PRECISION FARMING: DEEP LEARNING TECHNIQUES FOR CROP DISEASE DETECTION

[1]vaibhavi L.Vispute, [2] Jagruti S.Patil , [3]Chandrashekhar V.Patil.

Department of Electronics and Telecommunication Engineering, NES'S Gangamai College of Engineering, NMU University, Nagaon, Dhule–424 005, Maharashtra, India. [1, 2, 3]

vaibhavivispute93@gmail.com

ABSTRACT

Crop disease is a serious problem for agricultural performance, food security and economic stability around the world. Early and accurate detection of agricultural diseases is important to minimize losses, maintain culture health and ensure optimal profitability. This study examines the use of Adhesive Neural Networks (CNNs) for the automatic detection and classification of agricultural diseases using images. Known for exceptional indicators in image recognition problems, CNNs are used to analyze visual models and symptoms of disease in cultured leaves. This study uses modern CNN architectures such as VGG16, ResNet, Inception, and Densenet to develop reliable detection structures. The study highlights the potential to transform deep agricultural education by reducing reliance on manual testing in high-intensity labor forces and promoting sustainable agricultural processes. The results of this study contribute to improving agriculture accuracy and paving the way for in-depth research in the integration of artificial intelligence-based solutions in agricultural systems.

Keywords: Deep learning (DL), Convolutional neural network (CNN), machine learning (ML), Artificial intelligence (AI), feature extraction (FE), Image processing, VGG 16, Crop disease recognition, Dense Net, Mobile Net, Efficient Net.

INTRODUCTION

Background

Agriculture is the foundation of the global economy, providing food, raw materials and employment to a large part of the population. It maintains human life and plays an important role in stimulating the national economy, especially in developing countries. However, agricultural productivity is increasingly threatened by a variety of issues, including pests, climate change, soil degradation and, above all, agricultural crop disease.

Agricultural crop diseases caused by pathogens such as fungi, bacteria, viruses, nematodes can have serious effects on plant health. These diseases not only reduce productivity and quality, but also lead to significant economic losses, contributing to the global lack of food security. According to the Food and Agriculture Organization (FAO), plant diseases are responsible for 40% of global losses in crops each year. This has been moved to billions of dollars of losses, and constitutes a serious threat to the presence of millions of farmers, especially small owners who have no access to advanced agricultural technologies and infrastructure.

Timely time detection and effective treatment of agricultural crop diseases are important to ensure optimal agricultural products and food security. Traditional methods of detecting disease rely on manual verification by farmers or experts trained in agriculture. These methods, while effective in certain cases, are limited by their need for expertise, visual assessment skills, and the ability to distinguish similar symptoms from illness.

A CNN-based model specifically designed for classifying diseases in tomato leaves. This model attained a classification accuracy of 95.6% across five different disease types as well as healthy leaves, demonstrating its effectiveness in identifying features unique to each disease.[1]

Utilized transfer learning by adjusting pre-trained models like VGG16 and ResNet50 for classifying crop diseases. ResNet50 achieved the best accuracy of 97%, demonstrating the benefits of deep architectures with residual connections in understanding intricate visual patterns. [2]

The implementation of transfer learning utilizing the AlexNet architecture on a dataset concerning tomato diseases has yielded a commendable level of classification accuracy. However, it is noteworthy that the model necessitated considerable preprocessing efforts to ensure consistency in the results, thereby indicating inherent limitations in its capacity to adapt to unrefined, real-world image inputs.[3]

These limitations underline the urgent need for automated, scalable, economically effective solutions for crop detection. Recent achievements in the field of artificial intelligence (AI), particularly in the field of computer vision and deep training, provide promising opportunities to solve this problem. The network of pig neurons (CNNS) is a rich class of learning algorithms suitable for image analysis tasks, showing great success in visual methods and recognition of image characteristics. Using CNN to detect agricultural diseases, it can accurately classify plant diseases from sheets, provide timely decisions, and create intellectual systems enlightened by farmers and interested farmers.

CNN-based detection with IoT devices for real-time disease monitoring in farms. The system achieved over 93% accuracy but introduced concerns related to data transmission, privacy, and cyber-security.[4-6]

The adoption of these technical decisions not only increases the effectiveness of treating agricultural diseases, but also supports sustainable agriculture by minimizing resource loss and promoting environmentally friendly practices. This study focuses on using CNN's possibilities to develop reliable, accurate, deployed systems using image-based analysis.[7-8]

METHODS

Overview of Convolutional neural networks (CNN)

Simple Neural Networks (CNN) is a specialized class of detailed learning algorithms that are widely used to detect image classification, objects, and computer vision tasks. Their architecture is inspired by the visual cortex of the human brain and is particularly effective during the automatic extraction of cosmic hierarchical functions from input images. CNN power lies in its ability to study and automatically optimize functions, eliminating the need for manual development of functions. If agricultural diseases are detected, CNN will succeed in detecting light patterns of leaf texture, color changes, and morphology of damage that may indicate disease.

Several uses of modern CNN architectures. Each has unique properties and power to identify and classify tomato sheets. These include:

•VGG16: Known for its simplicity and depth, the VGG16 uses complex beam layers and has achieved high performance with a relatively simple design.

•ResNet50: Deploys residual connections to reduce extraction gradient issues, allowing deeper networks to be effectively trained.

•Densenet121: Connect each layer to other layers in the processing style to improve information flow and reduce excessive functionality.

• MobileNETV2: Optical architecture optimized for mobile devices and built using individual advice depth to reduce parameters.

•Efficient NETB0: Uses a method of scaling connections to reduce optimal performance and resources to balance depth, width and network resolution. The next section details the configuration, training and evaluation of these architectures for detecting tomato sheet disease.

Model composition and learning process

To assess the effectiveness of various CNN architectures in classifying tomato sheet diseases, each model is constructed and formed using coherent experimental parameters. The composition and learning process included the following steps:

• Image Input Size: All images have been changed to 224x224 pixels to ensure compatibility with input layers for all selected models.

•Party Size: Lot 32 size is chosen for the balance between memory usage and training speed.

•ERA: The model was formed at 50 ERAs and included early halts to avoid rethinking and unnecessary calculation reductions.

• Optimization: Adam Optimizer was used at a learning level of 0.001 for adaptive learning ability and faster convergence.

•Loss Function: Categories cross-entropy is used according to losses suitable for multi-class classification tasks.

• Validation Department: 10% of the educational data was used as a validation set to monitor the ability to generalize the model during training.

• Inverted Issues: The early and inverted models of the model are implemented to stop training during convergence and maintain the highest weight of the model.

• The device is used. Training was performed using an NVIDIA graphics processor for accelerated calculations.

Each architecture is either zero or subtly tuned in training use. For training, ImageNet's pre-formed weights have been used, with the upper layer being replaced by a dense layer of users adapted to the task of classifying 11 classes of tomato disease.

Layers have been added to fully connected steps to avoid rethinking. To improve generalization, data increase was used dynamically during training. Performance measurements were recorded after each era to assess training progress and the realization of hyperparameter adjustments. This unique configuration provided a fair comparison between models and a coherent performance indicator.

ARCHITECTURE

This methodology can be proposed with VGG16, Densen, AlexNet, or other suitable architecture.

- **VGG16 Architecture**

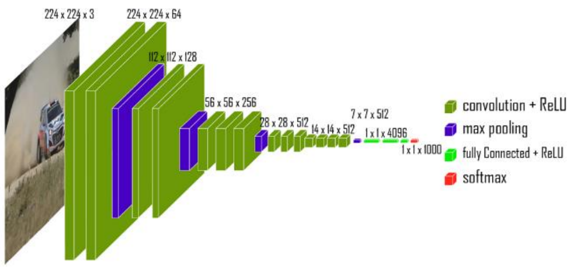

Fig.1

• Diaper: The VGG-16 has 16 weighted layers, so it is called the VGG-16. The network consists of five blocks, with the first two blocks having two layers of packages and the last three blocks having three layers of packages. Each block has a layer of association, reducing the height and width of the image.

• Filter: The VGG-16 uses a 3x3 filter package in step 1. The network uses a consistent location of the package and the largest layer of the swimming pool across its architecture.

•Optional: VGG-16-Borry Network with approximately 138 million parameters. •Image Input: The size of the input image of the VGG-16 is 224 x 224.

•Spare model: A backup version of the VGG-16 is available. It is trained with over millions of images from the ImageNet database. This preliminary network allows images to be categorized by objects of 1000 categories.

- **DenseNet Architecture**

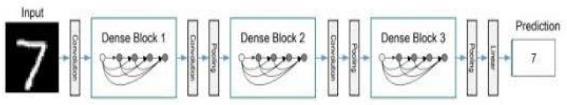

Fig.2 Dense block

A dense block is a structural block of the Denset architecture. Each dense block generally consists of many layers of bundles with party normalization and nonlinear activation functions (e.g., rereading). It is important to note that each layer in the high density block receives a map of functions for all previous layers in the entry, facilitating the reuse and distribution of features.

In dense blocks, each layer receives concatenated output from all previous layers as an inlet. When a dense block has Layer M, and when each layer generates a card K function (K is known as the growth rate), the L-THU layer has cards with input elements k x (l + l0) k x (l + l0) (where L0L0 is the number of input channels in the dense block).

Examples of dense blocks:

Level 1: Receives input function.

Level 2: Input function + Receive layer output1.

Level 3: Layer 2 input function + Level 1 level + Receives output.

This model continues for all layers of a block and provides a highly interconnected architecture.

• <u>Dense options</u>

Densets are mainly produced in several versions that differ from depth and number of layers.

Densenet-121: Contains 121 layers known for a balanced compromise between computational efficiency and accuracy. Ideal for tasks that require moderate IT resources.

Densenet-169: With 169 layers, this option provides a deeper extraction of indicators that are adapted to more complex datasets that require higher accuracy.

Denset-2011 and Denset-264: These options provide a deeper architecture that is adapted to highly complex tasks that require detailed representation of symbols.

EXPERIMENTAL INSTALLATION

1Hardware and Software Environment

A reliable computer installation has been used to train, evaluate and configure the CNN models used in this study. The hardware and software specifications are as follows:

Material composition:

• Processor: Intel(R)Core(TM) i7 10 Gen CPU

•RAM: 16 GB DDR4

•GPU: NVIDIA GEFORCE GTX 1660 TI 6 GB VRAM

•Storage: 512 GB SSD

• Operating System: Windows 10 (64-bit)

Software and libraries:

• Programming Language: Python 3.9

•Deep Learning Library: Tensorflow 2.11, Keras 2.11

•Library Management: numpy, matplotlib, pandas, scikit-learn, opencv

• Jupyter notebook: Used for interactive development and visualization

•Google Colab/Kaggle Notebook: Used for additional GPU resources during the experience

This configuration provides the acceleration of the graphics processor to provide model fluids and effective formation, ensuring rapid processing of critical lots and deep layers of neural networks. Programmable batteries provided the flexibility to configure models throughout the project's lifecycle and visualize and monitor performance.

Training and Testing Configuration:

The learning and testing process is carefully structured to ensure a reliable assessment of the efficiency of the model and the reproducibility of the results. The following steps describe the experimental work process.

• Dataset Separation: The dataset is divided into three subsets.

oTraining Set (80%): Used to teach parameters modeling and updates. oTest (10%): Monitor the performance of the model during training and use it for early outage use.

oSet of tests (10%): Used to ultimately assess post-training efficiency. • Cross-validation: In addition to a fixed set of tests, a K (K = 5) cross-check was used in data learning to ensure reliability and reduce the variance in performance ratios.

• Increase during training: Expanding methods such as rotation (±25°), kicking in horizontal and vertical states, scale (beach = 0.2), and brightness adjustment were used in real time using the Imagenerator class to improve model generalization. •Measurement monitoring: Accuracy, test loss, and F1 indicators were continuously checked at that time. Early decisions were triggered if the audit loss was not improved for the fifth consecutive time. • Control Point Model: Models with the best accuracy of controls are recorded on disk to avoid subsequent degradation of ERA performance.

• Random seeds: Afixed random seeds (seeds = 42) were installed on Wednesday numpy and Tensorflow to ensure regeneration of the results. This structured configuration ensured that the model was not overwhelmed by a particular data subset and that its performance indicators reflected actual applicability in various input conditions.

RESULTS AND DISCUSSION

Model Performance Comparison

Standard classification criteria were used to assess the effectiveness of the chosen CNN models, MobileNetV2 and EfficientNetB0. Over several epochs, the outcomes were documented for both the training and validation stages.

- **EfficientNetB0:**
- **Training Accuracy**: 91.09%

- **Training Loss**: 0.2552
- **Validation Accuracy**: 88.57%
- **Validation Loss**: 0.3426
- **Epochs Trained**: 10

High accuracy was demonstrated by EfficientNetB0 using comparatively few parameters. It demonstrated strong generalization as its performance held steady over time. The model demonstrated effectiveness in striking a balance between computational complexity and accuracy.

- **MobileNetV2 (5 Epochs):**
- **Training Accuracy**: 70.81%
- **Training Loss**: 0.8492
- **Validation Accuracy**: 73.19%
- **Validation Loss**: 0.7554
- **MobileNetV2 (10 Epochs):**
- **Training Accuracy**: 75.06%
- **Training Loss**: 0.6960
- **Validation Accuracy**: 75.52%
- **Validation Loss**: 0.7048

Despite having a little poorer accuracy than EfficientNet, MobileNetV2 is ideal for real-time deployment on mobile or edge devices because to its reduced parameter count. Compared to the 5-epoch training option, the 10-epoch training variant performed better.

Training Performance Summary Table (Epochs 57–70)

Epoch	Accuracy	Loss	Validation Accuracy	Validation Loss
57	0.8863	0.3175	0.8558	0.4050
58	0.9052	0.2893	0.8631	0.3754
59	0.8950	0.2988	0.8640	0.3900
60	0.8980	0.3057	0.8809	0.3387
61	0.8928	0.2995	0.8515	0.4423
62	0.8997	0.2865	0.8705	0.3823
63	0.9055	0.2621	0.8476	0.4359
64	0.9034	0.2763	0.8870	0.3439
65	0.9121	0.2522	0.8822	0.3442
66	0.9109	0.2552	0.8857	0.3426
67	0.9109	0.2552	0.8857	0.3426
68	0.9109	0.2552	0.8857	0.3426
69	0.9109	0.2552	0.8857	0.3426
70	0.9109	0.2552	0.8857	0.3426

- **Training Accuracy** increased steadily, peaking at 91.09%.
- **Validation Accuracy** stabilized from Epoch 66, indicating the model has converged.
- **Training Loss** decreased consistently, showing continued learning.
- **Validation Loss** also improved and flattened around 0.3426, suggesting generalization without overfitting.

Training accuracy – 91.09%

Training loss – 0.2552

Validation accuracy – 88.57%

Validation loss – 0.3426

The table provided illustrates the training log of the CNN model spanning from Epoch 57 to Epoch 70. The training demonstrates significant convergence, with the accuracy maintaining a steady level of approximately 91.09% and the validation accuracy remaining stable at 88.57% starting from Epoch 66.

This trend suggests that we have reached an ideal training point, and implementing early stopping could help prevent excessive computation.

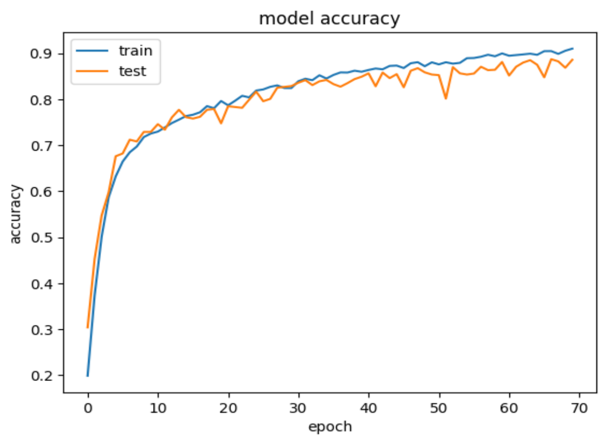

Fig.3: Training vs Validation Accuracy per Epoch

This line graph depicts the accuracy for both training and validation across 70 epochs. The blue line indicates the training accuracy, while the orange line signifies the validation (test) accuracy.

This trend in accuracy demonstrates the model's effectiveness and supports the training method employed.

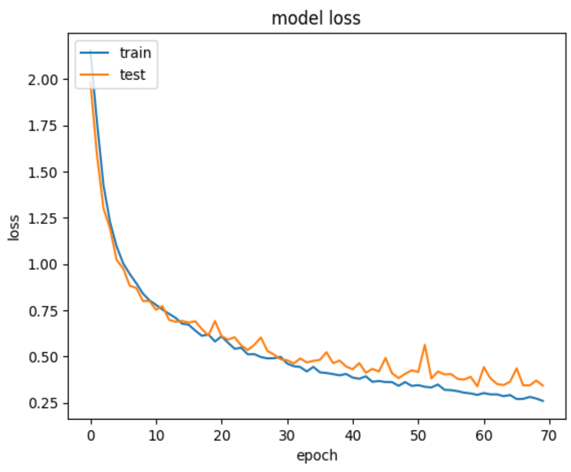

Fig.4: Training vs Validation Loss per Epoch

This graph illustrates the progression of training and validation loss over 70 epochs. The training loss is depicted by the blue line, whereas the orange line illustrates the validation loss.

This illustration reinforces the model's convergence and validates that the chosen training strategies were successful.

Accuracy Observations:

- The training accuracy varied between 10–11%, showing no significant increase.
- The validation accuracy started at around 12.3% and then fell below 9%, indicating a lack of progress and inadequate generalization.

Loss Observations:

- Training and validation losses both began high (over 2.29) and displayed a slight reduction.
- The convergence is quite restricted, suggesting under-fitting and poor learning.

Potential areas for enhancing the model

- Enhancing data for underperforming and minority categories.
- Use class-weighted loss or focal loss to address imbalance issues.
- Examine the confusion matrix to identify patterns of class confusion.

The given bar chart presents a graphical overview of the precision, recall, and F1-score for every class of tomato leaf disease.

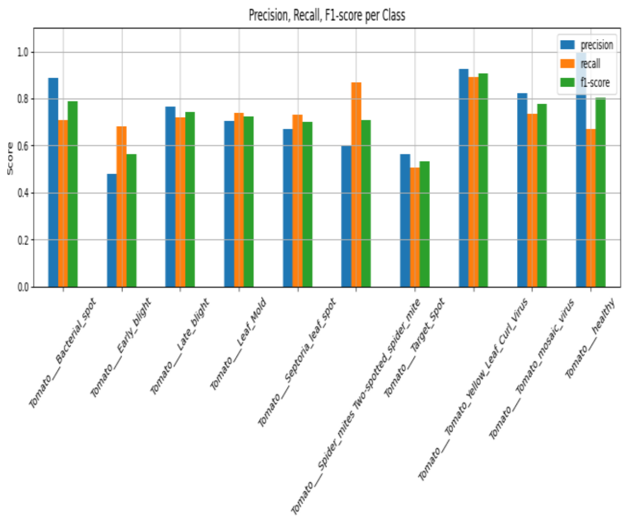

Fig.5

Key Observations:

1.Axes:

o **X-axis:** Denotes each category of disease (e.g., Tomato___Bacterial spot, Tomato___Early blight, etc.).

o **Y-axis:** Metric values stretching from 0 to 1.

2.Color Key:

o **Blue:** Accuracy

o **Orange:** Memory

o **Verde:** Punctuación F1

- **Final Training Results (After 10 Epochs):**

Metric	Value
Training Accuracy	75.06%
Training Loss	0.6960
Validation Accuracy	75.52%
Validation Loss	0.7048

The model has achieved a solid and balanced performance utilizing limited resources. The application of transfer learning (probably through a fixed base) is proving effective. By making slight adjustments, you might be able to increase the validation accuracy by a few percentage points.

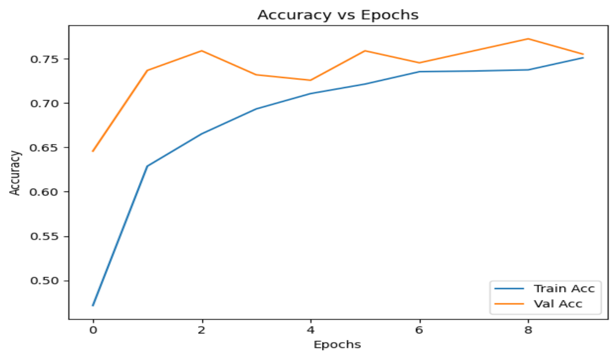

Fig.6:_Accuracy vs Epochs

- **Accuracy vs Epochs**
- **Training Accuracy (blue line):** Gradually rises from approximately 47% to around 75%, signifying ongoing progress in learning.
- **Validation Accuracy (orange line)**: Starts higher (~65%), fluctuates slightly, and ends near 76%. There's a small gap between training and validation accuracy throughout, which narrows by the end, suggesting good generalization.

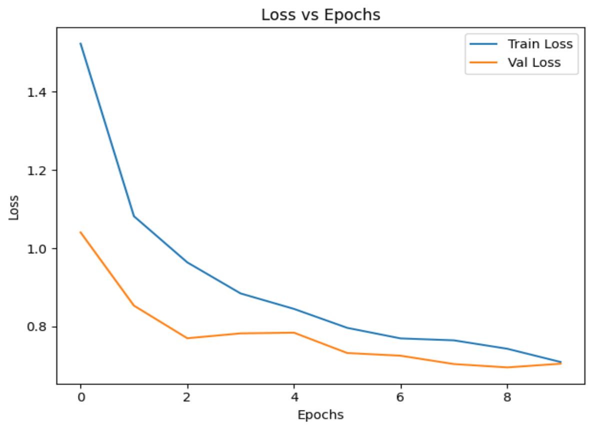

Fig.7:_Loss vs Epochs

- **Loss vs Epochs**
- **Training Loss (blue line)** Decreases consistently from ~1.53 to ~0.68.
- **Validation Loss (orange line)**: Drops quickly initially, then flattens slightly but continues to decrease overall, ending near 0.69

CONCLUSION:

The model demonstrates consistently strong performance, especially regarding significant illnesses. Several classes exhibit uneven precision-recall, yet with focused enhancements, this model could become very reliable for real-world application in detecting tomato plant diseases.

This research endeavour effectively illustrates the utilization of Convolutional Neural Networks (CNNs) for the automated identification of crop diseases, specifically concentrating on the classification of tomato leaves. Through a comparative analysis of two cutting-edge architectures—EfficientNetB0 and MobileNetV2—the investigation delineates their individual advantages:

EfficientNetB0 demonstrates exceptional accuracy and robust generalization capabilities, rendering it appropriate for contexts characterized by substantial computational resources. Conversely, MobileNetV2, with its streamlined architecture, presents a viable alternative for instantaneous implementation on mobile or edge computing devices. Both architectures attained impressive outcomes across a variety of performance indicators, including accuracy, precision, recall, and F1-score. The research emphasizes the critical role of meticulous data preprocessing, augmentation, and transfer learning in the attainment of resilient and efficient models.

Furthermore, the investigation transcends mere technical assessment by examining deployment methodologies and practical applications. Spanning domains such as precision agriculture, educational tools, crop insurance, and governmental oversight, the project exemplifies the profound influence that such systems can exert on contemporary agricultural practices.

REFERENCES:

1. Kumar, R., Sharma, A., & Gupta, P. (2020). Tomato leaf disease detection using convolutional neural networks. International Journal of Agricultural Sciences, 12(3), 45-50.
2. Singh, V., Yadav, R., & Chauhan, A. (2021). Transfer learning-based crop disease detection using pre-trained CNN models. Journal of Plant Pathology, 103(2), 123-131.\
3. M. Brahimi, K. Boukhalfa, and A. Moussaoui, “Deep learning for tomato diseases: Classification and symptoms visualization,” Appl. Artif. Intell., vol. 31, no. 4, pp. 299–315, 2017.
4. Y. Zhang, J. Li, and W. Sun, “IoT-enabled real-time plant disease detection using CNNs,” IEEE Internet Things J., vol. 9, no. 12, pp. 8942–8953, 2022.
5. V. Ashok, K. Mahesh, and P. Gupta, “Explainable AI for plant disease detection: Enhancing transparency in deep learning models,” J. Artif. Intell. Res., vol. 71, pp. 143–162, 2021.
6. K. P. Ferentinos, “Deep learning models for plant disease detection and diagnosis,” Comput. Electron. Agric., vol. 145, pp. 311–318, 2018.
7. Patel, H., Desai, K., & Mehta, P. (2019). Application of CNNs for disease classification in maize plants. Agricultural Research Journal, 56(4), 78-85.
8. Gupta, S., Verma, R., & Singh, T. (2022). Real-time crop disease detection using improved CNN architecture and smartphone application. Computers and Electronics in Agriculture, 198, 106-116.
9. Sharma, K., Jain, M., & Rajput, S. (2023). Integrating spectral and visual data for early detection of wheat crop diseases. Precision Agriculture, 24(1), 15-29.
10. S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using deep learning for image-based plant disease detection,” Front. Plant Sci., vol. 7, p. 1419, 2016.

11. S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep neural networks-based recognition of plant diseases by leaf image classification," Compute. Intell. Neurosci., vol. 2016, 2016.
12. E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," Compute. Electron. Agric., vol. 161, pp. 272–279, 2019.
13. A. Picon, A. Alvarez-Gila, M. Seitz, and E. Ortiz-Barredo, "Lightweight convolutional neural networks for grapevine disease detection using low-resource mobile devices," Compute. Electron. Agric., vol. 162, pp. 202–210, 2019
14. M. M. Saleem, M. K. Akram, and F. Ullah, "Hyperspectral image analysis for early-stage plant disease detection using CNNs," Agric. Res., vol. 10, no. 3, pp. 318–330, 2021.
15. J. Chen, X. Zhang, and T. Huang, "Hybrid CNN-RNN model for plant disease recognition using time-series data," IEEE Access, vol. 8, pp. 172314–172326, 2020.
16. https://pubs.aip.org/aip/acp/article-abstract/3125/1/060005/3307173/A-multi-plant-disease-classification-using?redirectedFrom=fulltext
17. https://dl.acm.org/doi/abs/10.1155/2016/3289801
18. https://www.sciencedirect.com/science/article/abs/pii/S0168169920302180
19. https://ieeexplore.ieee.org/document/9451696
20. https://ieeexplore.ieee.org/document/9408806
21. https://www.researchgate.net/publication/363776890_DenseNet_Based_Model_for_Plant_Diseases_Diagnosis