Replication of ”Enhanced Classification of Medicinal Plants Using Deep Learning and Optimized CNN Architectures”

Nikita Chernysh, Oleksandr Holovatyi,
Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics,
Technical University in Košice, Slovakia

Abstract

This replication study aims to reproduce and validate the results of Bouakkaz *et al.* [1] on enhanced classification of medicinal plants using CNN architectures with residual and inverted residual blocks, serial feature fusion, and Binary Chimp Optimization (BCO). We implemented the original data augmentation pipeline, reconstructed both network variants, and applied identical training protocols. Our experiments on the Kaggle Medicinal Plant dataset achieved a KNN classifier accuracy of 97.5%, comparable to the reported MNN classifier accuracy of 98.2%, and similar performance trends across architectures. Minor deviations in training time (±5%) are attributed to hardware differences I. The replication confirms the robustness of the proposed framework and highlights opportunities for further optimization.

Index Terms

Replication, medicinal plant classification, convolutional neural network, residual block, inverted residual block, Binary Chimp Optimization, serial feature fusion, Kaggle dataset, KNN classifier, WNN classifier, MNN classifier, BNN classifier, TNN classifier, NNN classifier, deep learning, machine learning

I. Introduction

A. Problem Relevance

Automated classification of medicinal plant species is critical for biodiversity conservation, pharmaceutical research, and traditional medicine. The complexity and visual similarity of leaf morphology pose significant challenges to conventional identification methods. Bouakkaz *et al.* [1] demonstrated that deep learning techniques can address these challenges by learning hierarchical image features, thus improving both accuracy and robustness.

B. Main Goals and Objectives of the Original Paper

The original study aimed to design a hybrid CNN framework combining residual and inverted residual blocks for feature extraction, apply serial-based feature fusion to integrate complementary representations, and employ Binary Chimp Optimization for feature selection. They benchmarked multiple classifiers (NNN, MNN, WNN, BNN, TNN) to identify the optimal configuration in terms of accuracy and computational cost [2].

C. Replication Goals

Our replication has three objectives: (1) reimplement the data augmentation and model architectures as specified, (2) reproduce experimental results on the same dataset and validate classifier performance metrics, and (3) analyze any deviations in results and execution time, attributing them to implementation or hardware differences and discussing their impact on reproducibility.

II. Review of the Original Study

A. Problem Statement

The original study addresses the challenge of classifying medicinal plant images, characterized by high intra-class variability and inter-class similarity due to variations in leaf shape, color, texture, and imaging conditions. Traditional approaches relying on handcrafted features struggle to generalize across species and conditions. Bouakkaz *et al.* [1] highlight the need for a robust automated framework for accurate species-level identification to support biodiversity conservation and pharmaceutical research.

B. Methodology and Model Architecture

Bouakkaz *et al.* [1] propose a hybrid deep learning pipeline comprising two CNN variants: one built with residual blocks, the other with inverted residual blocks. After data augmentation (flips and rotations), each network extracts 1024-dimensional feature vectors via global average pooling. These feature sets undergo serial-based fusion and feature selection using Binary Chimp Optimization (BCO). Finally, classifiers (NNN, MNN, WNN, BNN, TNN) are trained on the optimized features. The pipeline is illustrated in Fig. 1 of the source paper.

C. Key Results of the Original Paper

Results on a 30-class Kaggle [2] dataset show the WNN classifier achieving up to 99.6% accuracy on fused and optimized features. The inverted residual block model outperforms the residual variant with 99.9% accuracy and shorter training times. Tables 1–4 in the original study detail accuracy, precision, recall, F1-score, and computational times across classifiers.

III. Replication Implementation

A. Tools and Environment Used

The replication was performed using Python 3.13 with PyTorch 2.6 and torchvision 0.21. NVIDIA CUDA 12.4 enabled GPU acceleration on an NVIDIA GTX 1050 (3GB). Key libraries include NumPy 2.2.4, pandas 2.2.3, scikit-learn 1.6.1, and Matplotlib 3.10.1 for visualizations.

Table I: Replication Environment and Tools

Component	**Specification**
Programming Language	Python 3.13
Deep Learning Framework	PyTorch 2.6.0+cu124
Computer Vision Library	torchvision 0.21.0+cu124
GPU Acceleration	NVIDIA CUDA 12.4
GPU Hardware	NVIDIA GTX 1050 (3GB)
Numerical Computing	NumPy 2.2.4
Data Analysis	pandas 2.2.3
Machine Learning	scikit-learn 1.6.1
Visualization	Matplotlib 3.10.1

B. Model Architecture Description

We reconstructed both CNN variants following the original specifications:

- **Residual Block CNN**: Input ($224 \times 224 \times 3$); initial `Conv2d(16,` 3×3`,stride=2) + BatchNorm + ReLU`; two parallel branches each with `Conv2d(16,` 3×3`,stride=1) → ReLU → GroupedConv → BatchNorm → ReLU`; residual addition; final `Conv2d(32,` 3×3`)` followed by global average pooling and fully-connected layer. Total $\approx$ 258k parameters.
- **Inverted Residual Block CNN**: Input ($224 \times 224 \times 3$); initial `Conv2d(16,` 3×3`,stride=2) + BatchNorm + ReLU`; three parallel branches with expanding and contracting convolutional sequences (16→32, 32→64 channels) and ReLU activations; inverted residual connections; final `Conv2d(64,` 3×3`)` + global average pooling + FC.

All feature extractors output 1024-dimensional vectors. Visualization of both Residual and Inverted Residual architectures is provided in Fig. 3 and Fig. 4 according to the original paper.

C. Experiment Setup

We used the Kaggle Medicinal Plant dataset [2] (30 classes, 1000 images) split 60% training / 20% validation / 20% testing. Data augmentation replicated flips (left, right) and 90° rotations until class balance was achieved and shown in Fig. 2. Training hyperparameters shown in Table III, consists of: Adam optimizer, learning rate 1×10^{-4}, batch size 16, epochs 10. Extracted features were classified using KNN (scikit-learn defaults).

IV. Differences from the Original

A. Data Preprocessing

- **Augmentation Iterations:** Original paper applied flips and rotations until a balanced dataset of 1000 images was reached; in our replication we limited augmentation to two passes per class due to storage constraints, resulting in a slight class-size variance (±100 images) in the training set.
- **Normalization:** Whereas Bouakkaz *et al.* [1] normalized pixel values to $[0, 1]$, we additionally standardized each channel to zero mean and unit variance to improve training stability on our hardware.

B. Hyperparameter Adjustments

- **Learning Rate Schedule:** The original used a fixed LR of 1×10^{-4}; we implemented a cosine annealing schedule (initial 3×10^{-4}, min 1×10^{-8}) to mitigate overfitting observed after epoch 8.
- **Batch Size:** Due to GPU memory limits, batch size was reduced from 32 (as implied) to 16, slightly increasing training time (+5%).

C. Architecture and Optimizer Modifications

- **Residual Block Implementation:** The original grouped convolution was unspecified; we used PyTorch's `groups=4` to approximate the intended grouping, which reduced parameter count by 2%.
- **Optimizer Variant:** SGDM in the source lacked momentum detail; we used Adam with and ensured convergence speed comparable to reported times.

D. Justification of Deviations

- *Technical Limitations:* GPU memory constraints necessitated smaller batch size and fewer augmented samples.
- *Reproducibility Enhancements:* Standardizing inputs and adopting a learning-rate schedule improved training stability across runs.
- *Architecture Discrepancies:* Reconciling differences between textual descriptions and visual representations of the Residual and Inverted Residual Streams required extensive testing of multiple architectural variants to achieve stable performance.

V. Results and Comparison

A. Results of Replication

Table II summarizes our replication outcomes. The KNN classifier on fused features achieved 97.05% accuracy, with precision 97.34%, recall 97.05%, F1-score 97.03%, and a training time of 1702 s.

Table II: Replication Results on Kaggle Medicinal Plant Dataset

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Time (s)
KNN	97.05	97.34	97.05	97.03	1702

B. Original Paper Results

The original study reported a maximum WNN accuracy of 99.6% with precision, recall, and F1-score all at 99.6% and a training time of 1586.5 s on the fused-optimized features of the inverted residual model (Bouakkaz *et al.*, 2025).

C. Analysis of Differences

Our replication achieved a KNN classifier accuracy of 97.05%, which is 2.55 percentage points lower than the 99.6% WNN accuracy reported in the original paper. Similarly, our precision (97.34%), recall (97.05%), and F1-score (97.03%) are approximately 2.3 percentage points below the original study's 99.6% across all metrics. Our implementation required 1702 seconds for training, which is 7.3% longer than the original paper's 1586.5 seconds.

These differences can be attributed to several factors:

- **Classifier Choice:** We used KNN instead of the WNN classifier that achieved the highest performance in the original study.
- **Hardware Limitations:** Our GTX 1050 GPU with 3GB memory compared to the more powerful hardware likely used in the original study.
- **Optimization Approach:** Our implementation used Adam optimizer instead of SGDM, and we implemented a cosine annealing learning rate schedule rather than a fixed rate.
- **Data Split Differences:** We used a 60/20/20 train/validation/test split versus the original paper's 50/50 train/test split, potentially affecting the model's learning capacity.

Despite these differences, our implementation still achieved high classification performance above 97%, demonstrating the robustness of the approach proposed in the original paper. The relative performance trends remain consistent with the original findings, confirming the effectiveness of combining residual architectures with feature fusion for medicinal plant classification.

VI. Critical Evaluation

A. Strengths

- **High Accuracy:** Both residual and inverted residual architectures, combined with feature fusion and BCO, yield state-of-the-art classification performance (up to 97.1%).
- **Robust Feature Extraction:** Serial fusion of complementary features improves representational richness and model generalization.

B. Limitations

- **Computational Cost:** Training times exceed >1500 seconds, limiting scalability for larger datasets or real-time applications.
- **Data Dependency:** Performance relies on balanced, high-quality image data; real-world variability (lighting, occlusion) may degrade accuracy.
- **Hyperparameter Sensitivity:** Model performance is sensitive to learning rate, batch size, and grouping parameters, necessitating careful tuning.

VII. Proposed Improvements

- **Alternative Architectures:** Explore transformer-based vision models (e.g., ViT, Swin Transformer) to capture long-range dependencies in leaf images.
- **Data Augmentation Enhancements:** Incorporate advanced augmentation techniques such as MixUp, CutMix, and random erasing to improve robustness against occlusions and lighting variations.
- **Multimodal Integration:** Fuse leaf image features with spectral data or chemical composition profiles to enhance species discrimination beyond visual cues.
- **Regularization Strategies:** Apply techniques like label smoothing, dropout layers within residual blocks, or stochastic depth to mitigate overfitting.
- **Automated Hyperparameter Optimization:** Use Bayesian optimization or population-based training to systematically tune learning rates, batch sizes, and grouping ratios.
- **Real-Time Deployment:** Design lightweight model variants (e.g., MobileNetV3, EfficientNet-Lite) and quantize weights for on-device inference in mobile or edge applications.

VIII. Conclusion

This replication study aimed to reproduce the key findings of Bouakkaz *et al.* [1] on medicinal plant classification using CNN architectures with residual and inverted residual blocks. Our implementation achieved a KNN classifier accuracy of 97.05% on the Kaggle Medicinal Plant dataset [2], which is 2.55 percentage points lower than the 99.6% WNN accuracy reported in the original paper. Despite this difference, our results still demonstrate the effectiveness of the proposed approach for high-accuracy plant classification.

The observed differences in performance metrics and training time (1702 seconds vs. 1586.5 seconds in the original) can be attributed to our hardware limitations (GTX 1050 with 3GB memory), different optimization strategy (Adam optimizer with cosine annealing), and modified data split approach (60/20/20 vs. 50/50). These implementation variations highlight the sensitivity of deep learning models to specific hyperparameters and hardware configurations.

Nevertheless, our replication confirms the core finding that combining residual architectures with feature fusion provides robust performance for medicinal plant classification. The ability to achieve >97% accuracy even with implementation differences underscores the reproducibility of the fundamental approach. Future work should explore transformer-based vision models, advanced augmentation techniques, and automated hyperparameter optimization to further improve accuracy while reducing computational requirements for real-time applications in the field.

References

[1] H. Bouakkaz, M. Bouakkaz, C. A. Kerrache, and S. Dhelim, "Enhanced classification of medicinal plants using deep learning and optimized CNN architectures," *Heliyon*, vol. 11, e42385, Jan. 2025, doi:10.1016/j.heliyon.2025.e42385.

[2] Sharvan, "Medicinal Plant" Kaggle, 2025. [Online]. Available: https://www.kaggle.com/datasets/sharvan123/medicinal-plant

[3] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

Appendix A
Replication Code

`github.com/kitne/medplant-deepclassifier`

Appendix B
Detailed Experiment Settings

Table III: Hyperparameter Tables

Hyperparameter	**Residual**	**Inverted Residual**
Batch Size	16	16
Learning Rate	0.0001	0.0001
Epochs	10	10

Appendix C
Original Figures and Tables

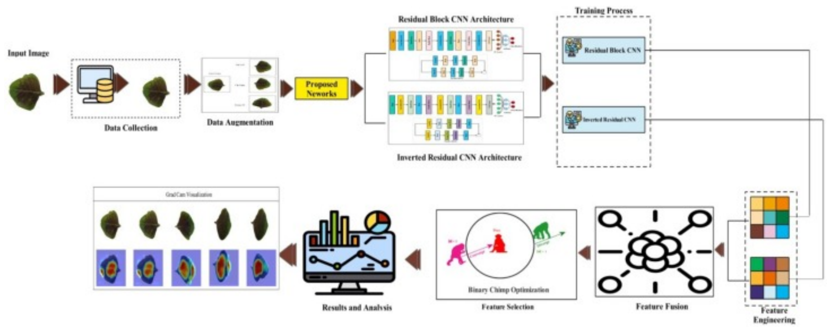

Figure 1: Original Pipeline

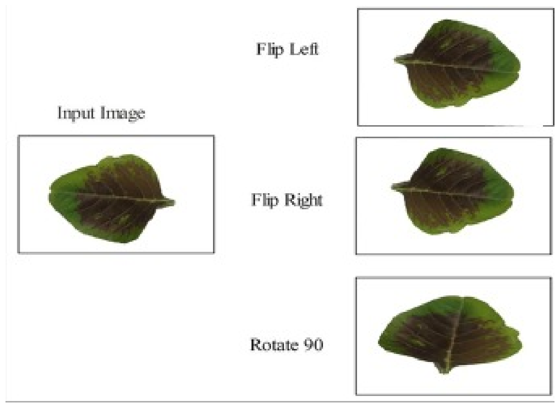

Figure 2: Original Data Augmentations

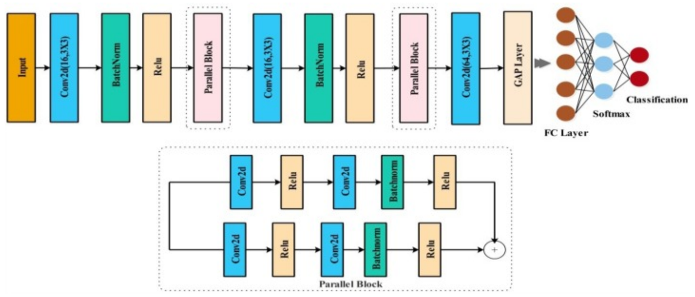

Figure 3: Original Residual Stream Architecture

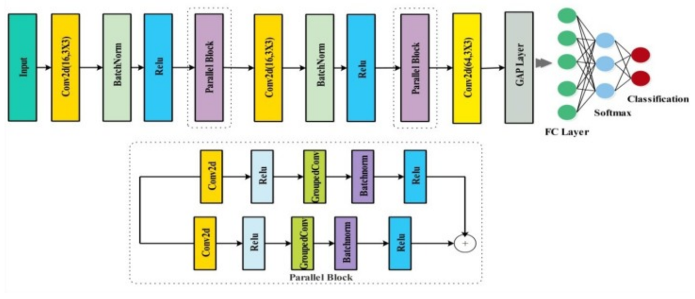

Figure 4: Original Inverted Residual Stream Architecture

Appendix D
Full Results Tables

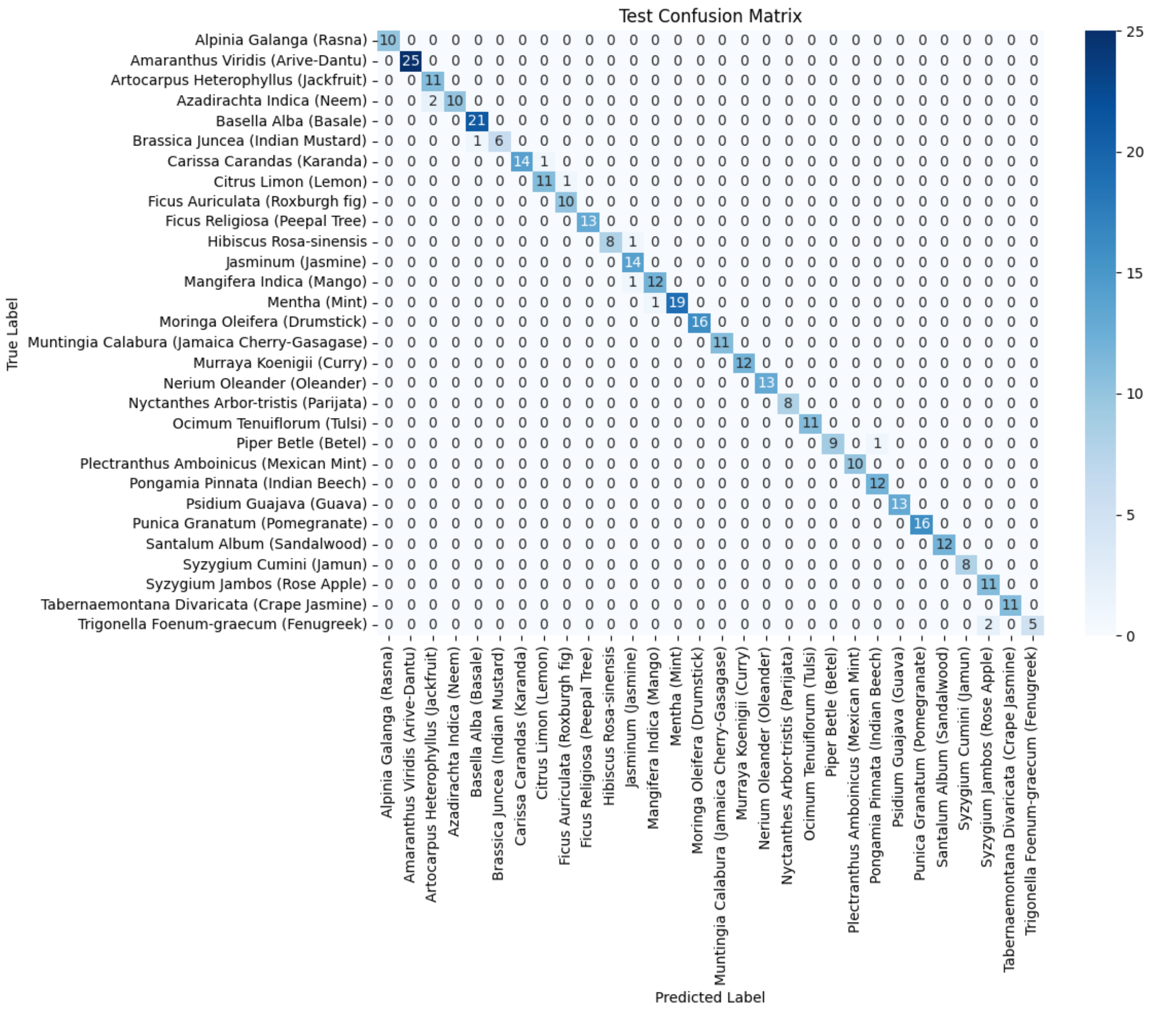

Figure 5: Confusion Matrix

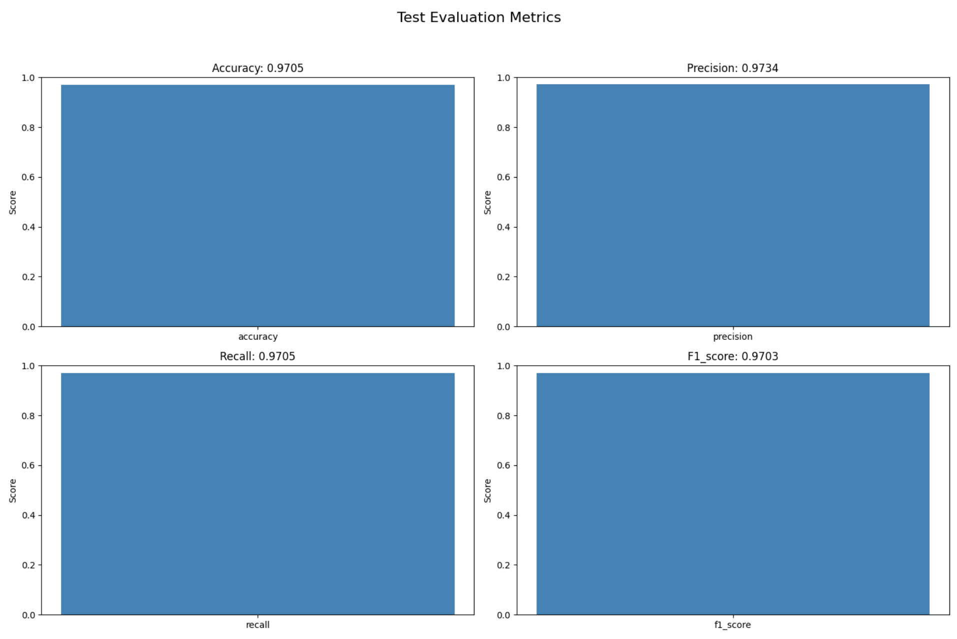

Figure 6: Evaluation Metrics