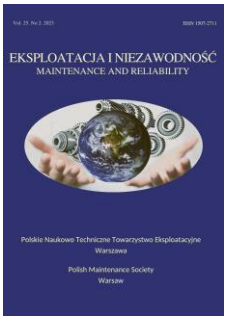


Eksploatacja i Niezawodnosc – Maintenance and Reliability
Volume 28 (2026), Issue 4
journal homepage: http://www.ein.org.pl

Article citation info:
Banupriya G, Sakthivel B, Duraichi N, A reliability-centered approach for solar panel fault detection and classification via autonomous drone system in solar farms, Eksploatacja i Niezawodnosc – Maintenance and Reliability 2026: 28(4) http://doi.org/10.17531/ein/220512

A reliability-centered approach for solar panel fault detection and classification via autonomous drone system in solar farms

Indexed by: Web of Science Group

Banu Priya GanapathyRaman[a,*], Sakthivel Balu[b], Duraichi Natarajan[c]

[a] Electronics and Communication Engineering, ULTRA College of Engineering & Technology, India
[b] Electronics and Communication Engineering, Pandian Saraswathi yadav Engineering College, India
[c] Electronics and Communication Engineering, Saveetha School of Engineering, saveetha Institute of Medical and Technical sciences, India

Highlights

- A drone-based system captures real-time images of solar panels for fault detection.
- The system achieves 96.2% accuracy in classifying faults.
- Residual boosting-based LSTM architecture.
- Enhanced reliability and maintenance.
- Practical implementation on drone and solar farms.

Abstract

In this work, a drone-based system is implemented for autonomous real-time fault detection in solar farms for improved reliability of power generation. To capture and analyze high-dimensional features from aerial images of solar panels, three powerful feature-extracting models are used. Then, for feature optimization, a new modified hybrid metaheuristic called Red Panda-Dhole Optimization (RPDO) is proposed. For accurate classification, a residual boosting-based Long Short-Term Memory (LSTM) architecture is proposed. The developed drone, based on the STM32F427VG microcontroller, follows a predefined flight path autonomously and captures images of the solar farm at regular intervals. These images are transmitted to a standalone system where they are processed and classified into different fault categories using a trained model. The proposed model is trained and tested on inspection videos recorded with drones in solar farms. The full model achieved the highest accuracy of 96.2% for the classification of solar panel status into Bird-drop, Clean, Dusty, Electrical-damage, and Physical-damage.

Keywords
drone, RPDO, LSTM, solar farm, fault categories

1. Introduction

As the need for power generation continues to rise, solar power has become important in power generation. It offers a reliable and eco-friendly way to generate power. Photovoltaic (PV) systems convert Solar energy into electrical power. Due to the benefits of renewable energy, reduced expenses, and technological advancements, it has become more popular in recent decades. In 2024, global solar generation surpassed 2,000 TWh, a 30% increase, and its share of the world's power reached about 7%, according to the International Energy Agency (IEA). PV farms are frequently located in remote locations with little sun shadowing, like plains and hills, and have massive capabilities (hundreds of MW, for example). They also encompass a vast geographic area. It's difficult to identify the faults in PV modules due to environmental causes such as Wind, snow and dust. It greatly reduces the PV module's output and results in corrosion and short circuits. It reduces the energy output of PV systems, shortens the service life of the modules, and decreases the return on investment. As a result, it is necessary to perform periodic inspection, adequate maintenance and timely resolution of any issues. On the other hand, due to

(*) Corresponding author.
E-mail addresses: Banu Priya Ganapathy Raman (ORCID: 0009-0008-8051-1746) banupriyag@ucetw.ac.in, Sakthivel Balu (ORCID: 0000-0003-0032-6775) 786sakthivel@gmail.com, Duraichi Natarajan (ORCID: 0009-0000-0349-5932) duraichin.sse@saveetha.com

the performance standards and the rapid development of large PV networks, early defect detection is promptly needed [1].

Towards these challenges, the need for intelligent, robust, cost-effective and reliable remedies to perform periodical inspection and maintenance operations is constantly highlighted, especially with the implementation of PV plants on increasingly large geographic scales. In recent advances inspection of PV installations took a competitive edge in using Unmanned Aerial Vehicles (UAVs) and image sensors [2].

Aerial imaging enables quicker and more reliable data acquisition, large-scale coverage and increased accessibility; it is a well-known efficient method for module inspection [3,4]. Recent research is moving on to autonomously detect defects and anomalies on solar panels, thereby increasing the accuracy and efficiency. The combination of UAVs and a Deep learning-based framework produces a revolutionary approach, addressing inherent challenges and limitations in traditional inspection methods. Furthermore, integration of UAVs enables real-time data acquisition and analysis. Also, it supports preventative maintenance and an excellent decision-making process. Deep learning-based PV module defect detection is crucial in identifying automated identification of surface defects such as dust and bird droppings, as well as physical and electrical defects on PV modules. By using the capabilities of UAVs, industries can achieve a well-organised and dynamic inspection workflow. It increases asset utilisation and maintains compliance and overall system performance.

Classical inspection methods for PV module inspection in visual systems use bounding boxes, instance segmentation, or traditional image processing methods to recognizes and highlight defects, but they require additional overhead for real-time imagery data. UAV-based PV inspection, the drone continuously records RGB/IR footage of solar modules while in flight and sends it to the ground station over a real-time wireless connection such as radio telemetry, 5G, or Wi-Fi. In order to extract defect-related features like hotspots, fractures, soiling, or discolouration, the ground station (or an onboard edge device) receives the video stream and processes each frame using deep learning models, usually CNNs, Transformers, or hybrid architectures. Real-time fault identification while the UAV is in motion is made possible by the classification or detection model's instantaneous analysis of these features, as shown in Figure 1. Without the need for post-flight analysis, this live processing pipeline enables quick decision-making, instantaneous quality evaluation, and effective large-scale PV plant monitoring.

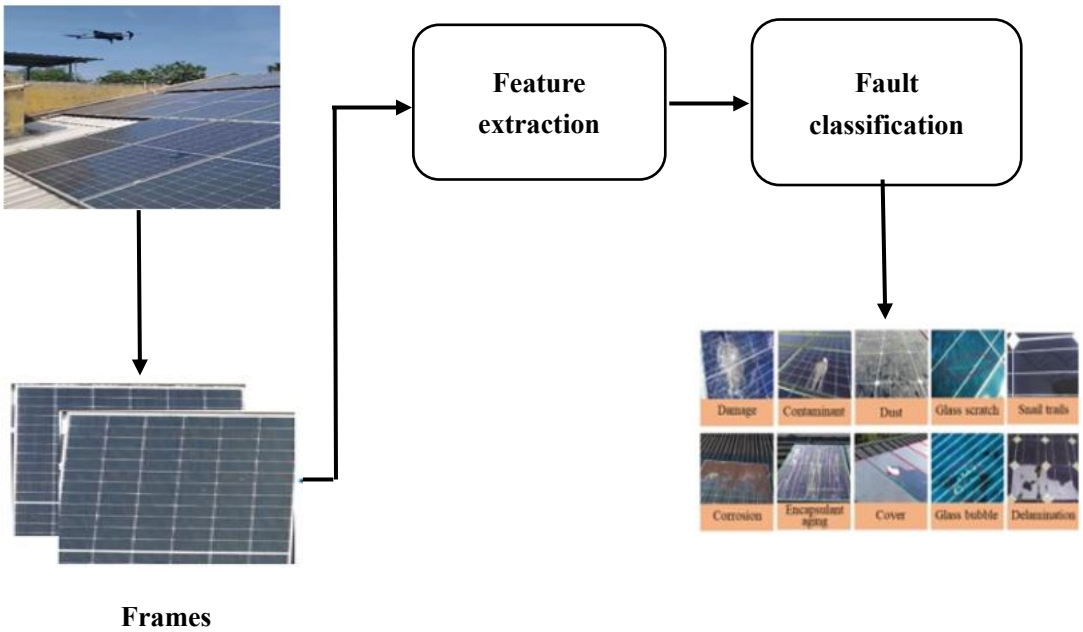


Figure 1. UAV-based PV fault analysis.

The benefits of RGB video over static visuals are highlighted in more recent studies. Defects can be seen from various perspectives and across consecutive frames thanks to the improved spatial and temporal information provided by continuous video taken during UAV flights. This increases visibility of little fractures or soiling patches as the drone moves,

improves defect localization, and lowers false positives brought on by glare, shadows, or reflections. Real-time processing via onboard edge devices or ground stations is also supported for UAV-based video streams. These video frames can be instantly inferred by contemporary deep learning models, such as CNNs, transformer-based networks, and YOLO versions. This eliminates the need to retain big information and allows for instant soil evaluation, dynamic defect tracking, and quicker maintenance decision-making.

It demonstrates that real-time RGB video analysis using UAVs provides a scalable, effective, and economical way to keep an eye on big PV installations. By offering quick coverage, accurate identification of obvious surface flaws, and real-time inspection capabilities, it overcomes the drawbacks of manual and sensor-intensive conventional methods. Because of this, UAV-RGB systems are becoming more and more useful for monitoring the operational phase of contemporary utility-scale PV projects.

In summary, we have made the following contributions:

- To allow for continuous aerial monitoring, a UAV-based system is used to record high-resolution real-time RGB videos of PV modules.
- A pipeline for defect analysis based on deep learning that uses multi-feature fusion to extract contextual and rich spatial information from video frames.
- An optimization approach that reduces computing overhead, increases model robustness, and chooses the best discriminative features.
- A hybrid classification system that uses continuous UAV video streams to accurately classify and identify numerous PV problems in real time.

This proposed solution optimizing resource consumption and operational expenses while enhancing PV system performance.

2. Related works

An overview of similar techniques for RGB-based real-time PV module inspection using UAVs is provided below. We evaluate current methods in terms of real-time classification accuracy, visual defect identification, feature fusion techniques, and video-based module detection, optimization and classification methods.

I) UAV based PV module inspection

Aerial object recognition is challenging in the identification of tiny objects, low illuminance conditions, and various angles of orientations. UAV-based inspections offer a non-contact, high-resolution, scalable, and automated approach to tackling physical issues such as soiling, bird's nest, encapsulate delamination, discoloration, and PV cell break. Li et al. [5] detect defects in PV modules such as soiling, discoloration, and dust using feature matching and Gaussian-based edge enhancement. Wang, J [6] et al proposed an autoencoder-based Support vector machine approach for finding defects such as burn marks, snail trails, discoloration, delamination, and glass breakage along with healthy modules.

UAVs provide consistent data collection and eliminate operator dependency when paired with autonomous flight planning by maintaining ideal flight altitude and camera angles. GPS-based geo-referencing frameworks to precisely identify the faults of particular PV modules are suggested by Kuo et al. [7]. This eliminates the difficulty of physically locating each defect in order to do maintenance. Ma, X et al [8] propose a hybrid optimization method that is a global path planning algorithm to plan the overall route with an updated DWA (Dynamic Window approach) to handle the narrow space and many obstacles in the UAV's flight path. For real-time UAV inspection Rodriguez-Vazquez et al. (9) suggest a TensorRT-optimised keypoint recognition model captures videos with 60 FPS and provides fast and accurate PV module detection. Moradi Sizkouhi [10] proposed optimized UAV flight route may effectively scan large PV plants in a reliable way, neglecting the need for localization. SIFT (Scale-Invariant Feature Transform) and RANSAC (Random Sample Consensus) methods combine the UAV captured IR and images and then a CNN detect and classifies the defects in PV modules, then GPS coordinates are used to locate the defects by pixel positions [11].

II) Deep learning-based Fault Detection

Using Wi-Fi, 4G/5G, or RF connectivity, a high-resolution RGB camera continuously transmits footage to the ground station while the UAV flies over the solar plant row by row during aerial inspection. Real-time processing and frame-by-frame decoding are applied to the video. After labelling the faults such as clean, Dusty, Bird’s nest, physical damage, electrical damage, a deep learning-based framework extracts

discriminative features from every frame. A CNN-based system was proposed by Usharani, M et al [12] for identifying dust and faults, using image processing steps like HSV conversion, color extraction, Gaussian filtering, and thresholding to identify solar panels, with difficulty in video processing. RGB-based PV defect detection relies heavily on CNNs, with popular models such as VGG16/19, ResNet-18/50, and EfficientNetV2. Using DenseNet-201 features, J48 (a decision-tree-based selector), and a WiSARD classifier, Sridharan et al. (13) achieved 100% accuracy in a quick testing time of 1.44 seconds. Kamal [14] used EfficientNetV2 structures in order to improve crack and hotspot identification over single-modality techniques. Di Tommaso et al. [15] achieved quick inference (~0.98 s per image) by using a two-stage YOLOv3 pipeline to detect both panels and faults from UAV RGB/IR images. In comparison to normal YOLOv8, Zhang, Li, Wang, and Chen [16] reported a 3–5% mAP improvement and faster inference using ESD-YOLOv8, which improves tiny defect identification with attention and refined feature fusion. Attention-based Vision Transformer (ViT) model treats images as a sequence of patches to extract high-level features, and autoencoders are also used to extract features effectively performs better.

III) Real time video analysis

Real-time UAV analysis offers continuous frame-by-frame monitoring, boosting coverage accuracy, reducing overlooked faults, and allows dynamic tracking of anomalies during flight. Faults detected in one frame are exactly recognized as the same defect in the upcoming frames, termed as temporal consistency [17], which is offered in real-time detection. A lightweight CNN–CNN-Transformer model designed for quick, real-time object recognition on UAV video streams with High-speed, on-board inference is provided by their network, allowing for continuous in-flight analysis, was presented by Ye et al [18]. These studies collectively demonstrate the great potential of real-time UAV systems for quick, precise, and effective PV plant monitoring.

IV) Multi-feature extraction

Feature extraction allows for quick and precise real-time object detection, tracking, and decision-making by transforming continuous video frames into concise and meaningful representations.

Feature extraction became far more powerful and automatic with the advent of CNNs. CNNs can accurately detect small cracks, hotspots, and surface differences in PV modules by directly learning hierarchical features from images, beginning with simple edges and working their way up to complex visual patterns. Convolution layers, pooling layers, and fully connected layers are some of the important layers that make up CNNs. Convolution layers extract edges, textures, and patterns at several levels using learnable filters. These feature maps are compressed by pooling layers to preserve important information while lowering noise, and fully connected layers combine the collected features for precise defect classification.

Deep CNNs can learn highly discriminative features end-to-end, as shown by Krizhevsky et al. [19]. By including skip connections, ResNet made it possible for very deep networks to train effectively without vanishing gradients, EfficientNet is accurate and lightweight because it jointly learns spatial and channel-wise representations using MBConv blocks. Vision Transformers [20] provide powerful feature extraction capabilities by breaking down images into fixed patches and modeling global relationships via self-attention. PV defect detection performance is significantly improved by contemporary models like EfficientNet, ResNet, and Vision Transformers that directly learn extensive spatial, structural, and textural data gathered from UAV Videos.

To maintain real-time UAV performance, lightweight backbones, feature-compression algorithms, pruning, and quantization are used. Multi-feature extraction is a very successful method for UAV-based PV inspection, as studies on multi-branch CNNs proposed by Zheng et al [21]. Compact feature maps are produced by the last convolution and pooling layers, summarizing all of the visual patterns that were learned. After flattening these feature maps, the retrieved data is combined and interpreted using fully connected layers. The final output of the network is shown as a classification vector, where each value represents the likelihood of a certain type of PV defect.

V) Feature Optimization

The multi-feature extraction strategy generates a single, enriched feature representation that is perfect for classification or detection tasks by combining complementary cues from the feature extraction module. All of these distinct features are merged into a single, all-inclusive vector after extraction. For

example, CTFuseNet (CNNTransformer Feature Fused Network) creates a single feature map for UAV picture segmentation and classification by combining local CNN features with global Transformer data [22]. Shang et al. [23] use a hybrid GWO (Grey Wolf Optimizer)–GA (Genetic Algorithm) strategy that combines global search with crossover refinement to achieve an optimal feature subset with reduced redundancy. Grasshopper Optimization Algorithm (GOA) gives a diabetes diagnosis by refining the high-level medical features, aiming to avoid repeated and irrelevant data that may confuse classification models [24]. Dhole-Inspired Optimization [25], Red Panda Optimization [26], each of which brings unique strengths in exploration and exploitation.

VI) Feature Classification

Optimization algorithms that select the high-level features and avoid redundancy from the feature set after feature extraction. These optimized features are then categorized into fault types like Bird's nest, dusty, electrical damage, physical damage or normal. According to Barraz et al. [27], deep learning models, such as CNNs, attention-based networks, and hybrid CNN–ViT architectures, are used to classify UAV-based PV modules. Alrifaey et al. [28] suggest a hybrid model that combines an Equilibrium Optimizer with LSTM-stacked autoencoders to improve features and accomplish precise real-time PV fault classification. Photovoltaic (PV) module faults are classified using CNN, Softmax layer predicts the label and finds the fault type from visual data [29]. Recurrent Neural Networks (RNNs) particularly Long Short-Term Memory (LSTM) network learns temporal patterns in sequential process data. The final layer assigns each time series predicts a process category, dynamic temporal features lead to excellent classification output in a real industrial process [30]. Optimized XGBoost classifier is used to detect brain tumors from MRI images, the gradient boosting technique combines many weak decision trees and hyperparameter tuning is done to perform strong classification [31].

3. Drone-based solar panel fault detection

To automate real-time fault analysis in solar farms, in this work, a drone-based system is proposed. The overall system architecture is shown in Figure 2. In this system, an autonomous drone is equipped with high-resolution cameras to capture detailed images of solar panels from different angles. For feature extraction, the different feature extractors like Vision Transformer (ViT), ResNet101, and EfficientNetB4 models are used. To optimize the feature extraction process, a hybrid optimizer called Red Panda-Dhole Optimization (RPDO) is proposed. This optimizer fine-tunes the extracted features by balancing the exploration and exploitation process. The most relevant features are selected for accurate fault detection. Then, a hybrid model based on gradient boosting LSTM is proposed for classification. The developed drone follows a predefined flight path autonomously and captures images of the solar farm at regular intervals. These images are transmitted to a standalone system where they are processed and classified into different fault categories using a trained model.

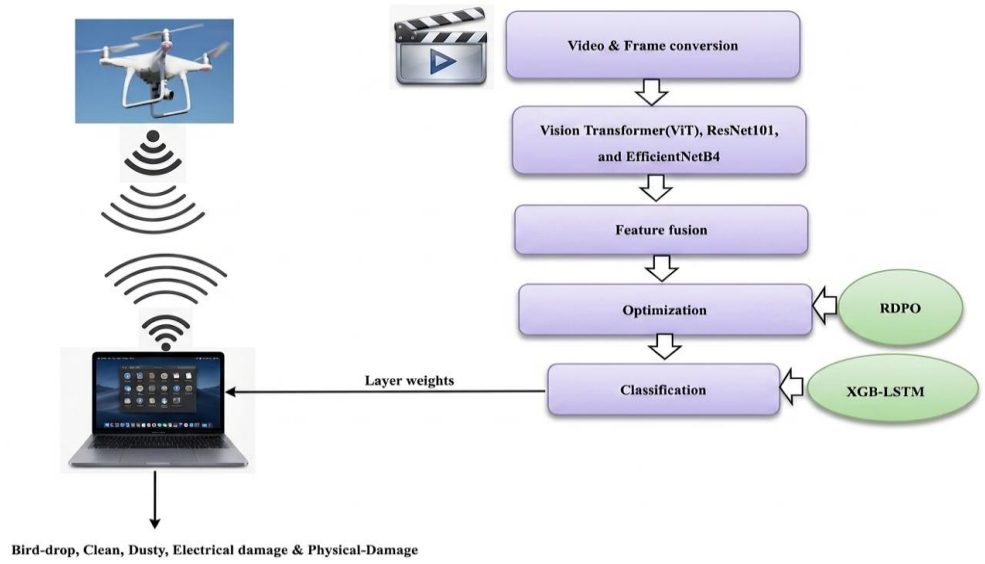


Figure 2. Overall system workflow.

The Vision Transformer (ViT) architecture [32] is based on the idea of treating an image as a sequence of patches. ViT breaks an image into fixed-size non-overlapping patches and processes them using transformer blocks. This block is known for its self-attention mechanism. This strategy is used for the model to capture long-range dependencies across patches.

Let $I \in \mathbb{R}^{H\times W\times C}$ be the input image, where H and W are the height and width, and C is the number of channels (such as 3 for RGB images). The image is divided into $N = (\frac{H}{P}) \times (\frac{W}{P})$ non-overlapping patches, each of size $P \times P$. Then, these patches are flattened into vectors which produces a patch sequence $X_{\text{patch}} \in \mathbb{R}^{N\times(P^2\cdot C)}$.

To give the patches a sense of position within the image, position embeddings $E_{\text{pos}} \in \mathbb{R}^{N\times D}$ are added to the patch embeddings. Thus, the input to the transformer blocks is expressed as follows:

$$X_{\text{input}} = X_{\text{patch\_embedded}} + E_{\text{pos}} \in \mathbb{R}^{N\times D} \quad (1)$$

Where D is the dimensionality of the patch embeddings. Then, the transformer block performs self-attention to model the relationships between the patches. For a given input X, the self-attention mechanism can be written as:

$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (2)$$

Where Q, K, and V are the query, key, and value matrices, respectively, and d_k is the dimension of the key vectors. This is used for the model to attend to different patches of the image at the same time. After multiple transformer blocks, the output is aggregated using Global Average Pooling (GAP):

$$X_{\text{output}} = \frac{1}{N}\sum_{i=1}^{N} X_i \quad (3)$$

ResNet101 Architecture

ResNet101 is a variant of the ResNet (Residual Network) [33]. This network uses residual connections to mitigate the problem of vanishing gradients when training deep neural networks. By using residual connections, the input skips certain layers in the network without degradation in performance. ResNet101 operates by stacking multiple Residual Blocks. The operation of the residual block can be expressed as follows:

$$\text{ResBlock}(x) = \text{Conv}(x) + x \quad (4)$$

Where x is the input to the residual block. In the above equation, the addition of x to the output of the convolution is used for the model to learn residual mappings rather than direct mappings. During deeper training, this residual mapping supports the model to learn the difference between the input and the output.

In ResNet101, each residual block uses Bottleneck Architecture, where the block consists of three layers: a 1x1 convolution, a 3x3 convolution, and a 1x1 convolution. This bottleneck design reduces the computational complexity of the model and retains its ability to learn powerful representations. The bottleneck block can be stated as follows:

$$\text{BottleneckBlock}(x) = \text{Conv1x1}(x)\text{->Conv3x3}(x)\text{->Conv1x1}(x) \quad (5)$$

The output from the bottleneck is passed through GAP to reduce the spatial dimensions of the feature maps to produce a single feature vector for classification:

$$x_{\text{GAP}} = \frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W} x_{ij} \quad (6)$$

Where H and W are the height and width of the feature map after convolutions.

EfficientNetB4 Architecture

EfficientNetB4 is a variant of the EfficientNet model based on the concept of compound scaling. The compound scaling is used to adjust depth, width, and resolution to optimizes model performance with fewer parameters. The depth rate defines the number of filters. The width rate defines the number of layers. The resolution defines the image size. Compared to traditional CNN architectures, EfficientNetB4 uses Depthwise Separable Convolutions which separate the convolutional operation into two parts: a depthwise convolution followed by a pointwise convolution (a 1x1 convolution). This significantly reduces the computational complexity. For a given input x, the depthwise separable convolution can be expressed as:

$$x_{\text{depthwise}} = \text{DepthwiseConv}(x) \quad (7)$$

Where DepthwiseConv applies a filter to each channel separately. Then, the output passed through a Pointwise Convolution:

$$x_{\text{pointwise}} = \text{Conv1x1}(x_{\text{depthwise}}) \quad (8)$$

In EfficientNet, these layers are stacked to balance width, depth, and resolution using a scaling coefficient α, β, γ for each:

$$\text{scale}_d = \alpha^{\phi}, \text{scale}_w = \beta^{\phi}, \text{scale}_r = \gamma^{\phi} \quad (9)$$

Where ϕ is the scaling factor. The output of the final convolutional layers is aggregated through GAP:

$$x_{\text{GAP}} = \frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W} x_{ij} \quad (10)$$

3.1. Red Panda-Dhole Optimization (RPDO)

Dhole-Inspired Optimization (DIO) is a bio-inspired metaheuristic algorithm based on the social and cooperative behaviors of dholes [34]. In DIO, this translates to a population-based search where the best solution called Lead Vocalizer guides the entire population of search agents. The algorithm emphasizes cooperative hunting where agents adjust their positions based on the Lead Vocalizer's location. DIO balances exploration and exploitation by adjusting the V parameter. This parameter decays over time to shift the algorithm's focus from broad exploration to more refined exploitation as the search progresses. Despite its innovative design, DIO has several drawbacks. The over-dependency on the Lead Vocalizer can cause premature convergence when the leader becomes stuck in a local optimum. This leads to limited exploration diversity. In addition, its boundary handling approach is simplistic and can disrupt the search process.

To address these issues, the Red Panda-Dhole Optimization (RPDO) hybrid is proposed. RPDO combines the DIO population structure and Lead Vocalizer concept with Red Panda Optimization (RPO)'s to balance exploration and exploitation phases [35]. The climbing strategy from RPO is used to perform local search for efficient fine-tuning of solutions. By combining DIO's social structure with RPO's clear separation of exploration and exploitation phases, the proposed RPDO becomes a more adaptable and efficient algorithm.

Initialization

The algorithm begins by initializing a population of Nsearch agents in an m-dimensional search space. Each agent represents a solution as follows:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} \dots x_{1,j} \dots x_{1,m} \\ \vdots \\ x_{i,1} \dots x_{i,j} \dots x_{i,m} \\ \vdots \\ x_{N,1} \dots x_{N,j} \dots x_{N,m} \end{bmatrix}_{N\times m} \quad (11)$$

Where, $X_i$is the i-th agent, and $x_{i,j}$is its position in the j-th dimension. Each agent is initialized randomly within the bounds:

$$x_{i,j} = lb_j + rand(0,1) \cdot (ub_j - lb_j) \quad (12)$$

for $i = 1,2,\dots,N$ and $j = 1,2,\dots,m$, where lb_j and ub_j are the lower and upper bounds for the j-th dimension, and $rand(0,1)$is a uniform random number. The fitness value $F_i = F(X_i)$is computed for each agent. The agent with the best fitness is designated as the initial Lead Vocalizer, X_{best}. The position of each agent is updated iteratively through two consecutive phases.

Phase 1: Cooperative Foraging (Exploration Phase)

This phase is designed to mimic a wide-ranging search for resources. This phase combines the red panda's strategy of seeking multiple food sources with the dhole's social cooperation. The main aim of this phase is to explore the search space globally to avoid premature convergence to local optima.

Step 1: Identify Promising Food Sources (PFS):

For each agent X_i, identify a set of all agents with better fitness with the Lead Vocalizer:

$$PFS_i = \{X_k \mid k \in \{1,2,\dots,N\} \text{ and } F_k < F_i\} \cup \{X_{\text{best}}\} \quad (13)$$

Step 2: Select a Target and Update Position:

A new candidate position is calculated for $Xi$by moving towards a weighted combination of a random agent from its PFS and the Lead Vocalizer:

$$X_{\text{new1},i} = X_i + r_1 \cdot ((w \cdot X_{\text{target}} + (1-w) \cdot X_{\text{best}}) - I \cdot X_i) \quad (14)$$

Where, $r_1$is a random number from $U(0,1)$, $X_{\text{target}}$is an agent randomly selected from PFS_i, $I$is a random integer from the set $\{1,2\}$, $w$is a dynamic weight for decentralization, calculated as:

$$w = w_{\max} - (\frac{t}{T}) \cdot (w_{\max} - w_{\min}) \quad (15)$$

Here, tis the current iteration, and Tis the maximum iterations. w_"max" and w_"min" are set to 0.9 and 0.1, which is used for the transition from exploring diverse targets to converging towards the leader.

Step 3: Greedy Selection:

The agent's position is updated only if the new position yields a better fitness:

$$X_i = \begin{cases} X_{\text{new1},i}, & \text{if } F(X_{\text{new1},i}) < F(X_i) \\ X_i, & \text{otherwise} \end{cases} \quad (16)$$

Phase 2: Skillful Climbing (Exploitation Phase)

After a broad exploration phase, the algorithm switches to a focused local search. This phase inspired by the red panda's precise climbing behavior. This phase is responsible to search around the current positions to refine solutions and achieve high precision.

Step 1: Local Position Update:

A new candidate position is generated by taking a small and randomized step around the current position. The step size

decreases as iterations progress:

$$x_{new2,i,j} = x_{i,j} + \left(lb_j + r_2(ub_j - lb_j)\right) \cdot \left(1 - \frac{t}{T}\right) \quad (17)$$

for j=1,2,…,m, where $r2$is a random number from $U(0,1)$, and $t$is the current iteration counter.

Step 2: Greedy Selection:

The agent moves to this new position only if it improves the fitness:

$$X_i = \begin{cases} X_{\text{new2},i}, & \text{if } F(X_{\text{new2},i}) < F(X_i) \\ X_i, & \text{otherwise} \end{cases} \quad (18)$$

Leadership Update and Termination

After all agents have moved in both the exploration and exploitation phases, the entire population is re-evaluated. The agent with the absolute best fitness value has completed the new Lead Vocalizer. This continuous update ensures that the guiding force for the population is always the best solution discovered so far. It dynamically refines the search direction as better solutions are found. The fitness of all agents is evaluated. The agent with the best fitness in the current population becomes the new Lead Vocalizer:

$$X_{\text{best}} = \arg\min_{X_i} F(X_i) \quad (19)$$

The algorithm increments the iteration counter $t = t + 1$and repeats from Phase 1 until the maximum number of iterations $T$is reached. The final $X_{\text{best}}$is presented as the solution.

Pseudocode of the RPDO Algorithm

Input: Population size N, Maximum iterations T, Problem bounds lb, ub

Output: Best solution Xbest and its fitness Fbest

```
Initialize population X using equation (Initial Position)
Evaluate fitness F for all agents
Identify the best agent Xbest
t = 1
while t <= T do
    // Phase 1: Cooperative Foraging
    for each agent Xi in X do
        Create PFSi using equation (PFS)
        Select a random Xtarget from PFS_i
        Calculate dynamic weight w
        Generate new candidate X_new1,i using equation (Cooperative Foraging)
        if F(X_new1,i) < F(Xi) then
            Xi = X_new1,i
        end if
    end for

    // Phase 2: Skillful Climbing
    for each agent Xi in X do
        Generate new candidate Xnew2,i using equation (Skillful Climbing)
        if F(Xnew2,i) < F(Xi) then
            Xi = Xnew2,i
        end if
    end for

    // Update Leadership
    Evaluate fitness F for all updated agents
    Update Xbest if a better solution is found
    t = t + 1
end while

Return Xbest, Fbest
```

3.2. Feature fusion & optimization

After feature extraction, the RPDO algorithm is applied for feature optimization. RPDO is used to fine-tune the weights associated with each feature extractor and optimize the fusion process by selecting the most relevant features and eliminating redundant or irrelevant ones. The weighted averaging is used for feature fusion. The selected and optimized feature set is passed through the XGB-LSTM classifier.

3.3. XGB-LSTM

In this work, a new architecture of XGB-LSTM is proposed for classification. This classifier integrates residual boosting within LSTM layers to increase the model's ability to handle sequential data and refine predictions iteratively. Compared to hybrid LSTM and XGBoost, the proposed XGB-LSTM combines the boosting mechanism directly into the LSTM architecture. This strategy is used for the model to capture temporal dependencies and progressively improve its predictions by adding residual corrections after each boosting stage.

Architecture of XGB-LSTM

The XGB-LSTM model consists of multiple key components: an initial LSTM layer, a series of residual boosting

blocks, and a final LSTM layer. Initially, the input sequence is processed using an LSTM layer. The LSTM captures temporal dependencies and learns patterns in the sequential data. The input sequence Xconsists of feature vectors for each time step which is represented as:

$$X = [X_1, X_2, \dots, X_T] \quad (20)$$

where $X_t \in \mathbb{R}^F$is the feature vector at time step t, and $T$is the total number of time steps. The initial output H_1at each time step $t$is computed by the LSTM layer as follows:

$$H_1 = \text{LSTM}_1(X_t) \quad (21)$$

The LSTM layer itself computes the following components for each time step:

Forget Gate:

$$f_t = \sigma(W_f X_t + U_f H_{t-1} + b_f) \quad (22)$$

Input Gate:

$$it = \sigma(WiXt + UiHt - 1 + bi) \quad (23)$$

Candidate Memory:

$$Ct = \tanh(WCXt + UCHt - 1 + bC) \quad (24)$$

Cell State:

$$Ct = ft \cdot Ct - 1 + it \cdot Ct \quad (25)$$

Output Gate:

$$ot = \sigma(WoXt + UoHt - 1 + bo) \quad (26)$$

Final Output:

$$Ht = ot \cdot \tanh(Ct) \quad (27)$$

The output at each time step $H_t$represents the learned feature for the sequence.

After LSTM layer, the multiple residual boosting blocks are applied iteratively. These blocks are used to learn the residuals (errors) between the previous predictions and the actual values. The residual boosting block output is defined as follows:

$$R_k = \text{LSTM}_k(H_{k-1}) + H_{k-1} \quad (28)$$

Where, $\text{LSTM}_k(H_{k-1})$is the output of the k-th residual block, $H_{k-1}$is the output of the previous residual block. Each residual block refines the prediction by adding the residuals to the output of the previous block. The residuals can be formulated as:

$$\Delta H_k = \text{LSTM}_k(H_{k-1}) \quad (29)$$

The final output of the k-th block is:

$$R_k = H_{k-1} + \Delta H_k \quad (30)$$

The main idea is that each subsequent LSTM block tries to learn the residual errors from the previous stage and adds these corrections to the output. This process is similar to the boosting algorithms of XGBoost.

After iterating through multiple residual boosting blocks, the output of the final residual block $R_K$is passed to the final LSTM layer. This layer consolidates the temporal features into a fixed-length vector. The output of this final LSTM layer is computed as:

$$H_{\text{final}} = \text{LSTM}_{\text{final}}(R_K) \quad (31)$$

This layer is used to combine all the features learned across multiple residual boosting rounds and prepares the data for the final classification. In this work, for multi-class classification, a Softmax activation is used to convert the final output into probabilities for each class. The Softmax function is defined as:

$$\text{Softmax}(Y) = \frac{\exp(Y_i)}{\sum_{j=1}^{C} \exp(Y_j)} \quad (32)$$

Where, $Y_i$is the output for class i, $C$is the number of classes. Then, the final classification output is then computed as:

$$Y = \text{Dense}_{\text{classification}}(\text{Softmax}(H_{\text{final}})) \quad (33)$$

4. Results and discussion

The drone is powered by an 11.7V 5000mAh battery and equipped with BLDC motors for efficient propulsion. It uses a 3DR 915 MHz Radio Telemetry kit for wireless communication. This telemetry is used for real-time data transmission like GPS positioning, system voltage, and waypoint navigation. The system operates on the MAVLink protocol and is configured through Mission Planner or APM Planner for full-duplex communication. The drone is controlled by a FlySky FS-i6 2.4GHz 6-channel RC transmitter. The flight controller integrates an STM32F427VG microcontroller based on the ARM Cortex-M4 core for high computational performance. It also incorporates an MPU9250 IMU for accurate motion tracking and stabilization. Also, it is combined with the HMC5983 magnetometer for orientation and navigation. The visualization of the implemented drone with its analyizs is shown in Figure 3.

The proposed model is trained and tested on inspection videos recorded with drones in Niranjana Weaving Mill, Madurai, Tamil Nadu, India (625008). The video is recorded for the five different fault classes of Bird-drop, Clean, Dusty, Electrical-damage, and Physical-damage. Each class contains 6-10 inspection videos with each video having 150-250 frames. The model is trained with frame-level labels. Each frame within a video is individually labeled according to the presence of

a specific fault. The model is trained to handle multi-class labels within each video. Each video clip is approximately 20 minutes long with the total number of frames in each class being 1,500 frames for training and nearly 450 frames reserved for testing. The sample frame images are shown in Figure 4.

Figure 3. Implementation of drone & view of Niranjana weaving mill.

Figure 4. Visualization of different fault classes.

Initially, the model is trained using video files in Google Colab. After successful training, the model weights are saved and stored in the local system. For live classification, the drone is responsible for capturing and transmitting live video data to the system. The local system with a loaded weight environment performs classification with the OpenCV environment. This allows the drone to offload computationally expensive operations and maintaining real-time performance.

The plot in Figure 5 displays the distribution of feature values extracted by the DL models ViT, ResNet-101, and EfficientNet-B4. The distributions for all three models are centered closely around zero and are approximately bell-shaped. The EfficientNet-B4 features show a wider spread and slightly heavier tails compared to the ViT and ResNet-101 features.

The plot in Figure 6 shows the optimization process of a model across 50 Iterations. The final iteration successfully reaches a fitness level with optimized feature set. By the feature optimization, the feature count reduced from 4,608 to 2,150 for model training. Likewise, the inference time is reduced by

34.5%.

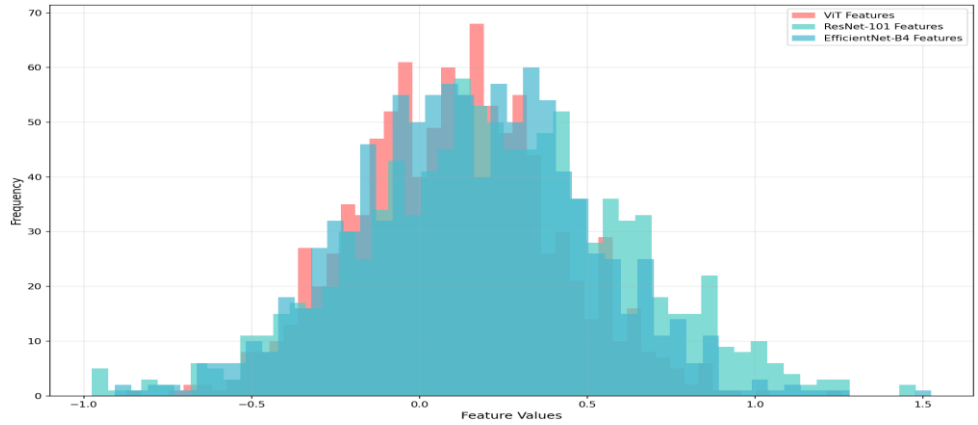


Figure 5. Visualization of feature distribution.

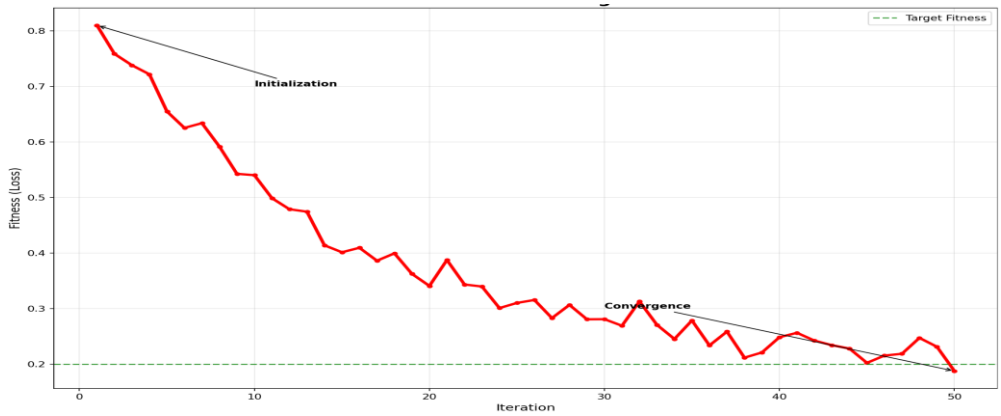


Figure 6. Feature optimization curve.

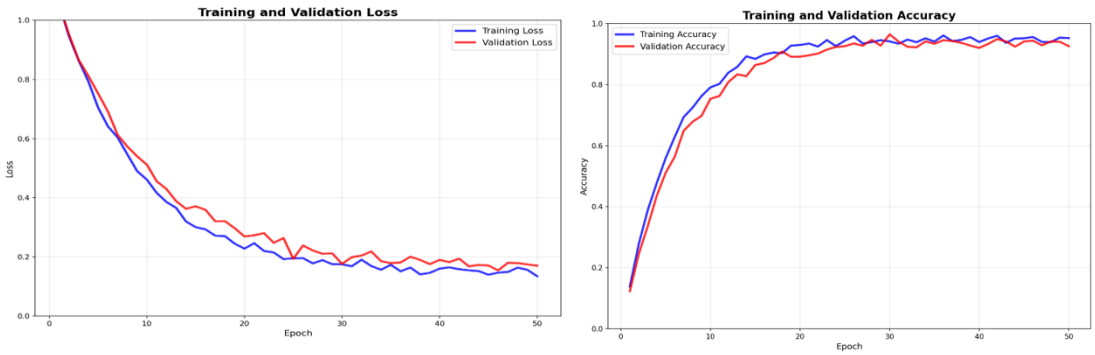


Figure 7. Model validations.

The model training and validation plots are shown in Figure 7. Both the training and validation loss curves show a sharp decrease in loss when the epoch increases. Similarly, the training accuracy and validation accuracy increase rapidly and converge to a high level. The sustained high validation accuracy indicates the model consistently performs well on unseen data.

The matrix Figure 8 shows the counts of True versus Predicted Labels. The diagonal cells represent correct classifications where the true and predicted labels match. The proposed model achieves high accuracy for all five categories. The off-diagonal cells represent misclassifications or errors.

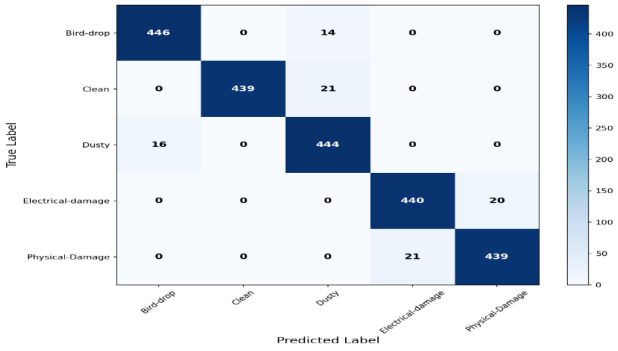


Figure 8. Confusion matrix plot of framewise classification.

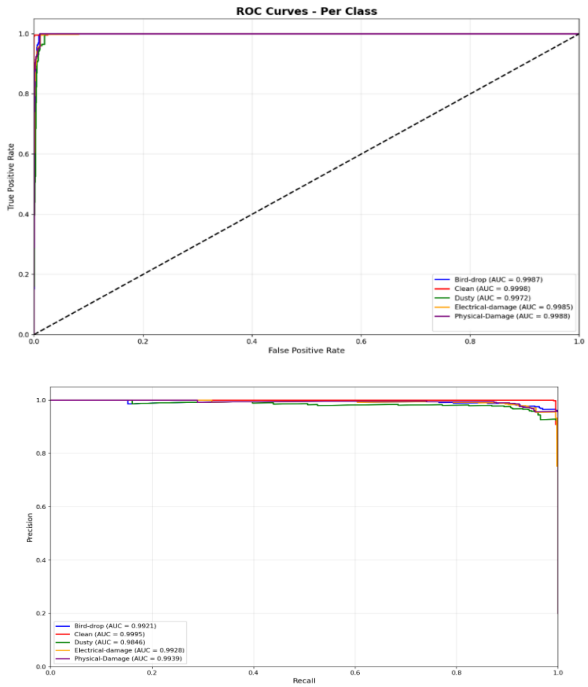


Figure 9. ROC and PR plot of the model.

The Receiver Operating Characteristic (ROC) curves for the proposed multi-class classification model are shown in Figure 9. All the curves are positioned extremely close to the top-left corner of the plot. The Area Under the Curve (AUC) values obtained range from 0.9972 to 0.9988. Likewise, the Precision-Recall (PR) Curves are shown in Figure 9. The closeness of the curves to the (1, 1) point indicates that the model maintains high precision even as the recall is also high. The class-wise accuracy of the model is given in Table 1. The model reaches strong values in all metrics for all classes. The results are graphically shown in Figure 10.

Table 1. Performance comparison.

Class	Precision	Recall	F1-Score
Bird-drop	0.9654	0.9696	0.9675
Clean	1.0000	0.9543	0.9767
Dusty	0.9269	0.9652	0.9456
Electrical damage	0.9544	0.9565	0.9554
Physical-Damage	0.9564	0.9543	0.9554

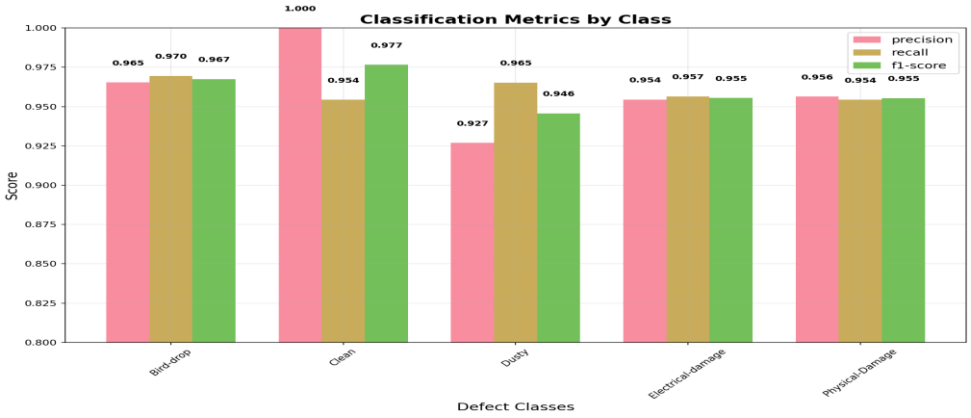


Figure 10. Performance analysis.

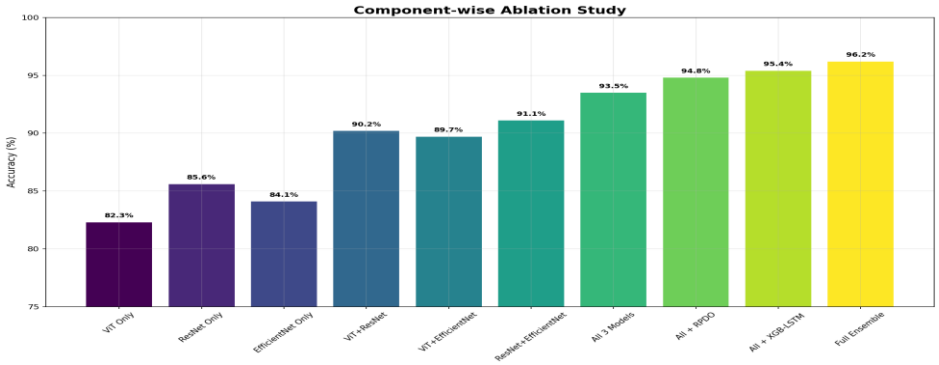


Figure 11. Ablation study of the model.

The component-wise ablation study of the model is given in Table 2. The ViT achieved an accuracy of 82.3%. The individual ResNet and EfficientNet obtained an accuracy of 85.6% and 84.1%, respectively. These results prove that each standalone model performs reasonably well for feature learning. The accuracy of the model is increased to 93.5% when models are combined. The integration of optimization increases the accuracy level to 95.4%. Finally, the full model achieved the highest accuracy of 96.2% for classification. The study is graphically shown in Figure 11.

Table 2. Ablation study.

Component / Model Combination	Accuracy (%)
ViT Only	82.3%
ResNet Only	85.6%
EfficientNet Only	84.1%
ViT + ResNet	90.2%
ViT + EfficientNet	89.7%
ResNet + EfficientNet	91.1%
All 3 Models	93.5%
All + RPDO	94.8%
All + XGB-LSTM	95.4%
Full Model	96.2%

Table 3. Comparison of XGB-LSTM with other classifiers.

Model/Classifier	Accuracy (%)
XGB-LSTM	95.9
Hybrid or ensemble XGBoost with	94.5
LSTM	93.3
XGBoost	94.2
Random Forest	93.0
SVM	91.2
Logistic Regression	85.4
K-Nearest Neighbors (KNN)	88.7
Decision Tree	89.5
Naive Bayes	82.8

Table 4. Comparison of RPDO with Other Optimizers.

Optimization Algorithm	Accuracy (%)	Convergence Time (Minutes)
Red Panda-Dhole Optimization (RPDO)	95.7	20-30
Particle Swarm Optimization (PSO)	93.5	15-20
Genetic Algorithm (GA)	92.8	25-40
Differential Evolution (DE)	92.2	15-25
Simulated Annealing (SA)	89.7	5-10
Cuckoo Search (CS)	91.3	15-25
Ant Colony Optimization (ACO)	90.5	20-30
Bayesian Optimization (BO)	94.0	30-50
Gradient-based Optimization	91.6	5-10

The performance comparison of the proposed classifier with other classifiers is given in Table 3. Due to residual boosting, the XGB-LSTM models achieve higher accuracy than the hybrid XGBoost with LSTM models. The performance comparison of RPDO with other optimizers is given in Table 4. The RPDO achieves the highest accuracy (95.4%) with a reasonable convergence time of 20-30 minutes. The computational overhead analysis of the model is given in Table 5. The proposed model has significantly higher parameters, FLOPs, and memory usage compared to simpler models. But its performance in terms of feature extraction and fault detection accuracy is much superior. Compared to the sole ResNet101 and ViT models, the proposed model shows minimal complexity increments. The overall complexity is within acceptable limits and proves the suitability for real-time implementation. By separating the training and inference phases, the drone operates efficiently in real-time without being burdened by the computational demands of model training. In this setup, the drone is responsible only for capturing high-resolution images of solar panels and sending them to a standalone system. The standalone system classifies the frames based on pre-trained weights. This architecture allows the drone to focus solely on capturing images and transmitting them without any delays caused by real-time processing. In this scenario, the hybrid Model provides a powerful solution for solar panel fault detection.

Table 5. Computational Complexity Analysis.

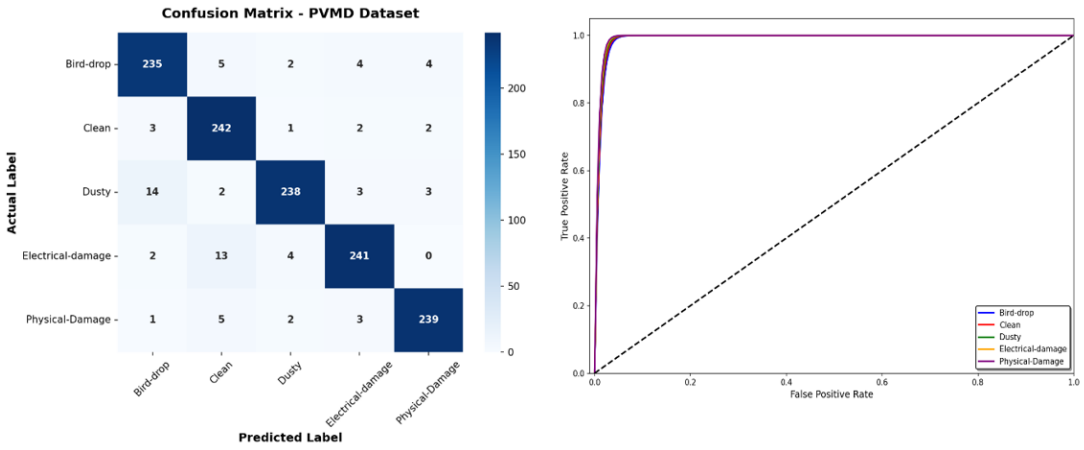
Model	Params (Millions)	FLOPs (Billions)	Inference Time (Avg ms)	Memory Usage (MB)	FPS	Complexity Score
Simple CNN	0.292	0.001	57.632	0.051	17.351	11.648
EfficientNetB0	4.708	0.009	60.811	0.039	16.444	14.052
ConVext	0.776	0.002	74.956	0.645	13.341	15.366
DenseNet121	7.565	0.015	74.979	0.004	13.337	18.027
EfficientNetB4	18.594	0.037	70.644	0.043	14.155	21.582
ResNet50	24.639	0.049	68.805	0.035	14.534	23.635
ViT (Vision Transformer)	38.791	0.078	67.160	1.180	14.890	29.090
ResNet101	43.710	0.087	85.610	0.609	11.681	34.693
Hybrid Feature Extractor	73.057	0.206	98.967	1.473	9.262	43.025

Table 6. 5-Fold Cross-Validation Performance on PVMD Dataset.

Fold	Precision	Recall	F1-Score	Accuracy (%)
1	0.9563	0.9502	0.9532	95.3
2	0.9634	0.9571	0.9602	96.1
3	0.9505	0.9558	0.9531	95.1
4	0.9642	0.9650	0.9646	96.5
5	0.9587	0.9603	0.9595	95.9
Average	0.9586	0.9577	0.9581	95.7

To validate and extend the fault detection capabilities of the model, the proposed model is applied to the PVMD dataset. It is collected from the Soshanguve South Campus. It consists of 1200 images with different faults. The 5-fold cross-validation is applied to verify the model's generalisation capabilities and ensure that the model is not overfitting to the training data. The obtained results are given in Table 6. It is observed that the model maintains high accuracy, precision, recall, and F1-score across different folds. The model shows strong robustness when

applied to different datasets. The confusion matrix and ROC plot for PVMD dataset is shown in Figure 12.a and 12.b. The diagonal line of dark blue squares indicates the better performance of the model. This suggests the model has a strong knowledge of the distinct visual features for each panel condition. This indicates that the model is highly effective at correctly identifying panel conditions and maintains an extremely low rate of false alarms.

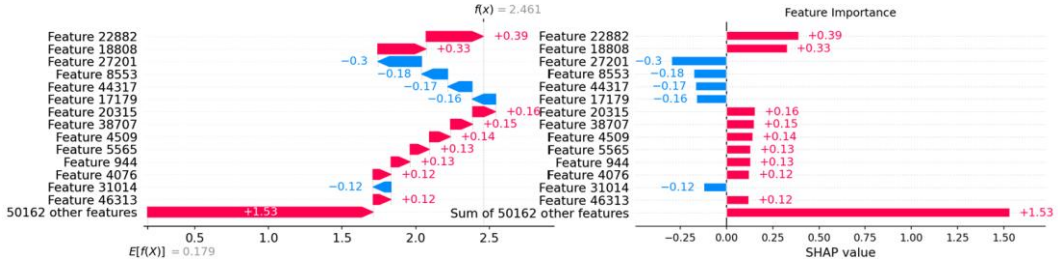


(a) (b)

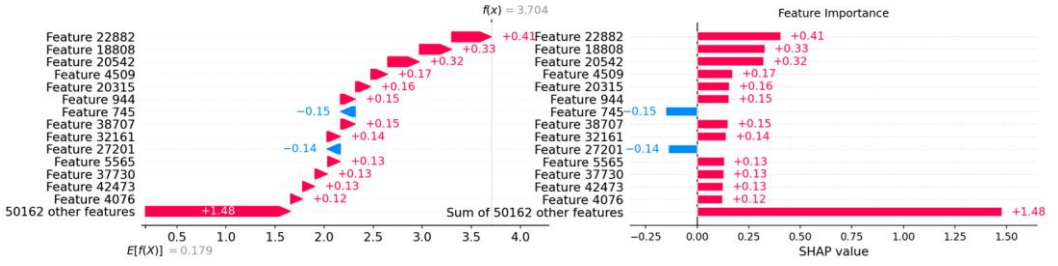
Figure 12. (a) Confusion matrix, (b) ROC Plot.

Figure 13 represents SHAP (SHapley Additive exPlanations) visualizations of the model for Normal and physical defect classes. It is used to interpret the predictions of the proposed model. The waterfall plots illustrate how the model's prediction is formed by starting from a base value and sequentially adding the contributions of individual features.

In Figure 13. a, the waterfall plot shows how the prediction starts from a base value and is influenced by different features. Positive contributions (red) from features such as Feature 22882 and Feature 18808 increase the prediction. The negative contributions (blue) like Feature 27201 are slightly reduced. These combined effects result in a final prediction value of approximately 2.461 which indicates a non-faulty condition. In Figure 13. b, the feature contribution represents a physical defect in the solar panel. The SHAP waterfall plot demonstrates a stronger cumulative effect of features pushing the prediction to a higher value of approximately 3.704.



(a)



(b)

Figure 13. (a) Normal condition, (b) Physical defect.

To validate whether the observed performance improvements are statistically meaningful, statistical hypothesis testing was conducted for both the optimization algorithms and classification models. The paired t-test and Wilcoxon signed-rank test are applied to the accuracy values obtained across multiple experimental runs (5-fold cross-validation). The statistical comparison between the proposed RPDO optimizer and other optimization techniques is given in Table 7.

The p-values obtained from both the paired t-test and the Wilcoxon test are less than 0.05 for all comparisons. It denotes that the performance improvement of RPDO over other optimizers is statistically significant. The high t-statistic values confirm that the difference in accuracy is not due to random variation. The statistical comparison of the proposed XGB-LSTM model with conventional classifiers is shown in Table 8. The XGB-LSTM classifier significantly outperforms existing methods with strong statistical confidence ($p < 0.05$).

To evaluate the balance between detection accuracy and processing speed, different model configurations are analyzed. The results are given in Table 9.

Table 7. Statistical significance analysis of RPDO vs other optimizers.

Optimizer	Mean Accuracy (%)	Std Dev	t-statistic	p-value	Wilcoxon p-value	Significance
RPDO	95.7	0.52	—	—	—	Reference
PSO	93.5	0.68	6.12	0.003	0.007	Significant
GA	92.8	0.74	7.45	0.002	0.005	Significant
DE	92.2	0.81	8.26	0.001	0.004	Significant
SA	89.7	0.95	10.84	0.0005	0.002	Significant
CS	91.3	0.77	8.92	0.001	0.003	Significant
ACO	90.5	0.83	9.76	0.0008	0.002	Significant
BO	94.0	0.61	4.85	0.006	0.009	Significant

Table 8. Statistical significance analysis of XGB-LSTM vs other classifiers.

Classifier	Mean Accuracy (%)	Std Dev	t-statistic	p-value	Wilcoxon p-value
XGB-LSTM	95.9	0.48	—	—	—
Hybrid XGB + LSTM	94.5	0.55	4.32	0.008	0.012
LSTM	93.3	0.63	6.75	0.002	0.006
XGBoost	94.2	0.58	5.18	0.005	0.010
Random Forest	93.0	0.66	7.02	0.002	0.006
SVM	91.2	0.79	9.64	0.001	0.003
Logistic Regression	85.4	1.02	14.21	0.0002	0.001
KNN	88.7	0.88	11.36	0.0005	0.002
Decision Tree	89.5	0.84	10.28	0.0007	0.002
Naïve Bayes	82.8	1.15	15.62	0.0001	0.001

Table 9. Accuracy vs FPS trade-off.

Model Configuration	Accuracy (%)	Inference Time (ms)	FPS	Remarks
Simple CNN	88.4	57.6	17.3	Fast but lower accuracy
EfficientNet-B0	91.2	60.8	16.4	Balanced performance
ResNet50	92.8	68.8	14.5	Moderate accuracy
ViT	93.6	67.1	14.8	Good feature learning
ResNet101	94.3	85.6	11.6	Higher accuracy, slower
Hybrid (No Optimization)	93.5	110.2	9.0	High complexity
Hybrid + RPDO	95.4	102.3	9.7	Optimized features
Proposed Full Model	96.2	98.9	10.1	Best accuracy-speed trade-off

Table 10. Latency breakdown of proposed model.

Processing Stage	Time (ms)	Percentage (%)
Feature Extraction (ViT + ResNet + EfficientNet)	62.4	63.1%
RPDO Feature Selection	18.7	18.9%
XGB-LSTM Inference	12.6	12.7%
Data Preprocessing (Resizing, Normalization)	5.2	5.3%
Total Latency	98.9 ms	100%

The results show that the Lightweight models achieve higher FPS with minimum accuracy. In contrast, the proposed hybrid model maintains the highest accuracy and sustains approximately 10 FPS. It is acceptable for moderate-speed

drone inspections. The RPDO optimization contributes to reducing feature redundancy and improving FPS without sacrificing accuracy. To better understand system performance, the total inference time is decomposed into individual processing stages. The results are given in Table 10.

The latency analysis shows that feature extraction dominates the computation. The RPDO module introduces moderate overhead and significantly increases efficiency by reducing feature dimensions. The XGB-LSTM classifier contributes minimal latency and supports high classification accuracy. The proposed model achieves an optimal balance between accuracy and processing speed. Future improvements can focus on model compression and hardware acceleration to further enhance real-time performance.

5. Conclusion

In this work, an autonomous drone-based system is proposed for real-time solar panel fault detection in solar farms. It involves three different stages: feature extraction and classification. The system achieves high accuracy in detecting and classifying faults. The integration of advanced technologies enables the system to continuously monitor solar panel performance. The use of drones allows for autonomous and cost-effective monitoring of large-scale solar farms. It reduces the need for manual inspections and minimises downtime. Future work could explore further optimization of the algorithms to reduce computational overhead in model training.

References

1. Masita K, Hasan A, Shongwe T, Abu Hilal H. Deep learning in defects detection of PV modules: A review. Solar Energy Advances 2025; 5: 100090. https://doi.org/10.1016/j.seja.2025.100090.
2. Manoj N, Sudhakar K, Samykano M, Jayaseelan V. On the technologies empowering drones for intelligent monitoring of solar photovoltaic power plants. Procedia Computer Science 2018; 133: 585–593. https://doi.org/10.1016/j.procs.2018.07.087.
3. Rahaman S A, Urmee T, Parlevliet D A. PV system defects identification using Remotely Piloted Aircraft (RPA) based infrared (IR) imaging: A review. Solar Energy 2020; 206: 579–595. https://doi.org/10.1016/j.solener.2020.06.014.
4. Abro G, Ali A, Memon S, Memon T, Khan F. Strategies and Challenges for Unmanned Aerial Vehicle-Based Continuous Inspection and Predictive Maintenance of Solar Modules. IEEE Access 2024; 12: 176615–176629. https://doi.org/10.1109/access.2024.3505754.
5. Li X, Yang Q, Chen Z, Luo X, Yan W. Visible defects detection based on UAV-based inspection in large-scale photovoltaic systems. IET Renewable Power Generation 2017; 11(10): 1234–1244. https://doi.org/10.1049/iet-rpg.2017.0001.
6. Wang J, Gao D, Zhu S, Wang S, Liu H. Fault diagnosis method of photovoltaic array based on support vector machine. Energy Sources, Part A: Recovery, Utilization, and Environmental Effects 2023; 45(2): 5380–5395. https://doi.org/10.1080/15567036.2019.1671557.
7. Kuo C F J, Chen S H, Huang C Y. Automatic detection, classification and localization of defects in large photovoltaic plants using unmanned aerial vehicles (UAV) based infrared (IR) and RGB imaging. Energy Conversion and Management 2023; 276: 116495. https://doi.org/10.1016/j.enconman.2022.116495.
8. Ma X, Zhang Y, Li Q. Obstacle avoidance path planning for UAV applied to photovoltaic stations based on improved dynamic window method. Electronics 2024; 14(10): 1963. https://doi.org/10.3390/electronics14101963.
9. Rodriguez-Vazquez J, Prieto-Centeno I, Fernandez-Cortizas M, Perez-Saura D, Molina M, Campoy P. Real-time object detection for autonomous solar farm inspection via UAVs. Sensors 2024; 24(3): 777. https://doi.org/10.3390/s24030777.
10. Sizkouhi A M M, Esmailifar S M, Aghaei M, Oliveira A K V, Rüther R. Autonomous Path Planning by Unmanned Aerial Vehicle (UAV) for Precise Monitoring of Large-Scale PV plants. Photovoltaic Specialists Conference 2019; 1398–1402. https://doi.org/10.1109/PVSC40753.2019.8980862.
11. Li X, Li W, Yang Q, Yan W, Zomaya A. Edge-Computing-Enabled Unmanned Module Defect Detection and Diagnosis System for Large-Scale Photovoltaic Plants. IEEE Internet of Things Journal 2020; 7: 9651–9663. https://doi.org/10.1109/jiot.2020.2983723.
12. Usharani M, Kavitha V P, Theivanathan G, Magesh V, Sakthivel B, Surendiran R. An Optimized Deep Learning Model Based PV Fault Classification for Reliable Power Generation. SSRG International Journal of Electrical and Electronics Engineering 2022; 9(9): 23–31. https://doi.org/10.14445/23488379/IJEEE-V9I9P103.
13. Sridharan N, Sugumaran V. Convolutional Neural Network based Automatic Detection of Visible Faults in a Photovoltaic Module. Energy

Sources, Part A: Recovery, Utilization, and Environmental Effects 2021; 47: 6270–6284. https://doi.org/10.1080/15567036.2021.1905753.
14. Kamal M, Shahbudin S, Rahman F. Photovoltaic (PV) Module Defect Image Classification Analysis Using EfficientNetV2 Architectures. IEEE 14th Control and System Graduate Research Colloquium (ICSGRC) 2023; 236–241. https://doi.org/10.1109/icsgrc57744.2023.10215491.
15. Di Tommaso A, Betti A, Fontanelli G, Michelozzi B. A multi-stage model based on YOLOv3 for defect detection in PV panels based on IR and visible imaging by unmanned aerial vehicle. Renewable Energy 2022; 193: 941–962. https://doi.org/10.1016/j.renene.2022.05.124.
16. Zhang Y, Li H, Wang X, Chen J. ESD-YOLOv8: An efficient solar cell fault detection model based on YOLOv8. Energies 2024; 17(3): 1125. https://doi.org/10.3390/en17031125.
17. Wang X, Chen R. Video Analysis and Frame Prediction Based on Improved Object Detection and ConvGRU. IEEE Access 2025; 13: 96069–96082. https://doi.org/10.1109/access.2025.3575491.
18. Ye T, Qin W, Zhao Z, Gao X, Deng X, Yu O. Real-Time Object Detection Network in UAV-Vision Based on CNN and Transformer. IEEE Transactions on Instrumentation and Measurement 2023; 72: 1–13. https://doi.org/10.1109/tim.2023.3241825.
19. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems 2012; 25: 1097–1105.
20. Bazi Y, Bashmal L, Rahhal M, Dayil R, Ajlan N. Vision Transformers for Remote Sensing Image Classification. Remote Sensing 2021; 13: 516. https://doi.org/10.3390/rs13030516.
21. Zheng H, Hu Z, Yang L, Xu A, Zheng M, Zhang C, Li K. Multifeature collaborative fusion network with deep supervision for SAR ship Classification. IEEE Transactions on Geoscience and Remote Sensing 2023; 61: 1–14.
22. Xiang J, Liu J, Chen D, Xiong Q, Deng C. CTFuseNet: A MultiScale CNNTransformer Feature Fused Network for Crop Type Segmentation on UAV Remote Sensing Imagery. Remote Sensing 2023; 15(4): 1151. https://doi.org/10.3390/rs15041151.
23. Shang Y, Zheng M, Li J, Zheng X. An effective feature selection approach based on hybrid Grey Wolf Optimizer and Genetic Algorithm for hyperspectral image. Scientific Reports 2025; 15. https://doi.org/10.1038/s41598-024-84934-8.
24. Kamel SR, Yaghoubzadeh R. Feature selection using grasshopper optimization algorithm in diagnosis of diabetes disease. Informatics in Medicine Unlocked 2021; 26: 100707. https://doi.org/10.1016/j.imu.2021.100707.
25. Nguyen B, Nguyen T, Vu Q, Hung T, Hai T, Binh H, Tran V. Dholes Hunting—A Multi-Local Search Algorithm Using Gradient Approximation and Its Application for Blockchain Consensus Problem. IEEE Access 2024; 12: 93333–93349. https://doi.org/10.1109/access.2024.3419172.
26. Givi H, Dehghani M, Hubalovsky S. Red Panda Optimization Algorithm: An Effective Bio-Inspired Metaheuristic Algorithm for Solving Engineering Optimization Problems. IEEE Access 2023; 11: 57203–57227. https://doi.org/10.1109/access.2023.3283422.
27. Barraz Z, Sebari I, Kadi K, Abdelmoula I. Towards a Holistic Approach for UAV-Based Large-Scale Photovoltaic Inspection: A Review on Deep Learning and Image Processing Techniques. Technologies 2025. https://doi.org/10.3390/technologies13030117.
28. Alrifaey M, Lim WH, Ang CK, Natarajan E, Solihin MI, Juhari MRM, Tiang SS. Hybrid Deep Learning Model for Fault Detection and Classification of Grid-Connected Photovoltaic System. IEEE Access 2022; 10: 13852–13869. https://doi.org/10.1109/ACCESS.2022.3140287.
29. Fonseca Alves RH, de Deus Júnior GA, Marra EG, Lemos RP. Automatic fault classification in photovoltaic modules using Convolutional Neural Networks. Renewable Energy 2021; 179: 502–516. https://doi.org/10.1016/j.renene.2021.07.070.
30. Mehdiyev N, Lahann J, Emrich A, Enke D, Fettke P, Loos P. Time series classification using deep learning for process planning: A case from the process industry. Procedia Computer Science 2017; 114: 242–249. https://doi.org/10.1016/j.procs.2017.09.066.
31. Tseng CJ, Tang C. An optimized XGBoost technique for accurate brain tumor detection using feature selection and image segmentation. Healthcare Analytics 2023; 4: 100217. https://doi.org/10.1016/j.health.2023.100217.
32. Sriwastawa A, Arul Jothi JA. Vision transformer and its variants for image classification in digital breast cancer histopathology: a comparative study. Multimedia Tools and Applications 2024; 83: 39731–39753. https://doi.org/10.1007/s11042-023-16954-x.
33. Srinivasan S, Rajakumar K. Hyperspectral image classification using efficientnet-B4 with search and rescue operation algorithm. International Journal of Information Technology 2023; 15: 1473–1479. https://doi.org/10.1007/s41870-023-01197-8.
34. Romeh AE, Snášel V, Mirjalili S. Dholes-inspired optimization (DIO): a nature-inspired algorithm for engineering optimization problems.

Cluster Computing 2025; 28: 853. https://doi.org/10.1007/s10586-025-05543-2.

35. Tomar V, Bansal M, Singh P. Metaheuristic Algorithms for Optimization: A Brief Review. Engineering Proceedings 2023; 59: 238. https://doi.org/10.3390/engproc2023059238.