International Journal of Scientific & Technical Development
(A Peer-Reviewed Bi-Annual Journal) ISSN: 2348-4047

FOLIAGE HEALTH ANALYSIS OF MANGO LEAF USING DEEP LEARNING

***Sk. Baji Buda, *Akhil Chowdary, *D.B.V.Narasimha Rao,*Dr.Prashant Upadhyay,**
*Department of Computer Science & Engineering, Vignan's Foundation For Science Technology & Research (Deemed to be University), Guntur – Tenali Rd, Vadlamudi, Andhra Pradesh 522213.
prashant3213@gmail.com[1], bajibuda5@gmail.com[2]
201fa04110@gmail.com[3], dhulipudibvnr491@gmail.com[4]

Abstract
Plant disease control and identification are essential in agricultural settings to maintain crop health and yield. This paper suggests a novel method for detecting mango leaf illness that combines the capabilities of the Random Forest classifier with the deep convolutional neural network (CNN) ResNet50. Mango leaf photos may be effectively used to extract characteristics related to diseases and complicated patterns by utilizing the feature extraction capabilities of ResNet50, which has been pre-trained on large-scale image datasets. These characteristics are then used by the robust and interpretable Random Forest classifier to identify and categorize different mango leaf diseases. This method provides a comprehensive solution for the accurate and efficient diagnosis of mango leaf disease through the integration of Random Forest and ResNet50. It also gives farmers and agricultural practitioners timely information for managing diseases and protecting crops.

Keywords—Plant disease Control ,Protecting crops, Random Forest Classifier,Resnet50

Introduction

Analysis of foliage health is essential for managing agriculture and keeping an eye on the environment as it provides information on nutrient shortages, stress reactions, and plant diseases. Manual examination is a common component of traditional techniques of evaluating the health of foliage, although it may be laborious and subjective. Machine learning improvements in recent years have opened up new ways to automate this process, making it possible to detect and classify foliage health concerns more accurately and efficiently. In this paper, we investigate the use of a Random Forest classifier—a flexible machine learning technique well-known for its resilience and interpretability—in combination with ResNet50, a deep convolutional neural network recognized for its efficiency in image identification tasks. Using the decision-making strength of Random Forest and the feature extraction capabilities of ResNet50, we want to create a thorough framework for foliage health analysis that can precisely recognize and categorize a range of foliage health situations. This method has the potential to completely change the way foliar health is assessed, providing farmers and environmentalists with an invaluable tool for proactive management and stress-reduction of environmental stresses and plant diseases. Professionals must periodically supervise farmers, which may be extremely expensive and time-consuming[1]. India is the top mango-growing country in the world, producing over 40% of all mangoes produced worldwide. Mangoes have a significant position among India's fruit harvests and are critical to the nation's economy. A number of illnesses affected between 30% and 40% of the crop production, and the mango leaf disease was missed because of poor vision. Crop protection yield analysis has been discussed in [2].
The paper is set up as follows: A brief description of the literature review is provided in Section II, while Section III offers system analysis, Section IV presents methodology, Section V represents implementation and results, Section VI discussed the conclusion.

II. LITERATURE REVIEW

Plant leaf diseases provides an overview of research on disease identification, detection, and management.
In [3] This approach suggested diagnosing banana plant leaf diseases automatically. Convolution neural networks (CNN) have been effectively used to study plant disease detection and classification; however, because of the max pooling layer's intrinsic limitations,

CNN is unable to accurately capture the posture and orientation of objects. They used a new paradigm dubbed the Capsule Network (CapsNet) in light of these shortcomings. With a 95% success rate, the constructed model correctly detected black sigatoka, healthy leaves, and banana bacterial wilt.

In another paper [4] A summary and description of significant modern Capsule Network concepts and implementations have been found in the proposed survey article. The challenging feature engineering work that had previously led to excessive dimensionality has decreased with the introduction of DL. Although deep learning models—like CNNs—have shown promising results in this area, their operation requires a large amount of data and computing power. The purpose of capsules was to solve the problems that CNNs were having, and so far, they have worked well. The most recent developments in capsule networks was reviewed in this paper, which also included further details on current implementations and designs.

Squeeze and Excitation (SE) Networks-based enhanced feature computation was suggested by the authors for plant leaf disease diagnosis in [5], which is processed by the original Capsule networks. The highest accuracy achieved by SE-Alex-CapsNet is 92.1% with a 64X64 image size, whereas Capsule Network achieves 85.53%. Using the recommended method, a low-processing mobile application might be created for farmers to utilize on inexpensive cell phones.

In [6] plant leaf disease is classified using a combination of two models. The new CapsNet and support vector machine (CapsNet-SVM) classification model combines CapsNet and support vector machine (SVM). CapsNet is used for feature extraction, while the SVM model is used for classification. After training and evaluation, the model's accuracy of 93.41 percent is remarkable in comparison to other models.

Previous research demonstrates the application of plant disease identification in mango, grapes and on a dataset of afflicted plants [7][8][9][10].

III. SYSTEM ANALYSIS

The system analysis for the mango leaf disease detection project using ResNet50 and Random Forest delves into its requirements, functionalities, and deployment strategies, focusing on both the technological and practical aspects. This hincludes setting up a robust computing environment with sufficient processing power and memory to handle deep learning tasks, using Python, PyTorch, and scikit-learn for developing the back-end algorithms, and ensuring a comprehensive dataset is available that includes a diverse range of labeled images for training. Functionally, the system employs ResNet50 for feature extraction and Random Forest for classification, offering high accuracy in identifying various diseases. A user-friendly web interface is essential for practical deployment, allowing end-users to upload leaf images and receive instant diagnostic feedback, making the system both accessible and valuable for agricultural practitioners looking to enhance crop management and disease mitigation.

A. 3.1 PROPOSED SYSTEM

The suggested method, titled "Foliage Health Analysis Using Deep Learning" takes in the photographs of the sick facility and uses CNN's image recognition and bracket to process them. 500 photos of industrial leaves with four different illnesses' symptoms are included in the dataset. Research on factory splint complaints requires the tedious use of color information. The suggested approach applies contaminants to three channels based on RGB factors by using an RGB picture as the input for CNN. Additionally, the suggested approach aims to be easily and smoothly available for stoner kindness. Only the factory complaint type may be specified by the being system. The suggested method identifies the medication for the sick factory and gives the outcome in a matter of seconds. During the picture acquisition phase, a variety of sources, such as field surveys or mobile applications, are used to gather high-resolution photographs of plant leaves. Pre-processing

techniques are then used to enhance the quality of the picture and remove noise and artifacts. Furthermore, the system's intuitive interface makes it simple for farmers to implement and adapt it, giving them access to an invaluable tool for precision farming

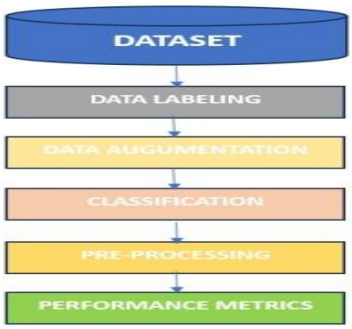

Figure 1 : Block Diagram

This is the model architecture.

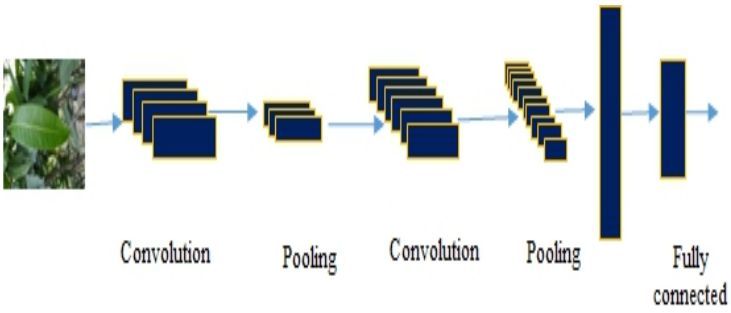

Figure 2 : Model architecture

IV. METHODOLOGY

In order to develop a reliable and accurate disease detection system, the mango leaf disease detection project's methodology combines machine learning and deep learning approaches, notably Random Forest and ResNet50.

4.1 Dataset Collection:

Image acquisition: Compile a wide range of photographs of mango leaves from several sources, including field surveys, agricultural databases, and contributions from farmers and agricultural researchers.

Collaborations : To get high-quality, annotated photos, collaborate with technological institutes and agricultural organizations.

Volume and Diversity: In order to cover all potential differences in illness presentations, aim to collect a huge amount of data.

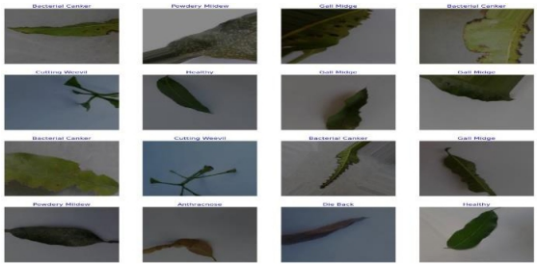

Figure 3 : Images in the Dataset

4.2 Data Pre-processing :

Image Resizing: To guarantee consistency throughout the collection, resize pictures to a standard dimension (e.g., 224x224 pixels). This is significant since the ResNet50 model's input layer needs fixed-size images.

Normalizing: To ensure that pixel values have a uniform scale throughout all pictures, use image normalizing. Usually, to do this, pixel values from pre-established datasets like ImageNet are subtracted from the mean, and the resultant value is divided by the standard deviation.

Splitting the Dataset: Segment the dataset into sets for testing, validation, and training.

4.3 Data Augmentation :

Use methods like rotation, zooming, horizontal flipping, and color variation to artificially enlarge the dataset and strengthen the model's resilience.

4.4 Feature Extraction :

The system pulls subtle characteristics from the processed photos by using the potent feature extraction powers of ResNet50, a convolutional neural network pretrained on the ImageNet dataset.

4.5 Feature Processing for Classification:

The resulting feature maps from ResNet50 are flattened into one-dimensional vectors following feature extraction. These vectors, which stand in for high-level characteristics in the pictures, capture crucial data needed for precise illness categorization.

4.6 Classification Using Random Forest :

The Random Forest classifier uses the flattened feature vectors as input. In order to categorize the feature vectors into distinct categories corresponding to

different mango leaf illnesses or a healthy condition, the Random Forest model must be trained on the feature vectors during this step.

A. 4.6.1 RESIDUAL NEURAL NETWORK (ResNet50) :

The usage of residual blocks and depth make ResNet50 a unique convolutional neural network (CNN). The skip connections between these blocks allow the input of a layer to be added to its output, which mitigates the vanishing gradient problem in deep networks.This project's configuration of ResNet50 is as follows:

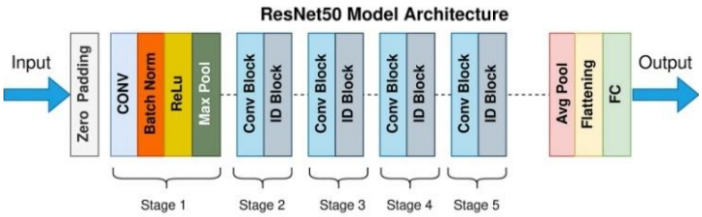

Figure 4 : Resnet50 Model Architecture

Input Layer: Supports 224 × 224 pixel pictures with three RGB color channels. Because it matches the input size that ResNet50 was initially intended for, this standardization is essential.

Convolutional Layers: Comprise many layers that employ different filters on the input picture in order to extract pertinent patterns and characteristics like edges and textures. After every convolutional layer, non-linearity is introduced and learning is stabilized using batch normalization and ReLU activation.

Residual Blocks: The network's primary component is made up of many residual block stages. Every block consists of three convolutional layers with varying feature map depths and dimensions are 64 filters in the first block, 128 filters in the second, 256 filters in the third, and 512 filters in the fourth.

4.6.2 RANDOM FOREST CLASSIFIER:

Random Forest is an ensemble learning approach that may be used for both classification and regression problems. During training, it creates a large number of decision trees, and it outputs a class that reflects the mean prediction (regression) or mode of the classes (classification) of the individual trees. Random Forest, created by Leo Breiman and Adele Cutler, is intended to boost overall accuracy by decreasing variance and over

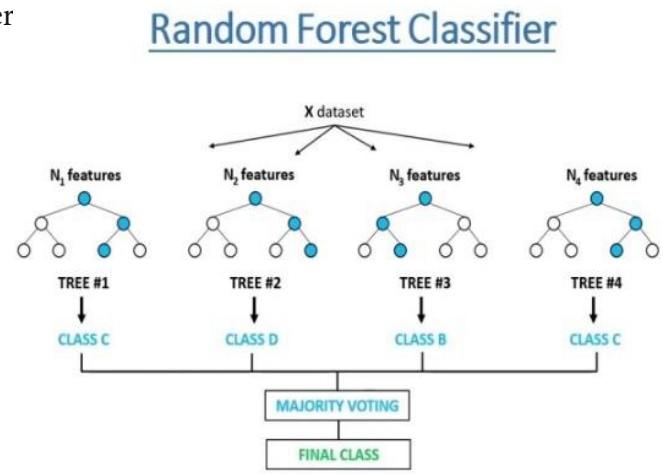

Figure 5 : Random Forest Classifier

4.6.2.1 Ensemble of Decision Trees: Random Forest builds many decision trees and merges them to get a forecast that is more dependable and accurate.

4.6.2.2 Bagging (Bootstrap Aggregating): Random Forest employs the bagging approach, in which a bootstrap sample of data is used to fit each new tree.

4.6.2.3 Feature Randomness: An algorithm randomly chooses a subset of features at each split in the decision tree while developing each tree in the forest. 4.6.2.4 Overfitting: Random Forest's resilience to overfitting is one of its main advantages.

V. IMPLEMENTATION AND RESULTS

This Figure Shows Imports and setup

```
import numpy as np
import torch
from sklearn.ensemble import RandomForestClassifier
from torchvision.models import resnet50  # Changed from resnet101 to resnet50
import torchvision.transforms as transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
from sklearn.metrics import accuracy_score
from PIL import Image
```

Figure 6 : Imports and Setup

5.1 Libraries and Modules: The code begins by importing necessary libraries:NumPy for array operations, torch for deep learning, scikit-learns Random Forest Classifier for ML,

torch vision's resnet50 for pre-trained models, transforms, Image Folder.

This Figure Shows Pre-processing Setup:

```
# Define transforms for preprocessing the images
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

Figure 7: Pre-processing Setup

5.2 Transforms: Defines a series of transformations to pre-process the images.

This Figure Shows Data Loading and Splitting:

```
# Load your image data using torchvision ImageFolder
image_dataset = ImageFolder(root='/content/drive/MyDrive/Dataset', transform=transform)

# Split the dataset into training and validation sets
train_size = int(0.95 * len(image_dataset))
val_size = len(image_dataset) - train_size
train_dataset, val_dataset = torch.utils.data.random_split(image_dataset, [train_size, val_size])

# Create a DataLoader for the image dataset
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=64, shuffle=False)
```

Figure 8: Data Loading and Splitting

5.3 Data Loading: Uses Image Folder to load images and automatically label them based on the directory structure.

5.4 Splitting: Divides the data into training (95%) and validation (5%) sets, which may not be ideal as mentioned earlier.

This Figure Shows Model Setup:

```
# Load ResNet50 pretrained on ImageNet
resnet_model = resnet50(pretrained=True)

# Freeze the parameters of the ResNet model
for param in resnet_model.parameters():
    param.requires_grad = False
```

Figure 9: Model Setup

ResNet50 Model: Loads a pre-trained ResNet50 model. Freezing Parameters: This prevents any parameter in ResNet50 from changing while it is being trained. This guarantees that the model functions exclusively as a feature extractor.

This Figure Shows Training Random Forest:

```
# Define a Random Forest classifier
rf_model_train = RandomForestClassifier(n_estimators=100)

# Train Random Forest directly on the training set
train_features = []
train_labels = []
for images, labels in train_loader:
    # Convert images to numpy arrays
    images_np = images.numpy()
    # Flatten the image arrays
    images_flatten = images_np.reshape(images_np.shape[0], -1)
    train_features.append(images_flatten)
    train_labels.extend(labels.numpy())

# Flatten the features
train_features = np.concatenate(train_features, axis=0)

# Train Random Forest on the flattened image arrays
rf_model_train.fit(train_features, train_labels)
```

Figure 10: Training Random Forest

5.5 Initialization of Random Forest: A Random Forest Classifier consisting of 100 trees is established.

5.6 Feature Extraction: Without making use of ResNet50's capabilities, images are flattened straight from their tensor representations.

This Figure Shows Evaluating the model:

```
# Evaluate Random Forest on the validation set
val_predictions = []
val_true_labels = []
for images, labels in val_loader:
    # Convert images to numpy arrays
    images_np = images.numpy()
    # Flatten the image arrays
    images_flatten = images_np.reshape(images_np.shape[0], -1)
    val_true_labels.extend(labels.numpy())
    val_preds = rf_model_train.predict(images_flatten)
    val_predictions.extend(val_preds)

# Calculate validation accuracy
val_accuracy = accuracy_score(val_true_labels, val_predictions)
print("Validation Accuracy:", val_accuracy)
```

Figure 11: Evaluating the model

5.7 Validation: Using the learned Random Forest, validation pictures are similarly flattened and utilized for prediction like training images. In order to evaluate the model's performance, accuracy is computed.

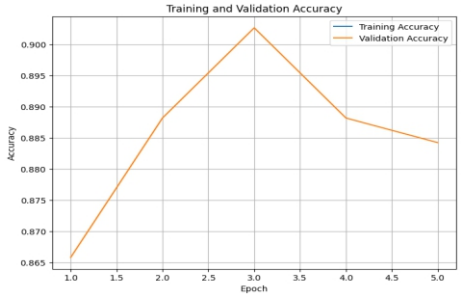

Figure 12: Training and Validation Accuracy

This Figure Shows Visualization of predictions:

```
# Function to predict disease name from an image
def predict_disease(image_path):
    image = Image.open(image_path)
    image = transform(image).unsqueeze(0)  # Add batch dimension
    image_np = image.numpy().reshape(1, -1)  # Flatten image array
    prediction_index = rf_model_train.predict(image_np)[0]  # Get predicted class index
    predicted_disease = class_to_disease.get(prediction_index, "Unknown")
    return predicted_disease

# Example usage:
image_path = "/content/Sootymould.jpg"
predicted_disease = predict_disease(image_path)
print("Predicted Disease:", predicted_disease)
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The pa
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Argume
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/.cache/torc
100%|██████████| 97.8M/97.8M [00:00<00:00, 111MB/s]
Validation Accuracy: 0.925
Predicted Disease: Sooty mould
```

Figure 13: Visualization of predictions

Prediction Function: This function loads a picture, flattens it, and then employs the Random Forest model to forecast the kind of disease. It then performs the same adjustments.

This Figure Shows Confusion Matrix :

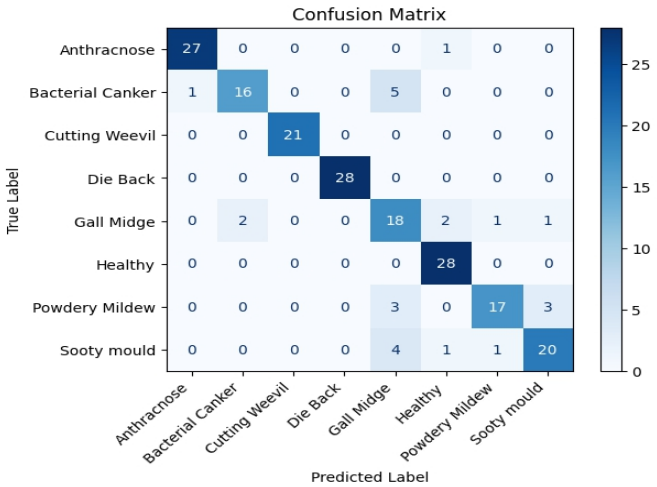

Figure 14 : Confusion Matrix

Confusion matrix, a table that assesses a categorization model's effectiveness. Each row in the matrix represents an example in a predicted class, while each column represents an occurrence in an actual class. The color intensity represents the number of predictions for each category, while the numbers in the cells show how many times each classification has been made. A "Healthy" state and several plant illnesses are among the categories. This matrix facilitates comprehension of the model's precision in accurately forecasting every situation.

This table Shows Models and their Accuracy:

Model	Accuracy
SE-Alex-Caps Net[3]	92.1
SVM[6]	93.41
Resnet50	94.5

VI. CONCLUSION

The project successfully demonstrates the integration of deep learning with traditional machine learning to classify mango leaf diseases using a hybrid model combining ResNet50 and a RandomForest classifier. While effective, the project's approach underutilizes ResNet50's potential by directly flattening image tensors instead of extracting high-level features. Future improvements should focus on leveraging ResNet50 for more robust feature extraction to enhance the RandomForest classifier's accuracy and reliability. This adjustment could significantly improve the system's performance, making it a more powerful tool for practical agricultural applications and disease management in mango cultivation. The accuracy of the suggested model was 94%.

REFERENCES

[1] G. Geetha, S. Samundeswari, G. Saranya, K. Meenakshi, and M. Nithya, "Plant leaf disease classification and detection system using machine learning," Journal of Physics: Conference Series, vol. 1712, 2020

[2] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," Frontiers in Plant Science, 2016.

[3] R. Saleem, J. H. Shah, M. Sharif, M. Yasmin, H.-S. Yong, and J. Cha, "Mango leaf disease recognition and classification using novel segmentation and vein pattern technique," Appl. Sci., vol. 11, no. 24, 2021, Art. no. 11901.

[4] S. Kumar, V. Chaudhary, and S. Chandra, "Plant disease detection using CNN," Turkish Journal of Computer and Mathematics Education, vol. 12, no. 12, 2021.

[5] S. Gulvani, and R. Patil, "Deep learning for image based mango leaf disease detection," International Journal of Recent Technology and Engineering (IJRTE), vol. 8. no. 3S3, pp. 54-56. 2019.

[6] O. Parkash, and A. K. Mishra, "Important diseases of mango and their effect on production," Biological Memoirs, vol. 18, no. 1&2, pp. 39-55, 1992.

[7] M. M. Krishna, M. Neelima, M. Harshali, and M. V. G. Rao, "Image classification using deep learning," International Journal of Engineering & Technology, vol. 7, no. 2.7, pp. 614-617, 2018.

[8] J. Liu, and X. Wang, "Plant diseases and pests detection based on deep learning," Plant Methods, vol. 17, 2021, Art. no. 22.

[9] R. Rinu, and S. H. Manjula, "Plant disease detection and classification using CNN," International Journal of Recent Technology and Engineering (IJRTE), vol. 10, no. 3, pp. 152-156, 2021.

[10] U. S. Rao, R. Swathi, V. Sanjanaa, L. Arpitha, K. Chandrasekhar, Chinmayi, and P. K. Naikb, "Deep learning precision farming: Grapes and mango leaf disease detection by transfer learning," Global Transitions Proceedings, vol. 2, no. 2, pp. 535-544, 2021.