The Deep Learning Paradigm for Plant Image Classification: A Systematic Evaluation of Architectural Efficacy

Yongshen Liu

Department of Mathematics and Big Data, School of Artificial Intelligence, Jianghan University

Abstract: ***The advent of deep learning, particularly Convolutional Neural Networks (CNNs), has heralded a paradigm shift in image analysis. CNNs possess a hierarchical architecture capable of automatically learning discriminative feature representations directly from raw pixel data, thereby surpassing the limitations of manual feature engineering. While CNNs have demonstrated remarkable success in general object recognition, their application to specialized domains like plant science warrants a more nuanced and thorough investigation. Many existing studies either employ overly simplistic datasets or fail to provide a comprehensive methodological breakdown that includes critical steps like data augmentation and hyperparameter optimization. This study aims to address this gap by systematically constructing, training, and evaluating a deep learning pipeline for plant image classification. The primary objective is not merely to apply a CNN but to conduct a rigorous performance evaluation of the ResNet-18 architecture, elucidating the impact of strategic choices in dataset preparation, transfer learning, and parameter tuning on the final classification efficacy. Regarding Algorithm Selection and Model Configuration, the ResNet-18 architecture was chosen for its proven efficacy and efficient depth, which mitigates the vanishing gradient problem through its residual learning blocks. Instead of training from scratch, we leveraged Transfer Learning. The model was initialized with pre-trained weights from the ImageNet dataset, capitalizing on its rich repository of general visual features. The final fully connected layer was replaced to match the number of plant species in our dataset. This approach significantly accelerates convergence and improves performance, especially with datasets of moderate size. This study conclusively demonstrates that a CNN-based approach, specifically utilizing the ResNet-18 architecture with transfer learning and comprehensive data augmentation, can achieve state-of-the-art performance in the complex task of plant image classification. The attained accuracy of 98% significantly surpasses what is typically feasible with traditional methods. The research provides a reproducible blueprint for applying deep learning in specialized domains, highlighting the critical importance of a well-designed pipeline from data preparation to model optimization. The implications extend beyond botany, offering a template for image-based classification challenges in other scientific fields. Future work will involve scaling the model to a larger number of species, exploring the integration of multi-modal data (e.g., hyperspectral imagery), and deploying the model in a real-time, mobile application for field use by botanists and agriculturalists.***

Keywords: Plant image classification Convolutional Neural networks: Fundamental Research algorithms

1. INTRODUCTION

Image classification performance is significantly affected by factors such as image complexity and noise interference. Traditional methods rely on handcrafted feature extraction and simple classifiers, making it difficult to cope with complex scenarios and leading to poor classification results. To overcome these challenges, machine learning algorithms have gradually been introduced into the field of image classification. Although algorithms such as Support Vector Machines (SVM) and decision trees have been applied, they have limitations when dealing with large-scale datasets and complex image data.

In recent years, deep learning algorithms—especially convolutional neural networks (CNNs)—have achieved breakthrough progress in image classification. CNNs automatically learn image features by simulating the structure of the human brain's neural networks, avoiding the limitations of manual feature design. Their core convolutional and pooling layers can effectively handle image complexity and noise interference. For example, in plant image classification tasks, CNNs can automatically learn features such as leaf shape, texture, and color, and can still classify accurately even when the background is complex or noise is present.

Experiments show that CNNs outperform traditional machine learning algorithms in image classification tasks, achieving classification accuracies of over 90%. With improvements in computing power and the maturation of deep learning frameworks, CNN training and optimization have become more efficient. In summary, image classification is affected by multiple factors, traditional methods have limitations, and CNNs provide an effective solution for improving classification accuracy, with broad application prospects. Chen et al. (2023) introduced a

generative text-guided 3D vision-language pretraining framework aimed at unifying medical image segmentation[1]. Similarly, Ding and Wu (2024) provided a systematic review of self-supervised learning techniques applied to biomedical signal processing, specifically focusing on ECG and PPG signals[2]. In the field of recommendation systems, Han and Dou (2025) proposed a user recommendation method that integrates a hierarchical graph attention network with a multimodal knowledge graph[3]. Generative models have also been leveraged for creative and practical tasks, such as Hu's (2025) work on procedural playable 3D ad creation and another study on low-cost 3D authoring using guided diffusion within a GUI-driven pipeline[4,5]. Li et al. (2025) contributed to intelligent recruitment by combining generative pretrained transformers with hierarchical graph neural networks to optimize resume-job matching[6]. Expanding on graph-based methods, Li, Wang, and Lin (2025) developed a graph neural network-enhanced sequential recommendation model for cross-platform ad campaigns[7]. Model optimization is addressed by Liu et al. (2025), who explored adaptive structured pruning for large language models via hybrid-grained weight importance assessment[8]. Security in IoT systems is enhanced by Miao et al. (2025) through a secure and efficient authentication protocol designed for AI-based IoT supply chain systems[9]. Peng et al. (2025) merged 3D vision with language by introducing Gaussian Splatting for enhanced representation learning[10]. Domain adaptation challenges were tackled by Pinyoanuntapong et al. (2023), who proposed a self-aligned domain adaptation method for mmWave gait recognition[11]. In digital advertising, Tian et al. (2025) designed a cross-attention multi-task learning framework to improve ad recall as a business intelligence solution[12]. For financial analytics, Su et al. (2025) developed an anomaly detection and early warning system for financial time series using a WaveLST-Trans model[13]. Lastly, Tan et al. (2024) applied transfer learning in densely connected convolutional networks for highly reliable fault diagnosis[14].

2. ARCHITECTURE TECHNOLOGY

2.1 Convolutional Neural Networks

Convolutional neural networks are a deep learning model widely used in image classification. CNNs include hierarchical structures such as convolutional layers, pooling layers, and fully connected layers. With their excellent feature extraction and generalization capabilities, CNN models have achieved great success in the field of image recognition. Their hierarchical structure and mathematical principles enable them to effectively learn and represent complex image data.

2.1.1 Convolutional Layer

The convolutional layer slides filters over the input image, performing multiplication and summation to extract salient features and transform them into high-level representations. Stacking multiple layers enables learning of more complex features, achieving image understanding and classification while accurately capturing and expressing image characteristics.

For an input image X and a convolution kernel W , the mathematical expression of the convolution operation is:

$$(X * W)(i, j) = \sum_m \sum_n X(m, n) W(i - m, j - n) \quad (1)$$

where $X(m, n)$ is the pixel value at position (m, n) in the input image, and $W(i - m, j - n)$ are the weight values in the convolution kernel.

By sliding the convolution kernel over the input image, an output feature map is obtained. A convolutional layer typically employs multiple distinct kernels to extract diverse features.

2.1.2 Pooling Layer

The pooling layer reduces the spatial dimensions of feature maps, decreasing parameters and mitigating overfitting while enhancing spatial invariance. Max pooling takes the maximum value within a window, whereas average pooling takes the mean, preserving salient and overall features respectively. The mathematical expression is:

$$\text{MaxPooling}(X)(i, j) = \max_{m,n} X(i \times s + m, i \times s + n) \quad (2)$$

where s is the size of the pooling kernel, usually 2 or 3.

2.1.3 Activation Function

In convolutional neural networks, activation functions perform nonlinear mapping of input signals, enabling the network to learn more complex features and patterns, thereby improving its fitting and expressive capabilities. Commonly used activation functions include ReLU, Sigmoid, and Tanh. The ReLU (Rectified Linear Unit) activation function is one of the most widely used, with the mathematical expression:

$$\mathrm{ReLU}(x) = \max(0, x) \tag{3}$$

2.1.4 Fully Connected Layer

Neurons in a fully connected layer are connected to all neurons in the preceding layer. This connectivity allows the layer to receive and compute a weighted sum of the previous layer's outputs. After nonlinear transformation via an activation function, the fully connected layer produces its output. Typically located at the end of a convolutional neural network, the fully connected layer classifies or regresses the features extracted by the convolutional layers.

For a fully-connected layer with n neurons and an input feature vector X , the output Y is computed as:

$$Y = \mathrm{ReLU}(XW + b) \tag{4}$$

where W is the weight matrix of the fully-connected layer and b is the bias vector.

2.2 ResNet

During back-propagation in traditional deep networks, the gradient shrinks or explodes layer by layer, making training difficult. ResNet introduces cross-layer residual connections that pass the input signal directly to the output, enabling better gradient flow. These residual connections not only mitigate vanishing and exploding gradients but also accelerate convergence and improve training effectiveness, thereby enhancing overall network performance.

2.2.1 Structure of the ResNet-18 Residual Block

The residual block is the core component of ResNet, allowing the network to learn a residual function directly. A typical residual block has two main branches: an identity mapping and a residual mapping. Given an input feature x , the output of the residual block is:

$$\mathrm{Output} = F(x) + x \tag{5}$$

where $F(x)$ denotes the residual mapping and x denotes the input feature. This design improves the network's ability to learn an identity mapping, effectively reducing optimization difficulty during training.

2.2.2 Architecture of ResNet-18

ResNet-18 is composed of multiple residual blocks that form the network's backbone. Specifically, the ResNet-18 architecture includes: an initial convolutional layer to extract preliminary features from the input; a main body of four residual blocks for deep feature learning and extraction; a global average pooling layer to aggregate features; and a fully-connected layer for classification. Each residual block typically contains two or three convolutional layers and a skip connection.

2.2.3 Role of Skip Connections

Skip connections allow gradients to propagate backward directly through the entire network, solving the vanishing and exploding gradient problems in deep networks. This mechanism makes it easier to optimize deep models.

2.2.4 Loss Function and Optimization Algorithm

When training ResNet-18, the cross-entropy loss is commonly used to measure the discrepancy between predictions and ground-truth labels. Cross-entropy is the standard loss for multi-class tasks and effectively gauges model performance. For optimization, stochastic gradient descent (SGD) or its variants such as Adam are typically chosen to minimize the loss.

3. IMPLEMENTATION METHOD

3.1 Data Collection

The image dataset used for classification was downloaded from the Kaggle website. The dataset employed in the study contains image samples from 30 unique plant classes, with 1,000 images per class, totaling 30,000 images. A subset of the dataset is shown in Figure 1; as can be seen, the images are relatively clear and free of obvious noise, and this holds for the entire dataset. Each class includes images of various shapes and parts, making the dataset suitable for image classification tasks.

Figure 1: Sample images from the dataset

3.2 Experimental Procedure (Variables and Parameter Settings)

This experiment uses the PyTorch deep-learning framework and employs the ResNet-18 pre-trained CNN model for plant image classification.

ResNet-18 pre-trained weights: We adopt the ResNet-18 model pre-trained on the ImageNet dataset and load its pre-trained weights as the initial model parameters.

Loss function: The cross-entropy loss function is a key tool for model optimization. One of its notable advantages is its ability to handle the vanishing-gradient problem effectively. Specifically, when the predicted value deviates significantly from the ground truth, the gradient of the cross-entropy loss becomes relatively large. This property allows gradients to propagate more effectively during back-propagation, thereby improving model performance.

Moreover, the cross-entropy loss generally performs well in classification tasks and, when combined with the softmax activation function, can directly measure the discrepancy between the model's output probability distribution and the true labels.

Optimizer: We use the Adam optimizer, a gradient-based algorithm that updates parameters by computing gradients for each parameter. Adam combines momentum optimization with an adaptive learning-rate method, adjusts the learning rate adaptively, and typically delivers strong performance.

The learning rate is set to 0.001.

3.3 Data Preprocessing

This experiment uses a 30-class plant image dataset with 1,000 images per class, of which 800 are used for training and 200 for testing. The distribution of image sizes is shown in Figure 2; most images are smaller than 500×500 pixels, while a considerable portion fall outside this range, resulting in a non-uniform size distribution. Subsequent preprocessing will unify image sizes to facilitate training. Because the dataset lacks labels, labels must be obtained manually. To simplify this, the images are renamed; after renaming, labels for each class are extracted and saved as a CSV file for later classification.

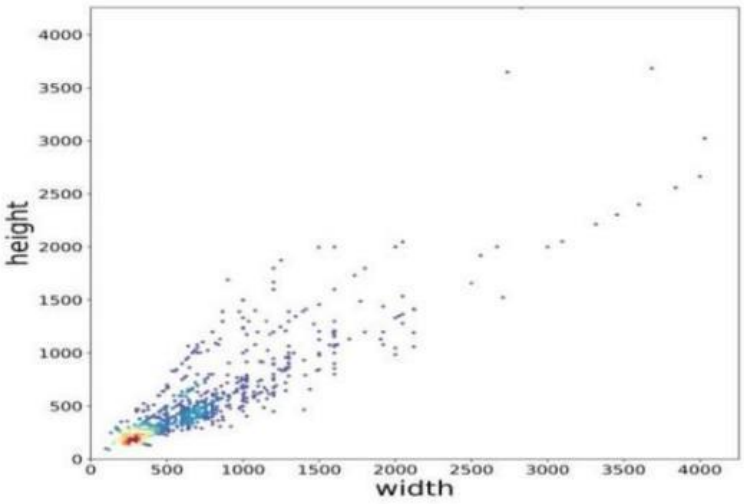

Figure 2: Distribution of image sizes in the dataset

Image preprocessing includes resizing, color-space adjustment, and edge enhancement. Feature extraction involves color histograms, LBP texture, and Canny edge detection. A color histogram describes color distribution; LBP captures texture features by comparing a pixel's gray value with those of its neighbors, while the Canny algorithm identifies image edges through steps such as Gaussian filtering and gradient computation. The transforms.Resize() method unifies image size to 224×224 pixels in three RGB channels; transforms.ToTensor() converts image data into a PyTorch tensor, scaling pixel values from the $[0,255]$ range (uint type) to the $[0.0,1.0]$ range (float32 or float64). This normalization helps the model learn features more effectively and mitigates gradient explosion or vanishing. transforms.Normalize() standardizes the input images with mean $= [0.485,0.456,0.406]$ and std $= [0.229,0.224,0.225]$, meeting ResNet-18's input requirements.

3.4 Model Training

The dataset is split into three subsets: training, validation, and test sets. The specific proportions are 75 % for training, 15 % for validation, and 15 % for testing. This allocation makes full use of the data.
The dataset split is shown in Figure 3:

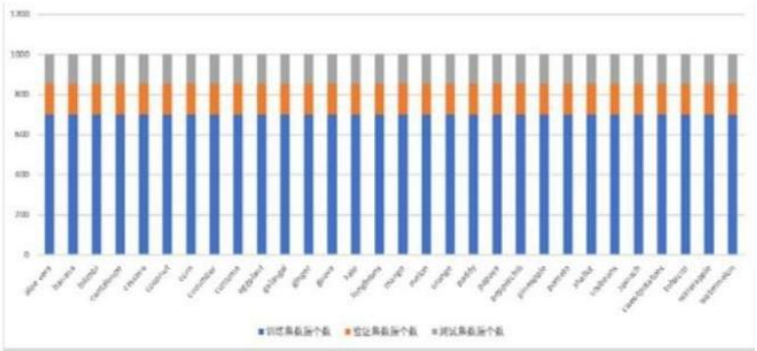

Figure 3: Image counts per category

We train the ResNet-18 model on the training set using cross-entropy loss for 10 epochs, transferring data to GPU for acceleration. During training, a scheduler combined with the Adam optimizer dynamically adjusts the learning rate: step_size=5, gamma=0.1, i.e., after every 5 epochs the learning rate is multiplied by 0.1. This scheduler helps control training, enabling faster convergence and better adaptation to the data. Training accuracy and loss on the training set are recorded.

To ensure optimal performance, we evaluate the model on the validation set using accuracy and loss, then fine-tune the parameters accordingly. The training and validation process is shown in Table 1:

Table 1: Training and validation process

Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy
1	0.8371	78.61%	0.8655	79.60%
2	0.276	90.89%	0.9122	81.78%
3	0.4009	87.68%	0.6691	85.29%
4	0.2585	91.79%	0.6161	84.27%
5	0.1726	94.30%	0.3794	95.40%
6	0.0387	98.77%	0.0831	97.29%
7	0.0139	99.59%	0.0934	97.82%
8	0.0127	99.77%	0.1015	97.71%
9	0.0035	99.91%	0.1087	97.67%
10	0.0018	99.95%	0.1219	97.87%
training_time: 2096-580204515457s				

training time: 2096.580204515457s.

Finally, the trained model's performance is assessed on the test set.

3.5 Evaluation Metrics for Experimental Results

This paper evaluates and compares the performance of different models using technical metrics such as accuracy and precision. Accuracy: Accuracy is the ratio of the number of samples correctly predicted by the classifier to the total number of samples. Its formula is:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{6}$$

where TP, true positive; TN, true negative; FP, false positive; and FN, false negative.

Precision: Precision is the proportion of true positives among the samples predicted as positive by the classifier. Its formula is:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{7}$$

Recall: Recall is the proportion of samples actually belonging to the positive class that are correctly predicted as positive by the classifier. Its formula is:

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{8}$$

F1-score: The F1 score is the harmonic mean of precision and recall; its formula is:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recal}} \tag{9}$$

4. CONCLUSION

As shown in Table 2, the ResNet18 model achieved an accuracy of 98.22%, precision of 98.229%, recall of 98.222%, and F1-score of 98.212% in this plant image classification experiment, with a total runtime of 2081.6057 s. These metrics indicate that the model demonstrates high performance on the plant image classification task, executing the classification accurately and reliably.

Convolutional neural network algorithms excel at image feature extraction and classification because they can automatically learn features within images and perform classification through multilayer networks. However, training and tuning convolutional neural networks require more time and computational resources. To improve training efficiency, techniques such as parallel computing and distributed training can be employed. Additionally, pretrained models can be leveraged to accelerate network convergence.

Table 2: Image classification results of the ResNet18 model

Model	Accuracy	Precision	Recall	F1	Total time(s)
Rensnet18	0.9822	0.98229	0.98222	0.98212	2081.6057

REFERENCES

[1] Chen, Yinda, et al. "Generative text-guided 3d vision-language pretraining for unified medical image segmentation." arXiv preprint arXiv:2306.04811 (2023).

[2] Ding, C.; Wu, C. Self-Supervised Learning for Biomedical Signal Processing: A Systematic Review on ECG and PPG Signals. medRxiv 2024.

[3] Han, X., & Dou, X. (2025). User recommendation method integrating hierarchical graph attention network with multimodal knowledge graph. Frontiers in Neurorobotics, 19, 1587973.

[4] Hu, Xiao. "GenPlayAds: Procedural Playable 3D Ad Creation via Generative Model." (2025).

[5] Hu, Xiao. "Low-Cost 3D Authoring via Guided Diffusion in GUI-Driven Pipeline." (2025).

[6] Li, Huaxu, et al. "Enhancing Intelligent Recruitment With Generative Pretrained Transformer and Hierarchical Graph Neural Networks: Optimizing Resume-Job Matching With Deep Learning and Graph-Based Modeling." Journal of Organizational and End User Computing (JOEUC) 37.1 (2025): 1-24.

[7] Li, X., Wang, X., & Lin, Y. (2025). Graph Neural Network Enhanced Sequential Recommendation Method for Cross-Platform Ad Campaign. arXiv preprint arXiv:2507.08959.

[8] Liu, Jun, et al. "Toward adaptive large language models structured pruning via hybrid-grained weight importance assessment." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 39. No. 18. 2025.

[9] Miao, Junfeng, et al. "Secure and Efficient Authentication Protocol for Supply Chain Systems in Artificial Intelligence-based Internet of Things." IEEE Internet of Things Journal (2025).

[10] Peng, Q., Planche, B., Gao, Z., Zheng, M., Choudhuri, A., Chen, T., Chen, C. and Wu, Z., 3D Vision-Language Gaussian Splatting. In The Thirteenth International Conference on Learning Representations.

[11] Pinyoanuntapong, Ekkasit, et al. "Gaitsada: Self-aligned domain adaptation for mmwave gait recognition." 2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS). IEEE, 2023.

[12] Q. Tian, D. Zou, Y. Han and X. Li, "A Business Intelligence Innovative Approach to Ad Recall: Cross-Attention Multi-Task Learning for Digital Advertising," 2025 IEEE 6th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), Shenzhen, China, 2025, pp. 1249-1253, doi: 10.1109/AINIT65432.2025.11035473.

[13] Su, Tian, et al. "Anomaly Detection and Risk Early Warning System for Financial Time Series Based on the WaveLST-Trans Model." (2025).

[14] Tan, C., Gao, F., Song, C., Xu, M., Li, Y., & Ma, H. (2024). Highly Reliable CI-JSO based Densely Connected Convolutional Networks Using Transfer Learning for Fault Diagnosis.