Plant Disease Classification Using Deep Learning

Faculty of Sciences of Sfax

Department of Computer Science and Communications

Academic Year: 2024-2025

Submitted by:
Bettaieb Selma

Supervised by:
Dr. Emna FENDRI
Dr. Sahar DAMMAK

Course: Deep Learning

Track: Computer Science Engineering (I2)

github:https://github.com/Ibrahimghali/plant-disease-recognition/tree/Selma

Table of Contents

- Key Findings and Recommendations
- Best Practice Recommendations

Comparative Analysis: Data Augmentation Impact

- Model Performance Visualization
- Training Dynamics Analysis
- The Critical Role of Data Augmentation and Regularization
- Analysis of Non-Augmented Model Behavior
- The Protective Role of Strong Regularization
- Data Augmentation as a Superior Solution
- Key Lessons for Small Datasets

The Pitfalls of Accuracy Metrics and Challenges of Custom CNN Architecture

- Why Accuracy Alone Is Misleading
- False Confidence Analysis
- Challenges in Custom Architecture Development
- Why Transfer Learning Offers a Better Approach

General Introduction

This project serves as a practical exploration of fundamental deep learning concepts through plant disease classification . we investigate how different methodologies affect model performance, particularly when working with limited domain-specific data. Through this comparative analysis, we seek to demonstrate the practical implications of theoretical concepts learned in deep learning and identify optimal strategies for similar computer vision tasks.

Dataset Overview

- Source: Plant Disease Recognition Dataset from Kaggle
- Total Images: 1,530 across three classes (Healthy, Powdery, Rust)
- Pre-split into training, validation, and test sets
- Image characteristics: RGB format, varying resolutions
- No missing data or corrupted images identified

Project Objectives

1. Compare and evaluate transfer learning strategies for plant disease classification
2. Conduct comparative analysis between VGG16 and ResNet50 architectures to determine optimal model selection for limited datasets
3. Assess the impact of data augmentation on model performance
4. Analyze the limitations of accuracy metrics in deep learning
5. Provide practical recommendations for similar computer vision tasks

1. Transfer Learning Strategies: VGG16 Implementation

Introduction

This study explores the effectiveness of different transfer learning strategies using VGG16 on a plant disease classification dataset, comparing three distinct approaches to layer freezing.

Experimental Setup

Strategy 1	Strategy 2	Strategy 3
Full Training	Partial Freezing	Full Freezing
All layers trainable	Last 4 conv layers trainable + classifier	Only classifier trainable

Common Configuration:

- Dataset: 1,530 images (3 classes)
- Base Model: VGG16 (pretrained on ImageNet)
- Dropout: (0.6, 0.5) in classifier
- Data Augmentation: rotation, shifts, flips, zoom

Experimental Results and Analysis

Performance Metrics Comparison

Metric	Frozen (S3)	Partial (S2)	Full (S1)
Test Accuracy	87.33%	98.00%	96.67%
Training Time	Full 50 epochs	Full 50 epochs	Stopped at epoch 29
Best Validation Accuracy	93.33%	100%	98.33%

Training Dynamics Analysis

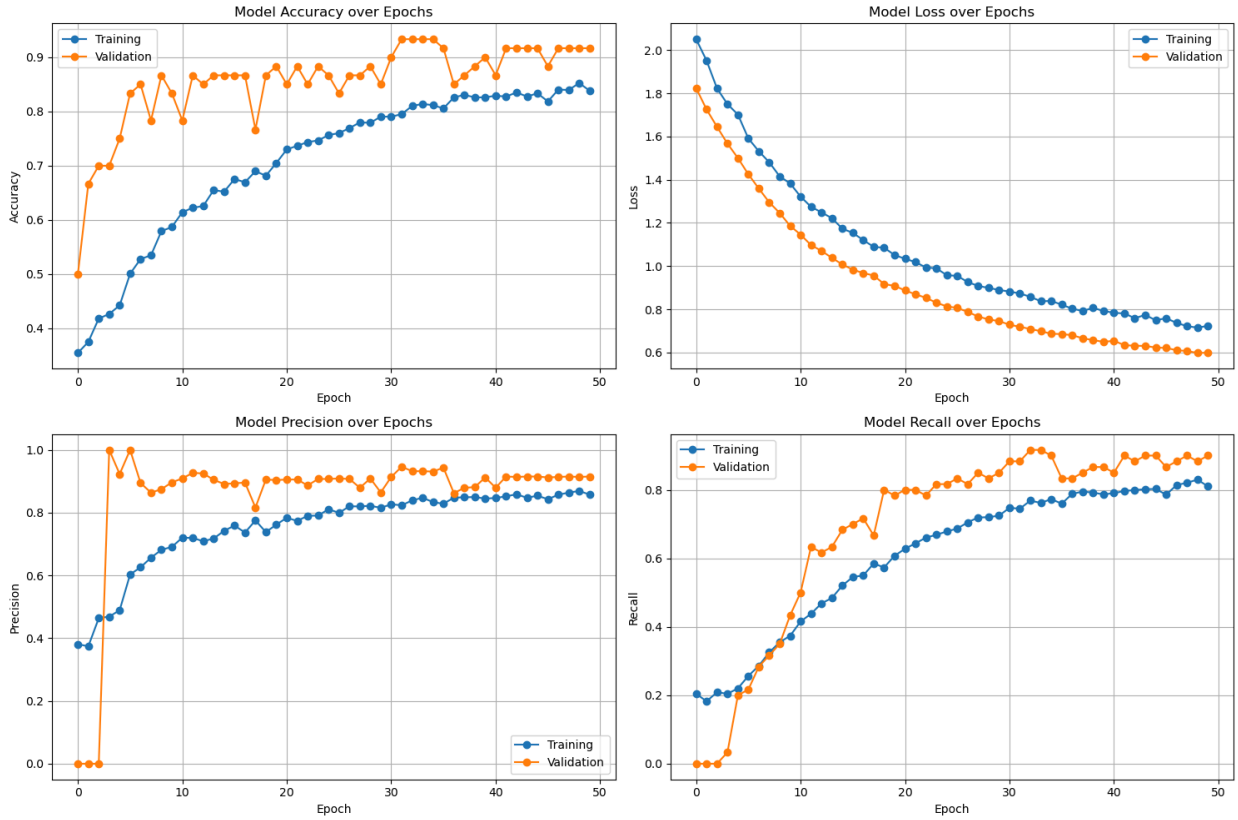


Figure 1: Training and validation metrics over epochs for full freezing _strategy3

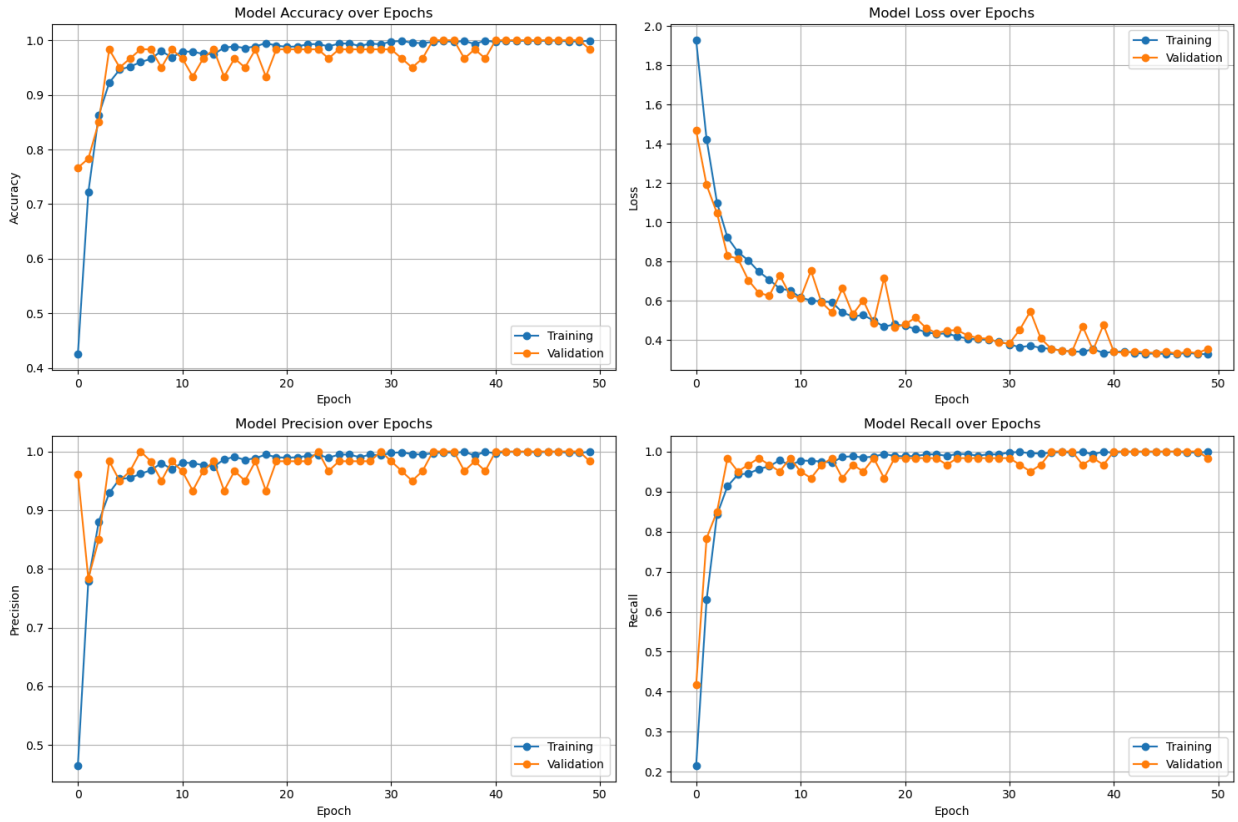


Figure 2: Training and validation metrics over epochs for partial freezing _strategy2

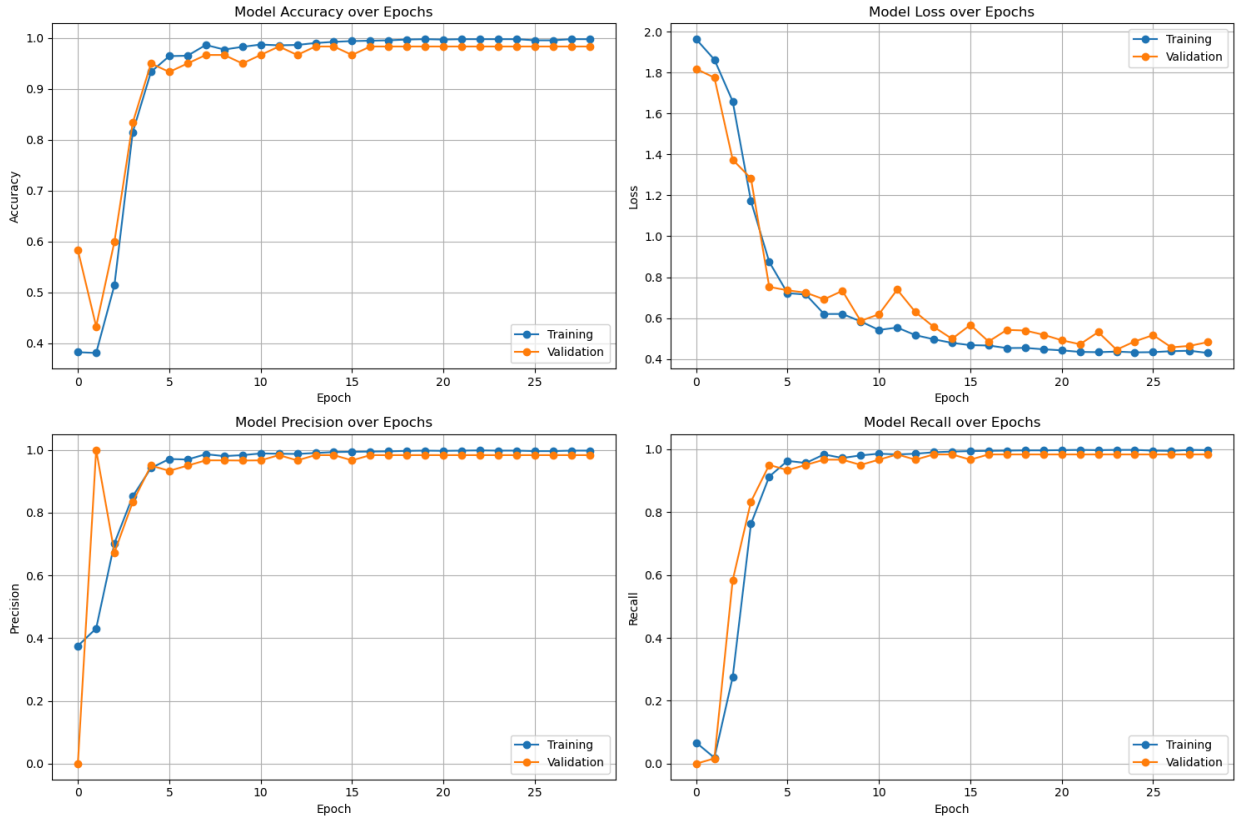


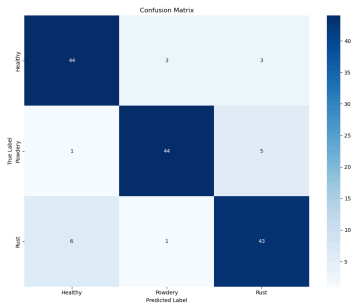
Figure 3: Training and validation metrics over epochs for full training _strategy1

The training curves reveal distinct patterns for each strategy:

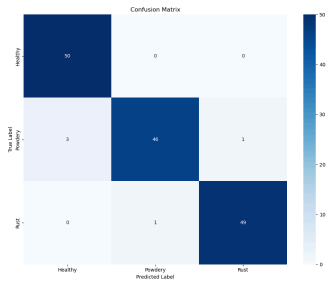
1. ***Frozen Model (Strategy 3):***
   - Steady but slower learning curve
   - Training accuracy plateaus around 85%
   - Most stable validation metrics
2. ***Partial Training (Strategy 2):***
   - Optimal convergence pattern
   - Smooth learning curve
   - Training/validation curves remain close
3. ***Full Training (Strategy 1):***
   - Rapid initial convergence
   - Validation accuracy becomes volatile
   - Clear gap between training/validation curves

- No signs of overfitting but clear underfitting
- Completed all epochs without early stopping
- Required early stopping intervention

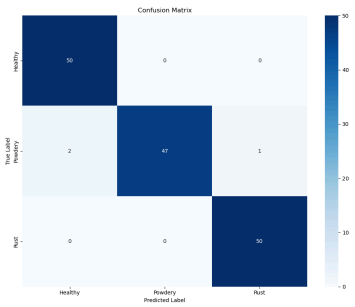
Confusion Matrix Analysis



ull freezing _strategy3



full training _strategy1



partial freezing _strategy2

1. **Frozen Model Performance:**

Healthy: 44/50 (88.0%)
Powdery: 44/50 (88.0%)
Rust: 43/50 (86.0%)

- Balanced but suboptimal performance across classes
- Consistent error distribution

2. **Partial Training Results:**

Healthy: 50/50 (100%)
Powdery: 47/50 (94.0%)
Rust: 50/50 (100%)

- Near-perfect class discrimination
- Minimal misclassifications
- Well-balanced across classes

3. **Full Training Outcomes:**

Healthy: 50/50 (100%)
Powdery: 46/50 (92.0%)
Rust: 49/50 (98.0%)

- Strong overall performance
- Slightly more misclassifications than partial
- Signs of potential overfitting

Per-Class F1-Scores

Class	Frozen	Partial	Full
Healthy	0.871	0.980	0.971
Powdery	0.898	0.969	0.948
Rust	0.851	0.990	0.980

The metrics clearly demonstrate the ***superiority of partial freezing (Strategy 2)*** in our specific case, achieving the best balance between feature reuse and adaptation.

The confusion matrices particularly highlight how partial freezing maintained high performance across all classes while avoiding the overfitting tendencies of full training and the underfitting limitations of full freezing.

Methodology and Theoretical Framework

Implemented Strategies:

1. **Full Model Training (Strategy 1):**
   - All layers trainable
   - Theoretically highest risk due to small dataset
   - Required careful regularization:
     - L2 regularization (0.001)
     - Dropout layers (0.5, 0.6)
     - Low learning rate (5e-5)
2. **Partial Layer Freezing (Strategy 2):**
   - Last 4 layers unfrozen
   - Theoretically optimal for our dataset size
   - Same regularization as Strategy 1
3. **Full Layer Freezing (Strategy 3):**
   - All convolutional layers frozen
   - Theoretically safest but limited
   - Only classifier layers trained

Results Analysis with Theoretical Context

Fully Frozen VGG16 (Strategy 3):

- Test Accuracy: 87.33%
- Aligned with theoretical expectations:
  - Stable training curves
  - Limited but consistent performance
  - No overfitting, but clear underfitting
- Most suitable when:
  - Very small datasets
  - High domain similarity
  - Limited computational resources

Fully Trainable VGG16 (Strategy 1):

- Test Accuracy: 96.67%
- Theoretical risks manifested:
  - Clear overfitting signs despite regularization
  - Required early stopping (epoch 29)
  - Volatile validation metrics
- Demonstrates why theory recommends against full training for small datasets

Partially Frozen VGG16 (Strategy 2):

- Test Accuracy: 98.00%
- Validates theoretical recommendations:
  - Best balance of adaptation and stability
  - No early stopping needed

    - Consistent validation metrics
- Optimal strategy for our dataset characteristics

Key Insights from Theory and Practice

1. **Strategy Selection Validation:**
    - Our results confirm theoretical guidelines about dataset size influence
    - Partial freezing's superior performance aligns with transfer learning best practices
    - Full training's overfitting issues validate theoretical warnings
2. **Learning Rate Impact:**
    - Low learning rate (5e-5) proved crucial
    - Helped preserve ImageNet knowledge
    - Particularly important for Strategies 1 and 2
3. **Practical Trade-offs:**
    - Full Freezing: Safest but performance-limited
    - Partial Freezing: Best performance/stability balance
    - Full Training: Highest potential but requires intensive management

Recommendations Based on Combined Insights

1. **For Similar Small Datasets (<1000 images/class):**
    - Start with Strategy 2 (Partial Freezing)
    - Use strong regularization
    - Maintain low learning rates
2. **When Computational Resources are Limited:**
    - Strategy 3 provides reliable baseline
    - Consider Strategy 2 with more frozen layers
    - Avoid Strategy 1 unless necessary

3. **For Maximum Performance:**
    - Strategy 2 offers best balance
    - Implement data augmentation
    - Monitor validation metrics closely

This integrated analysis demonstrates how ***our experimental results both validate and provide practical context to transfer learning theory***, offering valuable insights for similar computer vision tasks.

2.Comparative Analysis: VGG16 vs ResNet50

Introduction

This section compares the performance of VGG16 and ResNet50 architectures on our plant disease classification task, providing insights into how different model architectures and transfer learning strategies affect performance on our specific dataset.

Model Architectures Overview

1. **VGG16 Architecture:**
    - 16 layers deep (13 convolutional, 3 fully connected)
    - Simple, sequential structure
    - Large number of parameters in fully connected layers
    - Regular pattern of conv-pooling layers

1. **ResNet50 Architecture:**
    - 50 layers with skip connections
    - Complex residual learning blocks
    - Identity mappings for deep feature propagation
    - Designed for very deep learning on large datasets

- Strong feature extraction for textures and patterns
- Optimized for object recognition at multiple scales

Experimental Setup

Both models were tested with identical configurations:

- Dataset: 1,530 images across three classes (Healthy, Powdery, Rust)
- Batch size: 16
- Learning rate: 5e-5
- Regularization techniques:
  - L2 regularization (λ=0.001)
  - Dropout layers (0.6, 0.5)
  - Early stopping (patience=5)
  - Learning rate reduction (factor=0.3)
- Data augmentation:
  - Rotation (±20°)
  - Width/height shifts (20%)
  - Horizontal flips
  - Brightness variation (±20%)
  - Zoom range (20%)
- Transfer learning strategy:
  - Partial unfreezing (last 4 layers)
  - ImageNet pre-trained weights

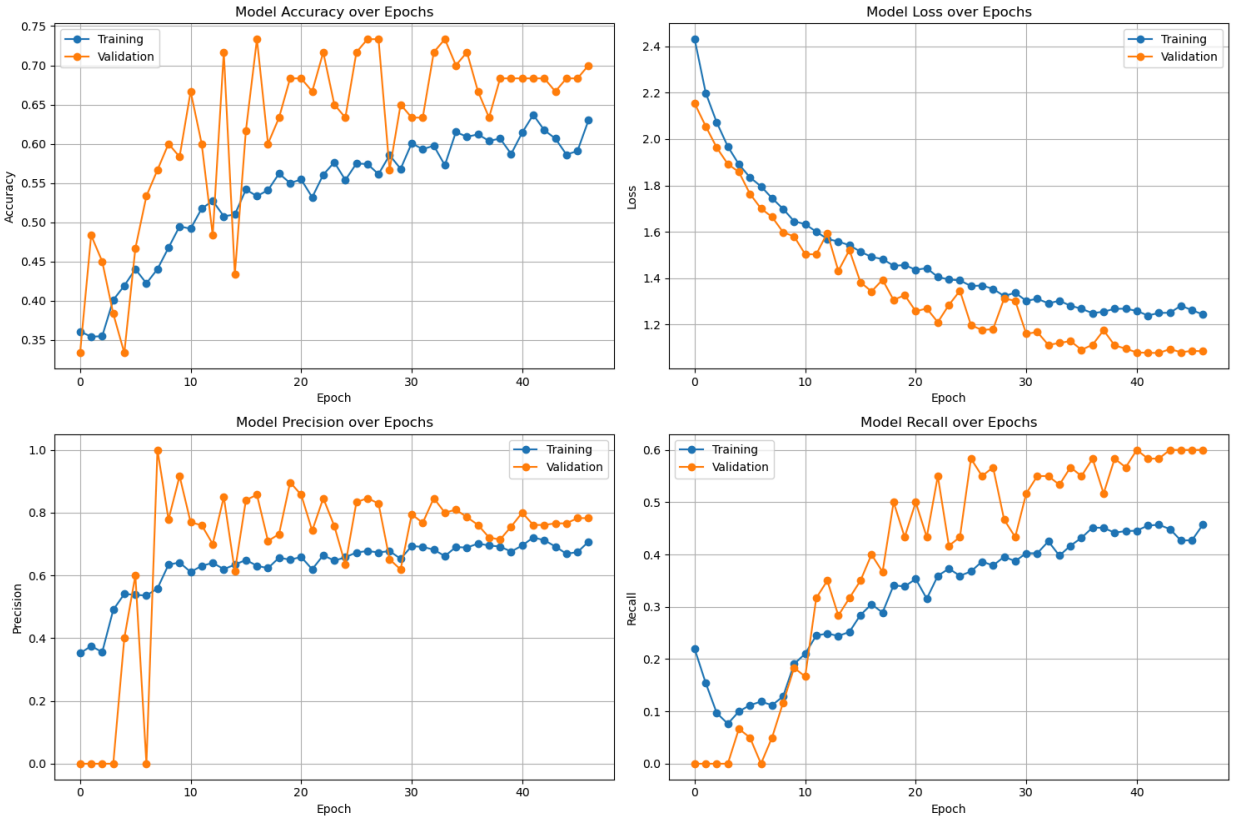


*Figure 1: Training and validation metrics over epochs for **ResNet***

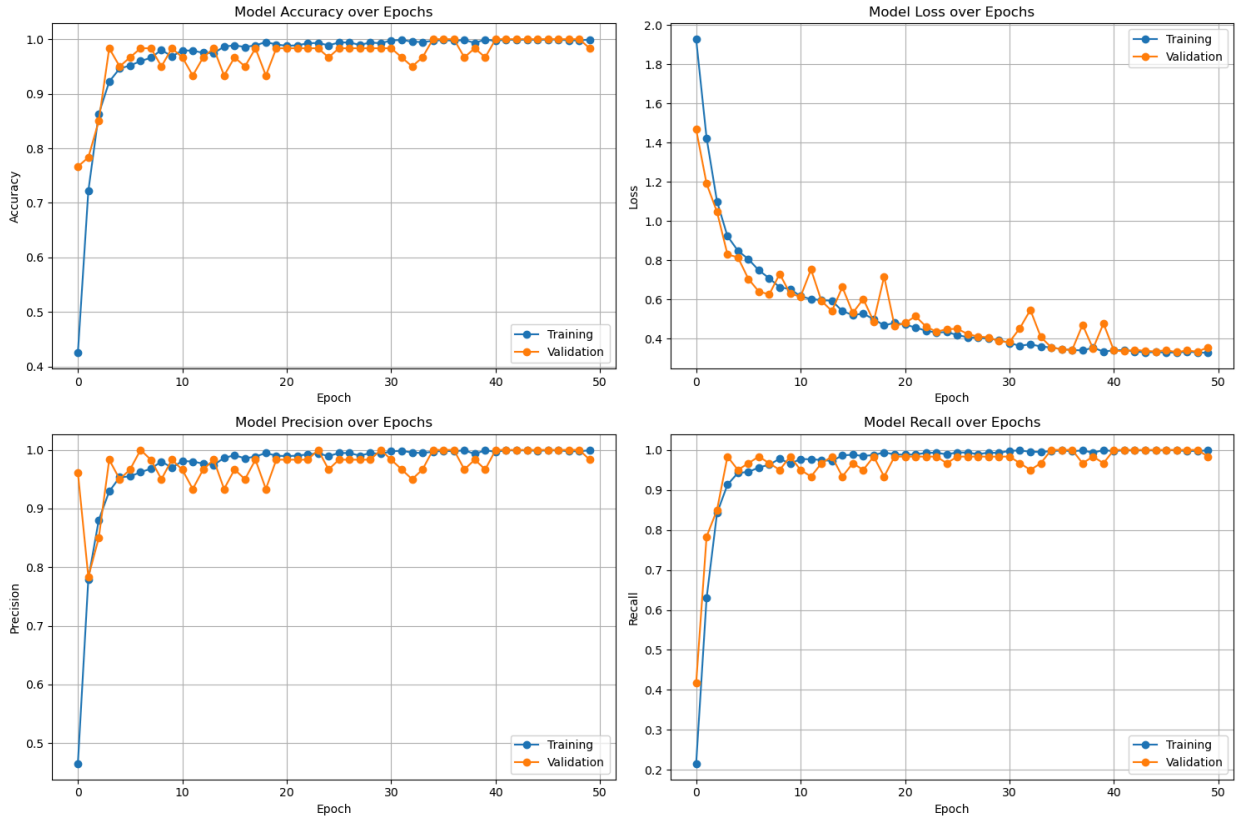


Figure 2: Training and validation metrics over epochs for VGG16

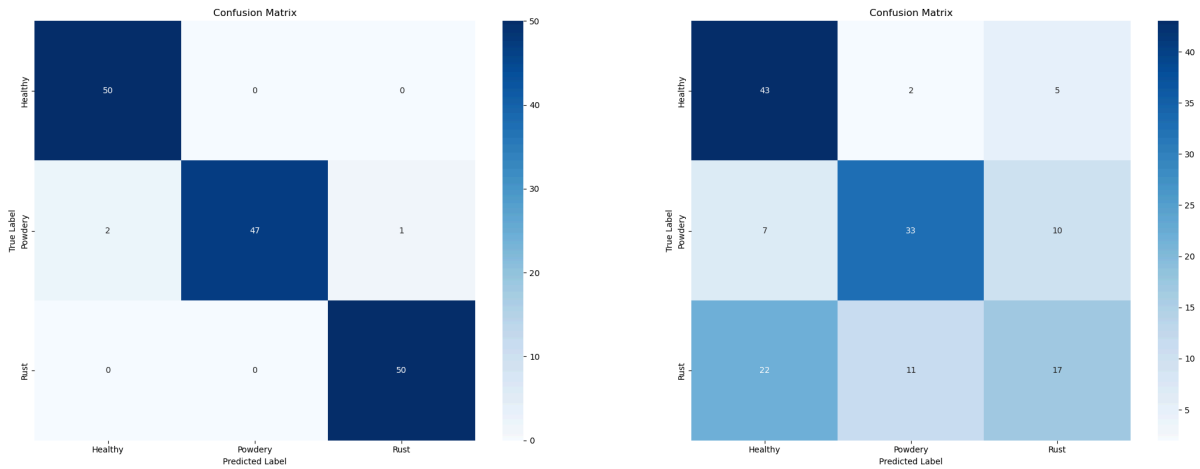
Results Analysis

VGG16 Performance:

- Test Accuracy: 98.00%
- Training convergence: Stable with smooth learning curves
- Final Loss: Training = 0.327, Validation = 0.355
- Per-class F1-scores:
  - Healthy: 0.980
  - Powdery: 0.969
  - Rust: 0.990

ResNet50 Performance:

- Test Accuracy: 62.00%
- Training convergence: Unstable with fluctuating metrics
- Final Loss: Training = 1.243, Validation = 1.085
    - Per-class F1-scores:
        - Healthy: 0.705
        - Powdery: 0.688
        - Rust: 0.415



VGG16 confusion matrix

ResNet confusion matrix

Figure 2: Confusion matrices comparing classification performance

Key Findings and Recommendations

1. **Transfer Learning Considerations:**
    - VGG16's simpler architecture (98% accuracy) outperformed ResNet50 (62%)
    - Success in transfer learning depends on similarity between source and target domains

- When source (ImageNet) and target (plant diseases) domains differ significantly, simpler architectures may transfer better
- Limited dataset size (1,530 images) favors less complex models

2. **Dataset-Architecture Alignment:**
   - Small, domain-specific datasets require careful model selection
   - Complex architectures (ResNet50) may be suboptimal when training data is limited
   - VGG16's linear structure proved more suitable for texture-based disease classification
   - Feature transferability decreases with architectural complexity in limited data scenarios
3. **Learning Strategy for Limited Data:**
   - Partial layer unfreezing (last 4 layers) prevents catastrophic forgetting
   - Conservative learning rate (5e-5) crucial for stable fine-tuning
   - Lower learning rates help preserve useful pre-trained features
   - Strong regularization essential for small datasets
4. **Overfitting Prevention Strategy:**
   - Comprehensive regularization crucial for limited data:
     - L2 regularization (0.001) controls weight magnitude
     - High dropout rates (0.6, 0.5) prevent co-adaptation
     - Early stopping prevents overspecialization
   - Data augmentation compensates for limited training examples

Best Practice Recommendations:

1. **Transfer Learning Strategy:**
   - Assess dataset size relative to model complexity
   - Consider domain gap between pre-trained and target tasks

- Start with simpler architectures for limited, domain-specific data
- Implement progressive fine-tuning approaches

2. **Training Optimization:**
   - Use conservative learning rates for transfer learning
   - Apply stronger regularization for smaller datasets
   - Implement data augmentation aligned with domain characteristics
   - Monitor validation metrics to detect overfitting early

⚠️ These findings emphasize that successful *transfer learning requires careful consideration of dataset characteristics*, domain differences, and model complexity. The superior performance of VGG16 demonstrates that simpler architectures, combined with appropriate regularization, can be more effective for limited, domain-specific datasets than more complex models.

---

3. Comparative Analysis: Data Augmentation Impact

Model Performance Visualization

Our experimental results provide a clear visual comparison between augmented and non-augmented approaches through multiple metrics:

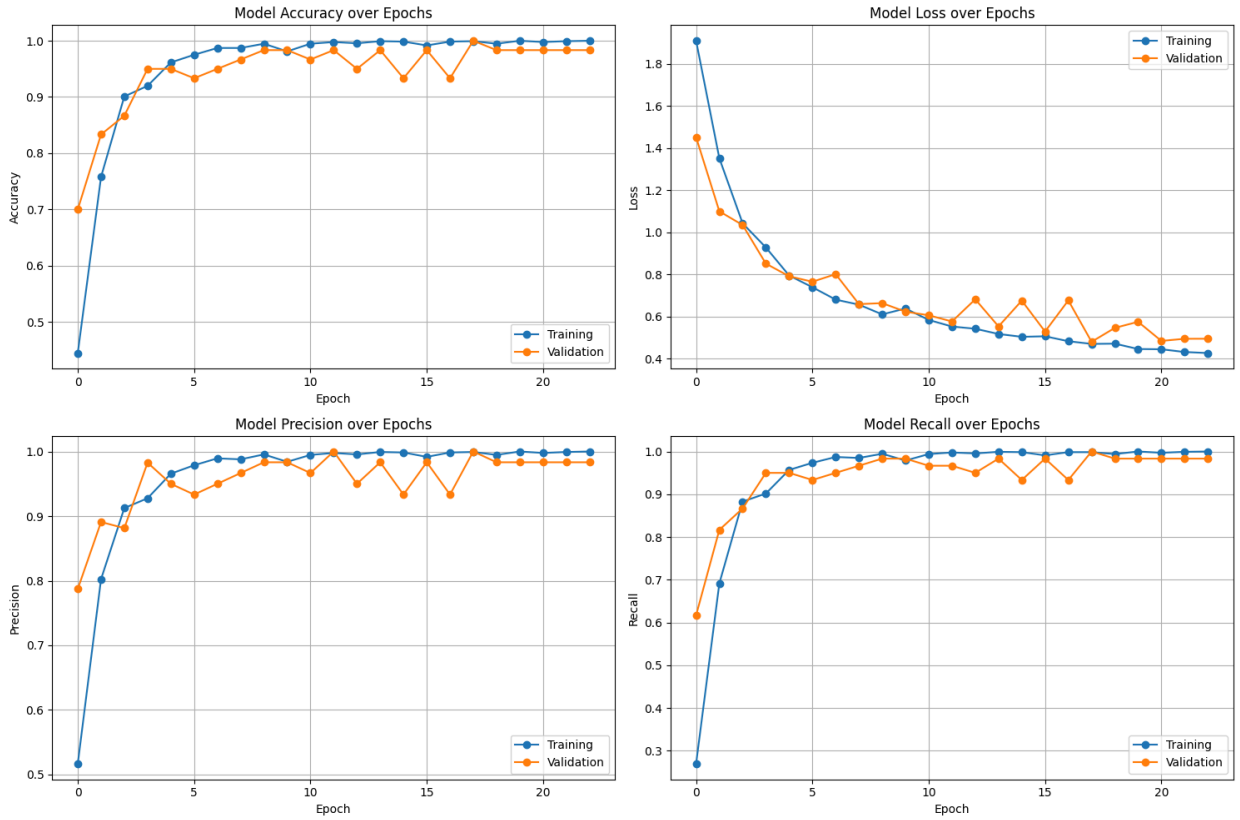


Figure 1: Training and validation metrics over epochs for VGG16 without data augmentation

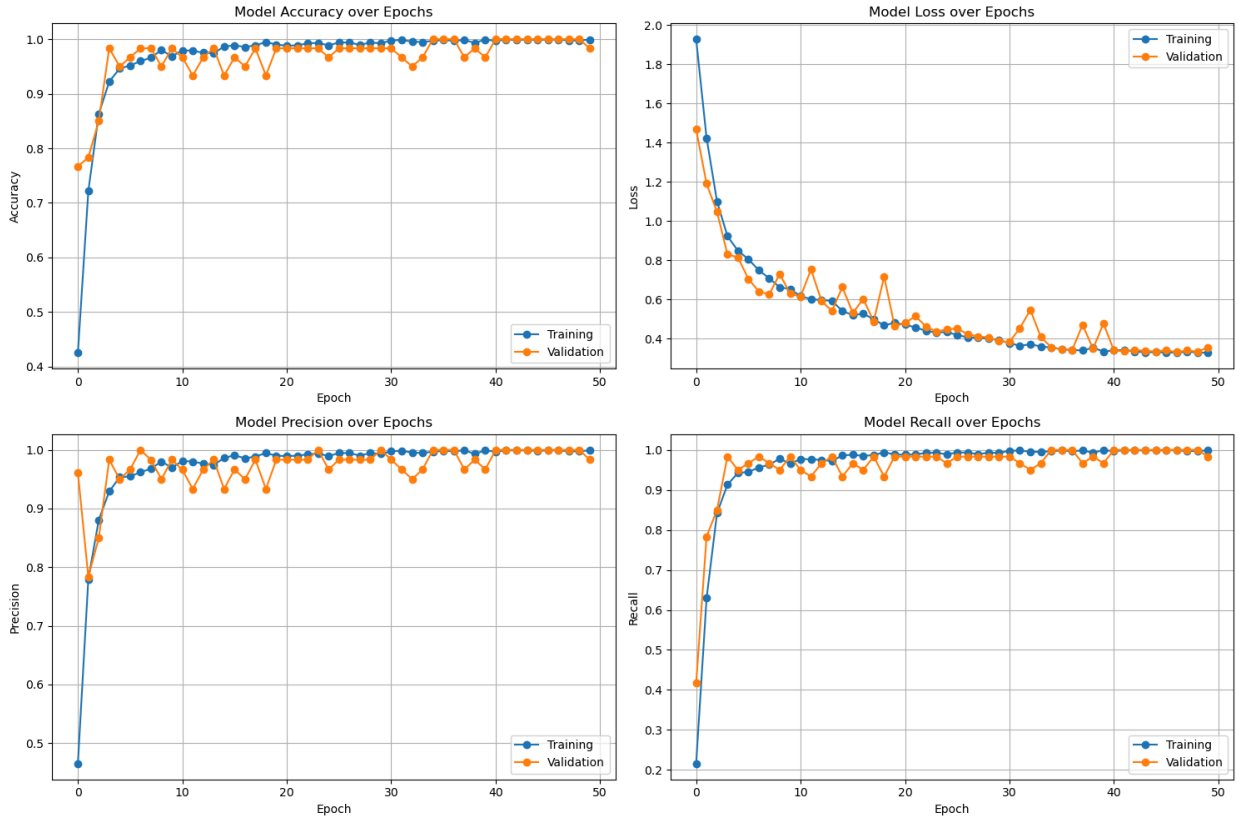
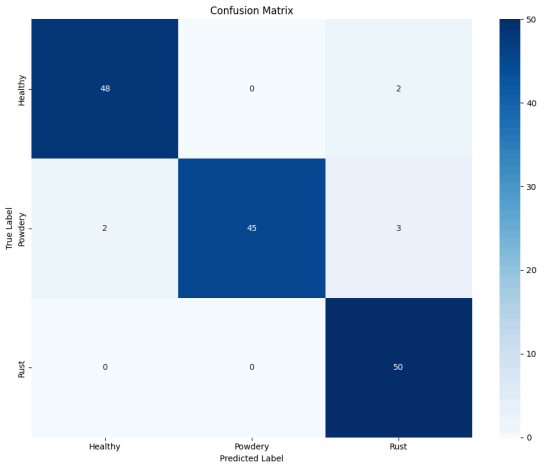
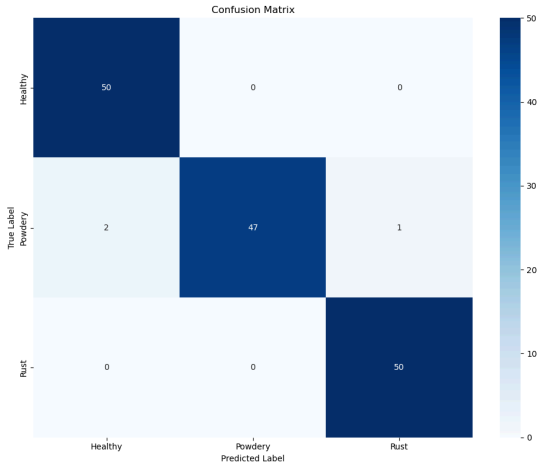


Figure 2: Training and validation metrics over epochs for VGG16 with data augmentation

1. **Training Dynamics (VGG16 with no data augmentation vs VGG16 with data augmentation):**

- Non-augmented model exhibits more volatile validation curves
- Augmented model shows smoother convergence patterns
- Early stopping triggered at epoch 23 for non-augmented model
- Augmented model sustained learning through all 50 epochs





VGG16 with no augmentation

VGG16 with data augmentation

2. **Confusion Matrix Analysis:**
    - Non-augmented model confusion matrix shows:
        - Healthy class: 48/50 correct (96%)
        - Powdery class: 45/50 correct (90%)
        - Rust class: 50/50 perfect (100%)
    - Augmented model demonstrates improved performance:
        - Healthy class: 50/50 perfect (100%)
        - Powdery class: 47/50 correct (94%)
        - Rust class: 50/50 perfect (100%)

Performance Metrics Comparison

Metric	Non-Augmented	Augmented	Improvement
Test Accuracy	95.33%	98.00%	+2.67%
Training Stability	Required early stopping	Completed full epochs	More stable

Class Balance	Good but variable	Excellent across classes	More balanced

The Critical Role of Data Augmentation and Regularization

Analysis of Non-Augmented Model Behavior

Our experiments with the non-augmented VGG16 model ***revealed several concerning patterns*** typical of models trained on limited datasets:

1. **Rapid Convergence Concerns**:
    - The model achieved perfect training accuracy by epoch 23
    - Early stopping was triggered to prevent further overfitting
    - The validation metrics showed higher volatility compared to the augmented model
    - Clear signs of memorization rather than generalization emerged
2. **Overfitting Indicators**:
    - Notable gap between training and validation metrics
    - Training accuracy consistently exceeded validation accuracy
    - Required aggressive early stopping intervention
    - Less stable validation performance across epochs

The Protective Role of Strong Regularization

Our implementation included robust regularization techniques that proved crucial:

1. **Architectural Safeguards:**
    - Dropout layers (0.5, 0.6) provided essential neuron deactivation
    - L2 regularization (0.001) effectively constrained weight growth
    - These measures partially compensated for the lack of data augmentation

2. **Performance Impact:**
    - Test accuracy reached 95.33% despite no augmentation
    - However, ***this success heavily relied on careful regularization***
    - Without these protective measures, overfitting would likely have been severe

Data Augmentation as a Superior Solution

The augmented model demonstrated clear advantages:

1. **Improved Training Dynamics:**
    - More gradual and stable learning curve
    - Closer alignment between training and validation metrics
    - Completed full 50 epochs without overfitting
    - Achieved higher final test accuracy (98.00%)
2. **Practical Benefits:**
    - Reduced reliance on complex regularization strategies
    - More robust feature learning through varied data exposure
    - Better generalization to unseen examples
    - More sustainable long-term training potential

Key Lessons for Small Datasets

⚠️ This comparison yields important insights for deep learning practitioners:

1. **Risk Management:**
    - While strong regularization can mitigate overfitting, it's a more fragile solution
    - Data augmentation provides a more robust and maintainable approach
    - Combined strategies offer the best protection against overfitting
2. **Best Practices:**

- **Always** implement data augmentation for limited datasets
- Consider augmentation as a **primary** defense against overfitting
- Use regularization as a **complementary** rather than primary strategy
- **Monitor** validation metrics closely for early signs of memorization

The superior performance and stability of the augmented model (2.67% higher accuracy) demonstrates that ***data augmentation should be considered a fundamental requirement for small dataset scenarios, not just an optional enhancement.***

4.The Pitfalls of Accuracy Metrics and Challenges of Custom CNN Architecture

Our custom CNN presents a classic case of why accuracy metrics can be deceptive. At first glance, the model appears highly successful:

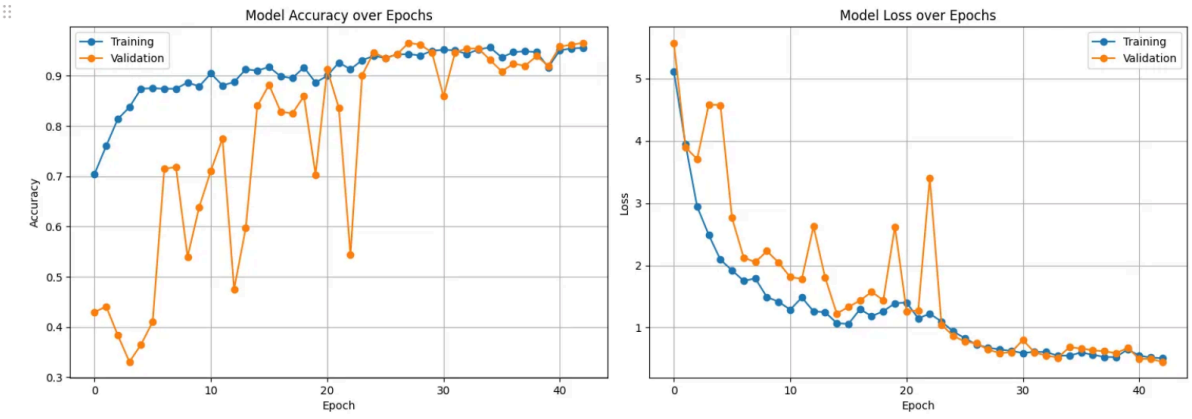
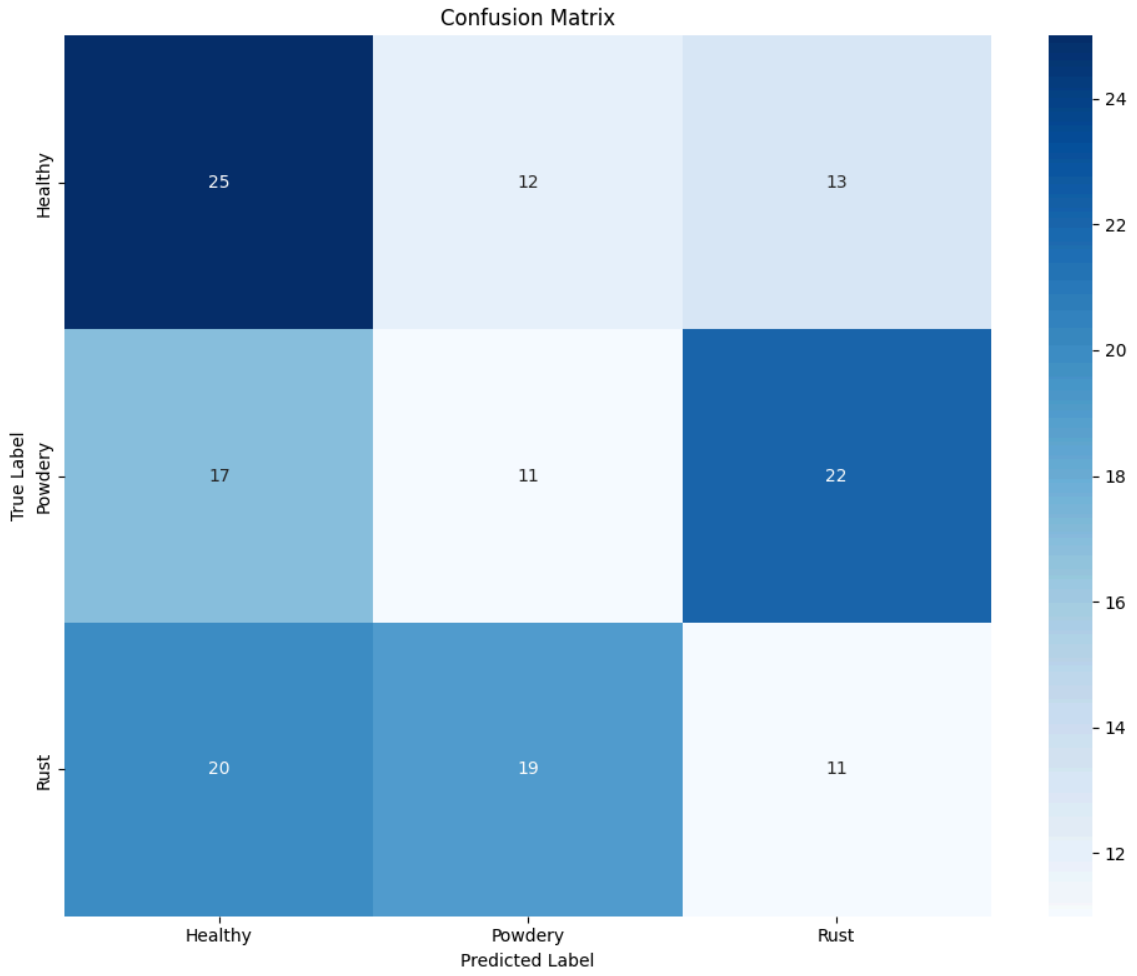


Figure 1: Training and validation metrics over epochs for our custom architecture

Why Accuracy Alone Is Misleading

- Training accuracy: 95.66%
- Validation accuracy: 96.58%
- Test accuracy: 92.00%

However, these impressive numbers mask serious underlying problems:



1. **Class Imbalance Impact:**
    - Looking at the confusion matrix, we see the model heavily favors certain predictions
    - Despite 92% overall accuracy, individual class **F1-scores are alarmingly low:**

    - Healthy: 0.446
    - Powdery: 0.239
    - Rust: 0.229
  - This suggests the model is biased and not truly learning to distinguish between classes

2. **False Confidence:**
  - The high accuracy primarily comes from the model's performance on majority classes
  - The confusion matrix reveals significant misclassifications
  - Example: The model frequently confuses Powdery with Rust diseases (22 misclassifications)

This dramatic disparity between overall accuracy and F1-scores reveals that our model is fundamentally flawed. While achieving 92% test accuracy, it's performing barely better than random chance for individual class prediction.

⚠️ This case study demonstrates why machine learning practitioners must:

- Always examine multiple performance metrics
- Look at per-class performance measures
- Consider the confusion matrix
- Analyze training dynamics

Challenges in Custom Architecture Development

1. **Overfitting Signs:**
  - Validation accuracy sometimes exceeds training accuracy
  - High volatility in validation metrics
  - Model required early stopping at epoch 43
2. **Resource Intensive Development:**

- Complex architecture design (4 conv blocks, multiple dense layers)
- Extensive hyperparameter tuning needed
- Long training cycles

Why Transfer Learning Offers a Better Approach

Transfer learning addresses these challenges by:

- Leveraging pre-learned feature hierarchies
- Providing more stable training
- Better handling of small datasets through proven architectures
- Reducing the risk of misleading performance metrics through more balanced feature extraction

The misleading accuracy metrics and development challenges of our custom CNN demonstrate why transfer learning is often more reliable, especially with limited data.