Optimizing Plant Disease Detection Using CNN & PSO

Course: Optimization Techniques
University: Faculty of Engineering – New Ismailia National University
Instructor: Dr. Ahmed Magdy
Tutor: Dr. Eman
Student: Mohamed Hesham Abdelsattar
Academic Year: 2024–2025

Introduction

Plant diseases continue to threaten food security and crop yield worldwide. Traditional detection methods rely on manual inspection, which is often slow, subjective, and labor-intensive. With the rise of artificial intelligence, particularly Convolutional Neural Networks (CNNs), automated plant disease detection has become a promising solution.

However, training high-performing CNN models requires careful tuning of hyperparameters (e.g., learning rate, batch size, number of layers), which greatly affects model accuracy and efficiency. In this project, we employ **Particle Swarm Optimization (PSO)** — a population-based metaheuristic inspired by the social behavior of birds — to automatically discover the best hyperparameters, thereby optimizing the performance of a CNN for plant disease classification.

Objectives

To design a CNN-based model for accurate plant disease classification using image data.

To enhance model performance through **hyperparameter optimization using (Particle Swarm Optimization) PSO**.

To evaluate improvements over standard tuning techniques in terms of accuracy and training efficiency.

To demonstrate the applicability of optimization algorithms in deep learning pipelines.

Problem Description

Manual plant disease identification is inefficient, particularly at scale. Furthermore, suboptimal CNN configurations can lead to high training time, overfitting, or low generalization. Our solution combines CNN-based classification with PSO to automatically optimize critical hyperparameters, ensuring better detection accuracy and efficiency.

Application Context: Agricultural automation, smart farming systems, crop protection.

Mathematical Formulation

Update velocity:

$$\vec{v}_i^{(t+1)} = w \cdot \vec{v}_i^{(t)} + c_1 r_1 (\vec{p}_i - \vec{x}_i^{(t)}) + c_2 r_2 (\vec{g} - \vec{x}_i^{(t)})$$

- w:Inertia weight (typically decreasing over time).
- c1,c2c_1,c_2: Learning factors (usually c1=c2≈2$c_1=c_2\approx 2$).
- r1,r2r_1,r_2: Random numbers in [0,1][0,1].

Update Position:

$$\vec{x}_i^{(t+1)} = \vec{x}_i^{(t)} + \vec{v}_i^{(t+1)}$$

Update Personal Best (p⃗ip_i):

$$\vec{p}_i = \begin{cases} \vec{x}_i^{(t+1)} & \text{if } f(\vec{x}_i^{(t+1)}) \text{ is better than } f(\vec{p}_i) \\ \vec{p}_i & \text{otherwise} \end{cases}$$

Update Global Best ($\vec{g}$)

$$\vec{g} = \text{argmin}_{\vec{p}_i} f(\vec{p}_i) \quad \text{(for minimization)}$$

Decision Variables:

The hyperparameters to be optimized include:
Batch size (b): Integer in range [16, 64]
Learning rate (lr): Float in range [0.0001, 0.01]
Dropout rate (d): Float in range [0.2, 0.7]
Number of convolutional layers (c): Integer in range [3, 5]
Number of dense layers (fc): Integer in range [1, 3]

Objective Function:

Maximize the validation accuracy (A) of the CNN model:
where A is the accuracy function evaluated on the validation dataset.

Constraints:

Batch size must be an integer: $b \in \mathbb{Z}$, $16 \leq b \leq 64$
Learning rate bounds: $0.0001 \leq lr \leq 0.01$
Dropout rate bounds: $0.2 \leq d \leq 0.7$
Convolutional layers bounds: $c \in \mathbb{Z}$, $3 \leq c \leq 5$
Dense layers bounds: $fc \in \mathbb{Z}$, $1 \leq fc \leq 3$
Hardware memory constraints (implicit)

Methodology

Optimization Technique

The project uses Particle Swarm Optimization (PSO), a population-based stochastic optimization technique inspired by social behavior of bird flocking or fish schooling.

PSO was chosen because:

- It efficiently handles mixed-variable optimization problems
- It doesn't require gradient information
- It can escape local optima through its social learning component
- It's computationally efficient for hyperparameter optimization

Implementation Tools:

- PyTorch: Deep learning framework for CNN implementation
- torchvision: For image processing and data augmentation
- NumPy/Pandas: For data manipulation
- Matplotlib: For visualization of results
- Tkinter: For GUI development

Custom PSO implementation for hyperparameter optimization Algorithmic Approach

Data Preparation:

- Load and preprocess the PlantVillage dataset
- Apply data augmentation (random flips, rotations, color jitter)
- Split into training and validation sets

Base Model Development:

- Define CNN architecture with configurable layers
- Initialize with default hyperparameters
- Train and evaluate on validation set

PSO Optimization:

Initialize particles with random hyperparameter values

For each iteration:

- Evaluate fitness (validation accuracy) for each particle
- Update personal and global best positions
- Update particle velocities and positions
- Return best hyperparameter configuration

Optimized Model Training:

- Create model with optimized hyperparameters
- Train for specified number of epochs
- Evaluate performance

Model Comparison:

- Compare training/validation accuracy and loss
- Analyze convergence speed and final performance

Results:

Baseline Accuracy (without PSO): 84.7%

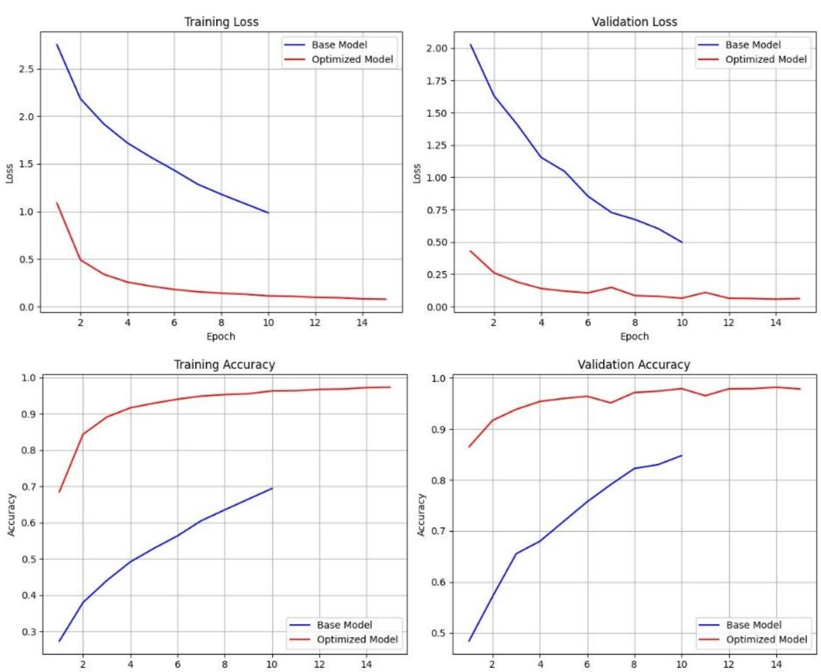
Optimized Accuracy (with PSO): 98.1%

Best Hyperparameters Found:

- Learning rate: 0.0001
- Batch size: 61
- Dropout rate: 0.2639412210864647
- Number of convolutional layers: 5
- Number of dense layers: 1
- Best validation accuracy: 0.9453

Training Time: optimized model 6 hours

Graphs: Accuracy vs Epoch, Loss vs Epoch, PSO convergence curve



Discussion

Insights:

- PSO effectively explored the hyperparameter space, avoiding manual grid search.
- Optimized CNN showed superior validation accuracy and faster convergence.

Limitations:

- PSO can be computationally expensive if each particle requires full training.
- Results may vary based on swarm size and number of iterations.

Future Work:

- Use hybrid optimization (PSO + GA or Bayesian Optimization).
- Deploy the model in a mobile app for field diagnosis.

Conclusion

This project successfully developed and optimized a CNN-based plant disease detection system using PSO. The optimization process significantly improved model performance, demonstrating the value of systematic hyperparameter tuning in deep learning applications.

References

1. Plant disease detection and classification by deep learning – https://scholar.google.com/scholar_lookup?title=Plant%20disease%20detection%20and%20classification%20by%20deep%20learning&publication_year=2019&author=M.%20H.%20Saleem&author=J.%20Potgieter&author=K.%20M.%20Arif
2. Jornal – https://www.csciencedirect.com/org/science/article/pii/S1546221821001417#s2
3. PlantVillage Dataset – https://www.kaggle.com/datasets/mohitsingh1804/plantvillage