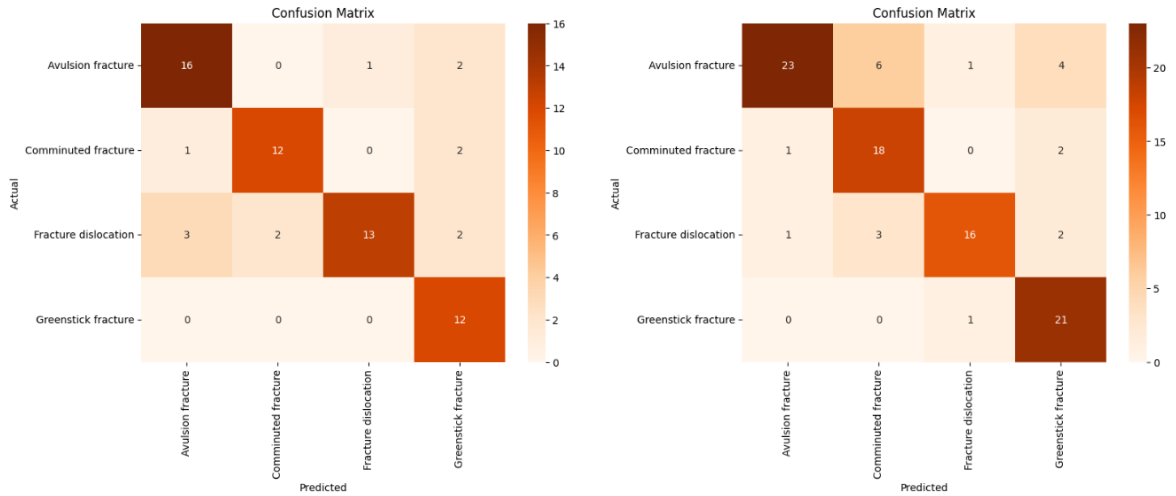
Table 4. Evaluation metrics in the 70:15:15 ratio

Class	Precision	Recall	F1-score
Avulsion fracture	0.71	0.79	0.75
Comminuted fracture	0.86	0.79	0.82
Fracture dislocation	0.82	0.56	0.67
Greenstick fracture	0.65	1.00	0.79

Table 5. Hyperparameter range

Hyperparameter	Value
Learning rate	$[1 \times 10^{-5}, 1 \times 10^{-2}]$
Weigh decay	$[1 \times 10^{-6}, 1 \times 10^{-3}]$
Dropout rate	$[1 \times 10^{-1}, 5 \times 10^{-1}]$

After training the model using the specified hyperparameter range, we obtained improved prediction accuracy. Figure 4 presents a comparison of model performance before and after optimization: Figure 4(a) for the 80:10:10 dataset split and Figure 4(b) for the 70:15:15 split. Meanwhile, Table 6 presents the optimal hyperparameter combinations that contributed to these improvements.



(a) (b)

Figure 4. Comparison of model performance after hyperparameter optimization, with (a) model performance using an 80:10:10 dataset split after optimization and (b) model performance using a 70:15:15 dataset split after optimization

Table 6. Best parameter combinations

Ratio	Learning rate	Weight decay	Dropout rate
80:10:10	9.5164×10^{-4}	1.6429×10^{-4}	1.994×10^{-1}
70:15:15	1.395×10^{-3}	4.758×10^{-4}	1.820×10^{-1}

The evaluation metrics in Tables 7 and 8 show an increase in model performance after hyperparameter optimization. Based on Tables 7 and 8, hyperparameter optimization improved model performance on both data divisions. In the 80:10:10 split, precision, recall, and F1-score each increased significantly, while in 70:15:15, the increase was minor, but the final result was still higher. Meanwhile, increasing accuracy can be seen in Figure 5.

Table 7. Rasio 80:10:10

Comparison	Precision	Recall	F1-score
Before optimization	0.73	0.73	0.72
After optimization	0.83	0.80	0.80

Table 8. Rasio 70:15:15

Comparison	Precision	Recall	F1-score
Before optimization	0.79	0.77	0.77
After optimization	0.82	0.79	0.79

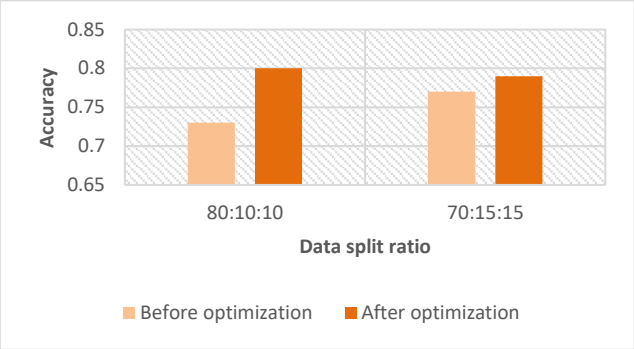


Figure 5. Accuracy comparison

In Figure 5, hyperparameter optimization positively impacts model performance in both data sharing scenarios, namely 80:10:10 and 70:15:15. In the 80:10:10 split, the accuracy increases from about 0.73 before optimization to 0.80 after optimization. This shows that the optimized combination of hyperparameters can enhance the model's capability to recognize data patterns. In the 70:15:15 split, the initial accuracy before optimization was 0.77, slightly higher than 80:10:10, and increased to 0.79 after optimization. Although the increase in 70:15:15 is minor, the final result still shows increased accuracy. Another experiment was conducted using MobileNet, which produced an accuracy 0.70. This demonstrated that EfficientNetB0 yields better result, with higher accuracy.

4. CONCLUSION

This study shows that the combination of EfficientNetB0 and GWO improves the model performance by 4.5% compared to the baseline model, especially on small and imbalanced datasets. The model can also be applied to larger datasets. Future development plans include: i) replacing the dataset with computed tomography (CT) scan or magnetic resonance imaging (MRI) images, ii) integrating basic visualization of classification decisions using gradient-weighted class activation mapping (Grad-CAM) or others, and iii) developing the model into a web application for automatic diagnosis by radiologists.

FUNDING INFORMATION

The authors declare that no funding was received for this research.

AUTHOR CONTRIBUTIONS STATEMENT

Author roles and contributions are presented in the following table, in accordance with the Contributor Roles Taxonomy (CRediT) guidelines.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Khusnul Khotimah	✓	✓	✓		✓	✓	✓	✓	✓		✓		✓	✓
Sugiyarto Surono		✓		✓	✓				✓	✓		✓	✓	
Aris Thobirin				✓	✓				✓	✓		✓	✓	

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data is available on Kaggle https://www.kaggle.com/datasets/pkdarabi/bone-break-classification-image-dataset/data.

REFERENCES

[1] S. Liu, W. Wang, L. Deng, and H. Xu, "Cnn-trans model: a parallel dual-branch network for fundus image classification," *Biomedical Signal Processing and Control*, vol. 96, Oct. 2024, doi: 10.1016/j.bspc.2024.106621.

[2] K. W. Goh *et al.*, "Comparison of activation functions in convolutional neural network for poisson noisy image classification," *Emerging Science Journal*, vol. 8, no. 2, pp. 592–602, Apr. 2024, doi: 10.28991/ESJ-2024-08-02-014.

[3] K. Man and J. Chahl, "A review of synthetic image data and its use in computer vision," *Journal of Imaging*, vol. 8, no. 11, Nov. 2022, doi: 10.3390/jimaging8110310.

[4] E. T. A. Albert, N. H. Bille, and N. M. E. Leonard, "A mathematical primer to classical deep learning," *Journal of Applied and Advanced Research*, vol. 9, pp. 15–25, Sep. 2024, doi: 10.21839/jaar.2024.v9.9169.

[5] A. Kaur and M. Kapoor, "An approach to recognize efficient deep learning model for pattern recognition," in *2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Mar. 2024, pp. 1–6, doi: 10.1109/ICRITO61523.2024.10522108.

[6] A. Lopes, F. P. dos Santos, D. de Oliveira, M. Schiezaro, and H. Pedrini, "Computer vision model compression techniques for embedded systems: a survey," *Computers & Graphics*, vol. 123, Oct. 2024, doi: 10.1016/j.cag.2024.104015.

[7] U. Samariya and R. K. Sonker, "Comparisons of image classification using LBP with CNN and ANN," *Journal of Applied Mathematics and Computation*, vol. 6, no. 3, pp. 343–346, Sep. 2022, doi: 10.26855/jamc.2022.09.006.

[8] S. Surono, M. Rivaldi, and N. Irsalinda, "Classification using u-net CN on multi-resolution CT scan image," *Fuzzy Systems and Data Mining X, A.J. Tallón-Ballesteros (Ed.)*, 2024, doi: 10.3233/FAIA241412.

[9] A. Meliboev, J. Alikhanov, and W. Kim, "Performance evaluation of deep learning based network intrusion detection system across multiple balanced and imbalanced datasets," *Electronics*, vol. 11, no. 4, Feb. 2022, doi: 10.3390/electronics11040515.

[10] F. A. Breve, "COVID-19 detection on chest X-ray images: a comparison of CNN architectures and ensembles," *Expert Systems with Applications*, vol. 204, Oct. 2022, doi: 10.1016/j.eswa.2022.117549.

[11] A. Sharma and D. Kumar, "Hyperparameter optimization in CNN: a review," in *2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Nov. 2023, pp. 237–242, doi: 10.1109/ICCCIS60361.2023.10425571.

[12] S. Surono, M. Y. F. Afitian, A. Setyawan, D. K. Eni Arofah, and A. Thobirin, "Comparison of CNN classification model using machine learning with bayesian optimizer," *HighTech and Innovation Journal*, vol. 4, no. 3, pp. 531–542, Sep. 2023, doi: 10.28991/HIJ-2023-04-03-05.

[13] M. Wojciuk, Z. Swiderska-Chadaj, K. Siwek, and A. Gertych, "Improving classification accuracy of fine-tuned CNN models: impact of hyperparameter optimization," *Heliyon*, vol. 10, no. 5, Mar. 2024, doi: 10.1016/j.heliyon.2024.e26586.

[14] C. J. Hellín, A. A. Olmedo, A. Valledor, J. Gómez, M. López-Benítez, and A. Tayebi, "Unraveling the impact of class imbalance on deep-learning models for medical image classification," *Applied Sciences*, vol. 14, no. 8, Apr. 2024, doi: 10.3390/app14083419.

[15] P. Jeevan and A. Sethi, "Which backbone to use: a resource-efficient domain specific comparison for computer vision,"*arXiv-Computer Science*, pp. 1–14, Jun. 2024, doi: 10.48550/arXiv.2406.05612.

[16] Y. C. Kuyu and N. Ozekmekci, "Grey wolf optimizer to the hyperparameters optimization of convolutional neural network with several activation functions," in *2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Oct. 2022, pp. 13–17, doi: 10.1109/ISMSIT56059.2022.9932838.

[17] L. V. Sari, R. P. Rosalin, and S. Uyun, "Classification fracture in X-ray images using VGG16 feature extraction and principal component analysis," *2024 12th International Conference on Cyber and IT Service Management, CITSM 2024*, pp. 1–6, 2024, doi: 10.1109/CITSM64103.2024.10775981.

[18] H. Min *et al.*, "Automatic classification of distal radius fracture using a two-stage ensemble deep learning framework," *Physical and Engineering Sciences in Medicine*, vol. 46, no. 2, pp. 877–886, 2023, doi: 10.1007/s13246-023-01261-4.

[19] L. Zou, H. F. Lam, and J. Hu, "Adaptive resize-residual deep neural network for fault diagnosis of rotating machinery," *Structural Health Monitoring*, vol. 22, no. 4, pp. 2193–2213, Jul. 2023, doi: 10.1177/14759217221122266.

[20] M. Tan and Q. V. Le, "EfficientNetV2: smaller models and faster training," *Proceedings of Machine Learning Research*, vol. 139, pp. 10096–10106, Apr. 2021, doi: 10.48550/arXiv.2104.00298.

[21] G. Zhang and W. Abdulla, "Optimizing hyperspectral imaging classification performance with CNN and batch normalization," *Applied Spectroscopy Practica*, vol. 1, no. 2, Sep. 2023, doi: 10.1177/27551857231204622.

[22] L. T. Duong, P. T. Nguyen, C. Di Sipio, and D. Di Ruscio, "Automated fruit recognition using EfficientNet and MixNet," *Computers and Electronics in Agriculture*, vol. 171, Apr. 2020, doi: 10.1016/j.compag.2020.105326.
[23] A. Aljohani, N. Alharbe, R. E. Al Mamlook, and M. M. Khayyat, "A hybrid combination of CNN attention with optimized random forest with grey wolf optimizer to discriminate between Arabic hateful, abusive tweets," *Journal of King Saud University - Computer and Information Sciences*, vol. 36, Feb. 2024, doi: 10.1016/j.jksuci.2024.101961.
[24] Q. Xie, Z. Guo, D. Liu, Z. Chen, Z. Shen, and X. Wang, "Optimization of heliostat field distribution based on improved gray wolf optimization algorithm," *Renewable Energy*, vol. 176, pp. 447–458, Oct. 2021, doi: 10.1016/j.renene.2021.05.058.
[25] R. Mohakud and R. Dash, "Designing a grey wolf optimization based hyper-parameter optimized convolutional neural network classifier for skin cancer detection," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 8, pp. 6280–6291, Sep. 2022, doi: 10.1016/j.jksuci.2021.05.012.
[26] P. M. Kitonyi and D. R. Segera, "Hybrid gradient descent grey wolf optimizer for optimal feature selection," *BioMed Research International*, vol. 2021, no. 1, Jan. 2021, doi: 10.1155/2021/2555622.
[27] G. Wolf and O. Gwo, *Advanced optimization by nature-inspired algorithms*, vol. 720. Singapore: Springer Singapore, 2018, doi: 10.1007/978-981-10-5221-7.
[28] M. C. Neves, J. Filgueiras, Z. Kokkinogenis, M. C. F. Silva, J. B. L. M. Campos, and L. P. Reis, "Enhancing experimental image quality in two-phase bubbly systems with super-resolution using generative adversarial networks," *International Journal of Multiphase Flow*, vol. 180, Nov. 2024, doi: 10.1016/j.ijmultiphaseflow.2024.104952.
[29] P. I. Ritharson, K. Raimond, X. A. Mary, J. E. Robert, and A. J, "DeepRice: a deep learning and deep feature based classification of rice leaf disease subtypes," *Artificial Intelligence in Agriculture*, vol. 11, pp. 34–49, Mar. 2024, doi: 10.1016/j.aiia.2023.11.001.
[30] Y. Wang *et al.*, "PGKD-Net: prior-guided and knowledge diffusive network for choroid segmentation," *Artificial Intelligence in Medicine*, vol. 150, 2024, doi: 10.1016/j.artmed.2024.102837.
[31] D. K. Saha, A. M. Joy, and A. Majumder, "YoTransViT: a transformer and CNN method for predicting and classifying skin diseases using segmentation techniques," *Informatics in Medicine Unlocked*, vol. 47, 2024, doi: 10.1016/j.imu.2024.101495.

BIOGRAPHIES OF AUTHORS

Khusnul Khotimah is an undergraduate student of Mathematics at Universitas Ahmad Dahlan. Her academic interests include statistics, machine learning, and deep learning. Currently, she focuses on computer vision but remains open to exploring other fields in artificial intelligence technology. She can be contacted at email: khusnul2100015009@webmail.uad.ac.id.

Sugiyarto Surono is a professor and senior lecturer in Mathematics at Universitas Ahmad Dahlan. He obtained his master's degree in Mathematics from Universitas Gadjah Mada and earned his Ph.D. from Universiti Malaysia Terengganu. His areas of expertise include optimization and control, queuing theory, fuzzy mathematics, machine learning, and deep learning. As a professor, he actively teaches and mentors students, integrating various scientific disciplines with data science applications. His goal is to raise students' awareness of the importance of mathematical science in technological development. Through this approach, he aims to make a significant contribution to preparing the younger generation for future technological challenges. He can be contacted at email: sugiyarto@math.uad.ac.id.

Aris Thobirin is a senior lecturer in Mathematics at Universitas Ahmad Dahlan. He earned his master's degree in Mathematics from Universitas Gadjah Mada. His areas of expertise include analysis and algebra, particularly in applying algebraic theory to problem-solving. He is also interested in the implementation of big data and machine learning. In addition, he actively provides training on innovative learning models. He can be contacted at email: aris.thobi@math.uad.ac.id.