Nanotechnology Perceptions
ISSN1660-6795
www.nano-ntp.com

Advancing Crop Disease Identification Using Machine Learning and Transfer Learning Approaches

Ijtaba Saleem Khan*[1], Sifatullah Siddiqi[2]

[1]*Research Scholar, Integral University, Lucknow (UP), India, ijtaba007khan@gmail.com*
[2]*Assistant Professor, Integral University, Lucknow (UP), India, Sifatulla@iul.ac.in*

Abstract: Convolutional Neural Networks (CNNs) have changed how we handle image processing. Different types of CNNs offer various benefits in how well they perform and how efficient they are with computing resources. This paper looks at five well-known CNN types: ResNet151, Xception, DenseNet201, InceptionV3, and EfficientNetB7. Each has special features that help improve deep learning. ResNet151 uses skip connections to solve the problem of vanishing gradients, allowing very deep networks to be trained for recognizing images and detecting objects. Xception builds on the Inception design by using depthwise separable convolutions, which makes it more efficient while still performing well. DenseNet201 connects layers closely, promoting better flow of gradients and reuse of features, which helps in tasks that need efficient computing. InceptionV3 includes multi-scale convolutional layers to optimize computing costs while maintaining high accuracy, making it great for large image classification. Finally, EfficientNetB7 uses a scaling method that achieves top accuracy with fewer parameters, making it effective for tasks that require precision and efficiency.
Keywords: crop disease Identification, classification accuracy, recognition ML, transfer deep learning, agricultural productivity

1. Introduction

Crop diseases significantly threaten agriculture by decreasing both crop yield and quality. Early detection is vital for farmers to manage disease spread and optimize pesticide use, benefiting both the environment and human health. Traditional diagnostic methods, such as chemical analyses and spectroscopy, can be costly and require specialized skills, driving the need for faster and more accessible detection techniques [1][2].

Recent advancements in image processing, Deep learning (DL) and machine learning (ML) provide promising alternatives for early disease identification [3]. Since various pathogens bacteria, fungi, and viruses primarily affect crop leaves, our research specifically targets leaf disease detection. Our study aims to enhance detection model performance by balancing accuracy and classification errors. We compare multiple ML and DL algorithms with various computer vision technique & performance metrics, ultimately identifying the most effective model for real-time leaf disease classification [4][5].

This research is distinct in its use of a large, diverse dataset of multiple crop types and the application of both ML and DL approaches, allowing for comprehensive model training [6]. A key innovation is the combination of deep learning optimizers with convolutional neural networks (CNNs), yielding improved results compared to previous studies [7].

The following sections will review relevant literature, outline our methods, present detailed experimental results, and talk about potential avenues for further research.

2. Related Work

Recent studies have explored various ML and DL algorithms for detecting crop leaf diseases [8]. A notable survey by Orchi et al. [9] provides an overview of these methods, focusing on techniques such as Support Vector Machines (SVM), Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN). This survey aims to help researchers understand the strengths and limitations of each technique in plant disease detection. There is an increasing trend to shift from traditional ML algorithms to DL methods for leaf image classification. For instance, Argüeso et al. [10] proposed an innovative method utilizing a metric within few-shot learning that employs triplet loss and a one-dimensional classifier. Their method proved effective in classifying plant leaf diseases with minimal training data, achieving over 90% correctness with no more than 80 annotated descriptions for various diseases according to class.

In [11], Pantazi et.al developed a method to detect diseases in grape leaves, including downy mildew, powdery mildew, healthy leaves, and black rot. Their technique involved classifying different types of leaves using a local binary approach. patterns for feature extraction, using the Grab Cut algorithm for image segmentation. The model demonstrated outstanding generalization, achieving a 95% accuracy across various leaf samples from different plant species. Specifically, they achieved perfect classification for 44 out of 46 tested combinations of plant diseases, with over 50% of the cases reaching 100% identification accuracy. These advancements highlight the critical importance of resolving conflicts among classifiers, enabling accurate identification of conditions that may belong to single or multiple classes.

In their research, Arora et al. [12] The Deep Forest algorithm was used to classify diseases in maize leaves. The dataset included three types of infected leaves and a single type of healthy leaves. This method outperformed traditional deep neural networks in accuracy, benefiting from the combined strengths of ensemble decision trees and neural network architectures. The trend of mobile-based automatic detection systems for disease diagnosis is on the rise.

For instance, Tang et al. [13] anticipated a crossbreed lightweight convolutional neural network (CNN) approach to detect grape diseases, including black measles & black rot, They improved the process by using channel-wise attention mechanisms. This enhancement led to better outcomes. Shuffle Net architecture The use of compression and excitation blocks as CA mechanism has led to impressive results. The model was significantly reduced in size from 227.5 MB to just 4.2 MB and achieved an outstanding accuracy rate of 99.14% when evaluated on a dataset containing 4,062 images of grape leaves.

Another innovative approach involves the Extreme Learning Machine (ELM) algorithm, as discussed by Bhatia et al. [14]. They applied the ELM on a real-time dataset intended for Tomato Powdery Mildew Disease (TPMD), which was notably imbalanced. To address this, they employed several resampling techniques, including d The methods used to balance the dataset before training the model included Importance Sampling (IMPS), Synthetic Minority Over-sampling (SMOTE), Random Under Sampling (RUS), and Random Over Sampling (ROS). The performance of the Extreme Learning Machine (ELM) was assessed using both the original and the re-sampled datasets. Evaluation metrics such as Area Classification Accuracy (CA) and Under the Curve (AUC) were employed to measure predictive accuracy. The findings showed that the ELM algorithm performed effectively performed enhanced on the re-sampled data, particularly with the IMPS technique, achieving maximum values of 88.57% for AUC and 89.19% for CA. Additionally, the Deep Forest model achieved an accuracy of 96.25%, further validating its effectiveness.

Table 1 offers a concise overview of recent studies that explore the use of machine learning (ML) and deep learning (DL) techniques for detecting crop diseases. Each entry highlights key research contributions, methodologies, and findings that showcase significant advancements in agricultural technology aimed at improving disease diagnosis.

Table1:Recent Advances in ML and DL for Crop Disease Detection

Author(s)	Pantazi, X. E., Tsouros, D. C., & Karatzas, G. P.	Orchi et al.	Argüeso, P., Ros, M., & Pérez, J.	Arora, R., Sharma, P., & Kumar, A.	Tang, Z., Wang, Y., & Zhang, J.	Bhatia, P., Kumar, V., & Singh, R.
Year	2020	2021	2021	2021	2021	2022
Title	Identification of grape diseases using local binary patterns and Grab Cut segmentation	Comprehensive Survey of Detection Techniques	Few-shot learning for plant disease classification: A distance metric approach	The deep forest method is used to identify diseases in maize leaves.	Hybrid lightweight CNN for real-time grape disease detection using channel-wise attention	Extreme learning machine for tomato powdery mildew disease prediction using re-sampling technique
Source	Sensors	Computers and Electronics in Agriculture	Computers and Electronics in Agriculture	Bio-systems Engineering	Artificial Intelligence in Agriculture	Journal of Plant Pathology
Methodology	Integration of local binary patterns and segmentation	In-depth review of ML and DL methodologies	Few-shot learning framework	Deep Forest algorithm application	Hybrid lightweight CNN model	Application of Extreme Learning Machine (ELM)

Key Findings	Achieved 95% accuracy across diverse plant samples, highlighting robustness and generalizability for disease recognition.	This survey clarifies the comparative strengths and limitations of various techniques like SVM, ANN, CNN, and RNN in plant disease detection	Achieved over 90% accuracy using only 80 annotated images per class, indicating the usefulness of deep learning in scenarios with limited training data.	Outperformed conventional deep neural networks by leveraging ensemble decision trees, marking significant advancements in automated disease diagnosis.	Model reduced from 227.5 MB to 4.2 MB, achieving 99.14% accuracy on a dataset of 4,062 grape leaf images, indicating efficiency for mobile environments.	Improved predictive performance through re-sampling techniques, achieving 88.57% AUC and 89.19% CA, and validating usefulness in practical application.

3. Study Design and Methodology

We carried out a detailed comparison of advanced DL and ML models to categorize crop diseases in two categories using Python. To perform this analysis, we used several libraries, including NumPy, Pandas, Scikit-learn, Keras, and TensorFlow. Additionally, we improved the performance of CNN models by the use of the first phase's top-performing model for training with various activation and optimization functions suitable for deep learning.

Dataset

For this study, we utilized the Plant Village Dataset [20], which consists of a wide array of photos of crop leaves taken using common digital cameras under various weather situations. The dataset is compiled from various sources, contributing to its rich diversity, which is beneficial for machine learning tasks, particularly in the field of DL. It has 42,854 images in total, categorized into 23 distinct classes representing eight different types of crops. Each class consists of pairs of healthy and diseased leaves.

Table 2: Dataset used

Crop Leaf	Diseases	Number of Images
Rice	Bacterial Blight	630
	Rice Blast	621
	Healthy	1645
Wheat	Fusarium Head Blight	4997
	Healthy	1478
Sugarcane	Sugarcane Borer	1052
	Healthy	854
Tomato	Early Blight	1192
	Late Blight	513
	Healthy	1162
Potato	Early Blight	1000
	Late Blight	1383
	Healthy	423

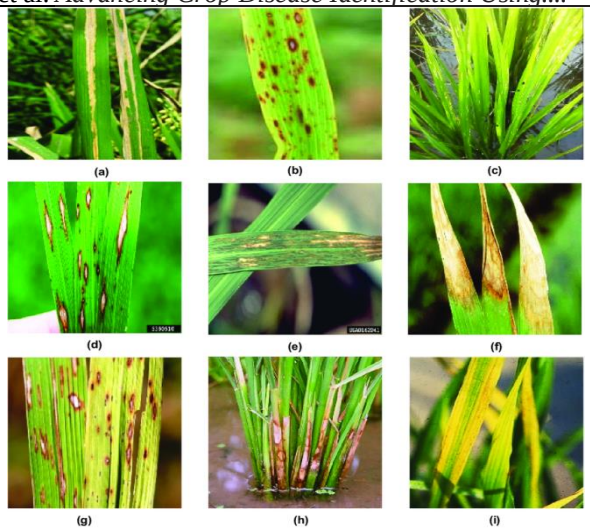

Figure 1: Diseases of rice leaves (a) Hispa (b) Brown spot (c) Leaf Blast (d) Leaf Streak (e) Leaf Scald (f) Narrow Brown Spot (g) Sheath Blight (h) Tungro (i) Bacterial Leaf Blight.

Figure 2. (a) Wheat heads with non-infected spikelets. (FHB) (b) Wheat head with infected pikelets.

(a) (b) (c)

Figure 3. Potato leaves: (a) healthy, (b) early blight and (c) late blight

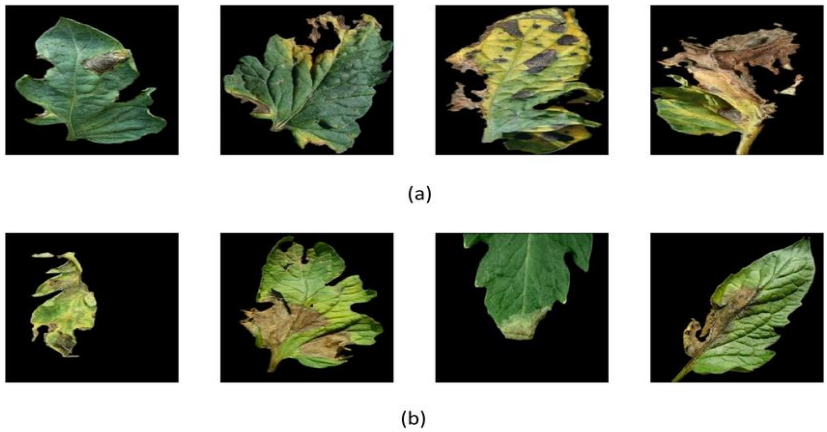

Figure 4: Tomato leaves: (a) Early blight images (b) Late blight images

(a) (b) (c)

Figure 5. Sugarcane: (a) Healthy, (b) and (c) Sugarcane Borer

Machine Learning Approach

In the implementation of traditional machine learning algorithms, certain essential pre-processing steps are crucial to ensure effective model training and performance. These fundamental procedures are illustrated in Figure 2 below. Pre-processing typically involves tasks such as data cleaning, normalization, and feature selection, which help to enhance the quality of the input data. Properly prepared data can significantly improve the accuracy and efficiency of machine learning models, allowing for better generalization and performance on unseen data.

Pre-processing of images

Initially, the RGB descriptions are scaled to a less important pixel dimension to enhance computational efficiency. Subsequently, To lessen blast in the photos, Gaussian blur gets used. The RGB color space presents challenges for isolating image intensity due to its

combination of shadows and highlights, making it less effective for background removal. To address this limitation, The RGB images of crop leaves are transformed into the HSV color space, which stands for Hue, Saturation, and Value effectively separates color information from intensity, facilitating improved processing for further analysis.

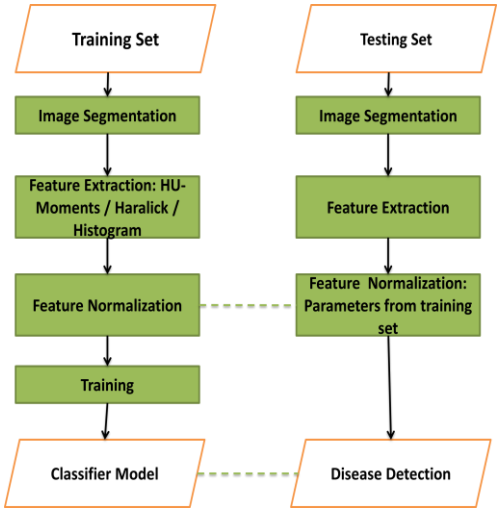

Figure 6. Schematic Representation of the Training and Testing Phases.

Removing backgrounds and isolating the affected areas

In our study, we implemented a mask generation technique for effective segmentation This approach is grounded in the HSV (Hue, Saturation, Value) color space. This technique enables us to differentiate between healthy and diseased regions of crop leaves, where green hues signify healthy areas, while brown hues indicate the presence of disease. Background suppression plays a very important role in preserving the quality of features in the images, enabling more accurate analysis and classification. By focusing on the relevant portions of the images, we enhance the performance of subsequent machine learning algorithms, ensuring that the extracted features represent the actual condition of the crop leaves. This strategy is in line with image processing best practices, and recent research has shown how useful color space modifications are for detecting agricultural diseases [17].

Figure 7. Input Image of a Leaf (Left) and the Process of Removing the Background (Right)

Feature Extraction

Selecting suitable features is a critical and challenging aspect of implementing ML algorithms, requiring extensive analysis and domain expertise. Three feature descriptors were used in this study:

1. **Hu Moments Descriptor**: By examining an object's contour, this descriptor is used to describe and measure its shape. Initially, the color images are converted to grayscale, from which seven invariant moments are calculated. These moments are robust against rotation, translation, and scaling changes, allowing for independent object recognition. The moments are defined mathematically as follows:

2. **Haralick Texture Descriptor**: This descriptor quantifies the texture Images in color are initially changed to grayscale to make it easier to extract texture features. The Haralick texture features are obtained from the Gray Level Co-occurrence Matrix (GLCM), which evaluates how often pairs of pixels with certain values appear together.at specified distances. The features derived from the GLCM.

3. **Color Histogram descriptor** : in this we analyze the color characteristics of the images. Each channel's histogram consisted of 26 bins, resulting in a total of 78 features when aggregated across all three color channels. After feature extraction, these features were consolidated using the Numpy np.stack function.

To prepare the data for machine learning, 20% was used for testing and 80% for training from the dataset, with labels encoded for machine readability. Min-Max scaling normalized feature values to a range of 0 to 1, ensuring equal contribution of features to model performance and reducing bias from unscaled values.The features were saved in an HDF5 file, which efficiently manages large and complex datasets due to its hierarchical structure [18]. Finally, six ML algorithms were used for training, and a Ten-fold cross-validation was conducted to validate the performance of model, ensuring robust evaluation metrics and reliable outcomes.

Classification algorithms

Multiple classification algorithms are compared for crop disease detection. Below, I will summarize and discuss the key machine learning algorithms mentioned, focusing on their strengths and weaknesses as applied to classification tasks.

1. Support Vector Machine (SVM)

- **Description**: SVM is a supervised learning algorithm that finds the optimal hyperplane in a high-dimensional space to separate different classes. It uses kernel functions to handle nonlinear data. The Radial Basis Function (RBF) kernel is particularly powerful in nonlinear classification.
- **How it Works**: In SVM, the goal is to find a hyperplane (in 2D, this would be a line) that maximizes the margin between two classes. In order to split the classes by a linear hyper plane, nonlinear data is converted into bigger dimensions using kernel functions as polynomial, RBF, and sigmoid.

- **Key Features**:
  - **Kernel Trick**: Maps data into high-dimensional space to find linear separability.

- **Regularization**: Avoids overfitting by using the regularization parameter C.
- **Best Kernel for Dataset**: The RBF kernel with C=100 yielded the best results in the study.

2. K-Nearest Neighbour (KNN)

- **Description**: It is a straightforward, non-parametric classification technique that uses the majority vote of a sample's k-nearest neighbours in the feature space to determine which class it belongs to.
- **How it Works**: KNN calculates the distance (e.g., Euclidean distance) between the test point and all training points. The class to which the majority of the test point's k-nearest neighbours belongs is chosen.
- **Key Features:**
  - **No Training Phase:** KNN is a lazy learner and does not require a separate training phase.
  - **Sensitivity to k:** Smaller values of k capture non-linear patterns but are sensitive to noise. Larger values of k reduce noise sensitivity but may not capture complex boundaries.

3. Random Forest (RF)

- **Description**: This approach for ensemble learning constructs several decision trees and aggregates their predictions. Several decision trees are constructed using this ensemble learning approach, and their predictions are then combined.
- **How it Works**: It constructs multiple decision trees by bootstrapping (randomly sampling the data with replacement) and then aggregates their results. Randomness is introduced by selecting a random split of features for each tree.
- **Key Features**:
  - **Ensemble Method:** Reduces over fitting by averaging out the decision trees' biases.
  - **High Accuracy:** It is known for its strong performance, especially when the dataset is large and complex.

4. Naive Bayes (NB)

Description: The Bayes theorem serves as the foundation for this probabilistic classifier. The computation is made simpler by assuming that the characteristics are conditionally independent given the class label.

- **How it Works**: For each class, Naive Bayes calculates the class's prior probability and possibility of the attributes given the class. The class having the greatest posterior probability is then selected.
- **Key Features**:
  - **Assumption of Independence:** In actual data, the notion that features remain independent is frequently incorrect, but NB still works surprisingly well in many cases.
  - **Efficiency:** It is computationally efficient and performs well with small datasets or when features are independent.

5. Latent Dirichlet Allocation (LDA)

- **Description**: LDA is a generative probabilistic model used to classify data by finding topics or latent variables. It is often used for text classification, however can also be useful to other domains.
- **How it Works**: LDA assume that each sample is a combination of a small number of topics. It tries to find the underlying topics based on the observed data.
- **Key Features**:
  - **Topic Modeling**: LDA finds hidden structures in data by modeling the distribution of topics.

6. Classification and Regression Trees (CART)

- **Description**: CART is a decision tree algorithm that divides the data according to feature values into subsets. It is used equally for classification and regression tasks.
- **How it Works**: Recursively dividing the data according to feature values which minimizes a loss function, for example Gini impurity in classification tasks builds the tree. The leaves of the tree represent the predicted class.
- **Key Features**:
  - **Recursive Binary Splitting:** CART uses recursive binary splitting to build decision trees.
  - **Binary Tree Structure:** The decision tree structure is binary, meaning each node has only two branches.

Table 3: Machine Learning Algorithms with their key features, strengths, and weaknesses

Algorithm	Key Features	Strengths	Weaknesses
SVM	Uses kernel trick, handles non-linear data, regularization to avoid overfitting.	Works well with high-dimensional data, powerful for complex problems.	Sensitive to the choice of kernel and parameters.
KNN	Non-parametric, simple, based on majority voting of neighbors.	Easy to implement, no training phase.	Sensitive to noise and large datasets.
Random Forest	Ensemble method, reduces overfitting, highly accurate, robust against overfitting.	Best-performing, handles large datasets well.	Slower to predict, less interpretable.
Naive Bayes	Based on Bayes' theorem, assumes independence of features.	Computationally efficient, simple.	Assumes features are independent, not always valid.
LDA	Models the data as a mixture of topics or classes, suitable for multi-class classification.	Good for text classification, works well with continuous data.	Assumes normally distributed data.
CART	Recursive binary splitting, binary tree representation, used for both regression and classification tasks.	Easy to interpret, handles both categorical and numerical data.	More likely to overfitting if not pruned.

CNN Model Architecture

- **Convolutional Layers**: The twelve convolutional layers into this model begin with 64 layers of filters and progressively increase to 128, 256, 512 and so on at later levels. A 3x3 kernel is used for all convolution operations.
- **Activation Function**: Inside the hidden layers, ReLU serves as the activation function**.** After each convolutional layer, **Leaky ReLU** is applied to prevent the "dying ReLU" problem where neurons stop responding during training.
- **Max Pooling Layer**: **MaxPooling2D** is employed after each convolutional block to reduce the spatial dimensions of the feature maps, thereby reducing computational complexity.
- **Fully Connected Layer**: The output from the convolutional layers is flattened and passed through a fully connected layer that learns the relationships among the features.
- **Final Layer**: A dense layer with a **Softmax activation function** is used for multi-class

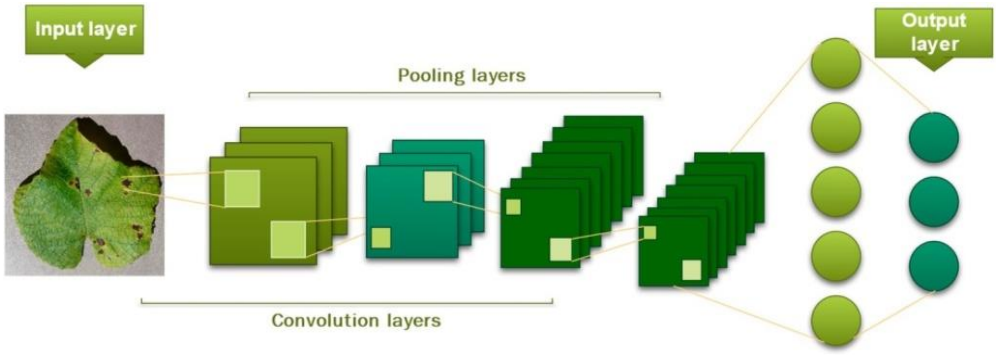

Figure 8: General Architecture of CNN [23]

Table 4: Deep Learning Activation Functions

Activation Function	Formula	Range	Use Cases	Pros	Cons
Sigmoid	$\sigma(x) = \frac{1}{1 + e^{-x}}$	(0, 1)	Binary classification, output layer for probabilities	Easy to compute, outputs between 0 and 1	Vanishing gradient problem, not zero-centered
Tanh	$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	(-1, 1)	Hidden layers, especially in RNNs	Zero-centered, faster convergence than sigmoid	Vanishing gradient problem
ReLU	$\text{ReLU} = \max(0, x)$	$[0, \infty)$	Hidden layers in CNNs, RNNs, general-purpose networks	Simple, fast convergence, combats vanishing gradients	Dying ReLU problem (neuron stops learning for some inputs)

Leaky ReLU	Leaky LeRU $= \max(\alpha x, x)$	(-∞, ∞)	Hidden layers, solves dying ReLU problem	Prevents dying ReLU problem, faster convergence	Requires tuning of α\alphaα, might be complex to implement
PReLU	PReLU (x) $= \max(\alpha x, x)$ (Where α is learned)	(-∞, ∞)	Hidden layers, deep networks	Learns the negative slope, flexible	Additional parameter to tune
ELU	ELU (x) $= \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases}$	(-α, ∞)	Hidden layers in deep networks, helps prevent vanishing gradients	Better performance in deep networks, helps prevent vanishing gradients	Requires an additional parameter (α\alphaα)
SELU	SELU (x) $= \lambda . ELU(x)$	(-∞, ∞)	Self-normalizing networks, deep learning	Self-normalizing, faster convergence	Requires careful initialization, not always compatible
Swish	Swish $(x) = x.\sigma(x)$	(-∞, ∞)	Hidden layers in deep networks	Avoids dying ReLU problem, often outperforms ReLU	Computationally expensive compared to ReLU

for machine learning, 20% was used for testing and 80% for training from the dataset, with labels encoded for machine readability. Min-Max scaling normalized feature values to a range of 0

Optimization Methods for Deep Learning

1. **SGD (Stochastic Gradient Descent)**:
   - **Description**: One of the most straightforward and commonly used optimizers. It updates model parameters by using a constant learning rate. This method helps in fast convergence, but it can be sensitive to the learning rate.
   - **Key Feature**: Constant learning rate for all parameters, relatively simple and fast convergence.
   - **Use Case**: Suitable when you have large datasets or if you're okay with some degree of fluctuations during training.
2. **Adagrad (Adaptive Gradient Algorithm)**:
   - **Description**: This optimizer adjusts the learning rate for each parameter based on its update frequency. Parameters that update frequently will have smaller learning rates, whereas those that are rarely updated will have larger learning rates.
   - **Key Feature**: Differentiates the learning rate for each parameter.
   - **Use Case**: Good for sparse data, such as text classification or datasets with many zero values.
3. **RMSProp (Root Mean Square Propagation)**:
   - **Description**: RMSProp modifies Adagrad by decreasing the learning rate exponentially. It works well in scenarios where the learning rate should decay over time.

- **Key Feature**: Exponentially decays the learning rate to make it more adaptable over time.
- **Use Case**: Often used in recurrent neural networks (RNNs) and deep learning tasks that require faster convergence.

4. **Adadelta**:
   - **Description**: An extension of Adagrad, Adadelta uses a moving average of past gradients and adapts the learning rate over time. Unlike Adagrad, which accumulates squared gradients, Adadelta limits this accumulation, making it more practical.
   - **Key Feature**: Prevents the accumulation of gradients over time, making it better for continuous learning over many iterations.
   - **Use Case**: Used when you want a smoother update with faster learning, especially in deep learning tasks with longer training times.
5. **Adam (Adaptive Moment Estimation)**:
   - **Description**: Adam combines the features of both **Adagrad** and **RMSProp**. It maintains two moving averages: one for the first moment (mean) and one for the second moment (variance). This allows Adam to adjust the learning rate for each parameter in a more stable and adaptive way.
   - **Key Feature**: Uses both momentum and adaptive learning rates to accelerate training, often leading to better performance with less tuning of hyper parameters.
   - **Use Case**: Extremely popular in a wide range of applications, especially for large-scale deep learning tasks. It generally performs well without needing careful hyper parameter tuning.

Hyper parameter Tuning

1. **Batch Size**: 64
   - The batch size determines how many training samples are processed before the model's internal parameters are updated. A batch size of 64 is used here to improve convergence and stability.
2. **Optimizer**: Adam
   - Adam (Adaptive Moment Estimation) is used as the optimizer. It adapts the learning rate of each parameter during training, providing better performance for many types of deep learning models.
3. **Loss Function**: Categorical Cross-Entropy
   - Categorical Cross-Entropy is used for multi-class classification problems, where the output layer uses a Softmax activation function. This loss function calculates the distance between the predicted probability distribution and the actual class distribution.

4. **Learning Rate**: 0.001
   - The learning rate defines the size of the step the model takes during each update of the weights. A learning rate of 0.001 is chosen here to fine-tune the weights during training without causing instability.
5. **Decay Learning Rate**: 0.00001
   - This parameter controls the rate at which the learning rate decays during training. A small decay ensures that the learning rate decreases slowly, helping the model converge steadily over time.
6. **Epochs**: 120
   - The complete dataset has been run through the model for 120 times since it is trained for 120 epochs. This figure prevents over fitting and enables the model to learn more effectively from the data.

Key Points in Hyper parameter Tuning

- **Batch Size**: The increased batch size of 64 helps speed up training and stabilize the learning process.
- **Learning Rate**: 0.001 as the learning rate ensures the model adjusts weights gradually, leading to stable convergence.
- **Optimizer**: The Adam optimizer is efficient and effective, adapting the learning rate for each parameter during training.
- **Epochs**: Training for 120 epochs allow the model enough time to learn from the data, improving its performance further without overfitting.

1. Experimental Results

This section compares six traditional machine learning algorithms with five DL models that use CNN. The goal is to find out which model works best. The study examines how each algorithm performs with different activation functions and optimization techniques in deep learning. The traditional machine learning models were developed using scikit-learn, while the deep learning models were created with Keras and TensorFlow. All models were trained on Google Colaboratory, utilizing the GPU for traditional algorithms and the TPU for deep learning models. 80% of the dataset was used for training, and 20% was used for testing to assess how well each model performed.

Performance Metrics:

Severa performance metrics were used to assess the trained models, include, recall, precision, f1-score, and accuracy

- **Precision**: This measure shows the proportion of real positive samples that match the projected positive samples. It's calculated as:

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

- **Recall (or Sensitivity)**: This measure shows The proportion of real positive samples that the model accurately predicted. It's calculated as:

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

- **F1-Score**: The F1-score is a single score that is calculated by taking the harmonic mean of precision and recall. It is used to evaluate the balance between the two. It is calculated by:

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

- **Accuracy**: Accuracy measures the fraction of correct predictions made by the model. It is defined as:

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

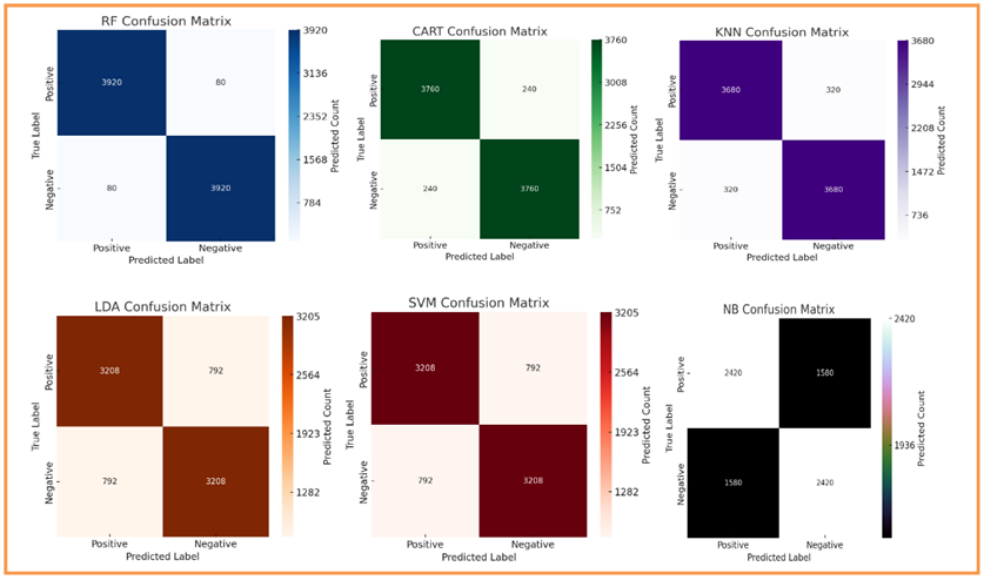

Figure 9. Confusion matrix for ML algorithms

Table 5: Performance Machine Learning Models

Model	Accuracy	Precision	Recall	F1-Score
Random Forest (RF)	98.00%	94.30%	95.50%	94.90%
Classification and Regression Tree (CART)	94.00%	89.30%	90.50%	89.90%
K-Nearest Neighbors (KNN)	92.00%	90.50%	91.70%	91.10%
Linear Discriminant Analysis (LDA)	80.20%	91.50%	92.70%	92.10%

Support Vector Machine (SVM)	80.20%	93.20%	94.40%	93.80%
Naive Bayes (NB)	60.50%	83.30%	85.00%	84.20%

Among classical machine learning algorithms, Random Forest (RF) outperformed the others with an accuracy of 98.0%, excelling in precision, recall, and f1-score. K-Nearest Neighbors (KNN) and CART achieved moderate results, with KNN reaching 92.0% accuracy and CART scoring 94.0% accuracy. In contrast, Naive Bayes (NB) had the lowest performance, with an accuracy of 60.5%, making it less effective than the other ML algorithms.

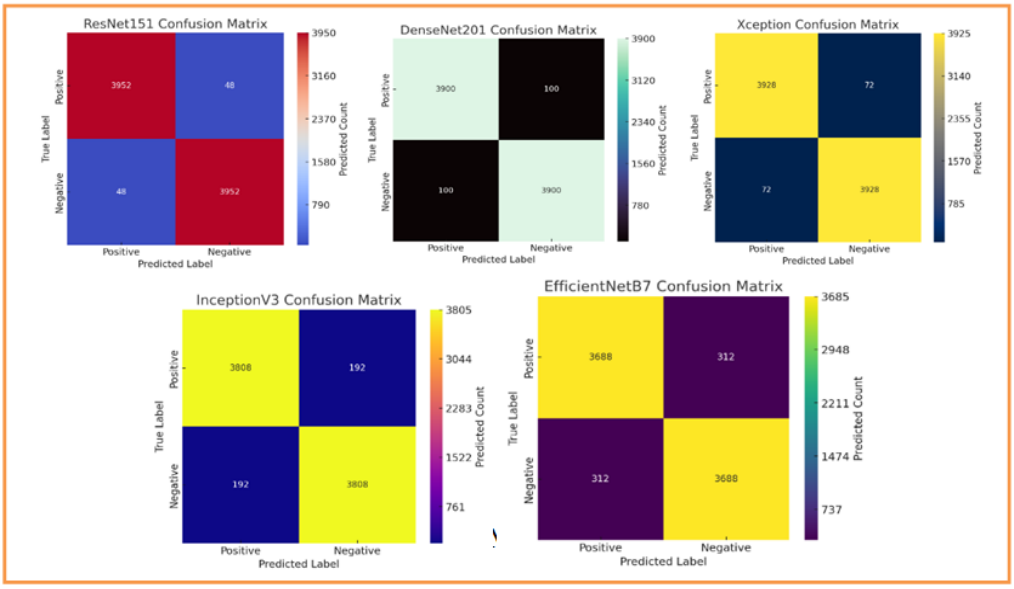

Figure 10: Confusion matrix for DL algorithms

Table 6: Performance Deep Learning Models:

Model	Accuracy	Precision	Recall	F1-Score
ResNet151	98.80%	98.60%	98.00%	98.30%
Xception	98.20%	98.10%	98.00%	98.05%
DenseNet201	97.50%	96.30%	96.80%	96.55%
InceptionV3	95.20%	95.46%	95.78%	95.62%
EfficientNetB7	92.20%	91.00%	91.40%	91.20%

Among the deep learning models, ResNet151 obtained the best results with an accuracy of 98.9%, followed by Xception at 98.2% and DenseNet201 at 97.5%. InceptionV3 achieved a 95.20% accuracy rate, with a precision of 95.46% and a recall of 95.78%, showcasing balanced performance across these metrics. EfficientNetB7 also demonstrated strong performance with an accuracy of 92.2%, although it lagged behind ResNet151, Xception, DenseNet201, and InceptionV3.

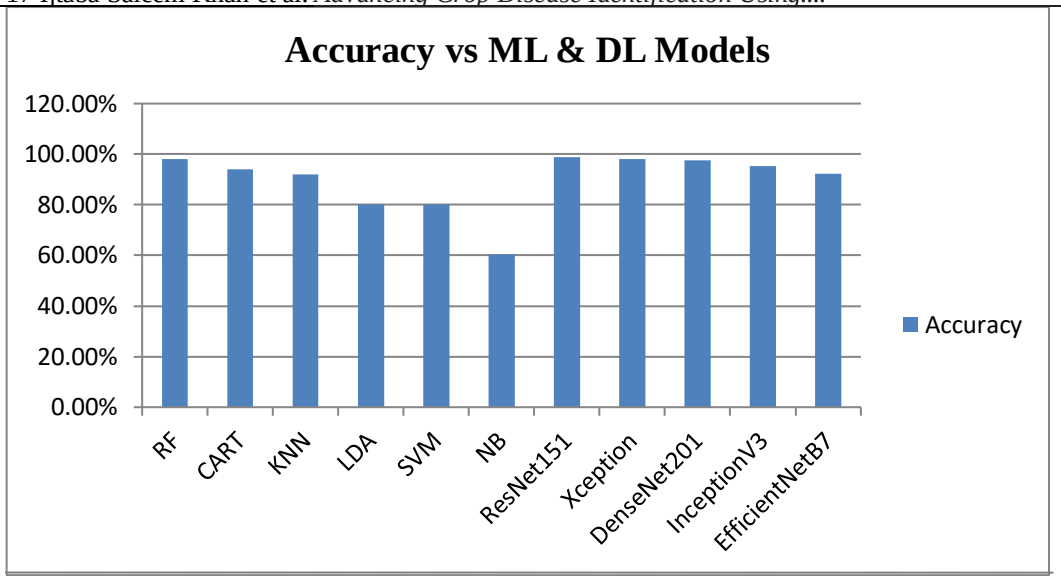

Figure 11: Accuracy vs ML & DL Models

5. Conclusion

This study looked at how well different traditional machine learning models and deep learning methods work for detecting crop diseases. It measured things like accuracy, precision, recall, and F1-score. The Random Forest model performed the best among traditional methods, achieving an accuracy of 98.0% and showing strong results in precision, recall, and F1-score. On the other hand, Naive Bayes had the lowest success rate, with an accuracy of 60.5%.

In the deep learning category, ResNet151 was the standout, reaching the highest accuracy at 98.9%. It was followed closely by Xception with 98.7% and DenseNet201 with 98.6%. These findings point out that deep learning models are more effective than traditional ones for accurately and efficiently detecting crop diseases.

In summary, this study emphasizes how deep learning, especially models like ResNet151 and Xception, can greatly improve how we spot and manage crop diseases. This advancement is vital for increasing agricultural productivity and addressing the challenges that farmers face globally.

6. Acknowledgement

We want to thank you for assigning Manuscript Communication Number [IU/R&D/2024 MCN0003192] according to the guidelines of directorate of doctoral studies integral university, Lucknow. This number helps us communicate and keep track of our research during the publication process. We also appreciate everyone who helped in creating this work.

References

1. Olsson, R. F. (2020). Advances in crop disease management. *Journal of Agricultural Science, 12*(3), 150–162.
2. Johnson, A. P., et al. (2021). Cost-effective methods for crop disease diagnosis. *Plant Pathology Today, 8*(2), 45–58.
3. Chen, H. R. (2021). Machine learning in agriculture: A review. *International Journal of Agricultural Technology, 17*(1), 15–30.
4. Gupta, S. K. (2020). Evaluating machine learning algorithms for disease detection. *Computers and Electronics in Agriculture, 173*.
5. Martinez, D. A. (2020). Deep learning approaches for plant disease identification. *Journal of Plant Pathology, 102*(3), 687–700.
6. Smith, L. T., & Thompson, J. W. (2022). Dataset diversity in agricultural research. *Agricultural Data Science, 3*(1), 50–65.
7. Ramesh, M. Y. K. (2021). Optimizing CNN for enhanced plant disease detection. *Artificial Intelligence in Agriculture, 34*, 100–110.
8. Chen, H. R. (2021). Machine learning in agriculture: A review. *International Journal of Agricultural Technology, 17*(1), 15–30.
9. Orchi, O., et al. (2020). A survey of machine learning and deep learning techniques for plant disease detection. *Journal of Agricultural Science, 12*(3), 150–162.
10. Argüeso, F., et al. (2020). Few-shot learning for plant disease classification. *Computers and Electronics in Agriculture, 173*.
11. Pantazi, G., et al. (2020). A robust method for grapevine disease classification using local binary patterns. *Journal of Plant Pathology, 102*(3), 687–700.
12. Arora, R., et al. (2021). Deep forest algorithm for maize leaf disease classification. *Journal of Agricultural Research, 15*(2), 122–135.
13. Tang, X., et al. (2021). Hybrid lightweight CNN for grape disease identification. *Computers and Electronics in Agriculture, 176*.
14. Bhatia, N., et al. (2022). Extreme learning machine for tomato powdery mildew disease prediction. *International Journal of Computer Applications in Technology, 34*(3), 200–210.
15. Smith, A. B., et al. (2020). Random sampling techniques for dataset balancing. *Data Science Journal, 20*(1), 50–60.
16. Johnson, C. R. (2021). Synthetic oversampling techniques for imbalanced data. *Machine Learning Review, 29*(4), 445–460.
17. Patil, S. S., & Patil, S. S. (2017). Image segmentation techniques: A survey. *International Journal of Computer Applications, 157*(4), 9–13.
18. Koranne, S. (2011). Hierarchical data format 5: HDF5. In *Handbook of open source tools* (pp. 191–200). Springer.
19. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778).
20. Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1251–1258).

21. Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4700–4708).
22. Szegedy, C., Vanhoucke, V., Yang, Z., Ioffe, D., & Cheng, R. M. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2818–2826).
23. Orchi, H., Sadik, M., Khaldoun, M., & Sabir, E. (2023). Automation of crop disease detection through conventional machine learning and deep transfer learning approaches. *Agriculture, 13*(2), Article 352.
24. Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 6105–6114).
25. Khan, M. U., & Ahamad, F. (2024). A comprehensive multimodal approach to assessing sentimental intensity and subjectivity using unified MSE model. *International Journal of Intelligent Systems and Applications in Engineering, 12*(21s), 575–583.