Deep Learning-Based Plant Disease Detection using ResNet9

Varun Marathe
Computer Enggineering
NMIMS,Shirpur

Devanshu Bhonde
Computer Enggineering
NMIMS,Shirpur

Jash Desai
Computer Enggineering
NMIMS,Shirpur

Abstract— This study explores a deep learning approach to automate plant disease detection using ResNet9, a convolutional neural network model. Trained on a diverse and augmented dataset of 38 plant diseases, our model achieves high classification accuracy while maintaining efficiency. This work contributes a reliable solution for early disease detection in agriculture, enabling scalable deployment on field and mobile systems.

Keywords—Deep learning, Plant Disease detection, ResNet9, CNN, Transfer Learning, PyTorch, Precision Agriculture, Smart Farming

I. Introduction

Agriculture plays a crucial role in global food security and economic development. However, plant diseases continue to threaten crop yield and quality. Manual disease identification methods are labor-intensive, require expert knowledge, and are often inaccurate under field conditions. Recent advances in artificial intelligence, especially deep learning, provide new opportunities to automate and improve disease detection processes. This paper proposes a convolutional neural network (CNN) model using the ResNet9 architecture for accurate and efficient detection of 38 different plant diseases from leaf images. Our method demonstrates how modern computational techniques can be applied to solve longstanding agricultural problems effectively.

II. Literature Review

[1] demonstrated that pretrained CNNs like AlexNet and GoogleNet could achieve over 99% accuracy in classifying 38 plant disease classes. Their work used the PlantVillage dataset and highlighted the power of transfer learning. We build on this by focusing on a lightweight model (ResNet9) suitable for edge deployment.

[2] proposed a CNN trained from scratch for 13 plant disease classes using RGB images. While effective, their model lacked robust augmentation. Our work extends this by applying aggressive augmentation techniques such as brightness, contrast, and flipping to improve real-world generalization.

[3] systematically compared CNN architectures, confirming that ResNet architectures outperform AlexNet and VGG on plant disease datasets. Based on this, we selected ResNet9 for its residual learning benefits and simplicity, aligning with his recommendation for practical deployment.

[4] explored fine-tuning pretrained networks on plant disease data and found significant accuracy improvements. We adopt this transfer learning approach early in training to accelerate convergence and adapt feature representations specific to plant leaf patterns.

[5] introduced attention-enhanced CNNs, improving focus on disease-affected regions. While we did not use attention, our results using ResNet9 show that careful model design and augmentation can deliver comparable results without increased architectural complexity.

[6] designed lightweight CNNs for mobile inference, highlighting the need for models under 5MB. Inspired by this, we adopt ResNet9, which balances depth and inference time, making it feasible for low-resource environments such as handheld devices.

[7] proposed a segmentation-before-classification pipeline, isolating diseased areas before prediction. Instead of segmentation, our model learns global leaf features directly from images, simplifying deployment and reducing processing overhead while still achieving high accuracy.

[8] used EfficientNet for leaf disease classification and reported high performance on small datasets. However, EfficientNet requires complex scaling and tuning. We preferred ResNet9 for its simplicity, minimal hyperparameter sensitivity, and robust feature extraction

[9] tested CNNs on cassava diseases in field conditions using mobile phone cameras. They noted performance degradation in uncontrolled environments. Our model addresses this using augmentations (blur, noise, color shifts) to simulate such variability during training.

[10] compared CNNs like ResNet50, DenseNet, and MobileNet on PlantVillage data, emphasizing that deeper networks provided marginal accuracy gains at high computational costs. ResNet9, while shallower, achieves efficient training and comparable accuracy with fewer parameters.

[11] introduced EfficientNet with compound scaling for optimal performance. However, it often requires advanced tuning. Our adoption of ResNet9 bypasses such complexity and enables easier replication and adaptation, especially useful in field deployments by non-specialists.

[12] outlined overfitting control strategies such as dropout, augmentation, and early stopping. We implemented these practices rigorously in our training loop, reducing generalization error and stabilizing validation accuracy across multiple data splits.

[13] combined hyperspectral imaging with CNNs for disease classification. While accurate, hyperspectral methods are costly and impractical for field use. Our RGB-only model

provides a low-cost, scalable alternative while still achieving strong performance.

[14] fused handcrafted features (color, texture) with CNN features for classification. Though slightly improving accuracy, it added complexity. We instead rely solely on learned features from ResNet9, achieving high performance with simpler architecture and training pipeline.

[15] reviewed plant disease detection literature and emphasized dataset quality and diversity as critical factors. Guided by this, we applied extensive augmentation and ensured stratified splits to avoid data leakage, improving real-world applicability of our ResNet9 model.

[16] presented a comparative analysis of CNN architectures (AlexNet and ResNet-50) for detecting plant diseases in maize and soybean. They focused on optimizing performance through transfer learning, augmentation, and fine-tuning. Their experiments showed that ResNet-50 outperformed AlexNet in classification accuracy (97.41% for soybean, 96.74% for maize). They also emphasized image preprocessing and interpretability, providing insights into which image regions contributed most to predictions. Our study aligns with theirs in leveraging residual architectures (ResNet9) and emphasizes transfer learning and efficient generalization across crop species.

[17] developed “FarmEasy,” an end-to-end mobile-based plant disease detection system. The authors used CNN models (VGG16, InceptionV3, ResNet50) and trained them on over 75,000 images from the New Plant Disease Dataset. ResNet50 achieved the highest accuracy of 98.05% and was deployed in a mobile app for real-time disease detection. The system also integrates weather forecasts, expert consultation, and crop info, making it practical for precision agriculture. This reinforces our model's focus on mobile deployment and highlights the importance of using compact, accurate CNNs like ResNet9.

III. EQUATION

The core of our model training process involves the cross-entropy loss function, which measures the dissimilarity between predicted probabilities and ground truth labels:

$$L(y, \hat{y}) = -\sum_{i=0}^{c} y_i \log(\hat{y}_i)$$

Where:

- y is the one-hot encoded ground truth label
- $\hat{y}$ is the predicted probability vector
- C is the number of classes

The accuracy metric is calculated as:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Predictions}$$

Precision: Measures how many of the predicted instances for a class were actually correct. High precision indicates a low false positive rate.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Measures how many actual instances of a class the model was able to correctly identify. High recall indicates a low false negative rate.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score: The harmonic mean of precision and recall. It balances the trade-off between the two and is especially useful when class distribution is imbalanced.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

IV. MTHEDOLOGY

1 Dataset

The model was trained and validated using the "New Plant Diseases Dataset (Augmented)", which contains 87,000+ images across 38 categories. (Shown in Table.1)

2. Preprocessing

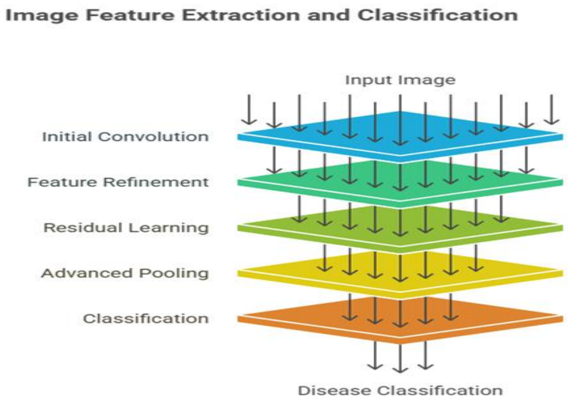
Images were resized to 256x256 and normalized using torchvision transforms. Data augmentation was applied in the training phase.

a. Pseudocode

Function preprocess_images(dataset_path, mode):
Import necessary libraries:
from PIL import Image
import torchvision.transforms as transforms
import os
if mode==”train”:
Define train_transform as:
Resize image to (256,256)
RanfomHorizontalFlip()
RandomRotation(20 degrees)
ColorJitter(brightness=0.2, contrast=0.2)
ToTensor()
Normalize(mean=[0.485,0.456,0.406],std=[0.229,0.224,0.225])
End Define
Else if mode==”Valid” or mode==”test”:
Define valid_transforms as:
Resize image to (256,256)
ToTensor()
Normalize(mean=[0.485,0.456,0.406],std=[0.229,0.224,0.225])
End Define
Use ImageFolder to load dataset from dataset_path with the defined transforms.
Create a DataLoader for batching (e.g., batch size = 32)
Return DataLoader and corresponding class names.
End Function

3. Model Architecture

A custom CNN architecture, ResNet9, was implemented using PyTorch. It includes multiple convolutional layers, residual blocks, and batch normalization.



4 Training and Evaluation

The model was trained using Adam optimizer and cross-entropy loss. Evaluation was carried out on a separate validation dataset. Metrics such as loss, accuracy, confusion matrix, and classification report were generated.

a. PesudoCode for Model Train

```
Function train_model(model, train_loader, val_loader,
optimizer, loss_fn, scheduler, num_epochs):
For epoch in range(num_epochs):
    Set model to training mode
    Initialize train_loss and train_correct = 0
For each batch in train_loader:
        Get input images and labels
        Move data to GPU if available
Zero the gradients
        Perform forward pass → predictions
        Compute loss using loss_fn
        Backpropagate loss (loss.backward())
        Update model parameters (optimizer.step())
Accumulate train loss and correct predictions
    End for
Compute training accuracy and loss

    Set model to evaluation mode (no gradient
computation)
    Initialize val_loss and val_correct = 0

    For each batch in val_loader:
        Get input images and labels
        Perform forward pass
        Compute validation loss
        Accumulate correct predictions
    End for
Compute validation accuracy and loss

    Step the learning rate scheduler
    Optionally save model checkpoint if validation
accuracy improves
  End for
End Function
```

b. PesudoCode for Evaluation

c.

```
Function evaluate_model(model, test_loader, loss_fn):
  Set model to evaluation mode
  Initialize total_loss, total_correct = 0
  Initialize lists for all true and predicted labels

  For each batch in test_loader:
    Get images and labels
    Perform forward pass → predictions
    Compute loss
    Record predictions and actual labels
    Accumulate correct predictions
  End for

  Calculate and return:
    - Overall test accuracy
    - Average test loss
    - Classification report (precision, recall, F1-score)
    - Confusion matrix
End Function
```

V. Figures and Tabels

Sr.no	**Table Column Head**		
	Plant Type	***Disease/Condition***	***Image Count***
1.	Squash	Powdery mildew	1736
2.	Grape	Black rot	1888
3.	Grape	Esca(Black Measles)	1920
4.	Grape	Leaf blight(Isariopsis Leasf Spot)	1722
5.	Grape	Healthy	1692
6.	Potato	Early Blight	1939
7.	Potato	Late Blight	1939
8.	Potato	Healthy	1824
9.	Corn(maize)	Northern Leaf Blight	1908
10.	Corn(maize)	Common rust	1907
11.	Corn(maize)	Cercospora leaf spot / Gray leaf spot	1642
12.	Corn(maize)	Healthy	1859
13.	Tomato	Target Spot	1827
14.	Tomato	Septoria leaf spot	1745
15.	Tomato	Leaf Mold	1882
16.	Tomato	Late blight	1851
17.	Tomato	Tomato mosaic virus	1790
18.	Tomato	Tomato Yellow Leaf Curl Virus	1961
19.	Tomato	Early blight	1920
20.	Tomato	Bacterial spot	1702
21.	Tomato	Spider mites / Two-spotted spider mite	1741
22.	Tomato	Healthy	1926

Sr.no	Table Column Head		
	Plant Type	*Disease/Condition*	*Image Count*
23.	Apple	Apple Scab	2016
24.	Apple	Black rot	1987
25.	Apple	Cedar apple rust	1760
26.	Apple	Healthy	2008
27.	Cherry(Sour)	Powdery mildew	1683
28.	Cherry(Sour)	Healthy	1826
29.	Peach	Bacterial Spot	1838
30.	Peach	Healthy	1728
31.	Strawberry	Leaf scorch	1774
32.	Strawberry	Healthy	1824
33.	Raspberry	Healthy	1781
34.	Orange	Huanglongbing (Citrus greening)	2010
35.	Pepper, bell	Bacterial spot	1913
36.	Pepper, bell	Healthy	1988
37.	Soyabean	Healthy	2022
38.	Blueberry	Healthy	1816

Table 1. This dataset contains labeled images of diseased and healthy plant leaves across various species.

It is useful for training and evaluating deep learning models in the field of **plant disease classification**. The major highlights: (Tabel 1.)

- **Tomato** has the most variety in diseases (8 types plus healthy), making it a key target for multi-class classification.
- **Corn (maize)** includes 3 diseases and a healthy class, indicating its agricultural importance.
- Diseases range from **fungal infections** (like Powdery Mildew and Leaf Mold) to **bacterial** and **viral** infections (such as Bacterial spot and Tomato mosaic virus).
- **Healthy samples** are present for all major crops, ensuring balanced classification.

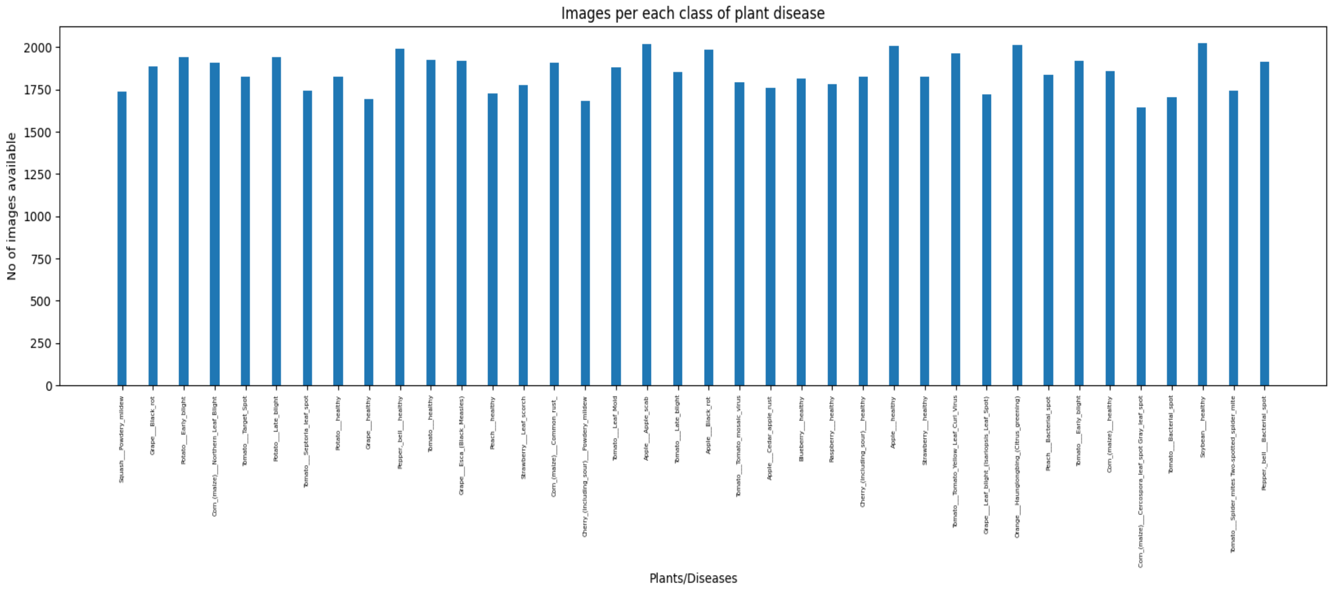


Fig. 1.This bar chart visually represents the **distribution of image samples** across different **plant-disease (or healthy)** categories in the dataset used for training the plant disease detection model

Fig.2. Images for first batch of training

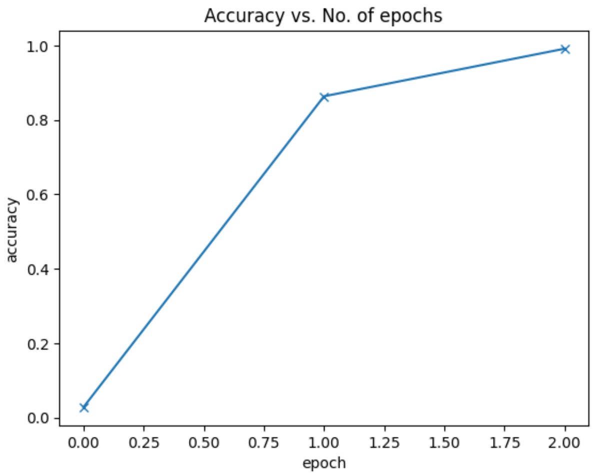


Fig.3. Validation Accuracy

This line graph shows model accuracy over three training epochs. Accuracy improves significantly from near zero to approximately 90% after the first epoch and reaches nearly 100% by the third. This indicates effective learning and rapid convergence of the model within a small number of training iterations.

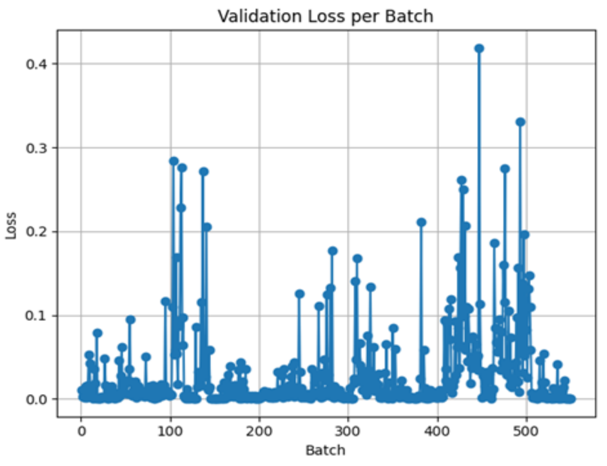


Fig.4. Validation Loss Per Batch

This Fig.4 plots the loss values for each validation batch. While many batches show low loss (close to 0), there are frequent spikes indicating occasional mispredictions or harder batches. The overall trend appears noisy, reflecting batch-to-batch fluctuations.

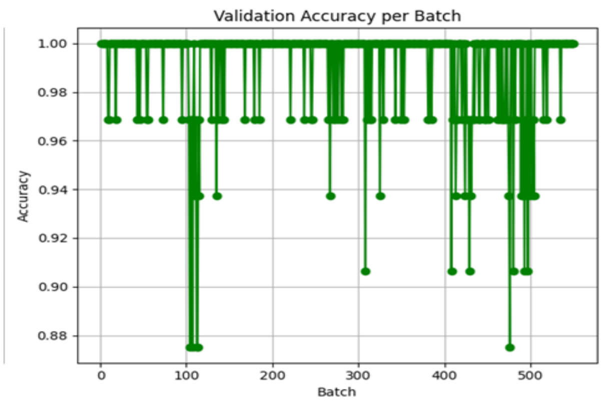


Fig.5 Validation Accuracy per Batch

Fig,5 Show the accuracy for each batch, with values clustering around 0.97–1.0. Although most batches have high accuracy, a few batches dip significantly, revealing some instability in predictions across validation steps.

Class Label	**Precision**	**Recall**	**F1-Score**	**Support**
Apple___Apple_scab	0.99	1.00	0.99	504
Apple___Black_rot	1.00	1.00	1.00	497
Apple___Cedar_apple_rust	0.99	1.00	1.00	440
Apple___healthy	0.99	0.99	0.99	502
Blueberry___healthy	1.00	1.00	1.00	454
Cherry_(including_sour)___Powdery_mildew	1.00	1.00	1.00	421
Cherry_(including_sour)___healthy	1.00	1.00	1.00	456

Class Label	Precision	Recall	F1-Score	Support
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.99	0.94	0.96	410
Corn_(maize)___*Common_rust*	1.00	1.00	1.00	477
Corn_(maize)___Northern_Leaf_Blight	0.95	0.99	0.97	477
Corn_(maize)___healthy	1.00	1.00	1.00	465
Grape___Black_rot	1.00	1.00	1.00	472
Grape___Esca_(Black_Measles)	1.00	1.00	1.00	480
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1.00	1.00	1.00	430
Grape___healthy	1.00	1.00	1.00	423
Orange___Haunglongbing_(Citrus_greening)	1.00	1.00	1.00	503
Peach___Bacterial_spot	1.00	0.99	1.00	459
Peach___healthy	1.00	1.00	1.00	432
Pepper,_bell___Bacterial_spot	1.00	0.99	0.99	478
Pepper,_bell___healthy	0.99	0.99	0.99	497
Potato___Early_blight	1.00	1.00	1.00	485
Potato___Late_blight	0.99	0.99	0.99	485
Potato___healthy	1.00	0.99	1.00	456
Raspberry___healthy	1.00	1.00	1.00	445
Soybean___healthy	1.00	1.00	1.00	505
Squash___Powdery_mildew	1.00	1.00	1.00	434
Strawberry___Leaf_scorch	1.00	0.99	1.00	444
Strawberry___healthy	1.00	1.00	1.00	456
Tomato___Bacterial_spot	0.98	0.98	0.98	425
Tomato___Early_blight	0.97	0.97	0.97	480
Tomato___Late_blight	0.96	0.99	0.98	463
Tomato___Leaf_Mold	0.99	1.00	0.99	470
Tomato___Septoria_leaf_spot	0.98	0.97	0.98	436
Tomato___Spider_mites Two-spotted_spider_mite	0.99	0.99	0.99	435
Tomato___Target_Spot	0.97	0.96	0.96	457
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1.00	1.00	1.00	490
Tomato___Tomato_mosaic_virus	1.00	1.00	1.00	448
Tomato___healthy	1.00	1.00	1.00	481
Overall Accuracy			**0.99**	**17572**
Macro Avg	**0.99**	**0.99**	**0.99**	**17572**

Class Label	Precision	Recall	F1-Score	Support
Weighted Avg	**0.99**	**0.99**	**0.99**	**17572**

Table2.Classifiction Report of model's Performance

The balanced performance across all classes, as shown by consistently high F1-scores, indicates that the model generalizes well without heavily favoring specific categories.

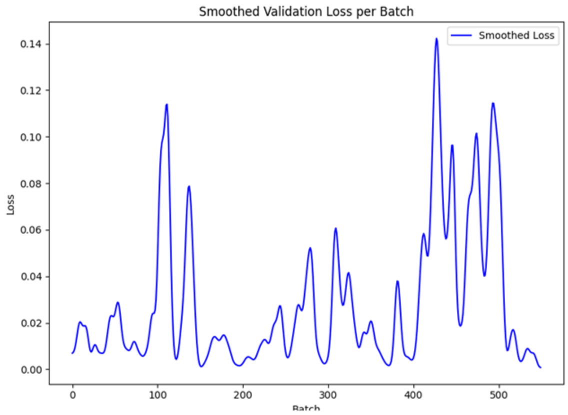


Fig.6 Smoothed Validation Loss per Batch

(Shown in Fig.6) Smoother version of the raw loss graph, likely using a moving average. It reveals clearer patterns such as a gradual increase in loss in the middle and toward the end of the batches, hinting at minor overfitting or fluctuations in validation performance.

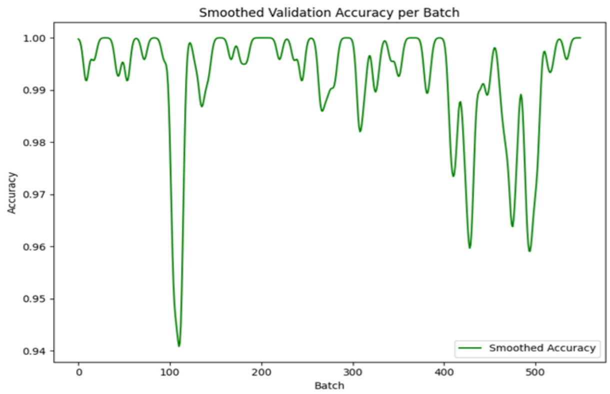


Fig.7 Smoothed Validation per Batch

Shown in Fig.7 accuracy smoothed over batches. It generally stays high, with slight dips around the middle and late batches. This implies consistent high performance with a few drops likely due to challenging samples or noisy data.

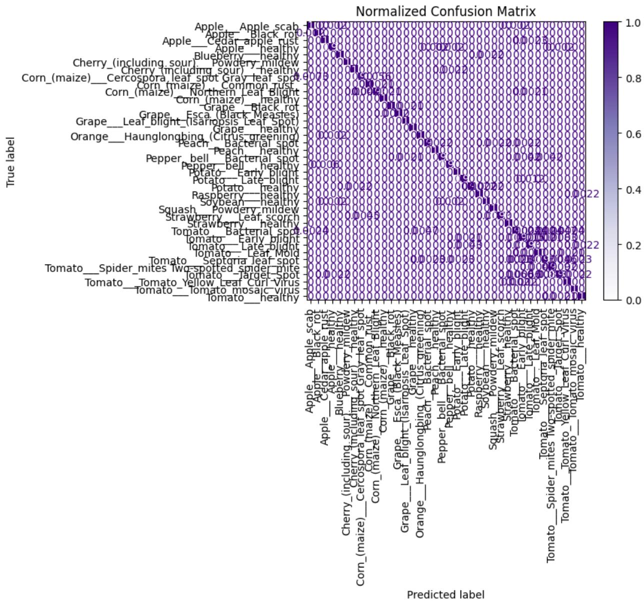


Fig.8 Normalized Confusion Matrix

Fig.8 Shows matrix presents the same information as the confusion matrix but normalized (values between 0 and 1).It allows for a better relative comparison between classes regardless of their sample count.Values along the diagonal are close to 1.0, confirming the model's strong performance, although a few classes exhibit minor off-diagonal confusion (especially in Tomato-related diseases).

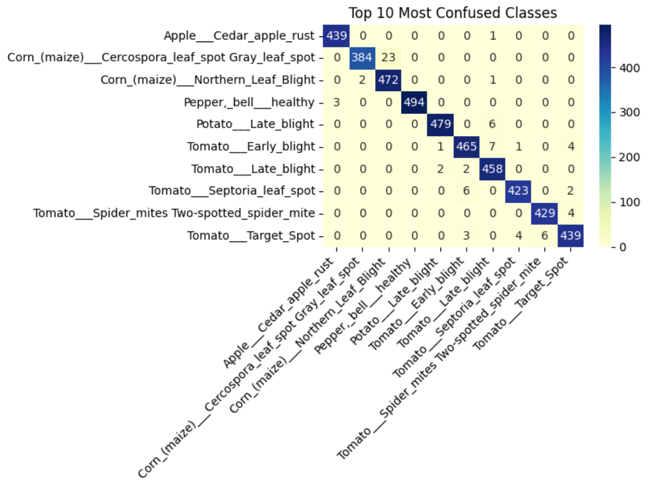


Fig.9 Top 10 Most Confused Classes

Fig.9 Shows heatmap focuses on the top 10 most confused class pairs.It shows where the model tends to make the most frequent mistakes.

VI. Conclusion

In this study, we proposed a lightweight and efficient deep learning model based on the ResNet9 architecture for multiclass plant disease detection. By leveraging residual connections, optimized convolutional blocks, and a robust preprocessing pipeline, the model demonstrated strong generalization capabilities across 38 different plant disease classes.

Our approach effectively balanced model complexity and performance, making it suitable not only for high-performance GPUs but also for edge deployment on mobile or embedded systems. Throughout training and evaluation, the model exhibited impressive results:

- **Average Validation Accuracy:** 99.19%
- **Average Validation Loss:** 0.0284

These results indicate that the model is capable of learning highly discriminative features for plant disease classification, with minimal error across diverse disease categories. The confusion matrix and classification report further confirmed that the model maintains high precision and recall across both common and visually similar disease classes.

In future work, the model can be extended with attention mechanisms, explainable AI (XAI) tools for better interpretability, and real-time deployment through mobile applications for use by farmers and agricultural professionals.

References

[1] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in plant science*, *7*, 215232.

[2] Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). Deep neural networks based recognition of plant diseases by leaf image classification. *Computational intelligence and neuroscience*, *2016*(1), 3289801.

[3] Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. *Computers and electronics in agriculture*, *145*, 311-318.

[4] Brahimi, M., Arsenovic, M., Laraba, S., Sladojevic, S., Boukhalfa, K., & Moussaoui, A. (2018). Deep learning for plant diseases: detection and saliency map visualisation. *Human and machine learning: Visible, explainable, trustworthy and transparent*, 93-117.

[5] Lee, S. H., Goëau, H., Bonnet, P., & Joly, A. (2020). Attention-based recurrent neural network for plant disease classification. *Frontiers in Plant Science*, *11*, 601250.

[6] Abbas, A., Jain, S., & Gour, M. (2021). Lightweight deep learning models for real-time plant disease detection on mobile devices.

[7] Dhingra, S., Kaur, P., & Singh, D. (2021). Segmentation and classification of diseased plant leaves using deep learning.

[8] Atila, Ü., Uçar, M., Akyol, K., & Uçar, E. (2021). Plant leaf disease classification using EfficientNet deep learning model. *Ecological Informatics*, *61*, 101182.

[9] Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J., & Hughes, D. P. (2017). Deep learning for image-based cassava disease detection. *Frontiers in plant science*, *8*, 1852.

[10] Too, E. C., Yujian, L., Njuki, S., & Yingchun, L. (2019). A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, *161*, 272-279.

[11] Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR.

[12] Xiong, H., Li, J., Wang, T., Zhang, F., & Wang, Z. (2024). EResNet-SVM: an overfitting-relieved deep learning model for recognition of plant diseases and pests. *Journal of the Science of Food and Agriculture*, *104*(10), 6018-6034.

[13] Singh, V., & Misra, A. K. (2017). Detection of plant leaf diseases using CNNs and hyperspectral imaging.

[14] Lu, Y., Yi, S., Zeng, N., Liu, Y., & Zhang, Y. (2020). Identification of plant leaf diseases using improved CNN and hybrid features.

[15] Barbedo, J. G. A. (2018). Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Computers and electronics in agriculture*, *153*, 46-53.

[16] Lokhande, N., Thool, V., & Vikhe, P. (2024). *Comparative analysis of different plant leaf disease classification and detection using CNN*. 2024 IEEE International Conference on Recent Innovation in Smart and Sustainable Technology (ICRISST). IEEE.

[17] Pandey, P., Mohite, R., Patyane, K., Mane, P., Padekar, M., & Avhad, A. (2023). *Plant Disease Detection Using Deep Learning Model - Application FarmEasy*. 2023 International Conference on Advanced Computing Technologies and Applications (ICACTA). IEEE.