Advance Sustainable Science, Engineering and Technology (ASSET)
Vol. 8, No.4, October 2026, pp. 0260406-01 ~ 0260406-015
ISSN: 2715-4211 DOI: https://doi.org/10.26877/asset.v8i4.3433

LDWFOX Optimization for Hyperparameter Tuning of Inception CNN in UAV-Based Vegetation Density Mapping

Ricardus Anggi Pramunendar[1,2*], Ashraf Alomoush[3], Dwi Puji Prabowo[1,2], Rama Aria Megantara[1,2], Farrikh Alzami[1,2], Nurul Anisa Sri Winarsih[1,2], Dewi Pergiwati[1,2], Guruh Fajar Shidik[1,2]

[1]Department of Informatics Engineering, Universitas Dian Nuswantoro, Semarang, Indonesia

[2]Center for Intelligent Distributed Surveillance and Security, Universitas Dian Nuswantoro, Indonesia

[3]Department of Data Science and Artificial Intelligence, Isra University, Amman, Jordan

*ricardus.anggi@dsn.dinus.ac.id

Abstract. Vegetation density classification from UAV imagery is a practical necessity in fire-prone landscapes, since fuel load on the ground directly informs risk management decisions. Convolutional neural networks handle this classification reasonably well, but good performance requires careful hyperparameter tuning, and manual trial and error produces results that are unstable and hard to reproduce. This study proposes LDW-FOX, a modified FOX metaheuristic using a Linearly Decreasing Weight mechanism to automate hyperparameter tuning for an Inception-based CNN. The original FOX algorithm applies fixed movement weights throughout optimization, causing search to stagnate early. LDW-FOX gradually reduces exploration intensity across iterations, pushing search toward exploitation as it converges. Five hyperparameters, namely learning rate, dropout rate, hidden layer size, activation function, and optimizer, were tuned on a balanced 3,000 image UAV dataset spanning three vegetation density classes. Manual tuning peaked at 61.00 percent test accuracy but varied considerably across epoch settings. LDW-FOX reached a peak test accuracy of 82.48 percent and a mean of 58.47 percent, outperforming the original FOX, whose mean was 55.30 percent. LDW-FOX showed a more consistent training-test gap than other swarm-based methods, with LDW variants beating unmodified counterparts under equal budgets. High variance across configurations indicates broader generalization needs testing.

Keywords: UAV remote sensing, vegetation density classification, metaheuristic hyperparameter optimization, peatland monitoring, vegetation density

(Received 2026-01-30, Revised 2026-04-18, Accepted 2026-06-10, Available Online by 2026-08-29)

1. Introduction

In 2022, Indonesia's forest cover was recorded at 95.97 million hectares, representing approximately 51.2% of its total land area of 187.588 million hectares. Between 2014 and 2022, forest cover fluctuated (Fig. 1), decreasing from 95.77 million hectares in 2014 to 95.97 million hectares in 2022. These variations highlight the importance of accurate land-cover and vegetation mapping to support quantitative analysis of landscape patterns influenced by vegetation dynamics, climate variability, and anthropogenic activities [1].

Figure 1. Land Cover Area in Indonesia (Source: BPS)

From a technical perspective, changes in forest and vegetation cover increase the complexity of environmental monitoring tasks and require reliable remote sensing methods that capture fine-scale spatial variability. In regions with limited ground-based observations, inaccurate vegetation mapping can increase uncertainty in downstream analyses, such as fire-risk assessment and land management modeling [2]. Consequently, efficient, data-driven land-cover classification approaches are essential, particularly for large, heterogeneous landscapes. Distinguishing vegetation density levels, such as bare land, sparse vegetation, and dense vegetation, remains challenging when relying on traditional imaging approaches [3]. Classical techniques, including vegetation indices and color-space transformations, have been widely applied but often require expert-designed features, manual thresholding, or specialized software, making them time-consuming and highly data-dependent [4]. In contrast, advances in satellite and unmanned aerial vehicle (UAV) imagery have significantly improved spatial resolution and data availability, enabling more detailed land-cover mapping and analysis [5]. These developments have accelerated the adoption of deep learning techniques for vegetation and land-cover classification tasks [6,7].

Recent studies have demonstrated the effectiveness of computer vision and deep learning methods for land-cover and vegetation mapping using both public [6–12] and private datasets [13–18]. While many of these approaches report high classification accuracy, often exceeding 90%, their performance is strongly influenced by the selection of model architecture and hyperparameter configuration [8,13,14,19]. In practice, hyperparameters are often selected through manual trial-and-error, which limits reproducibility and may lead to suboptimal generalization. Hyperparameter optimization methods can be broadly categorized into manual or heuristic tuning, probabilistic approaches such as Bayesian optimization, and population-based evolutionary or swarm intelligence algorithms. Manual, grid-based, and random search strategies remain common but are computationally expensive and scale poorly with increasing search dimensions. Bayesian optimization and adaptive resource allocation methods, such as Hyperband, offer improved sample efficiency but often struggle with highly nonconvex search spaces and require careful surrogate-model design. Evolutionary and swarm-based algorithms (such as Particle Swarm Optimization (PSO) and related variants) have therefore gained attention due to their flexibility

and gradient-free nature. However, many existing swarm-based HPO methods lack adaptive mechanisms to balance global exploration and local exploitation, which can lead to premature convergence or unstable optimization behavior.

To address these limitations, this study focuses on metaheuristic-based hyperparameter optimization for convolutional neural networks applied to vegetation density classification. An Inception-based CNN architecture is selected for its proven ability to capture multi-scale spatial features through parallel convolutional paths, making it well-suited to heterogeneous UAV imagery. Rather than modifying the network structure itself, this work concentrates on optimizing critical hyperparameters that directly influence learning dynamics and generalization, including learning rate, dropout rate, hidden-layer size, activation function, and optimizer choice. The FOX Optimization algorithm has recently been proposed as a swarm-based metaheuristic for solving complex optimization problems

[20]. However, the original FOX algorithm relies on fixed movement weights throughout the entire search process. This design does not account for the natural shift in search behavior that should occur as optimization progresses — broad exploration early on, followed by increasingly focused exploitation near convergence. Without this adaptive mechanism, FOX is prone to premature stagnation, particularly in high-dimensional hyperparameter spaces where the fitness landscape is nonconvex and irregular [21].

To overcome this limitation, this study introduces LDW-FOX, an enhanced FOX algorithm that incorporates a Linearly Decreasing Weight mechanism. The LDW strategy is well-established in swarm intelligence literature, most notably in PSO, where it has consistently improved convergence stability by dynamically adjusting the influence of movement updates [22,23]. Crucially, FOX's position update structure — which relies on directional jump mechanics governed by a movement vector — is architecturally compatible with weight-based scaling. Applying LDW to this vector allows the algorithm to perform wide-ranging exploration in early iterations and shift toward fine-grained local search as iterations progress, without introducing additional computational overhead or requiring changes to the core FOX behavioral model. This makes LDW a natural and non-trivial extension to FOX, distinct from simply porting LDW from PSO, because the underlying movement dynamics of FOX differ fundamentally from velocity-based updates in PSO.

The primary objective of this study is to identify optimal CNN hyperparameters for UAV-based vegetation density classification using LDW-FOX. Its effectiveness is evaluated through controlled experiments comparing manual hyperparameter tuning and alternative swarm-based optimization methods under identical computational budgets — same population size, maximum iterations, and total model evaluations — to ensure fair and unbiased comparison. The main contributions of this study are threefold: the integration of a Linearly Decreasing Weight mechanism into the FOX algorithm to improve convergence stability; the application of LDW-FOX for automatic hyperparameter tuning of an Inception-based CNN on UAV vegetation imagery; and a comprehensive experimental evaluation comparing manual tuning, original swarm optimizers, and LDW-enhanced variants under equal computational budgets, including analysis of training stability, generalization behavior, and statistical significance.

The remainder of this paper is organized as follows. Section 2 describes the proposed methodology and experimental setup. Section 3 presents the results and discussion. Section 4 concludes the paper and outlines directions for future study.

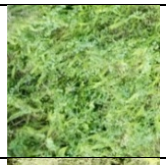
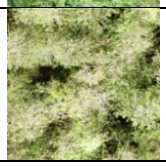
2. Methods

This study improves the FOX Optimization Algorithm by adding a Linearly Decreasing Weight (LDW) mechanism to tune hyperparameters of an Inception-based CNN for vegetation density classification, as shown in Fig. 2. The proposed LDW-FOX provides adaptive weighting approach that enhances optimization effectiveness and helps prevent premature convergence. Its performance is tested and compared against other swarm-based metaheuristic methods using publicly available datasets [24,25]. Results show that LDW-FOX consistently identifies competitive hyperparameter configurations, contributing to more stable classification performance across runs.

2.1. Data Collecting and Preprocessing

This study used the Vegetation Density of Peatland Drone Dataset, which consists of three vegetation density classes: bare, softly grazed, and heavily grazed. The dataset was collected from a surveyed area covering farmland, plantations, settlements, and weedy land. All images were captured with a drone-mounted camera at 4000 × 3000 pixels and resized to 224 × 224 pixels to match the input requirements of the Inception-based CNN (see Table 1). Each class contains 1,000 images, resulting in a balanced dataset. The dataset was divided into training, validation, and testing sets using a stratified split with proportions of 70%, 15%, and 15%, respectively. All experiments were conducted with a fixed random seed (seed = 42) and repeated across five independent runs, with results reported as the mean and standard deviation. Before training, images were normalized using the ImageNet mean and standard deviation. No data augmentation was applied. This decision was made deliberately: augmentation introduces additional stochasticity that can confound the evaluation of hyperparameter optimization methods. The dataset is publicly available at https://data.mendeley.com/datasets/tb26zy2jst/1 [24,25].

Table 1. Sample images dataset

Gambar	1	2	3	4
Bare				
Softly Grazed				
Heavily Grazed				

2.2. FOX Algorithm

The FOX algorithm begins by initializing a population X, which represents the positions of the fox agents. Each agent's fitness is evaluated using a predefined fitness function. The best solution $X_{\text{best}}^{t}$and its corresponding fitness value are determined by comparing all agents' fitness values at each iteration. To balance exploration and exploitation, a random variable is used to select one of the two phases with equal probability. This mechanism helps prevent premature convergence to local optima. Furthermore, a control variable (α^{t}) drives the search toward the current best solution, with its value gradually decreasing over iterations. During the position update process, the fitness function plays a crucial role in directing agents away from local optima. If an updated position does not improve fitness, the exploration phase is temporarily disabled to enhance exploitation effectiveness.

2.2.1. Exploitation

In the exploitation phase, when the random variable is $p > 0.18$, the fox searches for a new position to capture its prey. To perform this movement, several parameters are calculated, including the sound travel distance d_i^t, the distance between the fox and the prey r_i^t, and the jump height h_i^t.

2.2.1.1 Determining the sound travel distance

The sound travel time τ_i^t is randomly generated within the range [0, 1]. The sound travel distance from the fox is calculated by multiplying the speed of sound in air ($v = 343$ m/s) by the sound travel time, as shown in (1). In addition, the sound velocity can be approximated the best position found in the search population X_{best}^t, as shown in (2).

$$\mathrm{d_i^t} = v \times \tau_\mathrm{i}^t \tag{1}$$

$$v = \frac{\mathrm{X_{best}^t}}{\tau_\mathrm{i}^t} \tag{2}$$

2.2.1.2 Distance between the fox and the prey

The distance between the fox and its prey r_i^t is derived from the sound travel distance. In physics, the measured distance between a sensor and an object is half the total distance the sound wave travels because the signal is transmitted and received. Therefore, the fox–prey distance is calculated as (3).

$$r_\mathrm{i}^\mathrm{t} = \mathrm{d_i^t} \times 0.5 \tag{3}$$

2.2.1.3 Fox's jump height calculation

After the distance to the prey is determined, the jump height h_i^t is calculated using a free-fall motion model (4). Here, $g = 9.81$ m/s^2 denotes gravitational acceleration, and $\bar{\tau}$ represents the average sound travel time across all dimensions (5).

$$\mathrm{h_i^t} = \frac{1}{2}\mathrm{g}(\bar{\tau})^2 \tag{4}$$

$$\bar{\tau} = \frac{1}{\mathrm{D}}\sum_{j=1}^{D}\tau_{\mathrm{i,j}}^\mathrm{t} \tag{5}$$

2.2.2. Determining the fox's new position

The fox's new position is determined by multiplying the jump height h_i^t, the fox–prey distance r_i^t, and a direction constant. The updated positions are given in (6) and (7) and show the fox's new positions. Constants c1 and c2 are 0.18 and 0.82, respectively, guiding the fox's jump direction. If p>0.18, the fox jumps northeast, multiplying r_i^t and h_i^t by c1, which increases the likelihood of reaching the global optimum. If p≤0.18, the fox jumps in the opposite direction, multiplying by c2, thus decreasing the chance of reaching the prey (≤18%).

$$\mathrm{X_i^{t+1}} = \mathrm{r_i^t} \times \mathrm{h_i^t} \times c_1, \text{if p} > 0.18 \tag{6}$$

$$\mathrm{X_i^{t+1}} = \mathrm{r_i^t} \times \mathrm{h_i^t} \times c_2, \text{if p} \le 0.18 \tag{7}$$

2.2.3. Exploration

In the exploration phase, the fox performs a random search guided by the best position found so far. In this phase, the fox explores the search space without using the jump mechanism. A control factor α^t is used to regulate the randomness of the movement toward the best solution. The control factor is defined as (8). Equation (9) illustrates how the fox explores for new positions in the search space ($\mathrm{X_i^{it+1}}$); t denotes the current iteration, and T is the maximum number of iterations.

$$a = 2\left(1 - \left(\frac{t}{\mathrm{T}}\right)\right) \tag{8}$$

$$\mathrm{X_i^{it+1}} = \mathrm{X_{best}^t} \times random \times \bar{\tau} \times \alpha^\mathrm{t} \tag{9}$$

2.2.4. Integration of Linearly Decreasing Weight (LDW) into the FOX framework

The main novelty of this study is the integration of a Linearly Decreasing Weight (LDW) mechanism into the FOX optimization framework. While LDW has been widely adopted in PSO and related velocity-based algorithms, its application to FOX requires careful adaptation. Unlike PSO, where LDW scales the inertia of velocity updates, FOX relies on directional jump mechanics governed by a movement vector derived from fox behavioral dynamics. Applying LDW to this movement vector is therefore a structurally distinct modification — it directly modulates the magnitud of positional displacement rather than a velocity term, which aligns more naturally with FOX's physics-inspired update rule. This adaptation has not been explored in prior FOX implementations, making the integration both novel and architecturally motivated. The adaptive weight decreases linearly from a maximum to a minimum across iterations, as defined in (10). At early iterations, a larger weight (w_t) promotes global exploration, while smaller values near convergence encourage local exploitation. Accordingly, the standard FOX position update rule is reformulated as (11).

$$w_t = w_{max} - \frac{t}{T}(w_{max} - w_{min}) \tag{10}$$

$$X_i^{t+1} = w_t \times X_i^t + \Delta_i^t \tag{11}$$

where X_i^t denotes the current position of the agent i, and Δ_i^t represents the movement vector derived from fox behavioral dynamics. By incorporating the LDW factor, LDW-FOX adaptively adjusts its search behavior without increasing computational complexity, improving convergence stability, and preventing premature stagnation. This enhancement strengthens the applicability of FOX for Inception-based CNN hyperparameter optimization.

2.2.5. Pseudocode of FOX and LDW-FOX for Inception CNN Hyperparameter Optimization

To clarify the implementation of the optimization framework, Algorithms 1 and 2 present the pseudocode for FOX and LDW-FOX, as applied to hyperparameter optimization of an Inception-based CNN. In both algorithms, each agent represents a candidate hyperparameter configuration, and model performance is evaluated using validation accuracy. In addition to FOX and LDW-FOX, other swarm-based metaheuristic algorithms were implemented following the same optimization structure, population size, iteration limits, and evaluation protocol to ensure fair and consistent comparisons. The key distinction of LDW-FOX lies in its integration of a linearly decreasing weight mechanism that adaptively controls exploration and exploitation during optimization.

Algorithm 1: FOX Optimization for Inception CNN Hyperparameter Tuning

1. Initialize a population of fox agents, where each agent represents a candidate hyperparameter set for the Inception-based CNN.
2. Configure and train the Inception CNN using each agent's hyperparameters.
3. Evaluate each agent's fitness using validation accuracy.
4. Identify the best solution X_{best}.
5. For each iteration:
    - Randomly select the exploration or exploitation phase.
    - Update agent positions according to FOX movement rules.
    - Retrain the Inception CNN with updated hyperparameters.
    - Update X_{best} if a better solution is found.
6. Repeat until the maximum number of iterations is reached.

Algorithm 2: LDW-FOX Optimization for Inception CNN Hyperparameter Tuning

1. Initialize the fox population and LDW parameters.
2. For each iteration:
    - Update the linearly decreasing weight w_t.

- Update agent positions using LDW-modified FOX equations.
- Train the Inception CNN and evaluate validation accuracy.
- Update the global best solution.

3. Terminate when the convergence criteria or the maximum number of iterations are reached.

2.3. Inception Convolution Neural Network

This study uses an Inception-based CNN architecture for classification. Known for its ability to extract features across multiple scales simultaneously, the Inception module combines various convolutional operations (1×1, 3×3, and 5×5) with max pooling within a single block. This allows the model to better understand both spatial and texture information. We chose Inception V3 over V2 because it offers improved efficiency and accuracy, thanks to smaller convolutional kernels and a grid-size reduction strategy. These design features reduce computational load while still providing high-quality feature extraction.

Additionally, Inception V3 is pre-trained on large datasets like ImageNet, which supports transfer learning for adapting to vegetation-density classification. Critical parameters include the number of filters, kernel size, learning rate, activation function, and number of epochs. They are automatically optimized using the LDW-FOX algorithm. This enhances the model's accuracy and training stability. The structure of the Inception V3 model used here is shown in Fig 3.

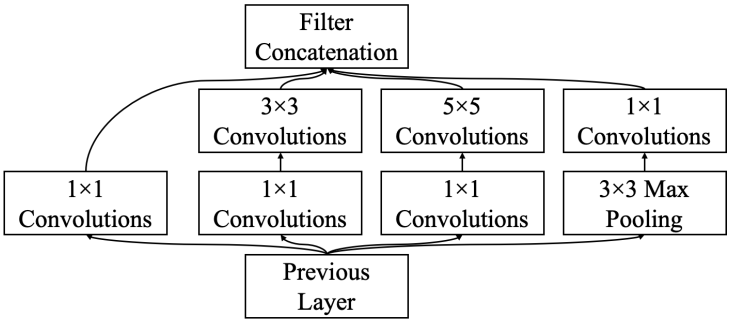


Figure 3 Model Inception V3

Table 2. Parameters of the Inception-based CNN

Parameter	Range/Typical Values
Epoch	1–10
Dropout	0.1–0.9
Hidden Layer Size	128, 256, 512, 1024
Learning Rate	1e-3 to 1e-1
Activation Function	ReLU, Sigmoid, Tanh, SiLU, ELU, CELU, SELU, GELU, Leaky ReLU, PReLU
Optimizer	Adam, Adagrad, SGD

Various parameters in the Inception-based CNN architecture significantly influence classification performance, including the number of filters, kernel size, activation function, learning rate, and number of training epochs, as these parameters directly affect feature extraction and model generalization. In this study, the hyperparameters listed in Table 2 were optimized simultaneously using the proposed LDW-FOX algorithm rather than through a staged or sequential tuning process. Continuous parameters, such as learning rate and dropout rate, were encoded as real-valued variables within predefined bounds. In contrast, discrete parameters, including activation function and optimizer, were represented using integer indices mapped to categorical choices. Hidden layer sizes were treated as discrete numerical values.

The number of training epochs is included as an optimizable hyperparameter within the range of 1 to 10. Although epochs are often treated as a fixed training schedule in standard deep learning pipelines,

this study treats them as part of the hyperparameter search space to examine whether the optimization algorithm can identify computationally efficient configurations that avoid unnecessary training. This range was selected based on preliminary experiments showing that the Inception-based CNN reaches stable validation performance within this interval on the given dataset, making a wider range redundant and computationally wasteful. Throughout the optimization process, all candidate solutions were constrained to the specified parameter ranges to ensure valid and consistent network configurations.

2.4. Performance Evaluation

Classification performance was evaluated using three metrics: training accuracy, validation accuracy, and test accuracy, all computed using the formula in (13). Training accuracy measures how well the model fits the training data, while validation accuracy guides the hyperparameter optimization process during the search. Test accuracy reflects generalization performance on unseen data and serves as the primary criterion for comparing optimization methods. Results across all three metrics are reported as the mean and standard deviation over five independent runs. In addition, a two-factor analysis of variance (ANOVA) was conducted to examine the statistical effects of population size and number of training epochs on model performance.

$$accuracy = \frac{total\ correct\ data\ (t)}{total\ number\ data\ (n)} \tag{13}$$

2.5. Research Design

The goal of this study was to find hyperparameter values that improve Inception-based CNN performance on a three-class vegetation cover task covering bare, lightly grazed, and heavily grazed land. The first phase used manual experiments to establish a baseline. Each parameter was varied sequentially while the others were held fixed, working through epochs from 1 to 10, dropout rates from 0.1 to 0.9, hidden layer sizes from 128 to 1024 neurons, learning rates from 1e-3 to 1e-1, eleven activation functions, and three optimizers: Adam, Adagrad, and SGD. The second phase handed that search over to LDW-FOX, which explored the same space automatically. Results were compared against the original FOX, standard PSO, PSOLDW, and several other swarm-based methods. Every method ran under the same population size, number of iterations, and total model evaluations. Performance was measured using training and validation accuracy. All experiments ran on a machine with an Intel i7-8700 processor, 64 GB RAM, and Windows 11.

3. Results and Discussion

This study used the Vegetation Density of Peatland Drone Dataset, which was classified using an Inception-based CNN architecture. The best hyperparameter configurations were obtained using the FOX metaheuristic method with a linearly decreasing weight. The first experiment used manually set parameters on a basic Inception-based CNN, while subsequent experiments applied metaheuristic optimization to the same architecture. In this study, model evaluation focuses on accuracy-based performance and optimization stability across different hyperparameter configurations, with particular emphasis on convergence behavior, generalization trends, and statistical significance among optimization strategies. Therefore, the evaluation emphasizes accuracy, variance across runs, and ANOVA-based validation rather than class-wise metrics.

3.1. Manual Setting of Parameter Value on Inception Parameter

Experiments were conducted with a fixed hyperparameter configuration: a dropout rate of 0.5, a hidden-layer size of 512, a learning rate of 0.01, ReLU activation, and the Adam optimizer. Figure 4 shows the training, validation, and test accuracies of the Inception-based CNN obtained with this manual parameter configuration across different training epochs. Overall, increasing the number of epochs does not consistently improve model performance. For most epoch values, the training, validation, and testing

accuracies remain within the range of approximately 50–55%, indicating that the learning process is generally suboptimal when manually selected parameters are used.

Testing accuracy under manual tuning stayed between 51% and 54% for epoch values from 5 to 20, while training time climbed from around 57.66 seconds to 228 seconds over the same range. Extending to 30 or 40 epochs pushed training time to roughly 455 seconds without moving accuracy in any meaningful direction. The best result came at 50 epochs, where training, validation, and test accuracies reached 64.98%, 64.60%, and 61.00%. Past that point the model stopped improving. Accuracy fell back to the 51% to 53% range at higher epoch counts while training time kept rising, eventually exceeding 1,140 seconds at 100 epochs. A few isolated spikes appeared, particularly at epochs 50 and 70, but they did not reproduce consistently across folds. Figure 4 makes the problem clear. Training accuracy continued to rise in several configurations while validation and test accuracy stayed flat or declined, which is the signature of a model fitting the training data rather than learning anything that transfers.

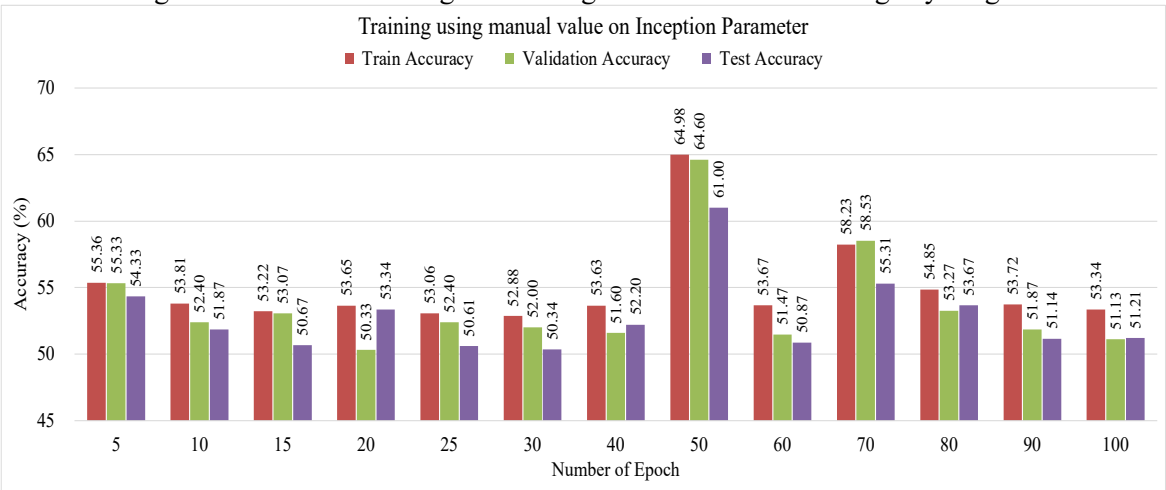


Figure 4 Average Accuracy using Inception from training, validation and testing data

3.2. Inception Parameters using LDW-FOX

The proposed LDW-FOX method uses a linearly decreasing weight to enhance optimization stability, accuracy, and adaptability while minimizing reliance on manual parameters. Results from training, validation, and testing across various metaheuristic setups are presented in Tables 3, 4, and 5. In these, ps5–ps25 indicate population sizes from 5 to 25, and ep1–ep10 refer to epochs one through ten.

3.2.1. Training Model

Table 3 shows how population size affects training accuracy. At a population size of 5, accuracy sits in a fairly narrow band between 81% and 82%. Moving to 10 raises it further, with a peak of 84.44%. Beyond that, the gains flatten out. Population sizes from 15 to 25 produce results that cluster between 82% and 83%, more consistent but not higher. Epoch count also plays a role. Ten epochs produced the most stable and highest training accuracy across population sizes, with shorter runs falling short on both measures. A two-factor ANOVA confirmed that both factors carry statistically significant weight, with F equal to 7.68 and p equal to 3.39×10^{-6} for epoch, and F equal to 6.52 and p equal to 0.00047 for population size.

Training time stayed near 11 seconds per epoch for most configurations. A few combinations ran longer, mostly due to optimizer selection rather than population size or epoch count. The low variance in training accuracy across conditions is consistent with what the LDW mechanism is meant to do. Gradually reducing exploration intensity keeps the search from bouncing between regions late in the process, when fine-tuning matters more than coverage. The configuration that worked best overall was a population size of 10 with 10 training epochs. Under those settings, LDW-FOX settled on a dropout rate of 0.1, a hidden size of 256, a learning rate of 0.001, ReLU activation, and the Adam optimizer.

Table 3. Training Accuracy based on LDW-FOX

	ps5	ps10	ps15	ps20	ps25
ep1	81.45	79.64	80.69	81.92	81.78
ep2	80.16	81.68	81.40	82.21	82.40
ep3	82.25	82.40	82.16	82.83	82.92
ep4	81.64	83.97	83.02	83.35	83.11
ep5	81.64	84.44	82.68	83.25	82.97
ep6	82.21	82.64	83.40	83.35	82.49
ep7	81.54	82.78	83.02	82.97	83.30
ep8	82.49	82.97	82.97	83.16	82.40
ep9	82.16	84.30	83.44	83.21	83.73
ep10	82.40	83.78	83.35	83.02	83.59

3.2.2. Testing and Validation Model

Each LDW-FOX experiment was evaluated on both validation and test sets under the same conditions to assess how well different population size and epoch combinations generalize. Table 4 shows that the highest validation accuracy came from ep7 with ps10 at 82.67%, followed by ep5 with ps20 at 82.33%. Smaller and larger population sizes produced more erratic results across configurations, suggesting the search is sensitive to agent count in ways that do not always favor extremes. Test results in Table 5 follow a similar pattern. The highest test accuracy was 82.48% at ep4 with ps20, while ps10 also performed competitively at ep6, reaching 72.85%. Some configurations failed considerably — ep6 with ps15 reached only 33.97%, indicating that certain hyperparameter combinations produce unstable learning regardless of the optimization method used.

Table 4. Validation Accuracy based on LDW-FOX

	ps5	ps10	ps15	ps20	ps25
ep1	56.44	63.44	60.56	61.89	41.78
ep2	55.44	58.89	58.22	51.56	53.11
ep3	53.11	51.78	69.22	57.56	40.22
ep4	58.89	53.33	53.11	80	54.44
ep5	52.89	63.11	54.89	82.33	54
ep6	52	78.11	34	64.44	60.44
ep7	58	82.67	81	63	62.78
ep8	58	51.78	55.56	55	51.56
ep9	60.33	62.11	77.22	68.44	62.44
ep10	56.44	66.22	52.00	62.78	60

The gap between training and test performance varied substantially across configurations. The highest training accuracy of 84.44% at ep5 with ps10 came with a test accuracy of only 63.24%, a difference large enough to indicate that the model was fitting the training data rather than learning representations that transfer. The ep4 with ps20 configuration showed a much tighter relationship, with 83.35% on training and 82.48% on testing. That consistency is a more meaningful indicator of optimization quality than peak training accuracy alone, and it is what the comparison across methods in the following section uses as its primary basis for evaluation.

Table 5. Testing Accuracy based on LDW-FOX

	ps5	ps10	ps15	ps20	ps25
ep1	52.82	69.52	52.82	63.31	38.85
ep2	57.99	61.24	62.79	53.48	53.92
ep3	52.82	53.48	67.53	61.91	37.96
ep4	60.13	57.25	53.04	82.48	56.14
ep5	55.48	63.24	58.58	63.31	53.48
ep6	53.26	72.85	33.97	64.79	60.13
ep7	57.47	49.05	81.27	59.55	64.42
ep8	56.81	51.93	59.25	57.33	52.82
ep9	57.11	62.43	51.71	69.67	63.02
ep10	59.47	68.34	53.48	59.25	60.58

3.3. Automatic Inception Parameters with Alternative Metaheuristic Methods

All baseline algorithms were evaluated under the same experimental conditions as LDW-FOX, using identical parameter settings, datasets, models, and computing environments. Table 6 shows the training and testing accuracies across methods. The original variants, namely ABCORI [26], GOAORI [27], GWOORI [28], SSOORI [29], WOAORI [30], and ZOAORI [31] all produced testing accuracies between 53.13% and 54.20%, a narrow band that suggests their unmodified search behavior does not translate well to this hyperparameter tuning task. PSOORI and FOXORI performed somewhat better at 54.79% and 55.30%, reflecting stronger base search dynamics in those two algorithms.

Table 6. Accuracy between Metaheuristic Methods

Method	Mean Training	Mean Testing
ABCORI [26]	70.29 ± 2.26	53.85 ± 2.80
GOAORI [27]	61.41 ± 9.12	53.44 ± 2.50
GWOORI [28]	67.22 ± 4.30	53.13 ± 4.20
SSOORI [29]	64.29 ± 11.08	53.94 ± 3.20
WOAORI [30]	70.36 ± 3.20	53.50 ± 5.60
ZOAORI [31]	71.67 ± 1.68	54.20 ± 3.40
PSOORI [32]	80.90 ± 1.98	54.79 ± 4.60
FOXORI [33]	81.99 ± 0.95	55.30 ± 8.00
PSOLDW [34,35]	80.90 ± 1.98	60.21 ± 8.30
LDW-FOX	82.61 ± 0.96	58.47 ± 8.90

Adding the LDW mechanism improved testing accuracy in every method it was applied to. PSOLDW reached a mean test accuracy of 60.21% and LDW-FOX reached 58.47%. On that metric alone, PSOLDW comes out ahead. However, LDW-FOX shows a more consistent relationship between training and testing performance, with a mean training accuracy of 82.61% against a test accuracy of 58.47%, while PSOLDW produces a wider and less stable spread. Under identical computational budgets, a tighter training-to-test gap is a more reliable signal of generalization than peak accuracy alone. A two-factor ANOVA supported this picture, showing a significant difference between training and testing accuracies at p less than 0.001, and a near-significant difference among optimization methods at p equal to 0.065. In every pairing, the LDW-enhanced variant outperformed its original counterpart in both testing accuracy and consistency.

3.4. Discussions

This study compared manual hyperparameter tuning and LDW-FOX on the Vegetation Density of Peatland Drone Dataset. Manual tuning produced low, unstable results across all three-accuracy metrics. Adding more epochs extended training time without improving performance, and ANOVA confirmed that epoch variation had no statistically significant effect when other parameters were held fixed. Deep learning models respond to the combined effect of all hyperparameters at once, not to any single one in isolation. A mismatched learning rate or optimizer makes additional epochs either redundant or counterproductive, and no amount of training time compensates for that mismatch.

LDW-FOX produced more stable training accuracy across tested population sizes and epoch settings. The weight in equation (10) starts high, pushing agents to cover the search space broadly, then decreases gradually, focusing the search on regions that showed promise in earlier iterations. The original FOX applies the same movement magnitude throughout, with no mechanism to distinguish early exploration from late refinement. When an agent is close to a good solution, a large, fixed step can overshoot it rather than settle. The LDW adjustment corrects this without modifying any other part of the algorithm.

Although convergence curves were not explicitly recorded in this study, the training accuracy trends in Table 3 provide indirect evidence of optimization dynamics. Accuracy across most population sizes stabilizes within the first few epochs and continues to improve modestly through epoch 10, suggesting that the search reaches a productive region relatively early in the process. This pattern is consistent with the expected behavior of the LDW mechanism, where the gradual reduction in weight progressively limits large positional updates as iterations advance, allowing the search to consolidate around promising configurations rather than continuing to explore. The absence of sharp accuracy drops between consecutive epoch settings further support the interpretation that LDW-FOX converges steadily rather than oscillating, which contrasts with the behavior that would be expected from a fixed-weight strategy operating in the same search space.

The peak test accuracy of 82.48% came from a single well-matched configuration, while the mean test accuracy of 58.47% with a standard deviation of 8.90% covers all tested combinations of population size and epoch. These two numbers measure different things. The peak shows what LDW-FOX achieves under favorable conditions. The mean shows how it performs across all conditions, including configurations that do not work well.

Some configurations failed badly. The ep6 ps15 setting produced a test accuracy of 33.97%, near-random for a three-class task, and the same failure pattern appeared in other methods as well. A high learning rate combined with an incompatible optimizer inside a tight epoch budget often produces a model that never converges. These degradation regions are a structural feature of the CNN hyperparameter space and population-based search does not reliably avoid them.

LDW-FOX showed a smaller gap between training and test accuracy for its stable configurations compared to manual tuning. The adaptive weight slows the search from locking onto solutions that fit training data well but do not hold up on unseen samples. The improvement is genuine but uneven across configurations.

LDW-based variants outperformed their original counterparts across all compared methods. The FOX versus LDW-FOX comparison in Table 6 acts as a controlled ablation since every other condition was held constant, and any performance difference between them traces back to the LDW mechanism alone. The high standard deviation across results is not a sign of algorithmic instability. It reflects the sensitivity of CNN training to discrete hyperparameter choices, where activation function and optimizer selections produce large accuracy swings regardless of which search method is used. The gains in mean accuracy and stability over FOXORI confirm that the LDW modification produced a concrete improvement in search behavior.

Fixed-weight strategies tend to struggle when the search space mixes smooth performance regions with sharp drop-offs, which is characteristic of CNN hyperparameter tuning on this dataset. LDW-FOX does not eliminate this problem, but the adaptive weight reduces how often the search ends in unstable territory. The remaining variability is largely a feature of the space being searched rather than the algorithm doing the searching. Adding an adaptive weight to a straightforward metaheuristic proved

sufficient to produce consistent gains in accuracy and stability at no additional computational cost, and that pattern held across every LDW-based variant tested in this study.

4. Conclusions

This study proposed LDW-FOX, an enhanced FOX metaheuristic with a Linearly Decreasing Weight mechanism, and evaluated it as an automatic hyperparameter optimizer for an Inception-based CNN applied to UAV-based vegetation density classification. Across the experiments conducted, adaptive weight modulation produced more stable and generalizable results than fixed-weight search strategies.

Manual tuning peaked at 61.00% test accuracy with considerable variation across epoch settings. LDW-FOX reached a mean test accuracy of 58.47% across all evaluated configurations and a peak of 82.48% under optimal conditions, consistently maintaining a tighter training-to-test gap than competing methods. The controlled comparison between FOXORI and LDW-FOX under identical computational budgets isolates the LDW mechanism as the source of these gains rather than differences in experimental conditions.

Several limitations constrain the scope of these findings. The high standard deviation across configurations indicates that LDW-FOX does not produce reliable generalization in every run, particularly when activation function and optimizer selections interact in ways that produce sharp performance variation. All experiments were conducted on a single dataset and architecture, which limits generalizability beyond the tested setting. Convergence curves were not recorded, which restricts direct assessment of how the search evolves across iterations and makes it harder to characterize optimization dynamics with precision.

Future work should test LDW-FOX on additional architectures, datasets, and remote sensing tasks including semantic segmentation and multi-site evaluation. Incorporating per-iteration fitness tracking would also allow more direct analysis of convergence behavior and provide stronger empirical grounds for comparing exploration and exploitation dynamics across optimization methods.

Declaration of AI and AI assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT to improve the language and grammar of the manuscript. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was funded by the Ministry of Education, Culture, Research, and Technology (Kemdiktisaintek) under research contracts No. 127/C3/DT.05.00/PL/2025, No. 028/LL6/PL/AL.04/2025, and No. 121/F.9.05/UDN-09/VI/2025.

References

[1] Manfré LA, Hirata E, Silva JB, Shinohara EJ, Giannotti MA, Larocca APC, et al. An Analysis of geospatial technologies for risk and natural disaster management. ISPRS Int J Geoinf 2012;1:166–85. https://doi.org/10.3390/ijgi1020166.

[2] Zhu Z, Qiu S, Ye S. Remote sensing of land change: A multifaceted perspective. Remote Sens Environ 2022;282. https://doi.org/10.1016/j.rse.2022.113266.

[3] Mendoza-Bernal J, González-Vidal A, Skarmeta AF. A Convolutional Neural Network approach for image-based anomaly detection in smart agriculture. Expert Syst Appl 2024;247. https://doi.org/10.1016/j.eswa.2024.123210.

[4] Lu T, Wan L, Qi S, Gao M. Land Cover Classification of UAV Remote Sensing Based on Transformer–CNN Hybrid Architecture. Sensors 2023;23. https://doi.org/10.3390/s23115288.
[5] Sharma SN, Ayuba D. Nature Based Solutions to Prevent Urban Flooding. 1st ed. Edupedia Publications Pvt; 2024. https://doi.org/http://dx.doi.org/10.5281/zenodo.10651295.
[6] I. Volke M, Abarca-Del-Rio R, A. Warner T. Cost-effective disaster-induced land cover analysis: a semi-automatic methodology Using machine learning and satellite imagery. Int J Remote Sens 2024;45:279–305. https://doi.org/10.1080/01431161.2023.2292015.
[7] Wambugu N, Chen Y, Xiao Z, Wei M, Aminu Bello S, Marcato Junior J, et al. A hybrid deep convolutional neural network for accurate land cover classification. International Journal of Applied Earth Observation and Geoinformation 2021;103. https://doi.org/10.1016/j.jag.2021.102515.
[8] Fayaz M, Nam J, Dang LM, Song H, Moon H. Land-Cover Classification Using Deep Learning with High-Resolution Remote-Sensing Imagery. Applied Sciences 2024;14:1844. https://doi.org/10.3390/app14051844.
[9] Cecili G, De Fioravante P, Dichicco P, Congedo L, Marchetti M, Munafò M. Land Cover Mapping with Convolutional Neural Networks Using Sentinel-2 Images: Case Study of Rome. Land (Basel) 2023;12. https://doi.org/10.3390/land12040879.
[10] Razafinimaro A, Hajalalaina AR, Rakotonirainy H, Zafimarina R. Land cover classification based optical satellite images using machine learning algorithms. International Journal of Advances in Intelligent Informatics 2022;8:362–80. https://doi.org/10.26555/ijain.v8i3.803.
[11] Kareem RSA, Ramanjineyulu AG, Rajan R, Setiawan R, Sharma DK, Gupta MK, et al. Multilabel land cover aerial image classification using convolutional neural networks. Arabian Journal of Geosciences 2021;14:1681. https://doi.org/10.1007/s12517-021-07791-z.
[12] Dewangkoro HI, Arymurthy AM. Land use and land cover classification using CNN, SVM, and Channel Squeeze & Spatial Excitation block. IOP Conf Ser Earth Environ Sci 2021;704. https://doi.org/10.1088/1755-1315/704/1/012048.
[13] Akshaya KG, Sathyanarayan Sridhar R. Prediction of land use and landcover changes in Tiruppur Tamilnadu using hybrid convolutional neural network. Global Nest Journal 2023;25:1–11. https://doi.org/10.30955/gnj.004716.
[14] Xia B, Kong F, Zhou J, Wu X, Xie Q. Land Resource Use Classification Using Deep Learning in Ecological Remote Sensing Images. Comput Intell Neurosci 2022;2022. https://doi.org/10.1155/2022/7179477.
[15] González RM, González MAB, Cruz AM, González AR, Pérez AL. Classification of land use and vegetation with convolutional neural networks. Rev Mex Cienc For 2022;13:97–119. https://doi.org/10.29298/rmcf.v13i74.1269.
[16] Tan J, Zuo J, Xie X, Ding M, Xu Z, Zhou F. MLAs land cover mapping performance across varying geomorphology with Landsat OLI-8 and minimum human intervention. Ecol Inform 2021;61:101227. https://doi.org/10.1016/j.ecoinf.2021.101227.
[17] Alhassan V, Henry C, Ramanna S, Storie C. A deep learning framework for land-use/land-cover mapping and analysis using multispectral satellite imagery. Neural Comput Appl 2020;32:8529–44. https://doi.org/10.1007/s00521-019-04349-9.
[18] Kalantar B, Mansor S Bin, Sameen MI, Pradhan B, Shafri HZM. Drone-based land-cover mapping using a fuzzy unordered rule induction algorithm integrated into object-based image analysis. Int J Remote Sens 2017;38:2535–56. https://doi.org/10.1080/01431161.2016.1277043.
[19] Mazzia V, Khaliq A, Chiaberge M. Improvement in land cover and crop classification based on temporal features learning from Sentinel-2 data using recurrent-Convolutional Neural Network (R-CNN). Applied Sciences (Switzerland) 2020;10. https://doi.org/10.3390/app10010238.
[20] Andono PN, Rohman MS, Megantara RA, Pergiwati D, Wahyudi F, Shidik GF, et al. Refining Parameter Values in Convolutional Neural Networks Using Weight Modification Particle Swarm Optimization for Enhanced Coral Reef Condition Classification. International Journal of

Intelligent Engineering and Systems 2024;17:779–90. https://doi.org/10.22266/ijies2024.1231.59.
[21] Clermont J, Woodward-Gagné S, Berteaux D. Digging into the behaviour of an active hunting predator: arctic fox prey caching events revealed by accelerometry. Mov Ecol 2021;9:1–12. https://doi.org/10.1186/s40462-021-00295-1.
[22] Qin P-F, Li W-H, Wang D, Huang G-L, Fan W, Sim C-Y-D. A Linear Decreasing Inertia Weight Particle Swarm Optimization Base on a K-means Clustering Chaotic Sampling for Antenna Design. 2022 Cross Strait Radio Science & Wireless Technology Conference (CSRSWTC), IEEE; 2022, p. 1–3. https://doi.org/10.1109/CSRSWTC56224.2022.10098311.
[23] Choudhary S, Sugumaran S, Belazi A, El-Latif AAA. Linearly decreasing inertia weight PSO and improved weight factor-based clustering algorithm for wireless sensor networks. J Ambient Intell Humaniz Comput 2023;14:6661–79. https://doi.org/10.1007/s12652-021-03534-w.
[24] Sari Y, Arifin Y, Novitasari, Faisal M. Implementation of Deep Learning Based Semantic Segmentation Method To Determine Vegetation Density. Eastern-European Journal of Enterprise Technologies 2022;5:42–54. https://doi.org/10.15587/1729-4061.2022.265807.
[25] Sari Y, Arifin YF, Novitasari N, Faisal MR. Effect of Feature Engineering Technique for Determining Vegetation Density. International Journal of Advanced Computer Science and Applications 2022;13:655–61. https://doi.org/10.14569/IJACSA.2022.0130776.
[26] Ibrahim AO, Elfadel EME, Hashem IAT, Syed HJ, Ismail MA, Osman AH, et al. The Artificial Bee Colony Algorithm: A Comprehensive Survey of Variants, Modifications, Applications, Developments, and Opportunities. Archives of Computational Methods in Engineering 2025;32:3499–533. https://doi.org/10.1007/s11831-025-10269-w.
[27] Pramunendar RA, Alomoush A, Alzami F, Wahyudi F, Mambang, Sari Y, et al. Robust GOA-Driven CNN Parameter Optimization for Effective Rice Leaf Disease Classification. JOIV : International Journal on Informatics Visualization 2026;10.
[28] Santosa PI, Pramunendar RA. A Robust Feature Construction for Fish Classification Using Grey Wolf Optimizer. Cybernetics and Information Technologies 2022;22:152–66. https://doi.org/10.2478/cait-2022-0045.
[29] Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software 2017;114:163–91. https://doi.org/10.1016/j.advengsoft.2017.07.002.
[30] Mirjalili S, Lewis A. The Whale Optimization Algorithm. Advances in Engineering Software 2016;95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008.
[31] Trojovska E, Dehghani M, Trojovsky P. Zebra Optimization Algorithm: A New Bio-Inspired Optimization Algorithm for Solving Optimization Algorithm. IEEE Access 2022;10:49445–73. https://doi.org/10.1109/ACCESS.2022.3172789.
[32] Andono PN, Shidik GF, Prabowo DP, Yanuarsari DH, Sari Y, Pramunendar RA. Feature Selection on Gammatone Cepstral Coefficients for Bird Voice Classification Using Particle Swarm Optimization. International Journal of Intelligent Engineering and Systems 2023;16:254–64. https://doi.org/10.22266/ijies2023.0228.23.
[33] Mohammed H, Rashid T. FOX: a FOX-inspired optimization algorithm. Applied Intelligence 2023;53:1030–50. https://doi.org/10.1007/s10489-022-03533-0.
[34] Prabowo DP, Rohman MS, Megantara RA, Pergiwati D, Saraswati GW, Pramunendar RA, et al. Adaptive Inertia Weight Particle Swarm Optimization for Augmentation Selection in Coral Reef Classification with Convolutional Neural Networks. JOIV : International Journal on Informatics Visualization 2025;9:216. https://doi.org/10.62527/joiv.9.1.2726.
[35] Shidik GF, Anggi Pramunendar R, Nurtantio Andono P, Arief Soeleman M, Pujiono P, Aria Megantara R, et al. Bird Species Classification Enhancement via Adaptive Inertia Weight Particle Swarm Optimization-Based Image Augmentation Selection. IEEE Access 2024;12:197048–60. https://doi.org/10.1109/ACCESS.2024.3521455.