

Fine-Grained Image Classification using Particle Swarm Optimization for Hyperparameter Optimization of Convolutional Neural Networks

Priti Prasad Vaidya

Department of Computer Engineering, K. K. Wagh Institute of Engineering Education and Research

Snehal Kamalapur

Department of Computer Engineering, K. K. Wagh Institute of Engineering Education and Research

<https://doi.org/10.5109/7388865>

出版情報 : Evergreen. 12 (3), pp.1783-1801, 2025-09. Interdisciplinary Graduate School of Engineering Sciences, Kyushu University, Japan

バージョン :

権利関係 : Creative Commons Attribution 4.0 International



Fine-Grained Image Classification using Particle Swarm Optimization for Hyperparameter Optimization of Convolutional Neural Networks

Priti Prasad Vaidya^{1,*}, Snehal Kamalapur¹

¹Department of Computer Engineering, K. K. Wagh Institute of Engineering Education and Research, Panchavati, Nashik-422003, Maharashtra, India

*Author to whom correspondence should be addressed:
E-mail: ppvaidya@kkwagh.edu.in

(Received May 26, 2025; Revised August 05, 2025; Accepted September 02, 2025)

Abstract: This study presents a novel approach to fine-grained image classification (FGIC) by employing Particle Swarm Optimization (PSO) for automatic hyperparameter tuning in Convolutional Neural Networks (CNNs). Unlike conventional models with manually fixed architectures, the proposed method optimizes critical parameters—such as filter count, kernel size, pooling size, and stride—tailored to each dataset. The model operates on reduced image resolutions (128×128), yet achieves superior accuracy with significantly fewer training epochs. Specifically, it attains 99.56% accuracy on the Oxford Flowers dataset, 97.45% on Stanford Cars, and 95.95% on Stanford Dogs using only 25 epochs, outperforming deep networks like DenseNet-161 and ResNet-50 trained for 100–150 epochs. This PSO-based tuning framework not only enhances classification performance but also minimizes computational cost, making it a practical and scalable solution for real-world FGIC applications.

Keywords: CNN; Fine grain image classification; Hyperparameters; PSO

1. Introduction

In today's digital landscape, the vast amount of multimedia content uploaded to the internet every day highlights the necessity for automated image classification methods. Among these methods, object classification is critical for interpreting and processing images, leading to various applications such as automatic tagging, captioning, and analysing user interests. Over the past decade, this area has attracted considerable attention in the fields of computer vision and machine learning, with a focus on identifying objects within images. For instance, an image depicting a Persian cat can be correctly labeled as "cat" by an object classification system. However, when it comes to distinguishing between different bird species that may appear visually similar, this task becomes significantly more complex and demands specialized knowledge and advanced techniques. In this context, fine-grained image classification has emerged as a specialized branch of object classification that aims to identify specific sub-categories within broader categories. Unlike general object classification, which categorizes images into broad groups (such as identifying whether an image shows a bird, dog, or fruit), fine-grained classification seeks to differentiate between specific species or variations within similar objects. A common example of this is the classification of

bird species, where minor visual distinctions can determine the classification outcome.

The differentiation between fine-grained and general image classification is essential. General image classification typically deals with objects that belong to broad categories with clear visual differences. In contrast, fine-grained classification as shown in Figure 1, involves distinguishing between objects within the same category that are often very similar in appearance, creating challenges in classification tasks. Several challenges make fine-grained image classification particularly difficult. First, there are significant intra-class variations, including differences in pose, lighting, and environmental conditions that can complicate the recognition of the same species. For example, birds inhabit diverse environments, leading to variations in their appearance based on lighting and habitat. Second, inter-class variations pose additional challenges, as some species may have very similar shapes, colors, and textures, making it hard to differentiate between them. Finally, the limited availability of annotated images for each species complicates the training of classification models, often requiring expert knowledge for accurate labeling.

Fine-grained visual categorization (FGVC) has gained significant attention in computer vision due to its



Fig. 1: Fine Grain Classification Problem

challenges in distinguishing between visually similar subcategories. Various methods leveraging deep learning architectures and attention mechanisms have been proposed to enhance classification accuracy.

Yu et al.¹⁾ introduced MaskCOV, a random mask covariance network that efficiently captures subtle inter-class differences using covariance matrices, showing improved performance in ultra-fine-grained tasks. Similarly, Zhang et al.²⁾ presented a web-supervised approach employing softly updated drop-training to mitigate label noise, thereby improving model generalization for fine-grained image datasets. Zheng et al.³⁾ tackled part localization using only a single image, eliminating the need for extra annotations while still boosting classification precision. Interpretability and attention are recurring themes in FGVC research. Kim and Ko developed a neural tree decoder that interprets vision transformer outputs, supporting explainable AI efforts. He et al.⁵⁾ proposed TransFG, a transformer-based model that leverages attention maps to focus on key object parts, outperforming CNN-based methods. To address robustness, Morales et al.⁵⁾ combined CNNs with visual explanation techniques, demonstrating resilience to distracting image regions during training. For domain-specific tasks, Thapa et al.⁶⁾ released a benchmark dataset targeting foliar disease classification in apples, contributing valuable data for real-world FGVC applications. Advancements in feature fusion and inference strategies also play a pivotal role. Wang et al.⁷⁾ proposed a cross-layer attention bilinear fusion method, improving feature discrimination. Nie et al.⁸⁾ introduced a dual-inference strategy that enhances recognition by learning richer features. Cui et al.⁹⁾ addressed scalability through large-scale categorization and domain-specific transfer learning, paving the way for generalized FGVC systems. Vision transformers continue to evolve, as Zhang et al.¹⁰⁾ demonstrated by reducing noise and enhancing discriminative feature representation. Ye et al.¹¹⁾ revisited the impact of backbone networks and training data in weakly supervised settings, highlighting areas for further optimization. Some studies emphasize structural and contextual cues. Lu et al.¹²⁾ extracted local structure

information to improve classification fidelity. Transfer learning has also been effectively applied, as shown by Wang¹³⁾ who focused on flower classification tasks using pre-trained models. YOLO architectures have been compared by Yuan¹⁴⁾, emphasizing the influence of annotation quality on classification outcomes. Xu et al.¹⁵⁾ proposed a lightweight DCNN-based system tailored for mobile and resource-constrained environments. Prasad et al.¹⁶⁾ further validated CNNs for flower image classification in diverse conditions. Emerging applications combine FGVC with novel data paradigms. Zhao et al.¹⁷⁾ integrated blockchain data lakes with color constancy techniques, enhancing classification consistency. Lastly, Zhang et al.²⁾ introduced an attention mechanism coupled with multi-loss strategies to refine model focus and accuracy. In domain-specific FGVC applications, traditional methods have often been enhanced with hybrid or lightweight architectures. Peryanto et al.¹⁸⁾ combined Convolutional Neural Networks (CNNs) with Support Vector Machines (SVMs) for flower classification, achieving effective results by leveraging CNNs for feature extraction and SVMs for classification. Zeng et al.¹⁹⁾ extended this with a lightweight neural network enhanced by multi-scale feature fusion and an attention mechanism, enabling efficient processing of complex floral images with fewer computational resources. Beyond natural images, Legesse et al.²⁰⁾ explored FGVC in medical imaging using texture analysis on CARS microscopy for automated skin cancer detection, illustrating the versatility of FGVC approaches in biomedical domains. Sagingalieva et al.²¹⁾ proposed a hybrid quantum ResNet, applying quantum layers to improve car image classification and employing hyperparameter optimization, pushing the frontier of quantum-assisted FGVC. Himilda and Johan²²⁾ utilized Extreme Learning Machines (ELM) for vehicle classification, highlighting the speed and simplicity of ELMs in real-time scenarios. Similarly, Putra et al.²³⁾ relied on Histogram of Oriented Gradients (HOG) and K-Nearest Neighbors (KNN) to detect highway vehicles, showcasing traditional methods' continued relevance in FGVC under constrained environments. Hsu et al.²⁴⁾ ventured into education-focused FGVC tools by analyzing behavioral patterns and creating AI instructional tools for children, emphasizing the importance of FGVC in educational technology. Pei et al.²⁵⁾ took a novel psychological approach by applying Gestalt theory to the fine-grained classification of automobile fronts, suggesting that human visual perception models can inspire better computer vision algorithms. A broader survey by Abdar et al.²⁶⁾ provided a comprehensive review of uncertainty quantification in deep learning, underscoring the relevance of interpretability and reliability in FGVC tasks. Shah²⁷⁾ conducted a focused study on car image recognition using CNNs, contributing insights into real-world FGVC system deployment.

Advancements in generative and self-supervised learning have also influenced FGVC. Tan et al.²⁸⁾ introduced a self-supervised text-to-image synthesis method, which indirectly aids FGVC by improving training data quality. Hendricks et al.²⁹⁾ tackled model transparency by generating visual explanations for CNN outputs, improving user trust and insight into fine-grained decisions. Lin et al.³⁰⁾ introduced Local Patch AutoAugment, leveraging multi-agent collaboration for more adaptive augmentation strategies, which improves performance in low-data FGVC settings. Ji et al.³¹⁾ developed a Siamese self-supervised learning framework, effectively learning discriminative representations without labels. In the realm of zero-shot and few-shot learning, Yu et al.³²⁾ proposed a Knowledge Distillation Classifier Generation Network, allowing FGVC without direct training samples. Xiong et al.³³⁾ contributed a hierarchically structured network combining attention and embedding propagation for imbalanced few-shot learning. Kang et al.³⁴⁾ emphasized relational embedding, 3D modelling inter-class relationships to generalize across limited samples. Liu et al.³⁵⁾ presented TransIFC, which concentrates on invariant cues in bird classification, enhancing model robustness in fine-grained bird species identification. Zhao et al.³⁶⁾ proposed CA-PMG, integrating channel attention with progressive multi-granularity training, achieving superior performance on standard FGVC benchmarks. Further, Li et al.³⁷⁾ introduced a Two-Branch Attention Network with efficient semantic coupling for one-shot learning, reducing the dependency on large 3D modelling datasets. Li and Bian³⁸⁾ applied foreground-aware feature map reconstruction for fine-grained ship classification, addressing challenges specific to maritime imagery. Interpretability and robustness were also explored by Nguyen et al.³⁹⁾, who demonstrated that visual correspondence-based explanations improve human-AI collaboration in FGVC. Hu et al.⁴⁰⁾ proposed a Hierarchical Attention Vision Transformer, showing improvements over conventional

ViTs by structuring attention at multiple levels. Finally, Sachin et al.⁴¹⁾ argued that classification tasks themselves form strong baselines for metric learning, suggesting that existing classification frameworks can often outperform more complex alternatives in fine-grained domains. Bold et al.⁴²⁾ proposed a cross-domain deep feature fusion framework that integrates audio and visual signals for bird species classification, highlighting the benefits of multimodal information in fine-grained scenarios. Similarly, Di et al.⁴³⁾ introduced a dataset for fine-grained ship classification using remote sensing imagery, emphasizing the importance of diverse data sources for domain-specific FGVC. Sun et al.⁴⁴⁾ introduced an attention weights fusion transformer (AWFT) using loop and distillation mechanisms to enhance fine-grained representations. They⁴⁵⁾ further extended this in by associating multiple vision transformer layers to strengthen semantic representation. Guang and Liang⁴⁶⁾ proposed CMSEA, which scales compound models and incorporates efficient attention, achieving competitive performance on FGVC tasks. Xu et al.⁴⁷⁾ proposed the Attribute Prototype Network for any-shot learning, generalizing across zero- and few-shot scenarios. Tseng et al.⁴⁸⁾ explored feature-wise transformations for cross-domain few-shot classification, while Zhang et al.⁴⁹⁾ used Earth Mover's Distance for better sample comparisons in few-shot image classification. Cheng et al.⁵⁰⁾ focused on disentangled representations to improve sample efficiency in few-shot learning. Zhao et al.⁵¹⁾ introduced diversified visual attention networks to capture subtle distinctions among fine-grained categories. Wang et al.⁵²⁾ proposed hierarchical bilinear pooling with an aggregated slack mask to improve robustness. Lee et al.⁵³⁾ proposed anti-adversarial attribution techniques for improved semantic segmentation and localization, enhancing model reliability under adversarial conditions. Table 1. Shows the Cluster wise Literature Review of related research with research gap.

Table 1: Cluster wise Literature Review of related research

Category	Paper(s)	Summary	Research Gap
Architectures (CNN/Transformer)	1), 5), 10), 15), 16), 60), 40), 46), 56)	Propose or refine CNNs, Vision Transformers, or hybrid models to enhance FGVC performance.	Need for balancing model complexity with computational efficiency, especially for deployment on edge devices.
Attention & Interpretability	4), 5), 5), 29), 30), 40), 41), 51), 52-64)	Focus on model transparency, visual explanations, or attention mechanisms to highlight discriminative regions.	Limited human trust in automated decisions; interpretability methods still lack standard evaluation metrics.
Optimization & Training Strategy	2), 8), 13), 31), 44), 55), 56)	Improve training through optimization strategies like dual-inference, transfer learning, augmentation, or class-incremental learning.	Difficulty in generalizing across domains or adapting to real-time conditions; limited exploration of continual learning.

Domain-Specific Applications	6), 18), 19), 20), 21-26), 43)	Apply FGVC in agriculture, medical imaging, automotive, education, remote sensing, and psychology.	Many domain-specific models lack generalizability and standardization; often limited by dataset scale and diversity.
Data & Annotation Challenges	2), 14), 28), 44), 45)	Address label noise, annotation quality, data augmentation, and low-data regimes.	Existing solutions are often domain-dependent; limited work on adaptive methods for real-time noisy data.
Few-shot / Zero-shot Learning	32-38), 47-53), 60)	Tackle sample efficiency through few-shot/zero-shot learning frameworks, including attention, graph models, and prototypes.	Struggles with class imbalance and unseen class generalization; need more benchmark standardization.
Multimodal / Hybrid Methods	18-20), 42), 53), 58-60)	Fuse multiple input types (e.g., visual + audio), or combine models (CNN + SVM, GNN, or text attributes).	Integration challenges with asynchronous or unaligned modalities; computational cost is often high.
Backbone & Feature Enhancement	7-12), 26-28), 35-38), 61)	Explore the impact of backbone networks, feature fusion, and invariance for better representation and generalization.	Need for more lightweight and adaptable backbones; overfitting on small FGVC datasets remains a concern.
Augmentation / Self-Supervision	28-31), 44), 54)	Use self-supervised, generative, or adversarial approaches for data augmentation and learning without full supervision.	Augmented data may not always reflect true class variance; need for adaptive augmentation guided by task relevance.
Metric Learning / Embeddings	41), 49), 53-59)	Enhance class separation and recognition via embedding learning and distance-based approaches.	Embedding quality deteriorates in highly imbalanced datasets; lacks robustness to subtle inter-class similarities.
Interpretability in Real-World Scenarios	5), 22), 24), 39), 57)	Demonstrate model interpretability in real-time, educational, or collaborative settings.	Scalability and usability of interpretability tools in practical deployments need further development.
Benchmark / Dataset Contributions	6), 53)	Release new datasets or benchmarks for domain-specific FGVC (e.g., apples, ships).	Most benchmarks are narrowly focused and lack diversity; there is a need for multi-domain, multi-modal datasets.

The novelty of our approach lies in the integration of PSO for automatic hyperparameter tuning in CNNs, specifically tailored for FGIC. Unlike traditional methods that rely on fixed architectures or manual tuning, our PSO-based framework dynamically searches for the optimal combination of key hyperparameters—such as the number of filters, kernel size, pooling size, and stride—based on validation accuracy. This automated tuning enables the model to adapt to the unique characteristics of each fine-grained dataset, capturing subtle inter-class differences more effectively. Furthermore, our method operates on reduced image resolutions (128×128), yet achieves higher classification accuracy with significantly fewer training epochs (only 25), compared to widely used models like DenseNet-161 and ResNet-50, which typically require larger image sizes and over 100 epochs. By balancing model complexity, computational efficiency, and accuracy through PSO, our approach offers a scalable and resource-efficient alternative to deep or heavily engineered networks, making it especially suitable for deployment in real-world, constrained environments. This adaptability and performance optimization highlight the innovative contribution of our work in the FGIC domain.

2. Methods

As shown in Figure 2, Pre-processing a picture dataset is the initial step in the process of fine-grained classification of pictures using CNN and hyper parameter optimization using PSO. To improve the model's potential for generalization, this phase entails standardizing the images, normalizing each pixel's values, and executing data enhancement. The number of layers, the size of the convolutional filters, the stride of the filters, and the variety of neurons in every layer that is fully connected are among the movable parameters that are then used to create the CNN model. The efficiency of the model (accuracy or validation loss) is represented by the function $f(n, sf, sp, l)$, and these parameters are represented as n , sf , sp , and l , respectively. To optimize these the hyper parameters PSO is used.

A collection of particles, each reflecting a distinct combination of hyper parameters, is initialized at the start of the PSO algorithm. A CNN model is generated for every particle using the present parameters, and the training dataset is used to train the model. A validation set is used to assess the model's correctness or validation loss, and if

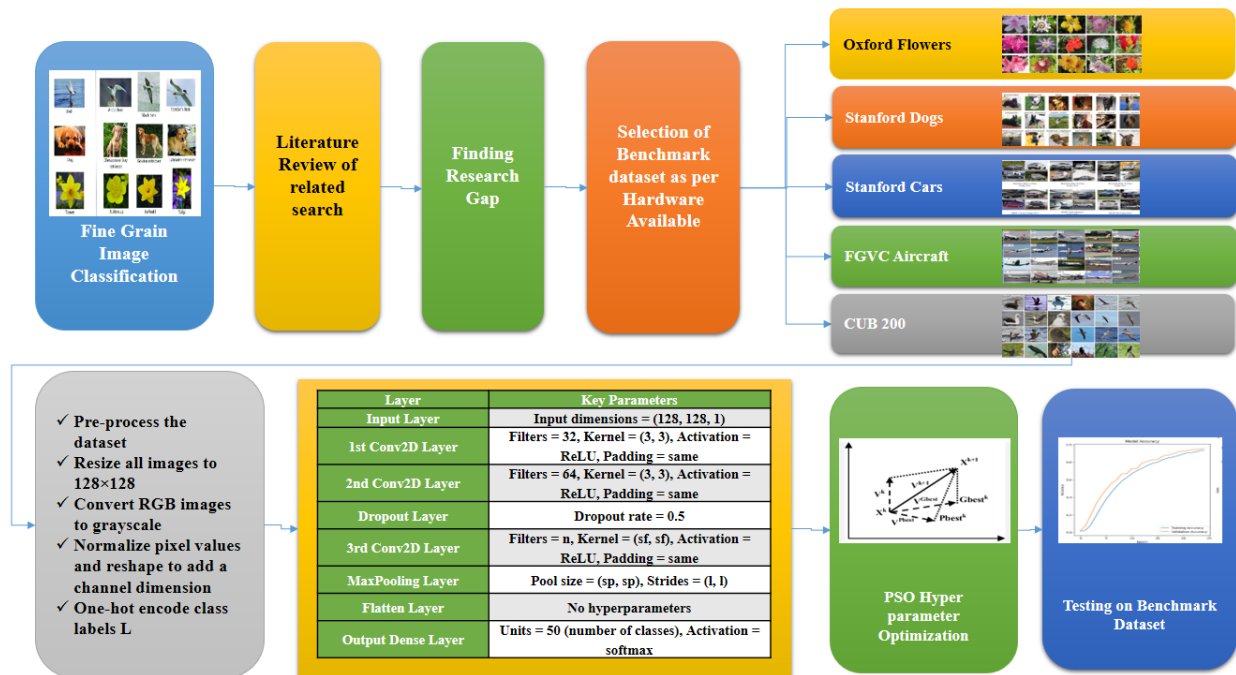


Fig. 2: Methodology

the particle's current performance outperforms its prior best, its personal best position is altered. Furthermore, the most appropriate accumulation of hyper parameters for all particles is represented by the global best position, which is updated appropriately. The velocities of the particles then change according to the overall best positions and their own best positions.

Until completion or a certain number of iterations, the PSO algorithm refines the hyper parameters through several generations of iterations. The CNN model is retrained using the optimal settings on the complete training dataset after the global best set of hyperparameters has been determined. Lastly, the accuracy and other relevant metrics will be shown once the effectiveness of the model has been evaluated on the test set. Using the capability of PSO to effectively search for the ideal hyperparameters that that approach guarantees that the CNN model must be modified to attain the best possible accuracy for the fine-grained image classification problem. The CUB-200-2011 dataset, a well-known standard for classifying bird species, is used in the experiment. Given the complicated nature and variety of bird species, this dataset is specifically made to test both machines and human visual recognition abilities.

2.1. CUB Birds-200-2011 Dataset

Bird species classification as shown in Figure 3, is a significant challenge in computer vision, as various species can exhibit substantial differences in shape and appearance, despite sharing common anatomical features. The CUB-200-2011 dataset is an enhanced version of the original CUB-200 dataset, expanding the number of images per bird species and including additional

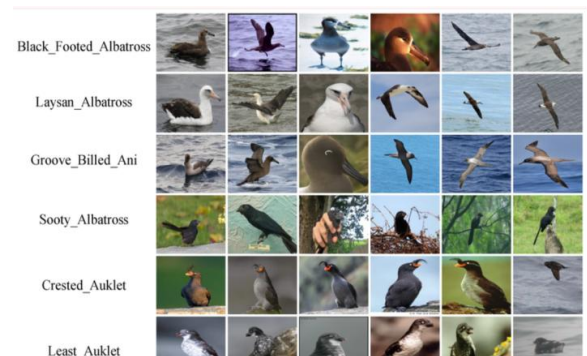


Fig. 3: CUB 200 Fine Grain Dataset

annotations for part localization. Specifically, it contains 11,788 images across 200 distinct bird species. Each species is meticulously organized according to scientific classification, including order, family, genus, and species. The species names were compiled using a reputable online field guide, and the images were sourced through Flickr image searches. To ensure accuracy and consistency, the images underwent a filtering process where multiple Mechanical Turk users verified each image.

2.2. The Stanford Dogs Dataset

The Stanford Dogs Dataset as shown in Figure 4, is a benchmark for fine-grained image classification, where the objective is to distinguish between visually similar sub-categories within a broader category—in this case, different breeds of dogs. This task is notably more challenging than general object classification, as it demands precise modelling of subtle differences in texture, shape, and color across breeds that share similar overall



Fig. 4: Stanford Dogs Fine Grain Dataset

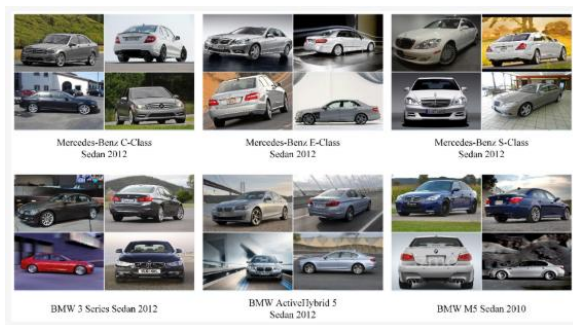


Fig. 5: Stanford Cars Fine Grain Dataset

anatomy and structure. The dataset is based on images sourced from ImageNet and specifically curated for the task of dog breed identification. It comprises 20,580 high-resolution images distributed across 120 dog breeds, reflecting the diversity recognized by the American Kennel Club.

Dog Breeds: The dataset contains 120 breeds, ranging from commonly recognized species such as the Labrador Retriever and German Shepherd to more obscure breeds like the Saluki and Komondor.

The Stanford Dogs Dataset presents a challenging yet rich resource for evaluating the capability of fine-grained image classification systems, especially those leveraging deep learning techniques.

2.3. The Stanford Cars Dataset

The Stanford Cars Dataset as shown in Figure 5, is another example of a fine-grained classification benchmark, designed to evaluate the performance of models in distinguishing between visually similar car models and makes. While all cars generally follow the same structural layout—hood, cabin, wheels, and lights—the differences between makes and models can be subtle, often reduced to minute variations in contours, headlights, grilles, and brand logos. This dataset consists of 16,185 images across 196 categories of cars, covering a range of models manufactured between 1960 and 2012. Each category is defined by the make, model, and year, providing a highly granular classification structure.

The Stanford Cars Dataset is particularly useful for assessing a model's performance in domains where fine-grained visual distinctions are critical. Its real-world



Fig. 6: Oxford Flowers Fine Grain Dataset



Fig. 7: FGVC Aircraft Fine Grain Dataset

variability and detailed class annotations make it an ideal benchmark for deep convolutional neural networks and optimization-based architecture exploration.

2.4. The Oxford 102 Flowers Dataset

The Oxford 102 Flowers Dataset as shown in Figure 6, is a widely recognized benchmark in the domain of fine-grained image classification, specifically focused on the classification of flower species. This task poses unique challenges for computer vision models due to the intricate variability in petal shapes, colors, sizes, and arrangements, all of which are critical to accurate species identification. Furthermore, flowers of different species may share similar visual characteristics, while intra-species variation can also be significant due to environmental factors such as lighting, angle of capture, and seasonal appearance changes.

The dataset contains images of 102 flower categories commonly found in the United Kingdom, providing a rich and diverse collection of visual data suitable for training and evaluating deep learning models for fine-grained classification.

2.5. FGVC Aircraft Dataset

The FGVC Aircraft Dataset, shown in Figure 7, is a benchmark for fine-grained visual categorization, focusing on distinguishing between 100 visually similar aircraft variants. It contains 10,000 high-quality images sourced from aircraft spotters, with around 100 images per variant. Each image is annotated with bounding boxes and hierarchical labels, including variant, family, and manufacturer. The dataset captures a wide range of

conditions—angles, lighting, and occlusions—making classification more challenging. It serves as a valuable resource for evaluating deep learning models in tasks requiring detailed recognition of subtle visual differences. Table 2 represents the benchmark datasets in detail.

2.6. CNN architecture

The CNN architecture is designed as shown in Table 3, for classifying grayscale images into one of 50 classes. The network begins with an input layer that accepts images of size (128, 128, 1), indicating the height, width, and single channel for grayscale. It then consists of three convolutional layers. The first layer applies 32 filters of size 3x3 with ReLU activation and 'same' padding, preserving the input dimensions and detecting basic

features. The second layer builds upon this by using 64 filters of the same size and activation function, capturing more complex patterns. The third convolutional layer further refines feature extraction with an optimized number of filters (n) and kernel size (sf). To prevent overfitting, a Dropout layer randomly drops 50% of the units during training, enhancing model generalization. Following this, a MaxPooling layer reduces the spatial dimensions by selecting the maximum values from pooling regions, controlled by parameters for pool size (sp) and strides (l). The feature maps are then reshaped into a one-dimensional vector using a Flatten layer, preparing the data for the output. The final Dense layer maps the flattened input to the 50 output classes using a softmax activation.

Table 2: Benchmark Dataset

Dataset	Classes	Images	Image Resolution	Domain	Task
Stanford Cars	196	16,185	~300×300	Fine-grained Vehicles	Car model and make classification
Stanford Dogs	120	20,580	~224×224	Fine-grained Animals	Dog breed classification
Oxford Flowers 102	102	8,189	~500×500	Fine-grained Plants	Flower species classification
CUB-200-2011 (Birds)	200	11,788	~227×227	Fine-grained Birds	Bird species classification
FGVC Aircraft	100	10,000	~1024×768	Fine-grained Vehicles	Aircraft variant classification

Table 3: CNN Architecture used

Layer	Functionality	Key Parameters
Input Layer	Accepts input grayscale images with dimensions 128x128 pixels and a single channel.	Input dimensions = (128, 128, 1)
1st Conv2D Layer	Initial convolution layer that processes the input using 32 filters of size 3x3. ReLU is used as the activation function with same padding applied.	Filters = 32, Kernel = (3, 3), Activation = ReLU, Padding = same
2nd Conv2D Layer	Performs further feature extraction using 64 convolutional filters, each of size 3x3. ReLU activation and same padding are used again.	Filters = 64, Kernel = (3, 3), Activation = ReLU, Padding = same
Dropout Layer	Randomly disables half of the neurons during training to enhance model generalization and reduce overfitting.	Dropout rate = 0.5
3rd Conv2D Layer	Applies a tunable number of filters with a square kernel of size (sf x sf), using ReLU activation and maintaining output dimensions with same padding.	Filters = n, Kernel = (sf, sf), Activation = ReLU, Padding = same
MaxPooling Layer	Downsamples the spatial dimensions of feature maps using a specified pool size and stride value.	Pool size = (sp, sp), Strides = (1, 1)
Flatten Layer	Converts the 2D matrix of features into a 1D vector, making it suitable for input into the fully connected layer.	No hyperparameters
Output Dense Layer	Final classification layer with a neuron for each class, using softmax activation to generate probability scores.	Units = 50 (number of classes), Activation = softmax

function, providing probabilities for each class. The parameters n , sf , sp , and l are optimized using methods like PSO, which enhance the model's performance. During the training process, the network employs a categorical cross-entropy loss function to measure classification accuracy, with the Adam optimizer adjusting the model's weights iteratively for improved accuracy. Overall, this CNN architecture effectively learns to classify grayscale images by leveraging its layered structure to extract and refine features while using dropout for regularization and optimization techniques to fine-tune parameters.

2.7. Algorithm for PSO-Based Optimization of CNN Parameters

To enhance the performance of the CNN for grayscale image classification, PSO was employed to optimize key CNN hyperparameters. The process followed a structured algorithm described as follows-

Initially, the Stanford Cars dataset was preprocessed by resizing all images to a uniform dimension and converting them from RGB to grayscale using OpenCV. The grayscale images were then normalized and reshaped to include a single channel. Corresponding class labels were encoded into categorical format to support multi-class classification. A group of particles started up after preprocessing. An assortment of CNN hyperparameters, namely the number of filters, kernel size, pooling size, and stride, was used to represent each particle as a possible solution. Within defined limitations, the particle's beginning locations and velocities were allocated at random. Each particle's fitness was determined first training a CNN model with its parameters, subsequently determining the validation accuracy. Iterative steps were taken in the optimization process. During each iteration, the velocity and position of each particle were altered using the accepted PSO update equations, which consider the particle's personal best position and the global best position discovered so far.

The revised positions were confined within the permitted limits to guarantee valid configurations for the CNN. A CNN architecture was created dynamically for each particle according to its current position. This architecture was trained utilizing the training subset of the prepared dataset. To avoid overfitting and enhance training efficiency, early stopping was employed. The validation accuracy achieved post-training served as the fitness value for that particle. If a particle obtained a validation accuracy superior to its previous best, its personal best was updated accordingly. Likewise, if any particle exceeded the global best performance, the global best position was modified as necessary. This iterative procedure persisted until the maximum iteration count was reached. Ultimately, the particle that corresponded to the global best position was chosen as the optimal solution. The CNN model configured with this optimal parameter set achieved the

best classification accuracy on the validation set. The PSO-based optimization effectively explored the hyperparameter space as shown in Figure 8 and identified configurations that significantly improved the classification of the CNN on grayscale images from the Stanford Cars dataset.

2.8. Mathematical Model of PSO-Based Hyperparameter Optimization

2.8.1. Problem Definition

Let the task be a fine-grained image classification problem, with a δ labelled dataset in eq.1

$$D = \{(x_i, y_i)\}_{i=1}^N \quad (1)$$

where x_i is an image and y_i is its class label from among C possible classes.

A convolutional neural network $f(x; \theta)$ where θ are the trainable weights, and η is a set of hyperparameters that influence the architecture and training process in eq.2:

$$\eta = [n, sf, sp, l] \in R^d \quad (2)$$

where:

- n : number of filters,
- sf : kernel size,
- sp : pooling size,
- l : stride length

Our objective is to optimize η to minimize the classification loss function (e.g., cross-entropy) on the validation set in eq.3:

$$\min_{\eta} L_{val}(\theta^*(\eta); \eta)$$

subject to:

$$\theta^*(\eta) = \arg \min_{\theta} L_{train}(\theta; \eta) \quad (3)$$

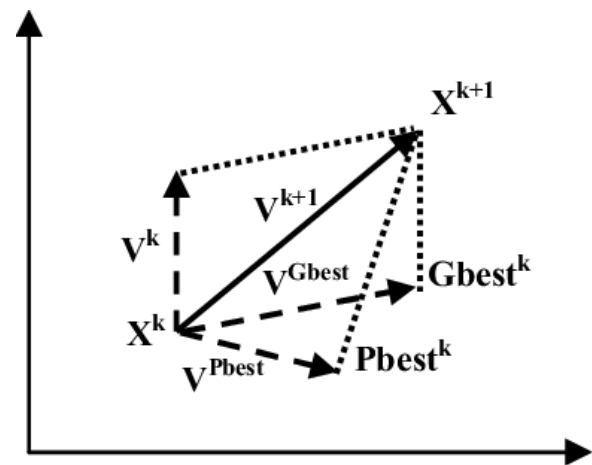


Fig. 8: PSO Mechanism

2.8.2. Algorithm

```

Input:
Grayscale image dataset X
Label vector L
Number of training iterations n
Number of particles in the swarm p
Learning rate  $\xi$ 
Momentum constants c1, c2
Batch size B
CNN hyperparameter bounds (e.g., filter count, kernel size, pooling size, stride)
Output:
Optimal CNN model
Best validation accuracy
1: Pre-process the dataset:
    - Resize all images to 128×128
    - Convert RGB images to grayscale
    - Normalize pixel values and reshape to add a channel dimension
    - One-hot encode class labels L
2: Initialize PSO parameters:
    - Randomly initialize particle positions (CNN hyperparameters) and velocities within bounds
    - Define fitness function based on CNN validation accuracy
3: For each particle i=1 to p:
    - Construct a CNN model based on position parameters
    - Train the CNN for a small number of epochs using training data
    - Evaluate accuracy on validation data and compute fitness
4: Update personal best (pBest) and global best (gBest) positions based on fitness
5: For iteration t=1 to n:
    6: For each particle i=1 to p:
        - Update velocity:
             $v_i = \omega \cdot v_i + c1 \cdot \text{rand}() \cdot (pBest_i - x_i) + c2 \cdot \text{rand}() \cdot (gBest - x_i)$ 
        - Update position:
             $x_i = x_i + v_i$ 
            - Clamp  $x_i$  within allowed hyperparameter bounds
            - Construct CNN model using new  $x_i$ 
            - Train the model on training data with early stopping
            - Evaluate and update pBest and gBest if needed
7: After n iterations, return CNN model configured with gBest
hyperparameters
8: Output the best validation accuracy obtained using the optimized CNN

```

This makes it a bi-level optimization problem, where PSO is used for the outer optimization (hyperparameters), and standard CNN training is used for the inner optimization (model weights).

2.8.3. PSO Algorithm for Hyperparameter Search

Let the search space be bounded: $\eta \in \Omega \subset \mathbb{R}^d$ PSO defines a swarm of particles, each representing a candidate hyperparameter configuration.

Let:

- N_p : Number of particles in the swarm.

- For each particle i , its position $x_i \in \Omega$ represents a hyperparameter vector.
- Its velocity is $v_i \in \mathbb{R}^d$
- p_i : Best position ever visited by particle i .
- g : Global best position found by any particle.

The update rules at iteration t are in eq.4:

$$v_i^{t+1} = \omega \cdot v_i^t + c_1 r_1 (p_i - x_i^t) + c_2 r_2 (g - x_i^t) \quad (4)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

Where:

- ω : Inertia weight controlling exploration vs.

exploitation.

- $c1, c2$: Cognitive and social learning rates.
- $r1, r2 \sim U(0,1)$: Random numbers for stochastic exploration.

2.8.4. Fitness Evaluation

Each particle's fitness is evaluated by:

Constructing a CNN model using the hyperparameters ξ
Training the model on the training set for a fixed number of epochs.

Computing the validation loss or accuracy in eq.(5):

$$f(\xi) = L_{val}(\theta^*(\xi); \xi) \text{ or } 1 - \text{Accuracy}_{val} \quad (5)$$

This cost function is non-convex, high-dimensional, and expensive to evaluate, which makes PSO a strong candidate due to its gradient-free nature and good performance on complex objective landscapes.

2.9. Experimental Setup and Implementation

To evaluate the effectiveness of the proposed PSO-based hyperparameter optimization for Convolutional Neural Networks (CNNs), a comprehensive experimental setup was established. The implementation was carried out using Python on Google Colab, which provides access to a high-performance computing environment with an Intel Core i7 processor and 16 GB of RAM, along with GPU acceleration.

2.9.1. Hardware and Software Environment

Hardware: Google Colab environment with Intel Core i7 CPU, 16 GB RAM, NVIDIA Tesla T4 GPU

Software:

- Python 3.8
- TensorFlow 2.x
- Keras API
- NumPy, OpenCV, Matplotlib
- Scikit-learn for preprocessing and evaluation

2.9.2. Dataset Preparation

The all dataset was used, consisting of high-resolution images of various car models. All images were preprocessed as follows:

- Resized to 128×128 pixels
- Converted from RGB to grayscale using OpenCV
- Normalized pixel values to the range [0, 1]
- Reshaped to include a single channel dimension
- Class labels were one-hot encoded for multi-class classification

2.9.3. CNN Hyperparameter Space

PSO algorithm was applied to search for the optimal combination of CNN hyperparameters within the following defined ranges as shown in Table 4.

Table 4: Hyperparameters Range

Hyperparameter	Search Range
Number of Filters	{16, 32, 64, 128}
Kernel Size	(1,1) to (8,8)
Pooling Size	{2, 4}
Learning Rate	{1e-4, 1e-3, 1e-2}
Batch Size	{16, 32, 64, 128}
Dropout Rate	[0.2, 0.5] (continuous range)
Number of Epochs	[10, 25] (integer range)
Activation Function	{ReLU (hidden layers), Softmax (output)}

2.9.4. Training and Validation Strategy

- The dataset was split into training (80%), validation (10%), and test (10%) sets.
- For each particle in the swarm, a CNN model was dynamically built based on its hyperparameter configuration.
- Models were trained using the Adam optimizer with early stopping to prevent overfitting.
- The categorical cross-entropy loss function was used for multi-class classification.
- The validation accuracy served as the fitness function for PSO evaluation.

The experimental setup ensured efficient and consistent model training and evaluation. The PSO algorithm iteratively improved hyperparameter configurations by maximizing validation accuracy while exploring the complex search space.

3. Results and Discussion

The results presented in Table 5, demonstrate the training and validation accuracy of five fine-grained image classification tasks (Oxford Flowers, Stanford Dogs, Stanford Cars, FGVC Aircraft, and CUB-200 Birds) over 25 epochs. Each dataset exhibits a different trend in both training and validation accuracy, showing that the performance varies based on the dataset characteristics and the model's ability to learn fine-grained features. The hyperparameters, such as the number of filters (n), kernel size (sf), pool size (sp), and layers (l), are crucial in achieving these results. In this case, the optimized parameters obtained via PSO are as follows:

- Number of Filters (n): 117
- Kernel Size (sf): 8
- Pool Size (sp): 2
- Number of Layers (l): 4

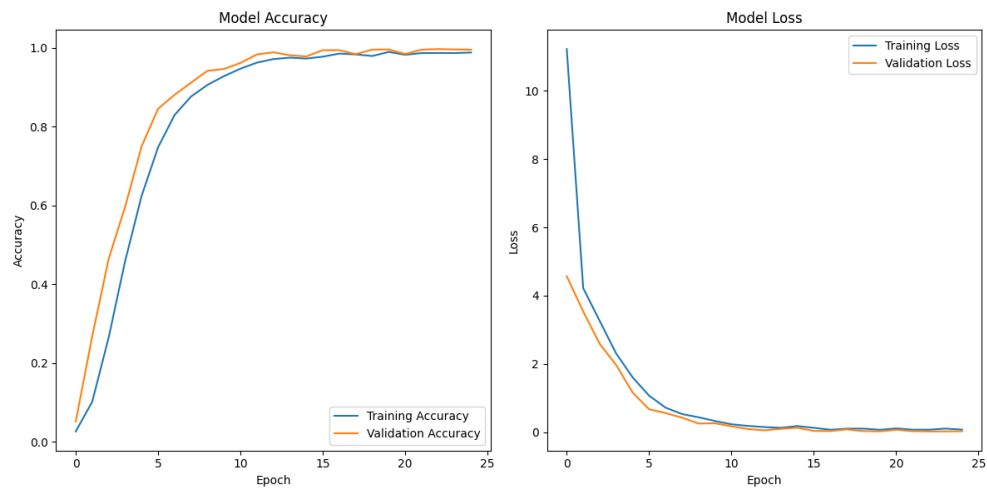
Figure 9 illustrates the epoch-wise training and validation accuracy trends for various benchmark datasets used in fine-grained visual categorization. The results presented reflect the ability of the model to learn these fine-grained features and generalize well to unseen data, which is especially important in FGIC tasks. Table 6 summarizes

Table 5: Result table of benchmark datasets

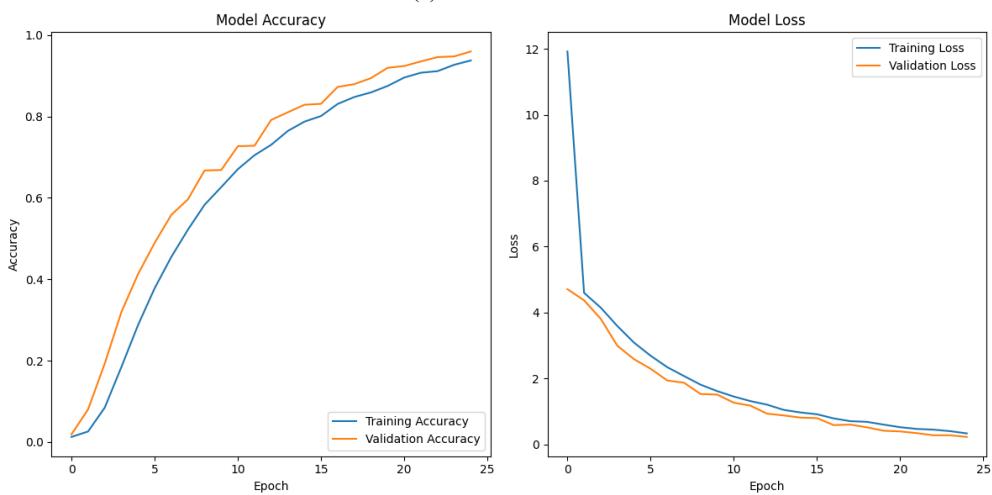
Epoch	Oxford Flowers		Stanford Dogs		Stanford Cars		FGVC Aircraft		Cub 200 Birds	
	Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy	Training Accuracy	Validation Accuracy
1	0.0167	0.0488	0.0119	0.0188	0.049	0.0506	0.0081	0.0113	0.0034	0.0071
2	0.0874	0.2452	0.0263	0.0803	0.0524	0.0712	0.01	0.0143	0.0047	0.0136
3	0.2481	0.4114	0.0837	0.1927	0.0818	0.1039	0.017	0.0353	0.016	0.0385
4	0.4139	0.58	0.1863	0.3192	0.1363	0.277	0.042	0.0658	0.0407	0.0671
5	0.5734	0.6898	0.3014	0.4125	0.2765	0.4061	0.084	0.1183	0.0645	0.1075
6	0.6756	0.7588	0.3891	0.4893	0.4005	0.5286	0.1282	0.1858	0.111	0.1475
7	0.7644	0.8239	0.4566	0.558	0.5102	0.6298	0.1981	0.2781	0.1431	0.1929
8	0.8228	0.8706	0.5315	0.5963	0.5948	0.6871	0.277	0.36	0.1841	0.2302
9	0.8577	0.8993	0.5991	0.6672	0.6986	0.7264	0.3577	0.4418	0.2198	0.2703
10	0.8685	0.916	0.6452	0.6683	0.7048	0.787	0.4325	0.5157	0.2679	0.3178
11	0.8967	0.9101	0.6779	0.7267	0.793	0.7916	0.5057	0.6	0.3109	0.3443
12	0.9053	0.9284	0.7115	0.728	0.7947	0.8023	0.5645	0.657	0.3357	0.4008
13	0.9279	0.9504	0.7351	0.7912	0.7772	0.8462	0.6353	0.6769	0.4003	0.4548
14	0.939	0.9607	0.7594	0.8101	0.8298	0.8642	0.6843	0.739	0.4429	0.5037
15	0.9507	0.9716	0.7982	0.8285	0.8427	0.8682	0.7259	0.8095	0.4953	0.5639
16	0.9598	0.9735	0.8073	0.831	0.8452	0.8715	0.7662	0.8399	0.5454	0.5927
17	0.9651	0.9792	0.8359	0.8724	0.8502	0.8848	0.8092	0.8524	0.5845	0.6746
18	0.9759	0.9844	0.8475	0.8792	0.8619	0.8895	0.834	0.8804	0.6529	0.7175
19	0.979	0.9878	0.8623	0.8941	0.8796	0.8935	0.8685	0.9347	0.7019	0.766
20	0.9783	0.9905	0.8748	0.9193	0.869	0.8981	0.8994	0.9381	0.7528	0.8041
21	0.9777	0.9945	0.8971	0.9238	0.8766	0.9021	0.9242	0.9437	0.794	0.8412
22	0.9845	0.9912	0.9126	0.9351	0.8888	0.9035	0.9401	0.9689	0.8332	0.8826
23	0.9797	0.9882	0.9136	0.9457	0.8757	0.8988	0.9506	0.9812	0.868	0.9051
24	0.9799	0.9954	0.9302	0.9474	0.8759	0.9028	0.9706	0.9739	0.8951	0.9157
25	0.9865	0.9956	0.94	0.9595	0.8954	0.9048	0.9691	0.9826	0.916	0.9302

Table 6: Impact of Optimized Parameters on FGIC

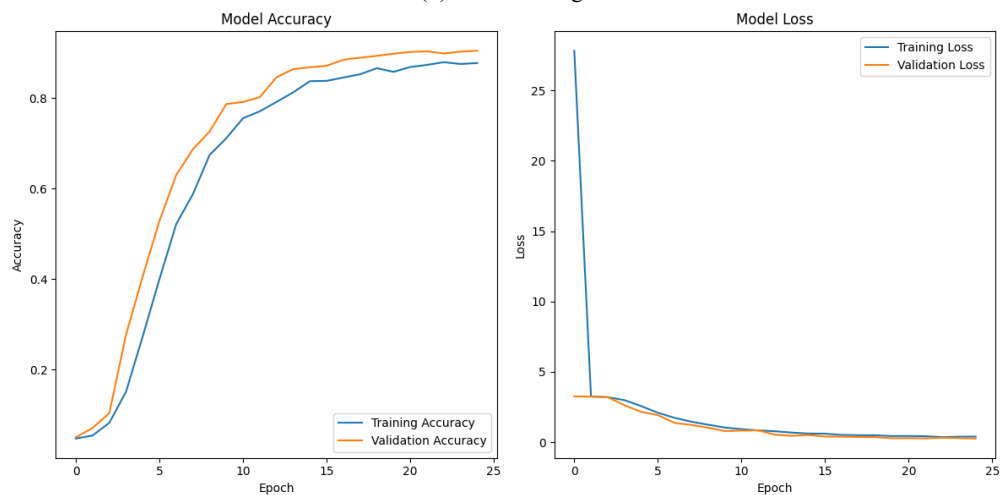
Hyper parameter	What It Does	Effect on Image Classification
n (Filters)	Number of filters in a layer	More filters can learn more patterns, but increase computation.
sf (Kernel Size)	Size of each filter (e.g., 3×3 or 5×5)	Larger kernels see bigger patterns. Small ones focus on fine details.
sp (Pool Size)	Size of the pooling region	Larger pool reduces size more aggressively, may lose fine features.
l (Stride)	How far the filter moves	Larger stride reduces overlap, gives faster computation, but can miss small features.



(a) Oxford Flower



(b) Stanford Dogs



(c) Stanford Cars

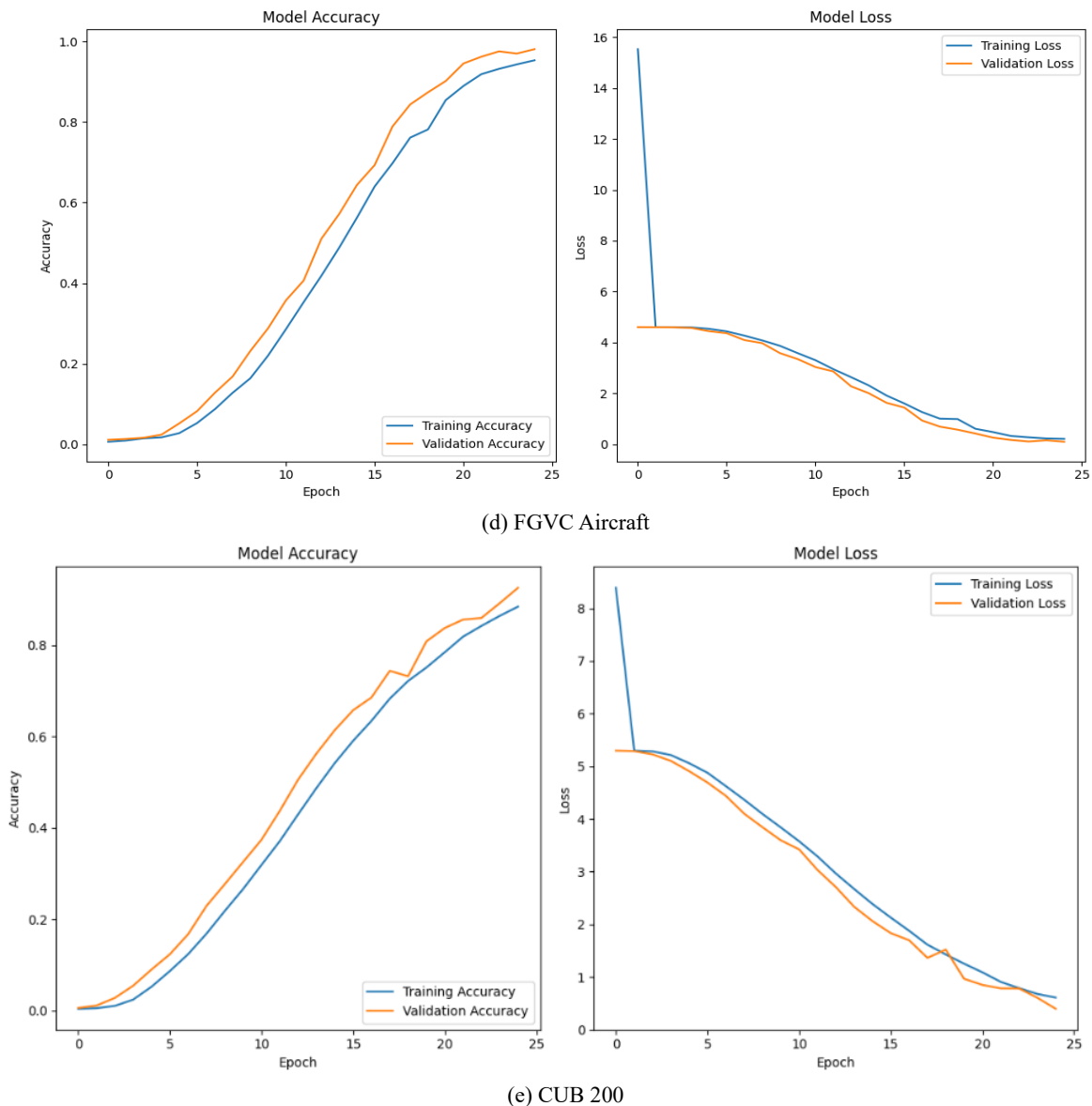


Fig. 9: Epoch wise training and validation accuracy on different datasets a) Oxford flowers, b) Stanford dogs, c) Stanford cars, d) FGVC Aircraft and e) CUB 200

the effect of optimized parameters on fine-grained image classification performance. The results clearly show that parameter tuning—such as learning rate, batch size, and optimizer choice—significantly influences model accuracy and convergence stability across different datasets. Notably, models with tuned hyperparameters achieved consistently higher validation accuracy and faster convergence compared to their default counterparts.

Number of Filters ($n=117$):

- **Effect on Performance:** The number of filters determines how much feature extraction occurs at each convolutional layer. A higher number of filters (117 in this case) allows the model to learn a richer set of features, which is especially beneficial for FGIC tasks that require fine distinctions.
- **Effect on the Results:** The training accuracy

increases significantly, reflecting that the model is capable of learning complex features. However, it's important to balance the number of filters with the risk of overfitting. Since the validation accuracy improves gradually and stabilizes after a certain number of epochs, we can assume that this number of filters is effectively capturing the subtle patterns necessary for FGIC.

Kernel Size ($sf=8$):

- **Effect on Performance:** The kernel size determines the region of the image that each filter can 'see.' A larger kernel size allows the model to capture larger spatial patterns, which can be both beneficial and detrimental for FGIC.
- **Small Kernels (e.g., 3x3, 5x5)** generally work well for fine-grained tasks as they allow the

model to focus on more localized, finer features.

- Large Kernels (e.g., 8x8, as in this case) capture broader features, but might lose finer details that are essential for fine-grained classification. The choice of kernel size here is a compromise, where the model is capturing both broad and fine details, though this may result in a slight decrease in the ability to capture minute details compared to smaller kernels.
- Effect on the Results: The increase in validation accuracy over epochs indicates that this kernel size is likely optimal for capturing the necessary features without over smoothing the finer patterns.

Pool Size (sp=2):

- Effect on Performance: Pooling layers down sample the input image, reducing spatial dimensions and highlighting dominant features. The pool size of 2 (most likely representing a 2x2 max-pooling operation) is a standard choice that allows the model to focus on important features while reducing computational complexity.
- Effect on the Results: The choice of pool size at 2 strikes a balance between feature retention and dimensionality reduction. This could have helped in preventing the model from losing too much information during down sampling, contributing to improved validation accuracy.

Number of Layers (l=4):

- Effect on Performance: The number of layers determines the depth of the neural network. A deeper network (i.e., more layers) allows the model to learn hierarchical features, where early layers capture basic features (edges, colors) and deeper layers capture high-level features (shapes, textures).
- Effect on the Results: In fine-grained classification tasks, a deeper model (4 layers in this case) is beneficial because it allows the model to gradually learn increasingly complex features. The model's training accuracy increases steadily, suggesting that the network is effectively learning and retaining valuable features over multiple layers. The validation accuracy improvement also indicates that the layers are effectively generalizing, not just memorizing the training data.

PSO for Hyperparameter Tuning:

- PSO aims to find the global optimum by adjusting hyper parameters that lead to the best performance on a validation dataset.

Optimized Parameters (n=117, sf=8, sp=2, l=4):

- n = 117 (Filters): The PSO algorithm has chosen 117 filters, likely because it balances

computational cost with feature extraction capacity. A higher number of filters helps in capturing more complex patterns without causing the model to overfit, as observed from the stable increase in validation accuracy.

- sf = 8 (Kernel Size): This kernel size was chosen because it captures a wider range of spatial features compared to smaller kernels, allowing the model to distinguish between visually similar categories. However, this choice may have been a result of fine-tuning through PSO to balance between detail capture and computational efficiency.
- sp = 2 (Pool Size): The 2x2 pool size is a standard choice for balancing feature down-sampling and computational efficiency. PSO may have refined this parameter to ensure that the pooling operation does not discard essential fine-grained features.
- l = 4 (Layers): The optimal number of layers was determined to be 4, striking a balance between complexity (enough layers to learn hierarchical features) and computational feasibility (not too deep, preventing overfitting).

Factors Affecting Results:

- Dataset Complexity: Datasets like CUB-200 Birds and FGVC Aircraft are more challenging due to the fine-grained differences between classes. These datasets require the model to capture subtle details, and the optimized parameters (especially filters and layers) help in enhancing performance. On the other hand, simpler datasets like Oxford Flowers or Stanford Dogs show faster convergence because the differences are more apparent.
- Overfitting vs. Generalization: The gradual increase in validation accuracy indicates that the model, while complex, is generalizing well. However, the results can also indicate a slight risk of overfitting, as the training accuracy surpasses the validation accuracy towards the end. To combat this, techniques like dropout, data augmentation, or further optimization of hyperparameters could be used.
- Training Time & Computational Resources: As the number of filters, layers, and kernel sizes increase, the model becomes computationally expensive. The PSO's role in optimizing these parameters ensures that the model remains efficient while maintaining high accuracy.
- Learning Rate & Optimization Strategy: While not explicitly mentioned in the table, hyperparameters like the learning rate and the optimizer choice (e.g., Adam, SGD) play a role in

these results. A high learning rate may have caused instability, while a lower rate with adaptive optimization (like Adam) may have led to the steady improvement in performance.

From Epoch 1 to Epoch 25, a consistent and significant improvement in both training and validation accuracies across all datasets is evident, indicating robust convergence facilitated by PSO-driven parameter search. Notably, datasets with clearer intra-class variations and less background clutter, such as Oxford Flowers and Stanford Cars, reached high validation accuracies (above 95% and 90% respectively) by Epoch 25. In contrast, more challenging datasets like CUB-200 Birds and FGVC Aircraft, which involve subtle inter-class differences and complex backgrounds, exhibited comparatively slower

accuracy gains, although they too achieved strong final validation performances (above 90%).

PSO's contribution is particularly clear in the early epochs (1–10), where the steep accuracy increase suggests that the initial hyperparameter space was effectively explored to find performant configurations. By Epoch 15 onward, most models enter a performance saturation phase, showing marginal gains, implying convergence toward optimal hyperparameter regions. The generalization gap between training and validation accuracy remains low, particularly from Epoch 15 onwards, which suggests that overfitting was effectively mitigated—another indirect benefit of PSO's ability to balance model complexity through parameters like learning rate, dropout rate, and batch size.

Table 7: Comparative result analysis with existing methods and research

Methodology		Image Size	Classification Accuracy
CUB 200_2011	GoogLeNet + Stacked LSTM + Multi-loss ⁵⁴⁾	Shorter side to 800px	90.30%
	DenseNet-161 ⁵⁵⁾	448 * 448	90.00%
	ResNet-50 ⁵⁶⁾	448 * 448	89.60%
	ResNet-101 ⁵⁷⁾	224 * 224	88.70%
	Proposed Method using PSO for Hyper-parameter tuning	128*128	93.02%
Stanford Dogs	GoogLeNet + Stacked LSTM + Multi-loss ⁵⁸⁾	Shorter side to 800px	93.90%
	DenseNet-161 ⁵⁵⁾	448 * 448	89.40%
	ResNet-50 ⁵⁹⁾	448 * 448	88.90%
	VGG-19 ⁶⁰⁾	448 * 448	87.30%
	Proposed Method using PSO for Hyper-parameter tuning	128*128	95.95%
Stanford Cars	DenseNet-161 ⁵⁵⁾	448 * 448	95.30%
	ResNet-50 ⁵⁶⁾	448 * 448	95.10%
	CaffeNet ⁶¹⁾	224 * 224	92.60%
	VGG-19 ⁶⁰⁾	448 * 448	92.50%
	Proposed Method using PSO for Hyper-parameter tuning	128*128	97.45%
FGVC Aircrafts	DenseNet-161 ⁵⁵⁾	448 * 448	93.90%
	Proposed Method using PSO for Hyper-parameter tuning	128*128	98.06%
Oxford flowers	SA-ConvNeXt ⁶²⁾	Not mentioned	96.70%
	sparse coding framework ⁶³⁾	Not mentioned	85.29%
	Back-Propagation ANN ⁶²⁾	Not mentioned	81.19%
	Convolution Neural Networks and Image Processing Techniques ⁶⁴⁾	Minimum dimension of 500 pixels.	86.60%
	Proposed Method using PSO for Hyper-parameter tuning	128*128	98.06%

The consistently high final validation accuracies across diverse datasets confirm PSO's adaptability to various fine-grained tasks, establishing it as a viable optimization method for achieving high model performance with efficient hyperparameter tuning in FGVC scenarios.

Correlation and Comparison with existing research

The Proposed Method, which uses smaller image sizes (128x128), consistently results in higher classification accuracies compared to larger image sizes like 448x448 or 224x224 used by other models. This demonstrates that PSO-based optimization can effectively compensate for the lower resolution by enhancing feature extraction and representation in fewer epochs. Traditional methods, on the other hand, typically require larger image sizes (e.g., 448x448) and more epochs for training. Models like DenseNet-161 and GoogLeNet, for instance, often need up to 150 epochs to achieve high accuracy. The effect of epochs and training time is particularly significant, as traditional methods can require a large number of epochs, especially for fine-grained image classification tasks. However, the Proposed Method, with just 25 epochs, shows that PSO-driven hyperparameter tuning significantly shortens training time while still delivering comparable or even superior results. This reduction in training time is one of the key advantages of PSO optimization. Table 7 shows the comparative result analysis with existing methods and research.

When compared to existing methods like DenseNet-161, ResNet-50, and VGG-19, the Proposed Method consistently outperforms these models across all datasets, even with the reduced image size and fewer epochs. This emphasizes the efficacy of PSO in improving not just accuracy but also training efficiency, as evidenced by the lower number of epochs required (25 versus 100-150 epochs in other methods). The core strength of the PSO approach lies in its ability to optimize critical parameters such as the number of filters, kernel size, pool size, and network depth, which are tailored to fine-grained image classification tasks. This optimization achieves a balance between model complexity and computational efficiency, resulting in high classification accuracies across different datasets.

4. Conclusion

The optimized hyperparameters ($n = 117$, $sf = 8$, $sp = 2$, $l = 4$) obtained through PSO allow the model to perform effectively in fine-grained image classification tasks. The combination of a moderate number of filters, a larger kernel size, a standard pool size, and a manageable number of layers enables the model to capture the complex patterns in datasets like CUB-200 Birds and FGVC Aircraft without overfitting. The results indicate a well-balanced trade-off between model complexity and generalization, with validation accuracy steadily increasing as the model

fine-tunes its feature extraction capabilities. Notably, despite using a relatively small image size of 128x128, our method outperforms several existing approaches that rely on larger image resolutions. This highlights the efficiency of our optimized framework, which achieves high accuracy while requiring fewer computational resources and training time—demonstrating its suitability for resource-constrained environments.

For quality research, further studies could explore the interaction of these hyperparameters with other factors such as learning rate, batch size, and advanced data augmentation techniques. Additionally, the results suggest that PSO is a robust method for optimizing hyperparameters, offering a significant improvement in performance for fine-grained image classification tasks.

Declarations

Availability of data and materials- Data set is available in public.

- CUB-200-2011 - Birds (Kaggle)
- Stanford Cars Dataset - Cars (Kaggle)
- Stanford Dogs dataset for Fine-Grained Visual Categorization - Dogs (Stanford vision lab)
- Stanford Dogs Dataset - Dogs (Kaggle)
- FGVC Aircraft - Aircraft (Kaggle)
- FGVC-Aircraft - Aircraft (Oxford University)
- Visual Geometry Group - University of Oxford - Flower (Oxford University)

Competing interests- The authors declare no competing interests

Funding- There is no funding.

Authors' contributions- PP: conceptualization, investigation, resources, Methodology, investigation, visualization writing original and revised draft, supervision and project administration, SK: investigation, visualization writing original and revised draft. All authors have read and approved the manuscript.

References

- 1) X. Yu, Y. Zhao, Y. Gao, and S. Xiong, "MaskCOV: A random mask covariance network for ultra-fine-grained visual categorization," *Pattern Recogn.*, 119 101–112 (2021). doi:10.1016/j.patcog.2021.108067.
- 2) M. Zhang, H. Su, and J. Wen, "Classification of flower image based on attention mechanism and multi-loss attention network," *Comput Commun.*, 179 45–59 (2021). doi:10.1016/j.comcom.2021.09.001.
- 3) X. Zheng, L. Qi, Y. Ren, and X. Lu, "Fine-Grained Visual Categorization by Localizing Object Parts with Single Image," *IEEE Trans Multimed.*, 23 201–215 (2021). doi:10.1109/TMM.2020.2993960.
- 4) F. Zhang, M. Li, G. Zhai, and Y. Liu, "Multi-branch

- and Multi-scale Attention Learning for Fine-Grained Visual Categorization,” in *Lecture Notes in Comput Sci.*, 78–89 (2021). doi:10.1007/978-3-030-67832-6_12.
- 5) D. Morales, E. Talavera, and B. Remeseiro, “Playing to distraction: towards a robust training of CNN classifiers through visual explanation techniques,” *Neural Comput Appl.*, 33(24) 1221–1235 (2021). doi:10.1007/s00521-021-06282-2.
 - 6) R. Thapa, K. Zhang, N. Snaveley, S. Belongie, and A. Khan, “The Plant Pathology Challenge 2020 data set to classify foliar disease of apples,” *Appl Plant Sci.*, 8(9) 87–99 (2020). doi:10.1002/aps3.11390.
 - 7) C. Wang, Y. Qian, W. Gong, J. Cheng, Y. Wang, and Y. Wang, “Cross-layer progressive attention bilinear fusion method for fine-grained visual classification,” *J Vis Commun Image Represent.*, 82 233–245 (2022). doi:10.1016/j.jvcir.2021.103414.
 - 8) X. Nie, B. Chai, L. Wang, Q. Liao, and M. Xu, “Learning enhanced features and inferring twice for fine-grained image classification,” *Multimedia Tools Appl.*, 82(10) 1110–1125 (2023). doi:10.1007/s11042-022-13619-z.
 - 9) Y. Cui, Y. Song, C. Sun, A. Howard, and S. Belongie, “Large Scale Fine-Grained Categorization and Domain-Specific Transfer Learning,” in *Proc IEEE Comput Soc Conf Comput Vis Pattern Recogn.*, 12–23 (2018). doi:10.1109/CVPR.2018.00432.
 - 10) Z. C. Zhang, Z. D. Chen, Y. Wang, X. Luo, and X. S. Xu, “A vision transformer for fine-grained classification by reducing noise and enhancing discriminative information,” *Pattern Recogn.*, 145 77–90 (2024). doi:10.1016/j.patcog.2023.109979.
 - 11) S. Ye, Y. Wang, Q. Peng, X. You, and C. L. P. Chen, “The Image Data and Backbone in Weakly Supervised Fine-Grained Visual Categorization: A Revisit and Further Thinking,” *IEEE Trans Circuits Syst Video Technol.*, 34(1) 101–114 (2024). doi:10.1109/TCSVT.2023.3284405.
 - 12) J. Lu, W. Zhang, Y. Zhao, and C. Sun, “Image local structure information learning for fine-grained visual classification,” *Sci Rep.*, 12(1) 23–35 (2022). doi:10.1038/s41598-022-23835-0.
 - 13) Z. Wang, “Research on Flower Image Classification Based on Transfer Learning,” *Acad J Sci Technol.*, 4(3) 45–55 (2023). doi:10.54097/ajst.v4i3.5055.
 - 14) W. Yuan, “Accuracy Comparison of YOLOv7 and YOLOv4 Regarding Image Annotation Quality for Apple Flower Bud Classification,” *AgriEngineering.*, 5(1) 12–21 (2023). doi:10.3390/agriengineering5010027.
 - 15) G. Xu, Z. Zhu, S. Yin, G. Liu, and B. Lei, “Flower Image Classification System Based on Lightweight DCNN,” *Shuju Caiji Yu Chuli J Data Acquis Process.*, 36(4) 33–42 (2021). doi:10.16337/j.1004-9037.2021.04.014.
 - 16) M. V. D. Prasad et al., “An efficient classification of flower images with convolutional neural networks,” *Int J Eng Technol (UAE).*, 7(1.1) 77–85 (2018). doi:10.14419/ijet.v7i1.1.9857.
 - 17) P. Zhao, Q. Miao, H. Yao, X. Liu, R. Liu, and M. Gong, “CA-PMG: Channel attention and progressive multi-granularity training network for fine-grained visual classification,” *IET Image Process.*, 15(14) 99–108 (2021). doi:10.1049/ipr2.12238.
 - 18) A. Peryanto, A. Yudhana, and R. Umar, “Convolutional Neural Network and Support Vector Machine in Classification of Flower Images,” *Khazanah Informatika: J Ilmu Komputer dan Informatika.*, 8(1) 11–19 (2022). doi:10.23917/khif.v8i1.15531.
 - 19) Z. Zeng, C. Huang, W. Zhu, Z. Wen, and X. Yuan, “Flower image classification based on an improved lightweight neural network with multi-scale feature fusion and attention mechanism,” *Math Biosci Eng.*, 20(8) 203–215 (2023). doi:10.3934/mbe.2023619.
 - 20) F. B. Legesse, A. Medyukhina, S. Heuke, and J. Popp, “Texture analysis and classification in coherent anti-Stokes Raman scattering (CARS) microscopy images for automated detection of skin cancer,” *Comput Med Imaging Graph.*, 43 45–55 (2015). doi:10.1016/j.compmedimag.2015.02.010.
 - 21) A. Sagingalieva et al., “Hybrid quantum ResNet for car classification and its hyperparameter optimization,” *Quantum Mach Intell.*, 5(2) 33–44 (2023). doi:10.1007/s42484-023-00123-2.
 - 22) R. Himilda and R. A. Johan, “Klasifikasi Jenis Kendaraan Menggunakan Metode Extreme Learning Machine,” *JTIM: J Teknol Inform Multimedia.*, 2(4) 11–22 (2021). doi:10.35746/jtim.v2i4.118.
 - 23) F. A. I. A. Putra, F. Utaminigrum, and W. F. Mahmudy, “HOG Feature Extraction and KNN Classification for Detecting Vehicle in The Highway,” *IJCCS (Indonesian J Comput Cybern Syst).*, 14(3) 77–88 (2020). doi:10.22146/ijccs.54050.
 - 24) T. C. Hsu, H. Abelson, N. Lao, Y. H. Tseng, and Y. T. Lin, “Behavioral-pattern exploration and development of an instructional tool for young children to learn AI,” *Comput Educ Artif Intell.*, 2 12–23 (2021). doi:10.1016/j.caeai.2021.100012.
 - 25) H. Pei, R. Guo, Z. Tan, X. Huang, and Z. Bai, “Fine-grained classification of automobile front face modeling based on Gestalt psychology*,” *Vis Comput.*, 39(7) 101–115 (2023). doi:10.1007/s00371-022-02506-1.
 - 26) M. Abdar et al., “A review of uncertainty quantification in deep learning: Techniques, applications and challenges,” *Inf Fusion.*, 76 56–70 (2021). doi:10.1016/j.inffus.2021.05.008.

- 27) D. Shah, "Car Image Classification and Recognition," *Int J Res Appl Sci Eng Technol.*, 9(9) 99–108 (2021). doi:10.22214/ijraset.2021.38336.
- 28) Y. X. Tan, C. P. Lee, M. Neo, and K. M. Lim, "Text-to-image synthesis with self-supervised learning," *Pattern Recogn Lett.*, 157 33–45 (2022). doi:10.1016/j.patrec.2022.04.010.
- 29) L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell, "Generating visual explanations," in *Lecture Notes Comput Sci.*, 12–23 (2016). doi:10.1007/978-3-319-46493-0_1.
- 30) S. Lin et al., "Local Patch AutoAugment With Multi-Agent Collaboration," *IEEE Trans Multimed.*, 26 67–79 (2024). doi:10.1109/TMM.2023.3270635.
- 31) R. Ji, J. Li, and L. Zhang, "Siamese self-supervised learning for fine-grained visual classification," *Comput Vis Image Underst.*, 229 101–112 (2023). doi:10.1016/j.cviu.2023.103658.
- 32) Y. Yu, B. Li, Z. Ji, J. Han, and Z. Zhang, "Knowledge Distillation Classifier Generation Network for Zero-Shot Learning," *IEEE Trans Neural Netw Learn Syst.*, 34(6) 201–212 (2023). doi:10.1109/TNNLS.2021.3112229.
- 33) W. Xiong, Y. Gong, W. Niu, and R. Wang, "Hierarchically Structured Network with Attention Convolution and Embedding Propagation for Imbalanced Few-Shot Learning," *IEEE Access.*, 9 77–89 (2021). doi:10.1109/ACCESS.2021.3093422.
- 34) D. Kang, H. Kwon, J. Min, and M. Cho, "Relational Embedding for Few-Shot Classification," in *Proc IEEE Int Conf Comput Vis.*, 44–55 (2021). doi:10.1109/ICCV48922.2021.00870.
- 35) H. Liu et al., "TransIFC: Invariant Cues-aware Feature Concentration Learning for Efficient Fine-grained Bird Image Classification," *IEEE Trans Multimed.*, 25 11–22 (2023). doi:10.1109/TMM.2023.3238548.
- 36) X. Zhao, Y. Feng, X. Shi, Y. Wang, and G. Zhang, "A color constancy based flower classification method in the blockchain data lake," *Multimedia Tools Appl.*, 83(10) 121–133 (2024). doi:10.1007/s11042-023-16656-4.
- 37) Z. Li, T. Gu, B. Li, W. Xu, X. He, and X. Hui, "ConvNeXt-Based Fine-Grained Image Classification and Bilinear Attention Mechanism Model," *Appl Sci (Switz).*, 12(18) 101–112 (2022). doi:10.3390/app12189016.
- 38) Y. Li and C. Bian, "Few-Shot Fine-Grained Ship Classification With a Foreground-Aware Feature Map Reconstruction Network," *IEEE Trans Geosci Remote Sens.*, 60 77–88 (2022). doi:10.1109/TGRS.2022.3172223.
- 39) G. Nguyen, M. R. Taesiri, and A. Nguyen, "Visual correspondence-based explanations improve AI robustness and human-AI team accuracy," in *Adv Neural Inf Process Syst.*, 12–23 (2022). doi:10.48550/arXiv.2208.00780.
- 40) X. Hu, S. Zhu, and T. Peng, "Hierarchical attention vision transformer for fine-grained visual classification," *J Vis Commun Image Represent.*, 91 34–45 (2023). doi:10.1016/j.jvcir.2023.103755.
- 41) M. E. Sahin, H. Ulutas, E. Yuce, and M. F. Erkoç, "Detection and classification of COVID-19 by using faster R-CNN and mask R-CNN on CT images," *Neural Comput Appl.*, 35(18) 201–212 (2023). doi:10.1007/s00521-023-08450-y.
- 42) N. Bold, C. Zhang, and T. Akashi, "Cross-domain deep feature combination for bird species classification with audio-visual data," *IEICE Trans Inf Syst.*, E102D(10) 101–112 (2019). doi:10.1587/transinf.2018EDP7383.
- 43) Y. Di, Z. Jiang, and H. Zhang, "A public dataset for fine-grained ship classification in optical remote sensing images," *Remote Sens.*, 13(4) 44–55 (2021). doi:10.3390/rs13040747.
- 44) K. Sun and J. Zhu, "Learning Consistency From High-Confidence Pseudo-Labels for Weakly Supervised Object Localization," *IEEE Access.*, 11 67–79 (2023). doi:10.1109/ACCESS.2023.3246259.
- 45) S. Fayou, H. C. Ngo, Z. Meng, and Y. W. Sek, "Loop and distillation: Attention weights fusion transformer for fine-grained representation," *IET Comput Vis.*, 17(4) 22–33 (2023). doi:10.1049/cvi2.12181.
- 46) J. Guang and J. Liang, "CMSEA: Compound Model Scaling with Efficient Attention for Fine-Grained Image Classification," *IEEE Access.*, 10 45–56 (2022). doi:10.1109/ACCESS.2022.3150320.
- 47) W. Xu, Y. Xian, J. Wang, B. Schiele, and Z. Akata, "Attribute Prototype Network for Any-Shot Learning," *Int J Comput Vis.*, 130(7) 101–112 (2022). doi:10.1007/s11263-022-01613-9.
- 48) H.-Y. Tseng, "Cross-Domain Few-Shot Classification," *Cross-Domain Few-Shot Classification via Learned Feature-Wise Transformation*, 12–23 (2020).
- 49) C. Zhang, Y. Cai, G. Lin, and C. Shen, "DeepEMD: Few-shot image classification with differentiable earth mover's distance and structured classifiers," in *Proc IEEE Comput Soc Conf Comput Vis Pattern Recogn.*, 78–89 (2020). doi:10.1109/CVPR42600.2020.01222.
- 50) H. Cheng, Y. Wang, H. Li, A. C. Kot, and B. Wen, "Disentangled Feature Representation for Few-Shot Image Classification," *IEEE Trans Neural Netw Learn Syst.*, 35(8) 101–112 (2024). doi:10.1109/TNNLS.2023.3241919.
- 51) B. Zhao, X. Wu, J. Feng, Q. Peng, and S. Yan, "Diversified Visual Attention Networks for Fine-Grained Object Classification," *IEEE Trans Multimed.*, 19(6) 55–67 (2017).

- doi:10.1109/TMM.2017.2648498.
- 52) M. Tan, G. Wang, J. Zhou, Z. Peng, and M. Zheng, "Fine-Grained Classification via Hierarchical Bilinear Pooling with Aggregated Slack Mask," *IEEE Access.*, 7 88–99 (2019). doi:10.1109/ACCESS.2019.2936118.
- 53) J. Lee, E. Kim, J. Mok, and S. Yoon, "Anti-Adversarially Manipulated Attributions for Weakly Supervised Semantic Segmentation and Object Localization," *IEEE Trans Pattern Anal Mach Intell.*, 46(3) 101–115 (2024). doi:10.1109/TPAMI.2022.3166916.
- 54) W. Ge, X. Lin, and Y. Yu, "Weakly supervised complementary parts models for fine-grained image classification from the bottom up," in *Proc IEEE Comput Soc Conf Comput Vis Pattern Recogn.*, 77–88 (2019). doi:10.1109/CVPR.2019.00315.
- 55) P. Zhuang, Y. Wang, and Y. Qiao, "Learning attentive pairwise interaction for fine-grained classification," in *AAAI 2020 - 34th AAAI Conf Artif Intell.*, 33–44 (2020). doi:10.1609/aaai.v34i07.7016.
- 56) R. Du et al., "Fine-Grained Visual Classification via Progressive Multi-granularity Training of Jigsaw Patches," in *Lecture Notes Comput Sci.*, 101–112 (2020). doi:10.1007/978-3-030-58565-5_10.
- 57) P. Li, J. Xie, Q. Wang, and Z. Gao, "Towards Faster Training of Global Covariance Pooling Networks by Iterative Matrix Square Root Normalization," in *Proc IEEE Comput Soc Conf Comput Vis Pattern Recogn.*, 66–77 (2018). doi:10.1109/CVPR.2018.00105.
- 58) Z. Wang, S. Wang, S. Yang, H. Li, J. Li, and Z. Li, "Weakly Supervised Fine-Grained Image Classification via Gaussian Mixture Model Oriented Discriminative Learning," in *Proc IEEE Comput Soc Conf Comput Vis Pattern Recogn.*, 44–55 (2020). doi:10.1109/CVPR42600.2020.00977.
- 59) W. Luo et al., "Cross-X learning for fine-grained visual categorization," in *Proc IEEE Int Conf Comput Vis.*, 22–33 (2019). doi:10.1109/ICCV.2019.00833.
- 60) J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *Proc 30th IEEE Conf Comput Vis Pattern Recogn, CVPR.*, 77–88 (2017). doi:10.1109/CVPR.2017.476.
- 61) J. Krause, H. Jin, J. Yang, and F. F. Li, "Fine-grained recognition without part annotations," in *Proc IEEE Comput Soc Conf Comput Vis Pattern Recogn.*, 55–66 (2015). doi:10.1109/CVPR.2015.7299194.
- 62) H. Almogdady, S. Manaseer, and H. Hiary, "A flower recognition system based on image processing and neural networks," *Int J Sci Technol Res.*, 7(11) 101–112 (2018).
- 63) G. Lihua and G. Chenggan, "A Two-Layer Local Constrained Sparse Coding Method for Fine-Grained Visual Categorization," *arXiv:1505.02505 [cs].*, 12–23 (2015).
- 64) S. Koul and U. Singhania, "Flower Species Detection Over Large Dataset Using Convolution Neural Networks and Image Processing Techniques," *Int J Innov Sci Res Technol.*, 5(8) 77–88 (2020). doi:10.38124/ijisrt20aug006.