

# DETERMINATION OF AUGMENTATION METHOD IN CONVOLUTION NEURAL NETWORK USING FOX OPTIMIZATION FOR FISH IMAGE CLASSIFICATION

FIRMAN WAHYUDI<sup>1</sup>, MOCH. ARIEF SOELEMEN<sup>2</sup>, RICARDUS ANGGI PRAMUNENDAR<sup>3\*</sup>, PULUNG NURTANTIO ANDONO<sup>4</sup>

<sup>1</sup>Program Studi Magister Teknik Informatika, Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, Semarang, Indonesia, 50131

E-mail: <sup>1</sup>p31202202500@mhs.dinus.ac.id, <sup>2</sup>m.rief.soeleman@dsn.dinus.ac.id,

<sup>3</sup>ricardus.anggi@dsn.dinus.ac.id, <sup>4</sup>pulung@dsn.dinus.ac.id

## ABSTRACT

The automatic identification of fish species plays a crucial role in various fields such as conservation biology, fisheries management, and biological research. Convolutional Neural Network (CNN) methods have become an effective solution for automating this process from digital images. However, achieving high accuracy requires careful consideration of factors such as the amount and quality of data, image processing methods, feature extraction techniques, classification algorithms, and optimization methods. This study addresses these challenges by proposing a CNN model optimized using the FOX algorithm to select the best augmentation method. The results show that selecting the appropriate augmentation techniques, such as Kmeans Color Quantization, Horizontal Flip, Voronoi, Elastic Transformation, and Contrast Normalization, can significantly improve fish species recognition accuracy up to 98.75% during training. The proposed model also demonstrated strong generalization capability, with a validation accuracy of 96.90%, indicating minimal overfitting. Although it requires intensive training time, this approach proves highly effective for applications demanding high accuracy and good generalization, thus enhancing the understanding and management of marine ecosystems in support of sustainable fishing practices.

**Keywords:** *Fish Image Classification, Convolutional Neural Network, Augmentation, Feature Selection, FOX Optimization Algorithm*

## 1. INTRODUCTION

The development of species identification techniques, including fish identification, plays a crucial role in various fields such as nature conservation, fisheries management, and biological research. Accurate fish species recognition is a vital first step in understanding marine ecosystems and ensuring the sustainability of fishery resources. With advances in technology, computational approaches, particularly Convolutional Neural Networks (CNN), have proven to be an effective solution for automating the fish species identification process from digital images [1].

Several studies have explored the automation of fish species recognition using artificial intelligence techniques, with the majority relying on CNN methods. However, differences in recognition performance across studies indicate that system effectiveness is highly dependent on factors such as the quantity and quality of data, image processing methods [2], feature extraction techniques [3]–[9],

the classification methods used, and optimization strategies to achieve optimal performance [10]. While many studies utilize underwater environmental data, few consider the impact of these environments when developing methods [11].

There is a limited number of studies that leverage data affected by underwater environmental conditions. Salman et al. [12] proposed a CNN-based approach combined with an SVM classifier to recognize 10 fish species, using 19,868 images from the fish4knowledge dataset, achieving an accuracy of 93.65%. A similar study by Qin et al. [13], using the same methods and dataset, attained an accuracy of 98.13% with 11,724 images from 23 fish species. Siddiqui et al. [1], unlike Salman et al. [12] and Hsiao et al. [14], applied a general feature detector before using CNN and used data influenced by underwater conditions rather than fish4knowledge data. Siddiqui et al. [1] compared their results with previous studies and achieved an accuracy of 94.3%, surpassing the performance of both Salman et al. [12] and Hsiao et al. [14].

Key challenges frequently discussed in fish species recognition include data heterogeneity and the influence of underwater environmental conditions on the data [1], [12]–[14]. The fish4knowledge dataset has been widely used due to its size and diversity, yet many methods do not adequately address the impact of underwater environments. Typically, these approaches involve feature extraction followed by classification methods for fish species recognition. Enhancing image quality and improving feature processing are critical for boosting recognition accuracy. Thus, both stages are essential to achieving high accuracy in fish species identification.

This study emphasizes the importance of image quality in improving fish identification accuracy. The proposed method utilizes a CNN optimized with the FOX algorithm for augmentation selection, carefully considering the impact of underwater environmental conditions. This results in a more robust and practical model for real-world applications, enhancing our understanding of marine ecosystems. The implications of this research extend to fields such as environmental monitoring, fishery resource conservation, and sustainable fisheries management, supporting the blue economy concept, which emphasizes the balance between marine resource use and conservation.

## 2. RELATED RESEARCH

Various studies [11], [12], [14]–[19] have examined fish species recognition, though few have utilized data influenced by underwater environmental conditions, such as in the research by Spampinato et al. [15], [18], Hsiao et al. [14], Boom et al. [17], Huang et al. [19], Salman et al. [12], Qin et al. [13], and Siddiqui et al. [1]. Spampinato et al. [15] employed artificial intelligence to recognize 13 fish species from 3,179 fish4knowledge images. Shape features were extracted using the invariant moment method, while texture features were derived from HOG, GLCM, Gabor filters, and Fourier descriptors. These feature combinations were classified using SVM, resulting in an accuracy of 86.32%. Boom et al. [17] used similar features and data but proposed BGOT classification based on SVM to address data imbalance, achieving a performance of 97.21%, surpassing Spampinato et al. [15]. Boom et al.'s [17] work was continued by Huang et al. [19], who added a Gaussian Mixture model (GMM). Tested on the fish4knowledge dataset with 24,150 images of 15 fish species, Huang et al. [19] achieved a 95% accuracy rate.

Hsiao et al. [14] used the fish4knowledge dataset with modifications to eigenfaces and fisherfaces, followed by sparse representation-based classification, achieving an accuracy of 81.8% on 1,000 images of 25 fish species. However, this study [14] did not compare its results with other classification methods from previous research. Spampinato et al. [18] proposed a different approach using SIFT and LTP feature extraction methods with SVM classification on the fish4knowledge dataset. The data consisted of 24,441 training images and 6,956 test images, achieving 91% precision. Salman et al. [12] utilized CNN and SVM to recognize 10 fish species from 19,868 fish4knowledge images, reaching 93.65% accuracy. Qin et al. [13] employed the same methods and data, achieving 98.13% accuracy from 11,724 fish images. Siddiqui et al. [1] applied a generalized feature detector before CNN on underwater data, achieving an accuracy of 94.3%, surpassing Salman et al. [12] and Hsiao et al. [14].

The key challenges in fish species recognition include data heterogeneity and the effects of underwater environments. Common methods involve feature extraction and classification but often fail to account for the impact of underwater conditions. Improving image quality and feature extraction processes is crucial for enhancing fish species recognition accuracy, as both are interrelated. Therefore, this study focuses on improving image quality through the selection of the best augmentation method.

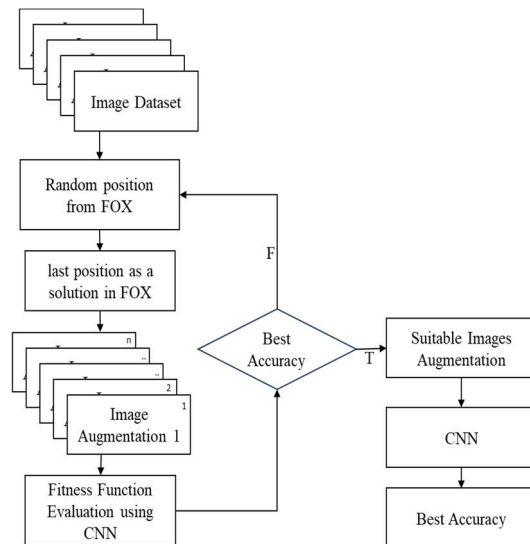


Figure 1: Proposed Research

## 3. PROPOSED RESEARCH

4. This study presents a novel approach to fish species identification by integrating a Convolutional

Neural Network (CNN) with the FOX optimization algorithm to select the most effective augmentation method, where the CNN evaluates the results of the FOX algorithm's fitness function based on accuracy, aiming to enhance fish recognition performance through the combination of advanced AI techniques and swarm-based optimization, as depicted in Fig 1.

### 3.1 Data Collection

Fish image data is collected from the public fish4knowledge dataset. This dataset includes 27,370 verified images extracted from live video, divided into 23 image labels [20]. The images depict variations in components, number, and shape of fins. The dataset is highly imbalanced, with the most frequently detected species appearing approximately 1,000 times more often than the least detected species [21]. Table 1 and 2 illustrates an overview of the fish4knowledge dataset.

Table 1. Dataset Fish4Knowledge



### 3.2 Augmentation

This study utilizes 60 augmentation methods from the Python "imgaug" package, including horizontal flipping, random cropping, and color space augmentation [22]. Only a specific subset of various image processing techniques from the library is employed in this research. The best augmentation methods are selected from Table 3 using the FOX Optimization Algorithm, which identifies the top five combinations from all available options.

Table 2. Dataset Fish4Knowledge

| ID                            | Species                          | Total images |
|-------------------------------|----------------------------------|--------------|
| 1                             | <i>Dascyllus reticulatus</i>     | 12112        |
| 2                             | <i>Plectroglyphidodon dickii</i> | 2683         |
| 3                             | <i>Chromis chrysur</i>           | 3593         |
| 4                             | <i>Amphiprion clarkii</i>        | 4049         |
| 5                             | <i>Chaetodon lunulatus</i>       | 2534         |
| 6                             | <i>Chaetodon trifascialis</i>    | 190          |
| 7                             | <i>Myripristis kuntee</i>        | 450          |
| 8                             | <i>Acanthurus nigrofuscus</i>    | 218          |
| 9                             | <i>Hemigymnus fasciatus</i>      | 241          |
| 10                            | <i>Neoniphon sammara</i>         | 299          |
| 11                            | <i>Abudefduf vaigiensis</i>      | 98           |
| 12                            | <i>Canthigaster valentini</i>    | 147          |
| 13                            | <i>Pomacentrus moluccensis</i>   | 181          |
| 14                            | <i>Zebrasoma scopas</i>          | 90           |
| 15                            | <i>Hemigymnus melapterus</i>     | 42           |
| 16                            | <i>Lutjanus fulvus</i>           | 206          |
| 17                            | <i>Scolopsis bilineata</i>       | 49           |
| 18                            | <i>Scaridae</i>                  | 56           |
| 19                            | <i>Pempheris vanicolensis</i>    | 29           |
| 20                            | <i>Zanclus cornutus</i>          | 21           |
| 21                            | <i>Neoglyphidodon nigroris</i>   | 16           |
| 22                            | <i>Balistapus undulatus</i>      | 41           |
| 23                            | <i>Siganus fuscescens</i>        | 25           |
| * source: Fish4Knowledge [20] |                                  |              |

### 3.3 FOX Optimization Algorithm

The FOX optimization algorithm begins by initializing a population X, which represents the positions of the foxes. The fitness of each agent is calculated using a benchmark function. The BestFitness and BestX values are determined by comparing the fitness of all agents during each iteration. To maintain a balance between exploration and exploitation, a random variable is used to allocate iterations between these two phases within

the FOX algorithm. This variable assigns a 50% probability for either exploration or exploitation, helping to prevent the algorithm from getting trapped in local optima. The variable *aaa* is used to enhance the search performance by reducing its value with each iteration based on BestX. Position updates are influenced by the fitness function, which helps agents avoid local optima. If the new position remains unchanged, the exploration phase is disabled, allowing the exploitation phase to take over, ensuring efficient convergence.

Table 3. Augmented Method

| No | Methods                           | No | Methods                  |
|----|-----------------------------------|----|--------------------------|
| 1  | Affine (Rotasi, Skala, Translasi) | 31 | Speckle Noise            |
| 2  | Horizontal Flip                   | 32 | Glass Blur               |
| 3  | Vertical Flip                     | 33 | Defocus Blur             |
| 4  | Contrast Normalization            | 34 | Zoom Blur                |
| 5  | Elastic Transformation            | 35 | Fog                      |
| 6  | Kmeans Color Quantization         | 36 | Frost                    |
| 7  | Superpixels                       | 37 | Snow                     |
| 8  | Voronoi                           | 38 | Spatter                  |
| 9  | Uniform Voronoi                   | 39 | Contrast                 |
| 10 | Regular Grid Voronoi              | 40 | Brightness               |
| 11 | Relative Regular Grid Voronoi     | 41 | Saturate                 |
| 12 | Gaussian Blur                     | 42 | Pixelate                 |
| 13 | Average Blur                      | 43 | Elastic Transform        |
| 14 | Median Blur                       | 44 | Solarize                 |
| 15 | Motion Blur                       | 45 | Equalize                 |
| 16 | Mean Shift Blur                   | 46 | Auto contrast            |
| 17 | Gamma Contrast                    | 47 | Enhance Color            |
| 18 | Sigmoid Contrast                  | 48 | Enhance Contrast         |
| 19 | Log Contrast                      | 49 | Enhance Brightness       |
| 20 | Linear Contrast                   | 50 | Enhance Sharpness        |
| 21 | All Channels Clahe                | 51 | Filter Blur              |
| 22 | Clahe                             | 52 | Filter Smooth            |
| 23 | Histogram Equalization            | 53 | Filter Smooth More       |
| 24 | Sharpen                           | 54 | Filter Edge Enhance      |
| 25 | Emboss                            | 55 | Filter Edge Enhance More |
| 26 | Edge Detect                       | 56 | Filter Find Edges        |

| No | Methods              | No | Methods        |
|----|----------------------|----|----------------|
| 27 | Directed Edge Detect | 57 | Filter Contour |
| 28 | Gaussian Noise       | 58 | Filter Emboss  |
| 29 | Shot Noise           | 59 | Filter Sharpen |
| 30 | Impulse Noise        | 60 | Filter Detail  |

### 3.3.1 Exploitation

During the exploitation phase, if the random variable *p* is greater than 0.18, the fox searches for a new position to capture its prey. For this, the distance traveled by the sound  $DistST_{it}$  the distance between the fox and the prey  $DistFoxPrey_{it}$ , and the jump value *Jump*. The sound travel time  $TimeST_{it}$  is determined randomly within the range of 0 to 1. The sound distance from the fox is calculated by multiplying the speed of sound in air ( $SpS = 343$  m/s) by  $TimeST_{it}$  [23], as shown in Equation (1). The number of iterations ranges from 1 to 500. Another equation calculates the speed of sound *SpS* based on the best position  $BestPosition_{it}$  in the search population, as per Equation (2).

$$DistST_{it} = SpS \times TimeST_{it} \tag{1}$$

$$SpS = \left( \frac{BestPosition_{it}}{TimeST_{it}} \right) \tag{2}$$

To calculate the sound distance, Equation (3) is used. As a result, the distance between the fox and its prey,  $DistFoxPrey_{it}$  can be computed using  $DistST_{it}$  in Equation (4). In physics, when calculating the distance between a sensor and an object, the sound distance is halved because the sensor is positioned halfway along the path traveled by the sound wave [23]. The sensor both sends and receives the sound wave signal, so the sound travel time is multiplied by 0.5 or divided by two. This operation accounts for the fact that only half of the sound wave's journey is relevant..

$$DistFoxPrey_{it} = DistST_{it} \times 0.5 \tag{3}$$

$$DistST_{it} = SpS \times TimeST_{it} \tag{4}$$

After determining the distance between the fox and its prey, the fox needs to calculate the jump height  $Jump_{it}$  using Equation (5) to find a new position that allows it to leap and capture the prey.

$$Jump_{it} = 0.5 \times 9.81 \times t^2 \tag{5}$$

The acceleration due to gravity is 9.81, and *t* represents the average sound travel time, squared due to the jump step. The transition time  $t_t$  is calculated from  $TimeST_{it}$  with *t* being half of  $t_t$  multiplied by 0.5. Gravity and *t* are then multiplied by 0.5 to calculate the jump height *Jump* along with the distance between the fox and the prey  $DistFoxPrey_{it}$  and a constant  $c_1$ .

The value 9.81 represents the acceleration due to gravity, and *t* is the average time taken by the sound wave, squared because of the upward and downward

steps involved in the jump. The transition time  $t_i$  is calculated by dividing the sum of  $TimeST_{it}$  by the number of dimensions. Equation (8) shows the calculations for  $t_i$  and  $MinT$ . The average time  $t$  is found by dividing  $tt$  by 2. Gravity and the average time are multiplied by 0.5 because the jump requires two different times for ascent and descent. The jump value  $Jump$  is then multiplied by the distance between the fox and prey  $DistFoxPrey_{it}$  and a constant  $c_1$ . The variable  $c_1$  ranges between  $[0, 0.18]$  when the fox jumps in the northeast direction..

$$X_{it+1} = DistFoxPrey_{it} \times Jump_{it} \times c_1 \quad (6)$$

Equations (6) and (7) are used to determine the fox's new location. Only one of these equations is executed in each iteration, depending on the condition of  $p$ . The main difference lies in the second part of the condition for  $p$  in Equation (5). If  $p > 0.18$ , Equation (5) is multiplied by  $c_2$  instead of  $q$ . If  $p \leq 0.18$ , the new position is calculated using Equation (7). The range of  $c_2$  is between  $[0.19, 1]$ .

$$X_{it+1} = DistFoxPrey_{it} \times Jump_{it} \times c_2 \quad (7)$$

The values of  $c_1$  and  $c_2$  are 0.18 and 0.82 respectively. These values are used in the fox's jump movement, either toward the northeast or in the opposite direction. If  $p > 0.18$ , the fox jumps toward the northeast, with  $DistFoxPrey_{it}$  and  $Jump_{it}$  multiplied by  $c_1$ , increasing the chances of reaching the global optimal position. However,  $p \leq 0.18$ , the fox jumps in the opposite direction from the northeast. In this case  $DistFoxPrey_{it}$  and  $Jump_{it}$  are multiplied by  $c_2$ , reducing the chances of reaching the prey ( $\leq 18\%$ ).

### 3.3.2 Exploration

The fox performs a random search based on the best position it has found. In this phase, the fox moves randomly to explore the search space for prey without using the jump technique. The variables  $MinT$  and  $a$  are used to control the random movement toward the fox's best position. Equations (8) and (9) describe the calculations of  $MinT$  and  $a$ , with  $MinT$  determined as the minimum value of  $t_i$ .

$$tt = \frac{\sum (TimeST_{it}(i,:))}{dimension} = Min(t_i), \quad MinT \quad (8)$$

The average  $TimeST_{it}$  is calculated based on the dimensions of the problem to find the minimum value of the average time  $t_i$ .

$$a = 2 \times \left( it - \left( \frac{1}{Max_{it}} \right) \right) \quad (9)$$

$Max_{it}$  represents the maximum number of iterations. The calculation of  $MinT$  and the variable

$a$  is crucial for the search phase toward the optimal solution. The fox uses  $rand(1, dimension)$  for stochastic exploration while searching for prey. The variable  $r$  is employed to balance exploration and exploitation. The best solution  $BestX_n$  significantly influences the fox's exploration strategy. Equation (9) describes the fox's exploration technique when searching for a new position within the search space  $X(i,:)$  which can be adapted to existing algorithms or used to develop new metaheuristic algorithms.

$$X_{it+1} = BestX_{it} \times rand(1, dimension) \times MinT \times a \quad (10)$$

The equations in both phases do not require modification, except for adjustments to suit specific problems when the fox is used to solve multidimensional space problems. Full details about the FOX algorithm can be found in Algorithm 1.

#### Algorithm 1. FOX

1. Initialize the red fox population  $X_i (i = 1, 2, \dots, n)$
2. While  $i < Max(it)$ :
  - Inialisasi variabel:  $DistST, SpS, TimeST, BestX, DistFoxPrey, Jump, MinT, a, BestFitness$
  - Calculate the fitness value for each search agent.
  - Select  $BestX$  and  $BestFitness$  among the fox population ( $X$ ) at each iteration.
  - if  $fitness_{it} > fitness_{it+1}$ 
    - $BestFitness = fitness_{it+1}$
    - $BestX = X(i, :)$
  - if  $r \geq 0.5$ 
    - if  $p \geq 0.18$ 
      - Initialize time randomly.
      - Calculate DistanceSoundtravels (1).
      - Calculate SpS (2).
      - Calculate the distance from the fox to the prey (3).
      - $T_t$  is the average time.
      - $T = \frac{T_t}{2}$
      - Hitung lompatan  $jump$  (5).
      - Temukan  $X_{it+1}$  (6).
    - If  $p < 0.18$ 
      - Initialize time randomly.
      - Calculate DistanceSoundtravels (1).
      - Calculate SpS from Equation (2).
      - Calculate the distance from the fox to the prey (3).
      - $T_t$  is the average time.
      - $T = \frac{T_t}{2}$
      - Calculate the  $jump$  (5).
      - Find  $X_{it+1}$  (7).
  - If  $r < 0.5$ 
    - Find  $MinT$  (8)
    - Explore  $X_{it+1}$  (10)
  - Check and correct the position if it exceeds the boundaries.

- Evaluate the search agents based on their fitness values.
  - Update BestX.
  - $i_t = i_t + 1$
3. Return *BestX* and *BestFitness*

FOX begins by randomly initializing the population of red foxes and selecting the best position based on the best fitness value within the population. Each iteration consists of either an exploitation or exploration phase, depending on the random value  $r$  and the condition  $p$ . The time complexity of FOX per iteration is  $O(n^2)$ , where  $n$  is the population size, with the same space complexity for the vectors and matrices in Algorithm 1.

### 3.4 Convolutional Neural Network as an Evaluation Method

This study employs the FOX approach to determine the optimal augmentation technique, utilizing Convolutional Neural Networks (CNN) to compute the fitness function based on accuracy. CNN, a mathematical model with a parametric design, consists of an input layer, several hidden layers, and an output layer [24], [25]. Each hidden layer is connected by adjustable weights and progressively represents increasingly complex aspects of the input image [26]. The traditional CNN framework includes convolutional layers, pooling layers, and fully connected layers [24], which together transform the initial input representation into a higher and more conceptual level. The CNN method used in this study is illustrated in Fig 2.

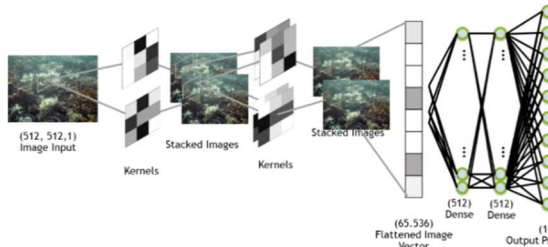


Figure 2. CNN Model as an Evaluation Method

CNN enhances useful input features while reducing irrelevant variations. Compared to feedforward neural networks, CNNs have fewer connections and parameters, making them easier to train. However, their optimal performance might be slightly lower. The capacity of CNNs can be adjusted by configuring their width and depth.

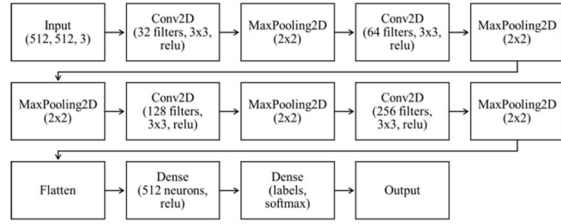


Figure 3. Model CNN

#### 3.4.1 CNN Model

The CNN model for fish classification accepts RGB images with a resolution of 128 x 128 pixels and three color channels. It consists of several sequential Conv2D and MaxPooling2D layers. The first Conv2D layer has 32 filters (3x3) with the ReLU activation function, followed by a MaxPooling2D layer (2x2). This process is repeated four times with filters of 64, 128, 128, and 256, respectively, with a MaxPooling2D layer after each Conv2D. The output is then flattened into a one-dimensional structure using a flatten layer. A dense layer with 512 neurons, using ReLU activation, follows. The final dense layer contains 23 neurons, using the softmax activation function (as shown in Fig 3). This structure enhances the model's ability to extract relevant features for classification.

#### 3.4.2 ReLU Function

The ReLU activation function is used in neural networks because it is simple and efficient, effectively addressing the "vanishing gradient" problem present in other functions like sigmoid or tanh [27]. However, ReLU has a drawback known as the "dying neuron" issue, where neurons produce a constant zero output, which hinders learning. The equation for ReLU is shown in (9).

$$f(x) = \max(0, x) \tag{9}$$

#### 3.5 Performance Evaluation

The performance of the classification algorithm is evaluated by calculating accuracy. Accuracy refers to the correct and precise categorization of all the obtained data [28]. The accuracy value is calculated using Equation (10), where  $t$  is the number of correctly identified sample data points, and  $n$  is the total number of sample data points.

$$accuracy = (t/n) \times 100 \tag{10}$$

#### 3.6 Research Design

This research includes the implementation and optimization stages of the fish identification model. The implementation follows guidelines using hardware such as an Intel Core i9 CPU, Nvidia GeForce RTX 3060 Ti graphics card, and 64GB





RAM. Optimization is performed to evaluate the performance results of the selected augmentation methods based on epochs, aiming to improve accuracy and efficiency. Model validation is conducted to test the results of the chosen optimal augmentation method. Accuracy is the primary metric used to evaluate the proposed method.

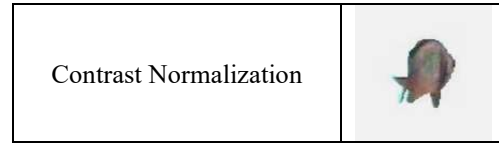
## 5. RESULTS AND DISCUSSION

### 4.1 Research Results

In this experiment, fish image classification was performed using Convolutional Neural Networks (CNN) on a dataset comprising 23 different species. Augmentation methods were applied to improve the quality of segmented images, while the FOX optimization method was used to select the best combination of various available augmentation techniques. The experimental results show that augmentation methods such as Kmeans Color Quantization, Horizontal Flip, Voronoi, Elastic Transformation, and Contrast Normalization achieved an accuracy of 98.75% on the training data. On the other hand, augmentation methods like Frost, Elastic Transformation, FilterEdgeEnhance, ShotNoise, and MotionBlur only reached an accuracy of 97.73% on the same data. This difference highlights the importance of selecting the appropriate augmentation method to improve the accuracy of the developed model. The performance improvements from each selected augmentation are shown in Tables 3 and 4. These results were used to combine augmentation methods in CNN classification.


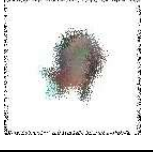
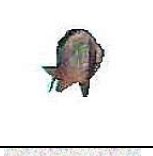
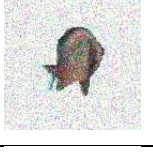
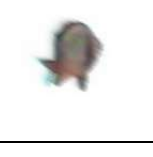
Table 3. Selected Best Augmentations

|                           |   |
|---------------------------|---|
| Kmeans Color Quantization |  |
| Horizontal Flip           |  |
| Voronoi                   |  |
| Elastic Transformation    |  |



In addition to presenting the augmentation results for each image, the performance accuracy for training is shown in Figure 4. The figure demonstrates that the model consistently improved in accuracy throughout the iterative process. Each iteration involved ten different trials, with the best result occurring in the eighth iteration. Initially, accuracy started at 98.10% in the first iteration, with slight variations in the early iterations, such as 98.28% in the second and third iterations, and a small drop to 98.20% in the fourth iteration. Although there was a decrease to 97.73% in the fifth iteration, the model recovered and showed significant improvement in subsequent iterations, reaching 98.75% in the eighth iteration. The ninth and tenth iterations demonstrated stable accuracy levels of 98.55% and 98.56%, respectively. Analysis indicates that the model tends to stabilize with an overall trend of increasing accuracy, despite minor fluctuations in some iterations.

Table 4. Selected Lowest Performing Augmentations

|                        |   |
|------------------------|---|
| Frost                  |  |
| Elastic Transformation |  |
| Filter Edge Enhance    |  |
| Shot Noise             |  |
| Motion Blur            |  |

The experimental results in Figure 5 indicate that the model maintained relatively stable variations in validation accuracy over the ten iterations conducted. Starting at 96.43% in the first iteration, the accuracy showed minor fluctuations during the initial iterations, reaching a peak of 97.06% in the second iteration. Small variations were observed in some iterations, such as 96.40% in the fourth iteration and 96.80% in the sixth iteration. Overall, despite these minor fluctuations, the model consistently maintained accuracy within the range of 96.40% to 97.06% throughout the experiment. This analysis suggests that the model exhibits stable performance with minimal variation, demonstrating good consistency in validation results across iterations.



Figure 4. Best Training Results for Each Iteration



Figure 5. Best Validation Results for Each Iteration

The difference between training accuracy (Fig 4) and validation accuracy (Fig 5), though not significant, suggests that the model may experience slight overfitting to the training data, especially considering that validation accuracy tends to be stable and not far from training accuracy. However, an ANOVA analysis revealed a significant difference between training and validation accuracy ( $F = 267.504$ ,  $p < 0.05$ ), indicating that the model tends to overfit the training data. The variability between iterations was not statistically significant ( $F = 2.321$ ,  $p > 0.05$ ), suggesting consistency in the model's performance across iterations.

#### 4.2 Discussion

From the experimental results, it is evident that the FOX optimization method successfully demonstrates its ability to combine various data augmentation techniques and achieve optimal

accuracy performance on the CNN model. Specifically, this method reached the highest training accuracy of 98.75%, indicating that the model was able to learn very effectively from the available training data. Furthermore, the validation accuracy of 96.90% shows that the model has excellent generalization capabilities on new, unseen data during training. All training time required to achieve these results was 103,413.83 seconds, indicating that the FOX method is computationally intensive. However, the performance results justify the long training time, as the high accuracy achieved proves the effectiveness of the method.

The small difference between training and validation accuracy suggests that the model does not suffer from significant overfitting, a condition that is often avoided in machine learning model training. The success of the FOX method in achieving ideal performance, where the gap between training and validation accuracy is minimal, makes it a highly effective approach for CNN model development. Therefore, despite the long training time, this method is highly recommended for applications that require high accuracy and strong generalization. This further reinforces the idea that the FOX optimization method excels not only in integrating augmentation techniques but also in producing a model that delivers consistent and reliable accuracy performance.

#### 4. CONCLUSION

The development of fish species identification techniques through computational approaches like Convolutional Neural Networks (CNN) offers an effective solution for automating this process from digital images. This study demonstrates that the FOX optimization method successfully identifies augmentation techniques that achieve top performance, such as Kmeans Color Quantization, Horizontal Flip, Voronoi, Elastic Transformation, and Contrast Normalization, which improved fish species recognition accuracy to 98.75% during training. However, other augmentation methods only reached 97.73% accuracy, highlighting the importance of selecting the right augmentation methods to enhance model performance. The results also show that the CNN model optimized by FOX for augmentation selection achieved consistent accuracy between training (98.75%) and validation (96.90%), with minimal performance degradation between these stages. Although this approach requires intensive training time, it has great potential for applications in fields such as nature conservation, fisheries management, and biological research, strengthening our understanding of marine



ecosystems and supporting sustainable fisheries resource management practices.

#### ACKNOWLEDGMENTS

This research was supported and funded by the Indonesian Ministry of Research, Technology, and Higher Education (DPRM-DIKTI) and Dian Nuswantoro University, specifically through the Research Center for Intelligent Distributed Monitoring and Security, with a special focus on Artificial Intelligence Studies in Nature Conservation and Natural Disaster Management.

#### REFERENCES:

- [1]. species classification in underwater videos: Exploiting pre-trained deep neural network models to compensate for limited labelled data,” *ICES Journal of Marine Science*, vol. 75, no. 1, pp. 374–389, 2018, doi: 10.1093/icesjms/fsx109.
- [2]. H. Tian, T. Wang, Y. Liu, X. Qiao, and Y. Li, “Computer vision technology in agricultural automation —A review,” *Information Processing in Agriculture*, vol. 7, no. 1, pp. 1–19, 2020, doi: 10.1016/j.inpa.2019.09.006.
- [3]. B. Peng, S. Wan, Y. Bi, B. Xue, and M. Zhang, “Automatic Feature Extraction and Construction Using Genetic Programming for Rotating Machinery Fault Diagnosis,” *IEEE Transactions on Cybernetics*, vol. 51, no. 10, pp. 4909–4923, 2021, doi: 10.1109/TCYB.2020.3032945.
- [4]. J. Ma and X. Gao, “Designing genetic programming classifiers with feature selection and feature construction,” *Applied Soft Computing Journal*, vol. 97, p. 106826, 2020, doi: 10.1016/j.asoc.2020.106826.
- [5]. J. Ma and G. Teng, “A hybrid multiple feature construction approach for classification using Genetic Programming,” *Applied Soft Computing Journal*, vol. 80, pp. 687–699, 2019, doi: 10.1016/j.asoc.2019.04.039.
- [6]. B. Tran, B. Xue, and M. Zhang, “Genetic programming for multiple-feature construction on high-dimensional classification,” *Pattern Recognition*, vol. 93, pp. 404–417, 2019, doi: 10.1016/j.patcog.2019.05.006.
- [7]. J. Liang, Y. Xue, and J. Wang, “Genetic programming based feature construction methods for foreground object segmentation,” *Engineering Applications of Artificial Intelligence*, vol. 89, no. August 2019, p. 103334, 2020, doi: 10.1016/j.engappai.2019.103334.
- [8]. A. Mahanipour and H. Nezamabadi-pour, “A multiple feature construction method based on gravitational search algorithm,” *Expert Systems with Applications*, vol. 127, pp. 199–209, 2019, doi: 10.1016/j.eswa.2019.03.015.
- [9]. A. Mahanipour, H. Nezamabadi-Pour, and B. Nikpour, “Using fuzzy-rough set feature selection for feature construction based on genetic programming,” *3rd Conference on Swarm Intelligence and Evolutionary Computation, CSIEC 2018*, pp. 1–6, 2018, doi: 10.1109/CSIEC.2018.8405407.
- [10]. J. Ma and X. Gao, “A filter-based feature construction and feature selection approach for classification using Genetic Programming,” *Knowledge-Based Systems*, vol. 196, no. xxxx, p. 105806, 2020, doi: 10.1016/j.knosys.2020.105806.
- [11]. G. I. Sayed, A. E. Hassanien, A. Gamal, and H. A. Ella, “An Automated Fish Species Identification System Based on Crow Search Algorithm,” *Advances in Intelligent Systems and Computing*, vol. 723, pp. 112–123, 2018, doi: 10.1007/978-3-319-74690-6\_12.
- [12]. A. Salman *et al.*, “Fish species classification in unconstrained underwater environments based on deep learning,” *Limnology and Oceanography: Methods*, vol. 14, no. 9, pp. 570–585, 2016, doi: 10.1002/lom3.10113.
- [13]. H. Qin, X. Li, J. Liang, Y. Peng, and C. Zhang, “DeepFish: Accurate underwater live fish recognition with a deep architecture,” *Neurocomputing*, vol. 187, pp. 49–58, 2016, doi: 10.1016/j.neucom.2015.10.122.
- [14]. Y. H. Hsiao, C. C. Chen, S. I. Lin, and F. P. Lin, “Real-world underwater fish recognition and identification, using sparse representation,” *Ecological Informatics*, vol. 23, pp. 13–21, 2014, doi: 10.1016/j.ecoinf.2013.10.002.
- [15]. C. Spampinato *et al.*, “A rule-based event detection system for real-life underwater domain,” *Machine Vision and Applications*, vol. 25, no. 1, pp. 99–117, 2014, doi: 10.1007/s00138-013-0509-x.
- [16]. S. Lee, S. Yun, J. H. Nam, C. S. Won, and S. W. Jung, “A review on dark channel prior based image dehazing algorithms,” *Eurasip Journal on Image and Video Processing*, vol. 2016, no. 1, pp. 1–23, 2016, doi: 10.1186/s13640-016-0104-y.
- [17]. B. J. Boom *et al.*, “A research tool for long-term and continuous analysis of fish assemblage in coral-reefs using underwater

- camera footage,” *Ecological Informatics*, vol. 23, pp. 83–97, 2014, doi: 10.1016/j.ecoinf.2013.10.006.
- [18]. C. Spampinato *et al.*, “Fine-grained object recognition in underwater visual data,” *Multimedia Tools and Applications*, vol. 75, no. 3, pp. 1701–1720, Feb. 2016, doi: 10.1007/s11042-015-2601-x.
- [19]. P. X. Huang, “Hierarchical Classification System with Reject Option for Live Fish Recognition,” in *Machine Vision and Applications*, 2016, pp. 141–159. doi: 10.1007/978-3-319-30208-9\_11.
- [20]. B. J. Boom, P. X. Huang, J. He, and R. B. Fisher, “Supporting ground-truth annotation of image datasets using clustering,” *Proceedings - International Conference on Pattern Recognition*, no. Icp, pp. 1542–1545, 2012.
- [21]. R. A. Pramunendar, D. P. Prabowo, D. Pergiawati, Y. Sari, P. N. Andono, and M. A. Soeleman, “New workflow for marine fish classification based on combination features and CLAHE enhancement technique,” *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 4, pp. 293–304, 2020, doi: 10.22266/IJIES2020.0831.26.
- [22]. A. B. Jung *et al.*, “imgaug,” <https://github.com/aleju/imgaug>, 2020. <https://github.com/aleju/imgaug>
- [23]. H. Mohammed and T. Rashid, “FOX: a FOX-inspired optimization algorithm,” *Applied Intelligence*, vol. 53, no. 1, pp. 1030–1050, Jan. 2023, doi: 10.1007/s10489-022-03533-0.
- [24]. F. Sultana, A. Sufian, and P. Dutta, “Advancements in image classification using convolutional neural network,” *Proceedings - 2018 4th IEEE International Conference on Research in Computational Intelligence and Communication Networks, ICRCICN 2018*, pp. 122–129, 2018, doi: 10.1109/ICRCICN.2018.8718718.
- [25]. C. P. Priya and S. Muruganatham, “Coral reef image classifications with hybrid methods,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 11 Special Issue, pp. 1247–1254, 2019, doi: 10.35940/ijitee.K1251.09811S19.
- [26]. B. M. Hopkinson, A. C. King, D. P. Owen, M. Johnson-Roberson, M. H. Long, and S. M. Bhandarkar, “Automated classification of three-dimensional reconstructions of coral reefs using convolutional neural networks,” *PLoS ONE*, vol. 15, no. 3, pp. 1–20, 2020, doi: 10.1371/journal.pone.0230671.
- [27]. G. F. Shidik *et al.*, “LUTanh Activation Function to Optimize BI-LSTM in Earthquake Forecasting,” *International Journal of Intelligent Engineering and Systems*, vol. 17, no. 1, pp. 572–583, 2024, doi: 10.22266/ijies2024.0229.48.
- [28]. R. A. Pramunendar *et al.*, “Integrating Grey Wolf Optimizer for Feature Selection in Birdsong Classification Using K-Nearest Neighbours Algorithm,” *International Journal of Intelligent Engineering and Systems*, vol. 16, no. 6, pp. 695–705, Dec. 2023, doi: 10.22266/ijies2023.1231.58.