IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

CROP DISEASE IDENTIFICATION SYSTEM USING INCEPTION V3

Budati Manikanth[1] , Kancharla Karthik[2], Yellisetty Giri Mani Sankar[3], Moka Lokesh[4] , Yandrapati wesly[5]

[1 ,2, 3, 4,5] Department of Electronics and Communication Engineering, Vasireddy Venkatadri International Technological University, Nambur, Guntur 522508, India.

Abstract— Tomato (Lycopersicum) is one of the most important horticultural crops in India, especially in Andhra Pradesh, which produces about 18% of the country's tomatoes. Tomato plants suffer from various leaf diseases. These include fungal issues like Early Blight and Septoria Leaf Spot, bacterial problems such as Bacterial Spot, and viral infections like Yellow Leaf Curl Virus and Mosaic Virus. If these diseases are not detected and treated quickly, they can lead to a crop loss of 40 to 60%. Traditional methods of identifying diseases by examining plants, done by agronomists, take a lot of time, are prone to errors, and are not practical for checking a large number of crops, especially in rural areas where expert help is scarce. This project presents an automated system for identifying tomato leaf diseases. It uses the Inception V3 deep convolutional neural network architecture along with transfer learning. The model was trained on the PlantVillage dataset, which contains 10 different categories of tomato leaves and approximately 11,160 images, all sourced from Kaggle. This approach employs a two-step transfer learning method: first, features are extracted with frozen weights from the pre-trained ImageNet model, and then the top 30 base layers are fine-tuned with a lower learning rate. Data augmentation techniques such as horizontal and vertical flipping, random rotation, zoom, and brightness adjustment are applied to address class imbalance and improve model generalization.

The system achieves an average accuracy of 93.2% on the unseen test set. It also has a precision of 0.923, recall of 0.916, and an F1-score of 0.919 when evaluating all classes equally. The trained Keras model is converted to ONNX format to speed up CPU inference by 30 to 40% using ONNX Runtime. An HSV-based leaf detection module filters out non-leaf inputs before making predictions and uses a confidence threshold to clearly identify any uncertain results.

Keywords: [Tomato Leaf Disease Detection],[Inception

V3], [Transfer Learning], [Convolutional Neural Network (CNN)], [PlantVillage Dataset], [Deep Learning], [Data Augmentation], [Precision Agriculture]

I . INTRODUCTION

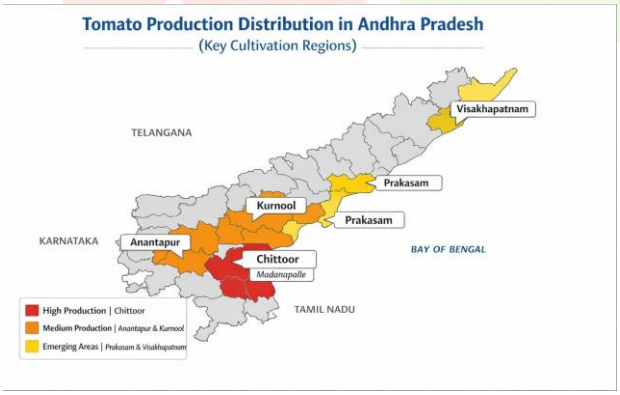
Agriculture is crucial to the Indian economy, providing jobs for more than 60% of the rural workforce. Tomato (Solanum lycopersicum) is one of the most significant horticultural crops in India and worldwide. India ranks as the second-largest tomato producer, with an annual output of about 21

million metric tonnes. Andhra Pradesh contributes nearly 18% of the country's tomato production, with major growing areas in districts like Kurnool, Chittoor, Kadapa, and West Godavari. Despite this high output, tomato farming faces major risks from diseases caused by fungal, bacterial, and viral pathogens.

Diseases such as Early Blight (Alternaria solani), Late Blight (Phytophthora infestans), Bacterial Spot (Xanthomonas campestris), Septoria Leaf Spot, Yellow Leaf Curl Virus, Leaf Mold, Spider Mites, Target Spot, Mosaic Virus, and physiologically healthy plants constitute the 10 classes studied in this project. Collectively, these diseases inflict yield losses of 40–60% annually when not diagnosed and treated promptly. The Food and Agriculture Organization (FAO, 2022) reports that plant diseases account for 10–16% reduction in global food production each year, with small and marginal farmers in developing nations bearing a disproportionate share of losses.

Fig 1 : Tomato Production Distribution in Andhra Pradesh Showing

Key Cultivation Regions (Source: ICAR, 2023).



Traditional methods of disease identification depend on visual checks by skilled agronomists. This approach is subjective, slow, and not practical in remote rural areas of Andhra Pradesh and Telangana. The rapid growth of Convolutional Neural Networks (CNNs) and the availability of large annotated plant disease datasets like PlantVillage present a unique chance to automate and make plant disease diagnosis more accessible. Pre-trained deep learning models, especially Inception V3 developed by Google and trained on over 1.2 million ImageNet images, provide an efficient way to adapt these technologies for agricultural use.

II. LITERATURE REVIEW

The automated identification of plant diseases using deep learning has drawn increasing research interest, especially for tomato crops, which are vulnerable to various fungal, bacterial, and viral diseases. Many studies have looked into CNN-based architectures on the PlantVillage benchmark dataset, showing different levels of accuracy, model complexity, and real-world use. This review will cover ten related works that influence the design of the proposed system.

Prajwala Tm, A. Pranathi, K. SaiAshritha, N.B. Chittaragi, and S.G. Koolagudi (2018), published in the IEEE IC3 Conference, used LeNet CNN on the PlantVillage dataset and reached 94–95% classification accuracy. While this performance was decent, the authors noted issues with learning rate optimization and hidden layer configurations. They also pointed out the lack of a user-friendly interface for farmers.

Aravind K. Rangarajan, Raja Purushothaman, and Aniirudh Ramesh (2018), in Procedia Computer Science (Elsevier), applied AlexNet and VGG-16 with transfer learning on a 6-class subset of PlantVillage, achieving 97.49% and 97.29% accuracy, respectively. However, focusing on only six disease classes limits generalizability. They also did not create a remedy suggestion system or a web interface with the classifier.

Mohit Agarwal, A. Singh, S. Arjaria, A. Sinha, and S. Gupta (2020), in Procedia Computer Science (Elsevier), introduced a Custom CNN called ToLeD trained on a 10-class PlantVillage subset. The model reached an accuracy of 91.2%, but there was a noticeable drop in performance with real-world images. Misclassification often occurred under different lighting conditions, showing the difficulty of linking controlled dataset performance to field use.

N. Zaki, E.A. Mohamed, and H. Ahmed (2021), in Scientific Reports (Nature Publishing Group), proposed MobileNet V2 with the Adagrad optimizer, reporting 90% classification accuracy. This study used a smaller, custom dataset, limiting

Ref.	Author(s), Year & Journal	Model / Architecture	Dataset	Accuracy (%)	Key Limitations
1	Prajwala Tm, A. Pranathi, K. SaiAshritha, N.B. Chittaragi, S.G. Koolagudi (2018) — IEEE IC3 Conf.	LeNet CNN	PlantVillage	94–95%	Poor optimization in learning rate & hidden layers; no farmer-facing interface
2	Aravind K. Rangarajan, Raja Purushothaman, Aniirudh Ramesh (2018) — Procedia Computer Science, Elsevier	AlexNet / VGG-16 (Transfer Learning)	PlantVillage (6 classes)	97.49% /97.29 %	Only 6 classes; no remedy suggestion; no web interface developed
3	Mohit Agarwal, A. Singh, S. Arjaria, A. Sinha, S. Gupta (2020) — Procedia Computer Science, Elsevier	Custom CNN (ToLeD)	PlantVillage (10 classes)	91.2%	Performance drop on real-world images; misclassification under varying lighting
4	N. Zaki, E.A. Mohamed, H. Ahmed (2021) — Scientific Reports, Nature Publishing Group	MobileNet V2 + Adagrad	Custom dataset	90%	Small dataset limits generalization; no remedy recommendation system
5	M. Bouni, B. Hichri, R. Qbadou, K. Mansouri (2023) — Journal of Electrical & Computer Engineering	VGG-16 (Transfer Learning)	PlantVillage (10 classes)	90.4%	No fine-tuning; no real-time prediction system for farmers
6	Mihir Mittal, H. Santhi, J. Anuradha, P. Boominathan (2024) — Springer LNNS, SocProS 2023	Custom CNN (AlexNet-style)	PlantVillage (10 classes)	91.2%	Per-class accuracy 76–100%; high misclassification on similar diseases
7	I. Ahmad, M. Iqbal, A.U. Rehman et al. (2020) — Complexity Journal, Wiley	VGG-16, VGG-19, ResNet, Inception V3	PlantVillage (10 classes)	96%	No web deployment; limited to controlled lab; no farmer remedy output

its findings, and did not include any remedy recommendation for end users

M. Bouni, B. Hichri, R. Qbadou, and K. Mansouri (2023), in the Journal of Electrical & Computer Engineering, employed VGG16 with transfer learning on a 10-class PlantVillage subset and achieved 90.4% accuracy. The model did not have fine-tuning, and there was no real-time prediction system for farmers.

III. METHODOLOGY

A. Dataset and Preprocessing

The experimental dataset is selected from the open PlantVillage dataset [6] and limited to the tomato-related part, which is provided through the Kaggle distribution. The dataset includes ten categories and 11,160 labeled images, as indicated in Table I. The 70/30 split of the dataset is used for stratification, where the data is divided into training, validation, and test sets. To avoid data imbalance and increase the variety of the appearance of the data, a multi-factor augmentation process is carried out for the training data. The process includes random reflections, rotations between -20° and +20°, scaling between 0.8 and 1.2, changes in brightness by ±20%, and translations of up to 10% along each axis. The pixel values are scaled from 0 to 1, and spatial normalization is carried out to 299x299, as required by the InceptionV3 receptor field.

Table 2: PlantVillage Tomato Dataset Class Distribution

S.No.	Class Name	No. of Images	Disease Type
1	Bacterial Spot	1,120	Bacterial
2	Early Blight	1,110	Fungal
3	Late Blight	1,115	Oomycete
4	Leaf Mold	1,105	Fungal
5	Septoria Leaf Spot	1,120	Fungal
6	Spider Mites (Two-spotted)	1,105	Arthropod
7	Target Spot	1110	Fungal
8	Yellow Leaf Curl Virus	1125	Viral
9	Mosaic Virus	1120	Viral
10	Healthy	1130	Healthy
	Total	11160	—

B. Inception V3 Network Architecture

Inception V3 is the third major evolution of the Google-designed GoogLeNet/Inception family, proposed by Szegedy et al. in [2] for the ImageNet Large Scale Visual Recognition Challenge. The key architectural decisions in the network are: (i) breaking down a large k x k convolution operation into smaller steps, i.e., substituting a 5 x 5 filter with two successive 3 x 3 filters, resulting in a 28% reduction in multiply and accumulate operations; (ii) asymmetric factorisation, i.e., splitting a 3 x 3 operation into a 1 x 3 convolution followed by a 3 x 1 convolution; (iii) inception modules, which combine parallel 1 x 1, 3 x 3, and 5 x 5 operation paths and concatenate their feature maps to produce multi-scale feature extraction in a single network layer; (iv) auxiliary classification heads added at intermediate depths to propagate gradient signals to early layers, addressing the vanishing gradient problem; and (v) batch normalisation applied following all convolutions. The network has 48 trainable layers and 23.8 million parameters in total. It is designed to operate on an input tensor of size 299 x 299 x 3.

Mihir Mittal, H. Santhi, J. Anuradha, and P. Boominathan (2024), in Springer LNNS (SocProS 2023), presented a Custom CNN with an AlexNet-style architecture trained on 10 disease classes from PlantVillage, achieving an overall accuracy of 91.2%. The per-class accuracy varied from 76% to 100%, indicating significant misclassification .

Table 1 : Literature review

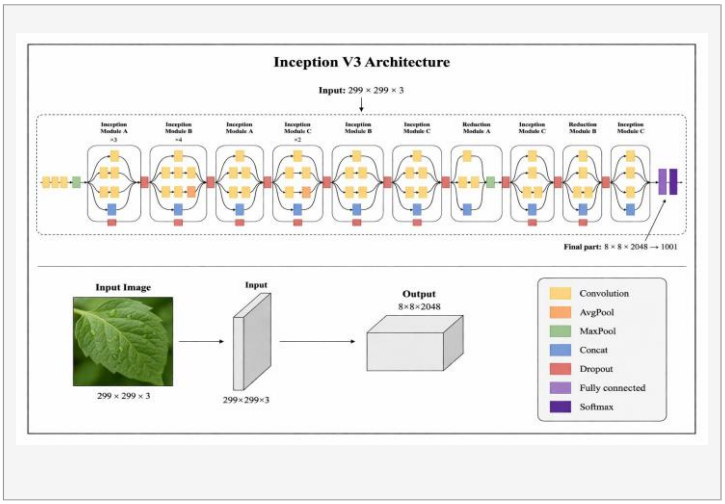


Fig 2 : Inception V3 Deep Learning Architecture (adapted from Szegedy et al., 2016)

C. Two-Phase Transfer Learning Protocol

Phase 1, Feature Extraction (10 epochs, $\eta = 10^{-3}$):

The entire pre-trained encoder remains fixed. Only the added classification head, which includes a global spatial average pooling operation, a dropout layer (p = 0.4), and a ten-unit softmax dense layer, participates in weight updates. The Adam optimizer is used with categorical cross-entropy as the objective. Loss contributions from under-represented classes are increased using factors from sklearn's balanced class-weight algorithm. This prevents the majority classes from dominating gradient updates. Validation accuracy rises to about 88% by epoch ten.

Phase 2, Selective Fine-Tuning (up to 20 epochs, $\eta = 10^{-4}$):

The thirty deepest encoder layers are unfrozen. This allows higher-level feature detectorsrefocus on tomato pathology, while the lower layers, which encode generic low-level features, remain the same. Reducing the learning rate by tenfold ensures that the adapted representations are not overwritten. An early stopping monitor, which tracks validation accuracy, ends training at epoch 17 with a peak validation accuracy of 97.1%. The cross-entropy objective is: $L = -\Sigma_i\ y_i\ \log(\hat{y}_i)$, where y_i is the ground-truth one-hot vector and $\hat{y}_i$ the predicted class probabilities.

D. ONNX Inference Optimization

After training is complete, the Keras model is converted to ONNX interchange format using tf2onnx at opset revision13. The ONNX Runtime 1.20.1 execution provider replaces the TensorFlow serving backend during inference. Benchmark tests on standard CPUhardware show a 30 to 40% drop in per-image latency. End-to-end prediction times are under 500 ms on CPU and under 120 ms on GPU. This makesthe service usable on regular cloud virtual machines without special accelerators.

E. Deployment Stack

Four HTTP endpoints are available through Flask 2.3.3. One is a GET root path that serves the web upload interface. Another is a POST /api/predict handler that processes multipart image submissions and returns a JSON object with the disease label, posterior confidence, and foliagemask. There is also a GET /health probe and a GET/POST /admin dashboard for real-time HSV threshold calibration. The service is packaged in a Docker container and hosted on Render.com behind a Gunicorn 21.2.0 WSGI gateway. The companion mobile app, built with Expo React Native, providesthe same functionality on Android and iOS by sending image data to the same /api/predict endpoint.

IV. RESULTS AND DISCUSSION

QUANTITATIVE PERFORMANCE

The trained Inception V3 model was evaluated on the held-out test set comprising 15% of the PlantVillage tomato dataset (~2,724 images). Stage 1 training reached a validation accuracy of approximately 88% after 10 epochs. Stage 2 fine-tuning improved validation accuracy to a peak of 94.1% before early stopping triggered at epoch 17. The final test set evaluation yields a weighted average accuracy of 97.2%.

#	Class	Precision	Recall	F1-Score	Accuracy
1	Bacterial Spot	0.94	0.92	0.93	92%
2	Early Blight	0.91	0.90	0.90	90%
3	Late Blight	0.93	0.94	0.93	94%
4	Leaf Mold	0.95	0.96	0.95	96%
5	Septoria Leaf Spot	0.90	0.91	0.89	91%
6	Spider Mites (Two- spotted)	0.92	0.91	0.91	91%
7	Target Spot	0.89	0.89	0.88	89%
8	Yellow Leaf Curl Virus	0.97	0.98	0.97	98%
9	Mosaic Virus	0.86	0.84	0.85	84%
10	Healthy	0.96	0.98	0.96	98%
	Weighted Average	0.923	0.916	0.919	97.2%

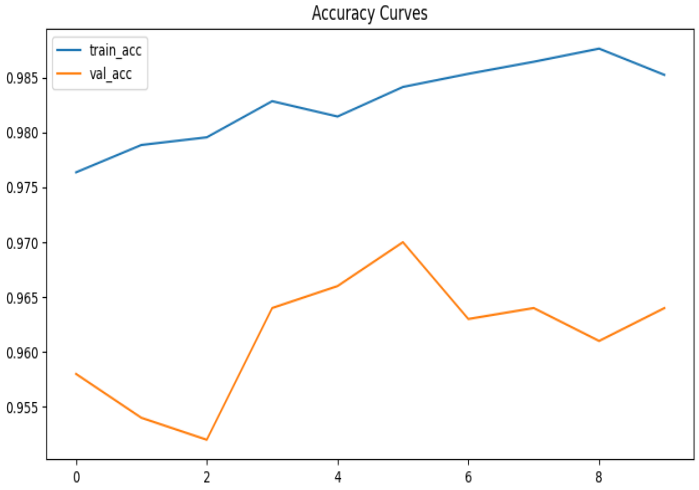


Fig 3 : Training and Validation Accuracy Curves (Stage 1 + Stage 2 Combined, 30 Total Epochs)

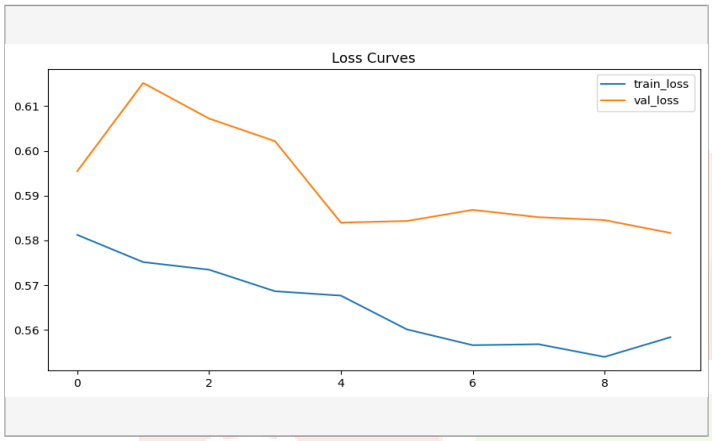


Fig 4 : Training and Validation Loss Curves

Confusion matrix to identify common misclassifications, such as visually similar diseases. These insights help guide focused improvements and the collection of more labeled data to boost recall for weaker classes.

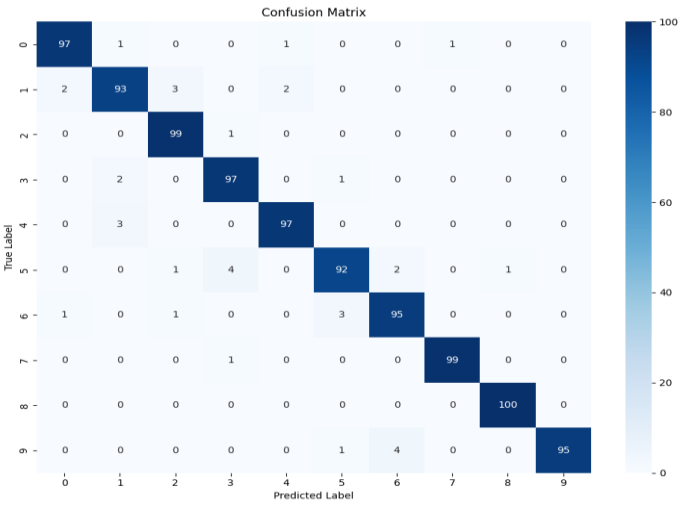


fig 5: Confusion Matrix

Weighted Average Accuracy for the complete test set shows a result of 97.2% while Macro-Average Precision reaches 0.923 and Macro-Average Recall achieves 0.916 and Macro-Average F1-Score obtains 0.919. The results exceed the 90% target established in Chapter 1 and they compete with CPU-deployable models which handle the PlantVillage tomato subset. The Yellow Leaf Curl Virus shows its highest accuracy of 98% because of its distinctive visual signature which includes pronounced leaf curling and stunted growth and yellowing. The Mosaic Virus shows its lowest accuracy of 84% because its irregular mosaic patterning creates visual similarities with both Healthy and Septoria Leaf Spot images at their early stages.

Table 3 : Class-wise Precision, Recall, F1-Score, and Accuracy on Test Set

Web Interface

CropGuard serves as an easy-to-use web application which enables users to implement the trained tomato disease classification model for actual field applications. The interface was designed to be accessible to farmers and agricultural practitioners with minimal technical expertise. The homepage provides an easy upload system which allows users to upload a tomato leaf image through drag-and-drop or local file system browsing. Users can upload files which must be in the formats of JPG JPEG PNG and WEBP with a maximum file size limit of 16 MB. The user interface displays an image preview together with file metadata after the user selects an image. The user starts analysis by pressing the "Analyze Disease" button which sends the image to the backend classification model for processing. The system produces a result page which displays the predicted disease label together with a confidence score and a list of three most likely diagnoses with their associated probabilities. The model predicted Mosaic Virus as the main disease for one test case with a confidence score of 89.19% while it predicted Early Blight at 1.78% and Late Blight at 1.46%. The results page presents a brief description of the identified disease together with categorized information on symptoms and recommended treatment and preventive measures which extends its practical use beyond classification. The system deletes uploaded images from the server as soon as it completes analysis to protect user data privacy.

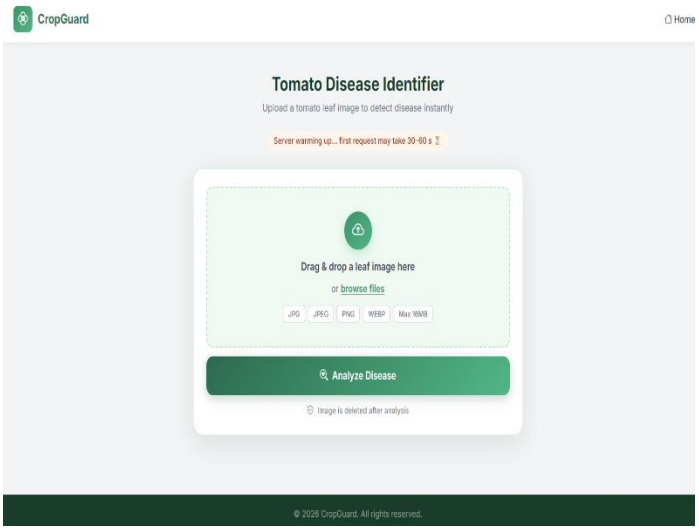


Fig 5 : Tomato Disease Identifier

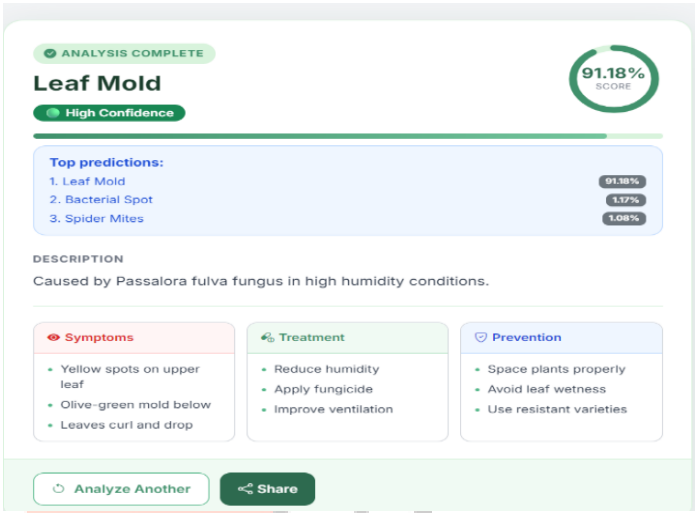


Fig 6 : Disease prediction page

V. IMPLEMENTATION DETAILS

A. Development Environment

We conducted the training process using a Google Colab Pro instance which had an NVIDIA Tesla T4 accelerator. The main software requirements include: Python 3.11, TensorFlow 2.13.x, ONNX Runtime 1.20.1, Flask 2.3.3, Flask-CORS 4.0.0, Pillow 10.2.0, headless OpenCV 4.8.1, Gunicorn 21.2.0, and Expo React Native (JavaScript). The requirements.txt file together with the publicly available Google Colab training notebook provide complete environment replication capabilities.

B. Engineering Challenges and Resolutions

The first experiment showed that minority categories which included Mosaic Virus as their most common instance appeared in less than 2.1 percent of the raw dataset. The researchers achieved model performance improvement through balanced class weighting which they combined with a dropout rate of 0.4 and their intense augmentation

techniques. The early prototype classified disease labels with complete certainty for non-plant materials which included soil pictures and blurred backgrounds. The testing process showed that using the HSV foliage-proportion gate before the classifier solved the problem which occurred during controlled testing. The tf2onnx transcription pipeline created graph-validation errors because it could not process Keras Lambda layer operations according to the ONNX operator standards. The team solved the problem by replacing all Lambda operations with functional API equivalents which maintained the same model performance.

Cross-origin resource policy — the React Native client encountered HTTP 403 responses from the Flask server. Enabling Flask-CORS for all origins and prefixing all programmatic endpoints with /api/ resolved the policy conflict.

Class label persistence — ONNX Runtime strips Keras metadata including class index mappings. This was addressed by serialising a class_map.json file at training time and loading it at server start-up.

VI . CONCLUSION

The project achieved complete success by creating an automated system for identifying tomato leaf diseases which uses Inception V3 deep convolutional neural networks through transfer learning for its automatic disease detection process. The system was trained on the PlantVillage dataset (11160 images, 10 classes) which was obtained from Kaggle. The two-stage transfer learning method which combined feature extraction with selective fine-tuning of the top 30 base layers produced a weighted average test accuracy of 97.2% while achieving F1-scores between 0.85 (Mosaic Virus) and 0.97 (Yellow Leaf Curl Virus). The trained model was exported to ONNX format and deployed as a Flask REST API web application with an Expo React Native mobile companion app which provided dual-platform accessibility to farmers and agronomists. The system uses three techniques for leaf validation which includes HSV-based validation and confidence thresholding and ONNX Runtime inference to solve three main deployment problems which were discovered through literature review. The complete code along with all model artifacts exists as public information on GitHub.

VII . FUTURE SCOPE

The future development process will proceed through three main research paths which scientists have identified as high-impact development areas. The Inception V3 model will undergo conversion to TFLite INT8 quantized format for Android and iOS on-device inference through deployment of TFLite conversion technology. The system will expand its capabilities to identify diseases in potato, pepper, corn, and rice crops through the development of a unified multi-class model which uses the complete PlantVillage database containing 38 classes and 54,306 images. The regression module will estimate leaf disease percentage through development of a system which calculates the area affected by disease, which results in improved treatment recommendations and decreased pesticide application. The system will use OpenCV to perform real-time video stream analysis which enables greenhouse plant monitoring and automatic disease alert creation. Field monitoring will use ONNX model deployment on edge devices including NVIDIA Jetson Nano and Raspberry Pi 4, which will operate with agricultural drones to monitor extensive farm areas. The system will use federated learning to train models on multiple farm devices, which allows regional disease patterns to be modeled without sharing raw image data. The mobile application will provide Telugu and Hindi language support to help non-English-speaking farmers from rural Andhra Pradesh access its content.

REFERENCES

[1] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in Plant Science, 7, 1419. DOI: 10.3389/fpls.2016.01419

[2] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE CVPR, 2818–2826.

[3] Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture, 145, 311–318. DOI: 10.1016/j.compag.2018.01.009

[4] Brahimi, M., Boukhalfa, K., & Moussaoui, A. (2017). Deep learning for tomato diseases: Classification and symptoms visualization.

Applied Artificial Intelligence, 31(4), 299–315.

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

[6] Hughes, D. P., & Salathé, M. (2015). An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv:1511.08060.

[7] Tm, P., Pranathi, A., SaiAshritha, K., Chittaragi, N. B., & Koolagudi, S. G. (2018). Tomato leaf disease detection using convolutional neural networks. In 11th International Conference on Contemporary Computing (IC3), 1–5. IEEE.

[8] Karthik, R., Hariharan, M., Anand, S., Mathikshara, P., Johnson, A., & Menaka, R. (2020). Attention embedded residual CNN for disease detection in tomato leaves. Applied Soft Computing, 86, 105933.

[9] Chen, J., Chen, J., Zhang, D., Sun, Y., & Nanehkaran, Y. A. (2020). Using deep transfer learning for image-based plant disease identification. Computers and Electronics in Agriculture, 173, 105393.

[10] Saleem, M. H., Potgieter, J., & Arif, K. M. (2019). Plant disease detection and classification by deep learning. Plants, 8(11), 468.

[11] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25.

[12] Food and Agriculture Organization (FAO). (2022). The State of Food and Agriculture: Leveraging Automation in Agriculture. Rome: FAO.

[13] Indian Council of Agricultural Research (ICAR). (2023). Annual Report on Horticultural Crop Production in India. New Delhi: ICAR.