Comparative Evaluation of Transfer Learning Models with Multiple Optimizers for Tea Leaf Disease Classification

Tanzila Akter
Department of CSE
City University

ABSTRACT
With the growing demand for smart farming, automated systems for plant disease detection have become increasingly important. This study presents a deep learning-based approach for tea leaf disease classification using five pre-trained models: ResNet50, InceptionV3, EfficientNetB0, VGG19, and Xception. Transfer learning is employed to evaluate the performance of these models using three optimizers: Adam, RMSprop, and Nadam. For each model–optimizer combination, training accuracy, training loss, validation accuracy, and validation loss are analyzed. The dataset is organized into class-based folders and divided into training, validation, and testing sets. Data augmentation, dropout, and global average pooling are incorporated to improve classification performance and reduce overfitting. Confusion matrices, prediction visualizations, and grouped bar charts are used to compare the experimental results. The experimental findings identify the best-performing model–optimizer combination for tea leaf disease classification, providing a practical and scalable approach to precision agriculture.

Keywords
Transfer Learning, Pre-trained CNN Models, Adam, RMSprop, Nadam, Tea Leaf Disease, Precision Agriculture.

1. INTRODUCTION
In addition to being an energy drink, tea is an important link between financial markets, history, culture, and clean-lines. Pests and illnesses associated with tea can influence output and quality in a number of ways, including lowering yields, lowering quality, and causing losses. Diseases have detrimental impacts on plant and agricultural environments [1]. Microorganisms, genetic anomalies, and infectious agents, including bacteria, fungi, and viruses, are the main causes of these disorders. The primary causes of illnesses affecting tea leaves are bacteria and fungi. This work uses a variety of pre trained deep learning models to identify illnesses in tea leaves. Seven kinds of tea leaf diseases from a publicly available dataset were utilized for categorization. Various optimizers were used to assess how well they increased the precision of disease detection. Manually identifying plant diseases is time-consuming and inefficient, especially on big farms. Visual examination and other traditional procedures are prone to errors and often require the help of a team of experts [1].

Furthermore, existing approaches cannot consistently guarantee early detection, which is essential for efficient disease control and prevention. In order to enable quick diagnosis along with effective disease management, a tea leaf disease detection method that is precise, productive, and computerized is obviously needed.

2. LITERATURE REVIEW
Zhang et al. [2] used 600 photos of cucumber leaves afflicted by six prevalent infections to introduce a Global Pooling Dilated Convolutional Neural Network (GPDCNN) for identifying cucumber leaf diseases. In comparison to other methods, this model showed stronger resilience and a higher accuracy of 94 %. However, a lot of factors are involved in its fully connected layers, which might slow down training and raise the possibility of overfitting.

Aditya Khamparia et al. [3] contains their research on a wide range of plant diseases and their improved techniques for diagnosing them. Next, utilizing different kernel functions, Support Vector Machines (SVMs) are used to identify and detect the type of illness affecting tomato plants.

The possibility of identifying tomato leaves with several diseases at various stages, including early or asymptomatic phases, was examined in this work [4] using a high-resolution portable spectral sensor. Four stages—healthy, asymptomatic, early stage, and late stage—were identified and gathered from a field, along with one healthy leaf and three diseased tomato leaves. Following procedures described in this work and published in earlier research, fifty-seven spectral vegetation indices (SVIs) were computed. SVIs were assessed using principal component analysis.

A machine learning-based rice leaf disease detection system is presented by S. Mehnaz et al. [5]. The present study identifies three prevalent diseases of rice plants: brown spot, bacterial leaf blight, and leaf smut. The input consisted of crisp pictures of the impacted rice leaves against a white backdrop. The dataset was trained using a variety of machine learning techniques, such as KNN (K Nearest Neighbour), J48 (Decision Tree), Naive Bayes, and Logistic Regression, following the required pre-processing. Following ten-fold cross-validation, the decision tree algorithm applied to the test dataset yielded an accuracy of more than 97%. Chandrala J [6] presents a technique that fine-tunes (transfer learning) pre-trained models, such as VGG19, to extract pertinent characteristics from the dataset. This model was trained using the ImageNet dataset, which consists of more than 15 million annotated high-resolution images in 22,000 categories. In VGG19, the only preprocessing that is done is to deduct each pixel's mean RGB value. When compared to AlexNet, the VGG19 model improved classification accuracy by substituting different 3*3 kernel-sized pooling for all of the large kernel-sized filters. This was done over a 2*2 window with a stride size of 2. The softmax layer is the final layer in the model. The ReLU function has been used to induce non linearity into each of the model's hidden layers. B. Tugrul et al. review [7] studied the CNNs used to automatically detect plant leaf diseases in order to assess the potential of these networks

for such uses. They are able to determine the problems in this field of study thanks to their survey. This study examines the various plant disease kinds, their causes, symptoms, and treatment strategies using image processing and classification techniques. The limits of previous research were also covered in this survey, which offers guidance for future advancements in the detection of plant diseases and their prevention.

3. DATASET DESCRIPTION

The TeaLeafBD– Tea Leaf Disease Detection dataset from Kaggle contains 5,226 high-resolution images of tea leaves that have been categorized into seven groups based on different illnesses and healthy conditions. These classes include Helopeltis, Tea Algal Leaf Spot, Red Spider, Green Mirid Bug, Brown Blight, and Grey Blight. The dataset is carefully divided into training, validation, and testing sets to ensure effective model training and accurate performance evaluation.

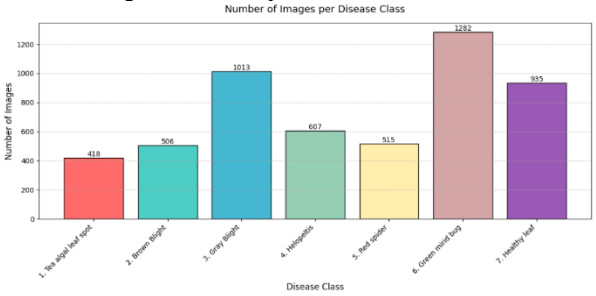


Figure 1: Images for each disease class

4. METHODOLOGY

The proposed framework for tea leaf disease classification follows a structured deep learning workflow, as illustrated in Figure 2. The complete process involves dataset preparation, data augmentation, model customization, training with multiple optimizers, and performance evaluation.

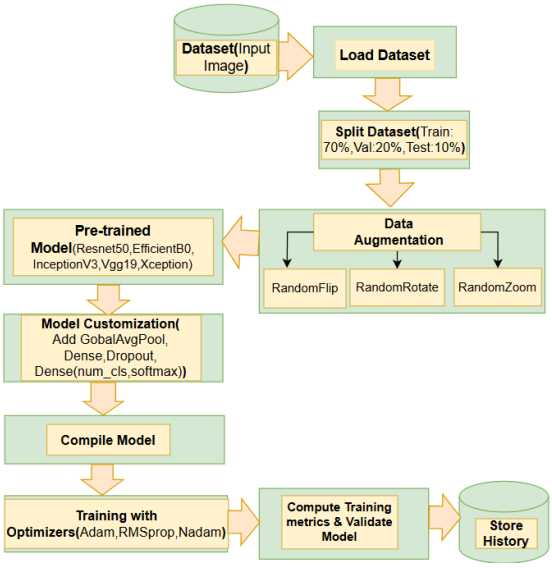


Figure 2: Model Architecture Flow Diagram

4.1 Load Dataset

The process begins with a dataset containing tea leaf images. These images are collected and stored in a structured format, usually organized into folders based on different disease categories. The dataset is loaded into the deep learning environment (e.g., TensorFlow/Keras). This step prepares the images for further processing and model training.

4.2 Split Dataset

During the preprocessing phase, the dataset on tea leaf disease is loaded from a designated directory and automatically divided into training, test, and validation subsets in an 80:20:10 relationship. For training, all photos are resized to 224 by 224 pixels and arranged in batches to guarantee effective processing. In order to determine the number of illness categories, class labels are retrieved. Prefetching is used on both the training and validation datasets using TensorFlow's AUTOTUNE function to optimize the input pipeline. This technique enables the data to be prepared in the background while the model is training, greatly increasing computational efficiency.

Class weights are calculated based on the distribution of labels in the training set to address class imbalance and guarantee that under-represented illness classes receive the proper training attention.

4.3 Data Augmentation

It is a process used in deep learning and machine learning to increase the quantity and diversity of data available for model training. Using transformations like flipping, rotating, scaling, or cropping, this method creates new data from pre-existing data. By exposing them to a wider range of training data, data augmentation aims to enhance the simplicity and performance of representations. Data augmentation can assist models in learning to identify things in various lighting situations and from various viewpoints by producing additional data with various variations. This method can also lessen overfitting by giving the model a wider variety of samples to work with. Zooming, color shifting, parallel flipping, upright flipping, and random cropping are a few examples of frequent data expansion techniques. These methods can be used for a variety of information types, such as text, audio, and images. In conclusion, by producing more training data with various variations and lowering overfitting, data expansion is an effective method for enhancing the caliber and resilience of machine learning models.

4.4 Pre-trained CNN Model Architectures

A machine learning (ML) model that has been prepared for a particular task by being trained on a sizable dataset is known as a pre-trained model. ML models are frequently developed using pre-trained models as a starting point because they offer a set of initial weights and biases that may be adjusted for a given task [8].

Utilizing pre-trained models has a number of benefits, such as enhancing model performance, saving time and money, and utilizing the expertise and experience of others. Large, varied datasets are frequently used to train pre-trained models, which are able to identify a broad variety of patterns and characteristics. They can therefore offer a solid basis for fine-tuning and greatly enhance the model's performance.

Object recognition models, which are trained to identify and classify objects in images or videos, are commonly built on top of CNNs, or region-based convolutional neural networks (R-CNNs). Recurrent neural networks (RNNs) and transformers are frequently used as the foundation for language models, which are trained to predict the next word in a sequence [8].

Five pre-trained CNN models are used: ResNet50, EfficientNetB0, InceptionV3, VGG19, and Xception. These models are chosen because they are highly efficient for image classification tasks and are trained on large datasets like ImageNet.

4.4.1 EfficientNetB0

Google AI created the EfficientNetB0 model, a deep convolutional neural network (CNN) that was unveiled in 2019 as a member of the EfficientNet family. In contrast to more conventional models like VGG, ResNet, and Inception, it may achieve high accuracy with fewer parameters and at a lower computational cost, which makes it popular for image classification applications. The baseline model, EfficientNetB0, is used to create the bigger variations (B1–B7) using the compound scaling approach. By optimizing all three dimensions simultaneously, EfficientNet offers a balanced scaling strategy that produces a highly efficient architecture in contrast to conventional scaling methods that enhance depth, breadth, or input resolution independently.

The pre-trained models are fine-tuned by adding custom layers.EfficientNet-B0 was created to solve this problem by employing the compound scaling approach, which uses a grid search to ascertain the link between the baseline network's various scaling dimensions while operating under a fixed resource limitation. This technology has made it possible to create seven distinct EfficientNet models and is a major improvement over conventional scaling techniques. The AutoML MNAS framework was used to create these models, which combine a mobile inverted bottleneck convolution (MBConv) with an extra Squeeze and Excitation (SE) block. The models use the TL approach to attain state-of-the-art accuracy on other datasets and perform well on ImageNet. This study makes use of EfficientNet-B0, a mobile-sized baseline network made up of seven MBConv blocks that is a member of the EffichientNet family [12].

4.4.2 Resnet50

A network that facilitates continuous learning is called a residual network or ResNet. Residual Network with 50 Layers is referred to as Resnet50. The pertained version of this network, which serves as the basis for numerous tasks, was trained on over 10 lakh images from the database. It was the winner of the ImageNet contest in 2015.ResNet50 is more accurate than models like CNN, InceptionV3, MobileNetV2, GoogLeNet and others.Resnet50 is the most effective model for detecting plant diseases, as evidenced by a large number of research and publications. Additionally, this ResNet50 model is among the top 4 pre-trained models for classifying pictures. Because the ResNet50 model considers these three criteria, it is a transfer learning model, meaning it does not require significant computing. The ability to train extremely deep neural systems using more than 150 films was the primary modernization brought about by ResNet [11].

4.4.3 Vgg19

The Image Network collection contains millions of images that are used to train VGG-19. Three dense layers, fc6, fc7, and fc8, and sixteen Conv layers make up VGG-19. In the last layer before the output, the activation function utilized for classification is called SoftMax. Each of the five blocks that make up the convolution layers has a maxpool layer after it. The size of fc6, fc7, and fc8 are therefore 4096, 4096, and 1000, respectively. With a dimension of 1000, the fc8 layer can categories the ImageNet dataset into 1000 different categories. Because the model has learnt rich representations and its parameters are very transferable, VGG-19 is very well-liked.VGG-19 is used by all five of the models in the experiment to initialize the parameters, which are subsequently adjusted based on the disaster datasets that each model uses.

The suggested Unsupervised Domain Adaptation network, which has VGG-19 as its backbone architecture, learns domain-invariant features by decreasing the difference between the source and target domains using the MMD metric and radial basis kernel.

4.4.4 InceptionV3

When the Inception v3 model was released in 2015, it had 42 layers total and a lower error rate than its predecessors. Let's examine the various improvements that enhance the Inception V3 model. Significant changes were made to the Inception V3.The model includes smaller convolutions using factorization, factoring spatial data into asymmetric convolutions. Advantages of supplementary classifiers: effective grid size decrease. Let's see how each optimization was applied and how the model was enhanced as a result [11].

An extension of the well-known GoogleNet, Inception-v3 has demonstrated strong classification performance in a number of biological applications through the use of transfer learning. After GoogleNet, Inception-v3 introduced an inception model that creates a new filter by concatenating many convolutional filters of varying sizes. Because fewer parameters need to be taught, this architecture lowers the computational complexity.

4.4.5 Xception

Xception is based on Inception-V3 and employs a linear stack of depth-wise separable convolution layers with residual connections to decrease time and space complexity. Xception's depth-wise separable convolution divides the learning of space-wise and channel-wise features. Additionally, the residual link creates a shortcut in the sequential network, which helps to tackle the issues of representational bottlenecks and disappearing gradients.

4.5 Model Customization

Model customization is the process of adding extra layers to a neural network that has already been trained in order to adapt it to a different job or dataset. The ability to reuse learnt features from a model trained on a big dataset (such as ImageNet), that is for a smaller or different dataset makes this particularly helpful in transfer learning.

This type of model customization involves adding unique layers to an existing model's structure in order to modify it for an entirely novel category goal.

First, a GlobalAveragePooling2D layer is attached to the base system's output, combining every feature map in order to produce an organized 1D vector of characteristics while reducing its spatial dimensions. Next comes a 128-unit dense layer with a ReLU activation function, which adds non-linear nature and enables the model to recognize increasingly intricate patterns. Then, to help avoid overfitting, a Dropout layer with a 0.5 rate is introduced, which randomly deactivates half of the neurons during training. Lastly, the probability estimates for various classes are generated using a dense layer as the output layer, which has num_classes neural networks and a softmax activation. By keeping the learnt features from the previously trained base, this customized head allows the model to function well on the new dataset.

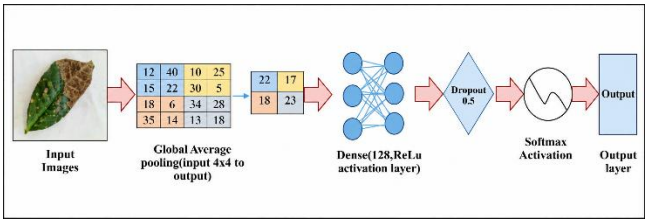


Figure 3: Model Customization

4.5.1 Global Average Pooling (GlobalAvgPool)

Global average pooling is intended to replace fully linked layers in conventional CNNs. The final conv layer is intended to produce a single feature map for each category that corresponds to the classification task.

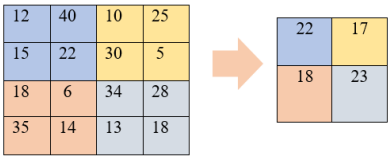


Figure 4: Global Average Pooling

4.5.2 ReLU and Softmax Activation Functions

CNNs commonly use the Rectified Linear Unit (ReLU) as an activation function. By adding nonlinearity it enhances the network's capacity to represent intricate connections between input and output data. The neuron can only process positive values thanks to the ReLU function. The presence of a positive input causes the neuron's electrical output to be equivalent to the input value; a negative input causes the resultant value to represent zero [9].

The formula for the ReLU function is f(y) = max{0,y}

A popular mathematical function in artificial neural networks, especially for multi-class classification tasks, is the softmax activation function. It is employed in the output layer of a neural network to generate a probability distribution across the conceivable classes.

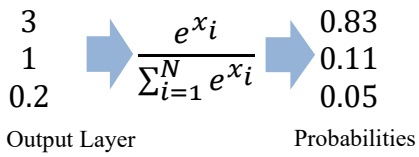


Figure 5: Softmax Activation Function

4.6 Model Optimizer

Optimizers are methods or algorithms used to minimize an error function, sometimes called a loss function, or to maximize production efficiency. Optimizers are mathematical operations that rely on the learnable parameters of the model, such as weights and biases. Optimizers are methods or algorithms used to minimize an error function, sometimes called a loss function, or to maximize production efficiency.

4.6.1 Nadam Optimizer

Timothy Dozat presented the optimization algorithm known as Nesterov-accelerated Adaptive Moment Estimation (Nadam) in his 2016 work, "Incorporating Nesterov Momentum into Adam." In order to increase convergence speed and performance when training deep learning models, Nadam integrates Nesterov momentum into the Adaptive Moment Estimation (Adam) optimizer. This approach has been used to optimize neural network training in a variety of fields, such as computer vision and natural language.

The advantages of momentum and adaptive learning rates are combined in Adaptive Moment Estimation (Adam), which calculates individual adaptive rates using the first and second moments of gradients. Adam's adaptive learning rate mechanism may cause it to converge to poor solutions at times, and it may slow down before achieving global optima. Nadam improves on Adam by adding Nesterov momentum, assessing the gradient at a predicted future parameter location, and increasing the accuracy and speed of convergence [10].

Key Components and Formula:

Gra_t=Gradient at time
Momentum decay rate=α,
Rate of deterioration for RMS scaling=β,
average of historical squared gradients
that decays exponentially=$Moment_1$,
Average of previous squared gradients with exponential decay
$= Moment_{t2}$

1. **1st Momentum**

$$Moment_1 = \alpha . Moment_{t-1} + (1-\alpha).\ Gra_t$$

2. **2nd Momentum**

$$Moment_2 = \beta . Moment_{t2-1} + (1-\beta).\ Gra^2{}_t$$

3. **Bias correction:**

$$\widehat{Moment}_{t1} = \frac{Moment_t}{1-\alpha^t}$$
$$\widehat{Moment}_{t2} = \frac{Moment_t}{(1-\beta)}.$$

4. **Update with Nesterov momentum integrated**

$$theta_{t+1} = theta_t - \text{Learning rate}\ ((\widehat{Moment}_{t2} / \sqrt{\hat{v}_t} + \epsilon))$$

4.6.2 RMSprop Optimizer

One popular optimization technique in deep learning is called Root Mean Square Propagation, or RMSProp. It is an enhancement on the Adaptive Gradient Algorithm (AdaGrad) that fixes some of its main drawbacks, particularly for models that work with sparse data. RMSProp tracks an exponentially decreasing average of the squared gradients. RMSProp employs a moving average in place of AdaGrad's accumulation of all past squared gradients. Because it gives more weight to recent gradients, RMSProp can adapt more successfully without drastically reducing the learning rate [10].

***Key Components and Formula*:**

Let:

$$\theta_{theta} = parameter(weights)$$

Gra_t=Gradient at time
$Expo(Gra^2)_t$=Moving average of squared gradients with exponential weighting.

Step-1: Moving average of squared gradient

$$Expo(Gra^2)_t = \text{decay rate}.\ Expo(Gra^2)_{t-1} + (1\text{-decay rate}).Gra^2{}_t$$

Step-2: Parameter update

$$\theta_{t+1} = \theta_t - \frac{\text{learning rate}}{\sqrt{Expo(Gra^2)_t + \text{small constant}}}\ .\ Gra_t$$

4.6.3 Adam Optimizer

Adam (Adaptive Moment Estimation) is a complex optimization method that is often used to train deep learning models. It combines the advantages of two other well-known optimizers: RMSProp, which modifies learning rates based on a moving average of recent gradients, and AdaGrad, which modifies learning rates for each parameter based on the frequency of updates.

For each parameter, Adam keeps two moving averages: The initial moment estimate or the average of the slopes. The uncentered variance of the gradients, or the second moment estimate. Particularly in the early time steps, these estimations are skewed towards zero. Adam fixes this by adding bias-correction terms, which guarantee more precise and steady convergence, especially in the beginning of training [10].

5. EXPERIMENTS

All five pre-trained CNN models (ResNet50, InceptionV3,

EfficientNetB0, VGG19, and Xception) were trained using three different optimizers (Adam, RMSprop, and Nadam) and their training and validation accuracy and loss were documented. In order to evaluate convergence behavior and generalization capabilities, model performance was tracked throughout several epochs during the training phase. Each model was trained using the specified hyper parameters. To maintain reproducibility, random seeds were fixed, and details of the training environment, including Tensor Flow/Keras version and GPU specifications, were documented. The training process used ModelCheckpoint to save the best performing weights based on validation loss and Early Stopping to prevent overfitting.

5.1 Training Procedure

The training and validation accuracy of ResNet50, InceptionV3, EfficientNetB0, Xception, and VGG19 across 20 epochs using the Adam optimizer is shown in Figure 6. All five models saw an increase in training accuracy, with EfficientNetB0 and ResNet50 reaching the highest values—nearly 0.90. On the other hand, validation accuracy shows more volatility, especially for ResNet50 and Xception, suggesting less consistent validation performance. The validation accuracy of EfficientNetB0 shows a steadier rising trend, reaching about 0.85 by the last epoch.

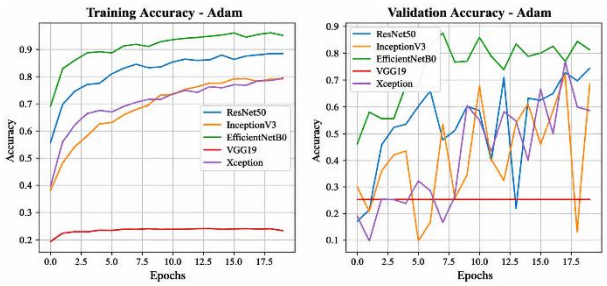


Figure 6: Training and validation accuracy plots for Adam

All models shown in Figure 7 a consistent drop in training loss across 20 epochs, with VGG19 displaying a moderate decline. ResNet50 and EfficientNetB0 demonstrating the most notable reduction. For the majority of models, validation loss stays modest and consistent, with the exception of ResNet50.

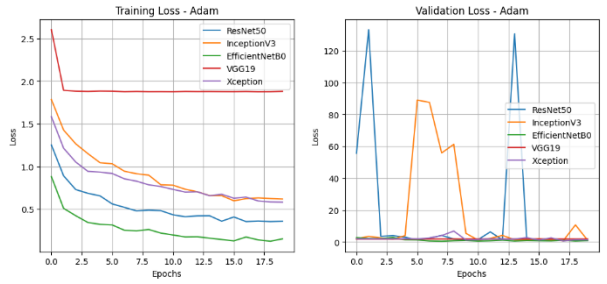


Figure 7: Training and validation loss plots for Adam

In the Nadam training accuracy Figure 8 VGG19 exhibits a notable decline before rebounding, whereas ResNet50 and EfficientNetB0 reach roughly 0.95-1.0. The validation accuracy varies; EfficientNetB0 has the highest and most stable value.

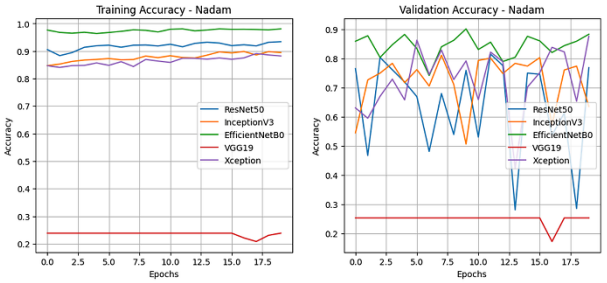


Figure 8: Training and validation accuracy plots for Nadam

Using Nadam training loss generally decreases for the ResNet50, InceptionV3, EfficientNetB0, Xception, and VGG19 models; except for VGG19, which shows a dramatic increase around epoch 16, indicating overfitting, validation loss is negligible for the majority of models but ResNet50 peaks at 13,17.

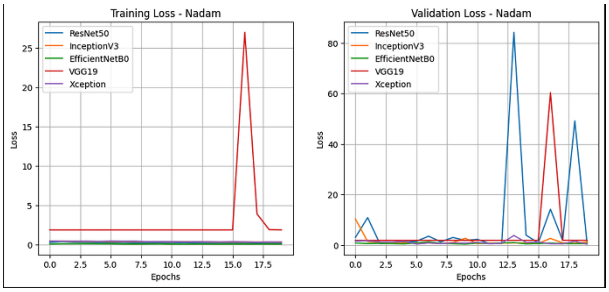


Figure 9: Training and validation loss plots for Nadam

The RMSprop optimizer is used to evaluate training and validation accuracy over twenty epochs for the 5 models. The results show in the Figure 10 a steady increase in training accuracy, with VGG19 lower while EfficientNetB0 and ResNet50 approach 1.0. EfficientNetB0 remains stable at about 0.9, while other validation accuracy model, including VGG19, exhibit instability and uneven generalization.

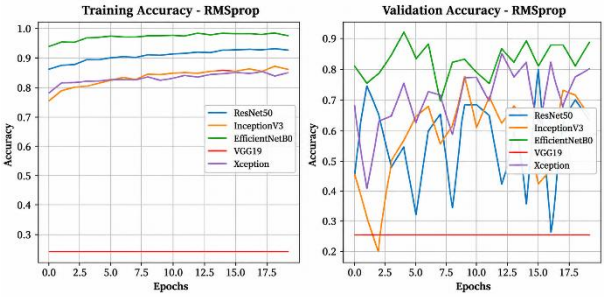


Figure 10: Training and val accuracy plots for RMSprop

All models exhibit a general decline in training loss, except VGG19 using RMSprop.

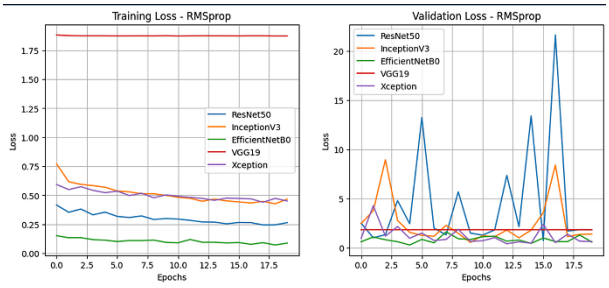


Figure 11: Training and validation loss plots for RMSprop

5.2 Training and Validation Performance

The five CNN models—ResNet50, InceptionV3, Efficient NetB0, Xception, and VGG19—were trained using fixed hyper parameters. EfficientNetB0 and ResNet50 achieved faster convergence and higher training accuracy, while VGG19 exhibited slightly slower convergence due to its deeper architecture. Figure 12 illustrates the training and validation curves for all models with their respective best-performing optimizers.

5.3 Best Model–Optimizer Selection

The best-performing optimizer for each model was selected based on the highest validation accuracy. Table I summarizes the results. EfficientNetB0 achieved the highest validation accuracy with Nadam, whereas ResNet50 also performed best using Nadam.

Table 1. Best Performing Model–Optimizer Combination

Model	Best Optimiizer	Train Accuracy	Validation Accuracy
ResNet50	Nadam	0.9339	0.7492
InceptionV3	Nadam	0.8946	0.6852
Efficient NetB0	Nadam	0.9867	0.8929
Xception	Nadam	0.8824	0.8767
VGG19	Nadam	0.423	0.321

5.4 Confusion Matrix Analysis

To better evaluate classification performance, confusion matrices were generated for each model with its best optimizer. As shown in Figure 13 EfficientNetB0 and Xception achieved minimal misclassifications, indicating better generalization, while VGG19 showed slightly more confusion among visually similar classes.

5.5 Test Accuracy Comparison

Table II summarizes the test accuracy of all five CNN models using their best optimizers. EfficientNetB0 achieved the highest accuracy, while VGG19 had the lowest performance among the tested architectures. Adam, RMSprop, and Nadam, to evaluate their impact on model performance. A well-curated dataset of tea leaf images was used, and extensive data augmentation techniques were applied to improve generalization. The evaluation was carried out using training accuracy, validation accuracy, training loss, validation loss, test accuracy, confusion matrices, and classification reports.

5.6 Classification Reports

The classification reports summarize per-class precision, re-call, and F1-scores for each model. For example, Efficient-NetB0 optimized with Adam achieved the highest overall accuracy and balanced performance across all disease classes.

Table 2. Test Accuracy for Different Models

Model	Test Accuracy
Efficient NetB0+Adam	92.16
VGG19+Nadam	24.91
InceptionV3+Nadam	73.92
Xception+Nadam	87.86
ResNet50+Nadam	78.86

6. CONCLUSION

This study presented a comprehensive analysis of five pre trained deep convolutional neural network (CNN) models—ResNet50, InceptionV3, EfficientNetB0, Xception, and VGG19—for the classification of tea leaf diseases. The experiments were conducted using three different optimizers Adam, RMSprop, and Nadam, to evaluate their impact on model performance. A well-curated dataset of tea leaf images was used, and extensive data augmentation techniques were applied to improve generalization. The evaluation was carried out using training accuracy, validation accuracy, training loss, validation loss, test accuracy, confusion matrices, and classification reports. Among all the models, EfficientNetB0 combined with the Nadam optimizer demonstrated the highest overall accuracy and the best generalization capability, achieving a test accuracy of 92.16%. ResNet50 also showed competitive performance, particularly in achieving high training accuracy, while InceptionV3 performed reasonably well but exhibited slightly lower generalization compared to EfficientNetB0. Xception, optimized with Nadam performed moderately, showing good capability in distinguishing certain complex disease categories, while VGG19 showed comparatively slower convergence due to its larger parameter size. The analysis of the confusion matrices revealed that EfficientNetB0 and Xception achieved minimal misclassification across most disease classes, reflecting their strong generalization capabilities.

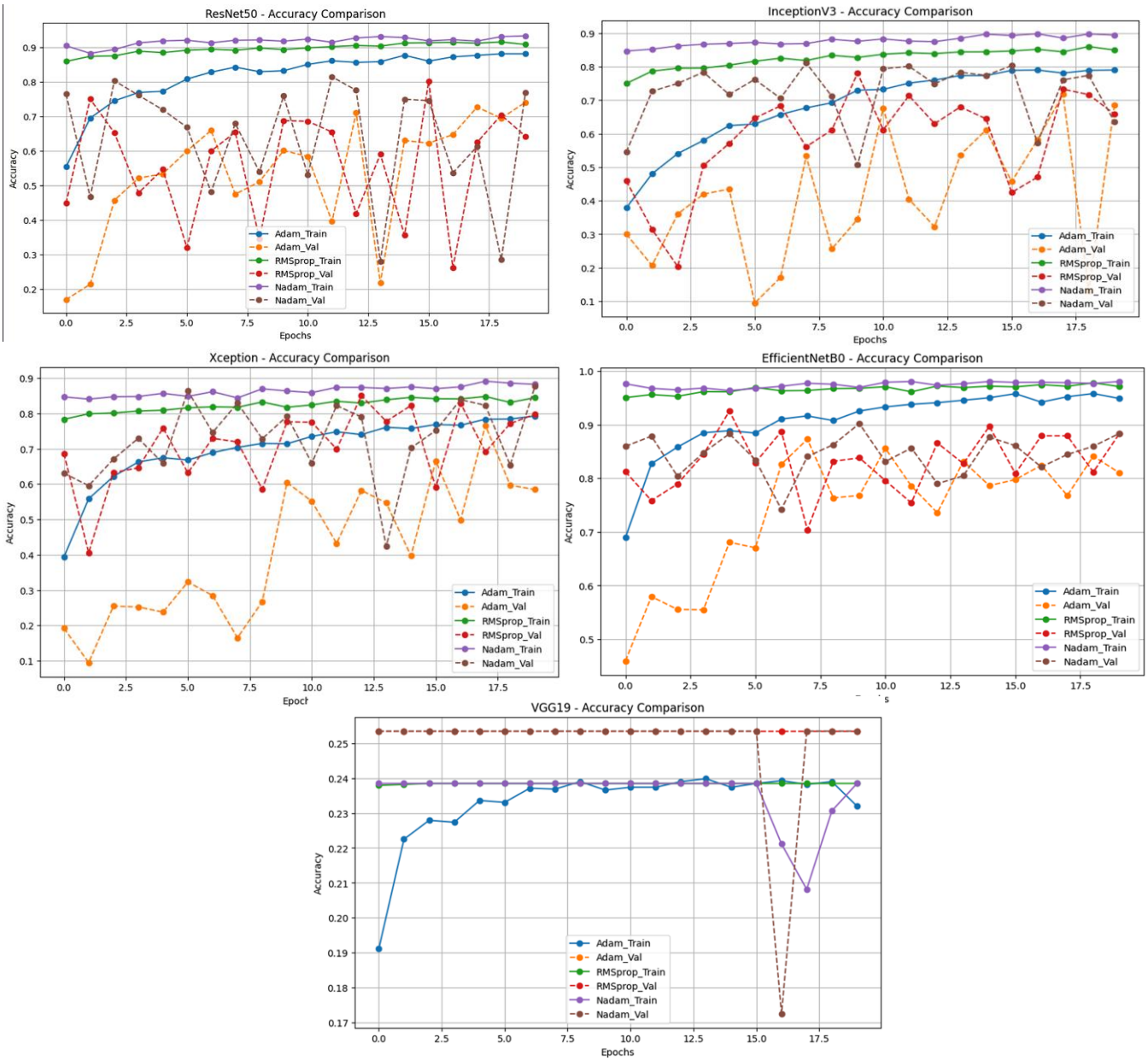
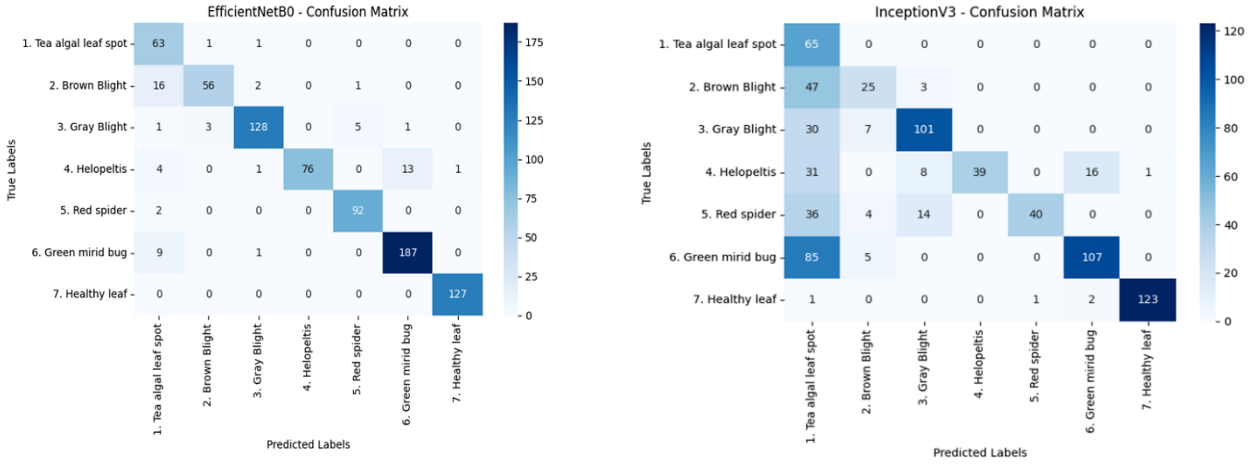


Figure 12: Training and validation accuracy plots for three optimizers across models



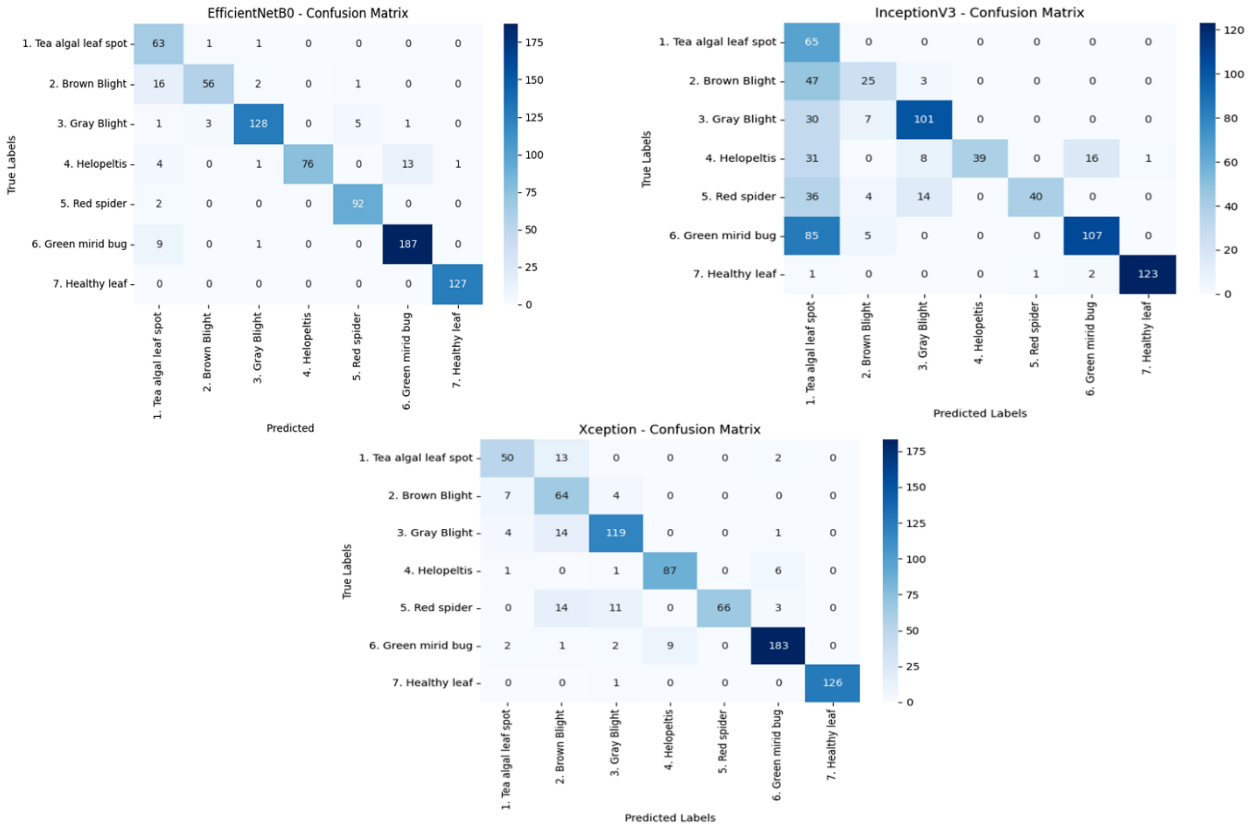


Figure 13: Confusion matrices on test set for each model (best optimizer)

The classification reports further validated these findings, with EfficientNetB0 consistently achieving the highest precision, recall, and F1-scores among all tested models. Overall, the results demonstrate that selecting an appropriate optimizer significantly impacts the performance of CNN models in disease classification tasks. The Nadam optimizer proved to be the most effective for the majority of models. The findings highlight the potential of deep learning-based transfer learning techniques for automating plant disease detection, which can assist farmers and researchers in early diagnosis and sustainable crop management.

7. FUTURE WORK

Although the proposed approach achieved promising results, there is room for further improvement. Future research could focus on:

- Expanding the dataset with more diverse samples under varying environmental conditions.
- Incorporating attention based architecture (e.g., vision Transformer) to improve feature extraction.
- Applying ensemble learning techniques to combine multiple models for enhanced robustness.
- Developing a lightweight real-time detection system for deployment on mobile and IoT-based devices to assist farmers directly in the field.

This study demonstrates the effectiveness of transfer learning for tea disease classification and sets a strong foundation for deploying AI-powered tools in precision agriculture.

8. REFERENCES

[1] X. Deng, T. Chen, C. Photong (2024, June).Automated Identification of Tea Leaf Disease. Operational Research in Engineering Sciences: Theory and Applications (pp. 315-336).

[2] S. Zhang, Chulanlei Zhang, X. Wang, Cucumber leaf disease identification with global pooling dilated convolutional neural network (pp.422-430).

[3] Aditya Khamparia, G. S (2020). Seasonal Crops Disease Prediction and Classification Using Deep Convolutional. Encoder Network.Springer (pp. 818-837).

[4] J. Lu, R. Ehsani, Y. Shi, A.I. Castro, S. Wang (2018). Detection of multi tomato leaf diseases (late blight, target and bacterial spots) in different stages by using a spectral-based sensor.

[5] S. Mehnaz, Md. Touhidul Islam (2025, Jan).Rice Leaf Disease Detection: A Comparative Study between CNN Transformer and Non-neural Network Architectures.

[6] Chandrala J (2017, September).Transfer learning: Leveraging Pretrained Models for New Tasks (pp. 809-815).

[7] B. Tugrul, E. Elfatimi, R. Eryigit (2022). Convolutional Neural Networks in Detection of Plant Leaf Diseases: A Review (pp.1-22).

[8] Roopa R, R. Lingam, S. Kumar, A. Suresh, P. Balaji (2025, June). A Generalized Deep Learning Approach for Cross-Crop Plant Disease Detection Using the Plant Village Dataset (pp.1-17).

[9] Park, H. N (2020) Unsupervised image translation using adversarial networks for improved plant disease recognition. Computers and Electronics in Agriculture, (pp. 105-117).

[10] Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, George E. Dahl (2019, October). On Empirical Comparisons of Optimizers for Deep Learning.

[11] Liana Trihardianingsih, A. S., & T. H. (2023). Classification of Tea Leaf Diseases Based on ResNet-50 and Inception V3. Sinkron: Jurnal dan Penelitian Teknik Informatika, (pp. 1564–1573).

[12] T.L. Rama, V.Praveen , G. Jayanth , S. Jaya. Plant Leaf Disease Detection Using Machine Learning Algorithms. AICTE (pp. 1-80).