RESEARCH ARTICLE OPEN ACCESS

Agri Care: An AI-Based Image Processing System for Early Detection of Plant Diseases Using Deep Learning

Prof. Amol Jadhav*, Sanket Khapake**, Prasanna Jadhav***,
Prasad Hajgude****, Gaurav Jaiswal *****

*Information Technology , Zeal Collage of Engineering And Research Pune, and Pune
** Information Technology , Zeal Collage of Engineering And Research Pune, and Pune
*** Information Technology , Zeal Collage of Engineering And Research Pune, and Pune
**** Information Technology , Zeal Collage of Engineering And Research Pune, and Pune
***** Information Technology , Zeal Collage of Engineering And Research Pune, and Pune

ABSTRACT
Finding plant leaf diseases early is very important for keeping crops healthy and cutting down on production loss. We have made AgriCare, a smartphone app that helps people by using pictures to sort plant leaf diseases. The EfficientNetB3 model architecture for identifying plant diseases uses the PlantVillage dataset, which has 33 different types of plant diseases. Before being sent to the trained model for processing, images of leaves that were taken or uploaded are normalized and scaled to 224 by 224 pixels. During training, the system was rotated and flipped to make it work better and make it more flexible in different situations. We changed the model into TensorFlow Lite format, which we will then add to a mobile app that is based on Flutter. The suggested Disease Prediction makes the possible results look like they happen a little faster and works on the device without using mobile data. It also helps people keep their information safe. The app shows the expected name of the disease, the level of confidence, and some basic advice on how to avoid getting sick. Firebase is just used to keep track of prediction history and handle user logins. The established system is useful for practical agricultural support because it gives consistent and reliable results, which is especially important in remote areas where internet access may be limited.

Keywords: - Plant Leaf Disease, Image-Based Classification, EfficientNetB3, Mobile Application, On-Device Processing, Agriculture

I. INTRODUCTION

Globally, agriculture is essential to maintaining food security and promoting economic expansion. Crop quality, production, and farmer income are all directly impacted by crop health. Plant leaf infections are one of the many issues in farming that have a significant impact on productivity. Diseases can spread swiftly and cause significant crop damage and financial loss if they are not identified early.

Traditionally, farmers or agricultural specialists manually inspect plants to identify diseases. Experience and visual observation play a major role in this approach. Manual monitoring becomes laborious and occasionally inaccurate in big farming regions. Inadequate treatment and decreased crop yield could result from delayed identification.

Mobile-based systems are now a feasible method for diagnosing plant diseases due to the increasing use of smartphones and improvements in image processing. Images of leaves can be categorized into various disease classes using image classification methods. Mobile applications can benefit from the great performance that architectures like EfficientNet have shown in picture classification tasks while maintaining computational efficiency.

Nevertheless, a lot of current mobile systems mostly rely on cloud-based servers for forecasting. Stable internet access is necessary for these systems, although it is frequently scarce or erratic in rural agricultural areas. Continuous reliance on distant servers may also result in slower response times and lower dependability. The general applicability of disease detection software in actual farming conditions is impacted by these difficulties.

Reducing reliance on high-speed or continuous internet access for plant disease categorization is the main goal of this study. A system that can function well even in situations with limited network availability while preserving accessibility and speed is required.

In order to meet this need, this study presents AgriCare, a smartphone application that effectively classifies plant diseases. Utilizing the EfficientNetB3 architecture, the system is incorporated into a Flutter-based application after being transformed into TensorFlow Lite format. Predictions can be made with little reliance on the internet thanks to the categorization process's optimization for the device. The main purposes of internet access are user authentication and Firebase services' synchronization of prediction history.

The creation of a mobile-based system for classifying plant diseases that works well in settings with spotty or poor internet connectivity is the main contribution of this work. The technology increases accessibility for farmers in rural locations, improves data privacy, and speeds up response times by reducing reliance on cloud-based prediction. The suggested

solution provides a dependable and useful method for assisting contemporary farming methods.

II. RELATED WORK

Different ways to use images to find plant leaf diseases have been suggested over the years. Examining prior research aids in comprehending current methodologies, their advantages, and their practical constraints.

Methods for Machine Learning and Traditional Image Processing :

In the past, plant disease detection systems mostly used traditional image processing methods along with machine learning classifiers like Support Vector Machines (SVM), kNearest Neighbours (k-NN), and Random Forest algorithms. Using colour analysis, texture measurement, and shape detection, these methods manually pulled features from images of leaves.

These methods worked well in controlled settings, but their performance was often affected by changes in lighting, background complexity, and the way the leaves looked. The system relied heavily on domain knowledge because features had to be designed and chosen by hand. It was also not easy to scale for large or varied datasets.

Approaches Based on Convolutional Neural Networks :

As image classification techniques got better, Convolutional Neural Networks (CNNs) became the most popular way to find plant diseases. Models based on CNNs can automatically learn image features during training, which means that manual feature extraction is less necessary. Research utilising datasets like PlantVillage has indicated elevated classification accuracy through the application of deep network architectures.

But deeper models usually need more memory and processing power. This makes it hard to use them directly in mobile apps, especially in places where device resources are limited.

VGG16 for Classifying Plant Diseases :

Several studies on classifying plant diseases have used the VGG16 architecture. It can get detailed image features because it has a deep layered structure and a uniform convolutional design. Numerous researchers have attained robust classification performance employing VGG16 through transfer learning.

VGG16 works well, but it has a lot of parameters, which makes the model bigger and costs more to run. This can make the model less efficient when it's being used in mobile systems that need faster inference and less memory.

Models Based on EfficientNet :

EfficientNet models came up with a balanced scaling method that changes the depth, width, and resolution of the network at the same time. Some versions, like EfficientNetB3, have shown better performance while still needing less computing power. EfficientNet models are thought to be good for mobile environments when they are properly optimized because they strike a good balance between accuracy and efficiency.

When deploying a model on a mobile device, you need to use model conversion and optimization techniques to make sure it works well on smartphones with limited hardware.

Apps for Plant Diseases on Mobile Devices :

Smartphone cameras can be used to find plant diseases with the help of a number of mobile apps. Most of these apps use server-side processing, which means that images are sent to cloud systems to be sorted. This method lets you use powerful hardware to make predictions, but it needs a stable internet connection and could cause delays. Network availability is often spotty in rural farming areas, which makes it less useful. Also, sending pictures to servers far away could make people worry about the safety and privacy of their data.

Even though past research has helped a lot to make plant disease classification better, it is still hard to make a system that works well on mobile devices with little or no network dependence. This research proposes the AgriCare system, which centers on the implementation of an optimized EfficientNetB3 model converted to TensorFlow Lite format for mobile integration. The system works better in real-world agricultural settings because it doesn't rely on constant cloudbased prediction. This makes response times faster, privacy better, and gives a practical solution.

III. PROPOSED METHOD

The AgriCare system uses a mobile app to classify plant leaf diseases into multiple classes. The full method includes preparing the dataset, preprocessing the images, making the model, training it, testing it, and putting it on mobile devices.

The workflow starts with getting the image, then preprocessing it, using the trained model to classify it, and finally showing the results in the mobile app interface.

1. GATHERING DATASETS:

The dataset utilized for this research is sourced from the publicly accessible PlantVillage repository. There are 54,306 plant leaf images in this collection, divided into 33 groups. Eight of these groups are for healthy plants and 25 are for sick plants from different types of crops. The dataset is split up like this for testing:

- Training Set: 80% of all the pictures
- Validation Set: 20% of all images

This split makes sure that the model learns correctly and that the performance evaluation is fair.

2. IMAGE PREPROCESSING:

To keep things the same, all images are resized to 224 × 224 pixels, which is what the chosen architectures need for input.

Using min-max scaling, pixel values are made to be the same:

$$I_{normalized} = \frac{I}{255}$$

where I is the original value of the pixel's intensity.

The following augmentation methods are used during training to make generalization better and overfitting less likely:

- Rotation (±25°)
- Width shift (0.2)
- Height shift (0.2)
- Shear transformation (0.15)
- Zoom (0.2)
- Horizontal flipping

This is a multi-class classification problem with 33 output categories, so we use one-hot encoding to encode the labels.

3. MODEL STRUCTURES :

Three different models were used and tested to find the best architecture.

A. Custom CNN

A basic Convolutional Neural Network was made with three convolutional blocks and fully connected layers after that. The structure has:

- Conv2D (32 filters) + ReLU
- Conv2D (64 filters) + ReLU
- Conv2D (128 filters) + ReLU
- MaxPooling layers
- Flatten layer
- Dense layer (128 units)
- Dropout (0.5)
- Output layer (33 neurons with Softmax activation)

This model is the standard architecture for comparing performance

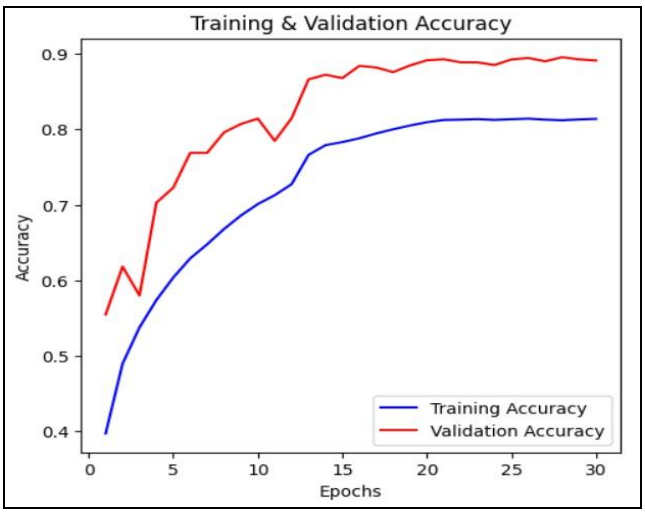

Figure 1: Custom CNN Accuracy Plot

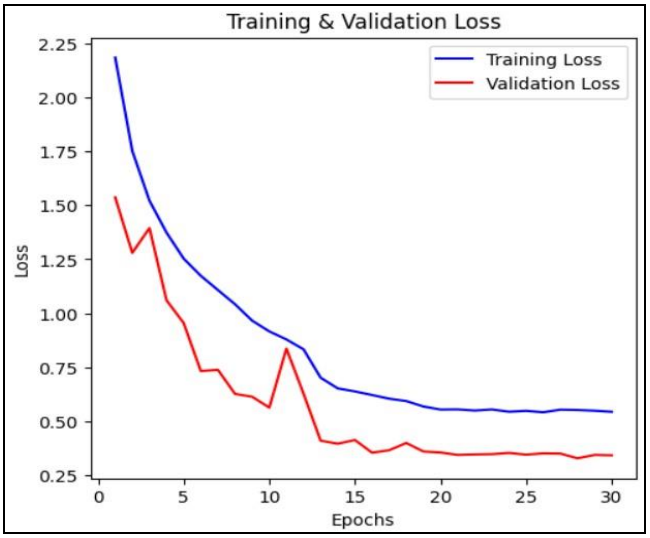

Figure 2: Custom CNN Loss Plot

B. VGG16

We used transfer learning with pre-trained ImageNet weights to build the VGG16 architecture. The convolutional base was kept for feature extraction, and the final classification layers were changed to work with 33 different disease types.

This model gives a more detailed representation of features than the custom CNN, but it has more parameters.

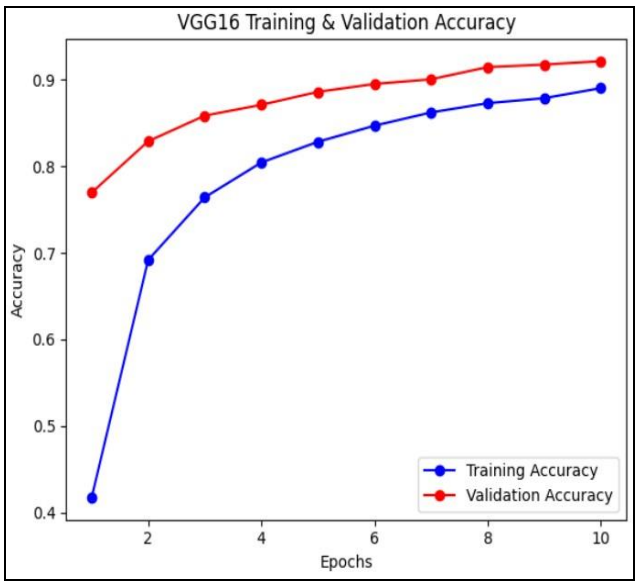

Figure 3: Vgg16 Accuracy Plot

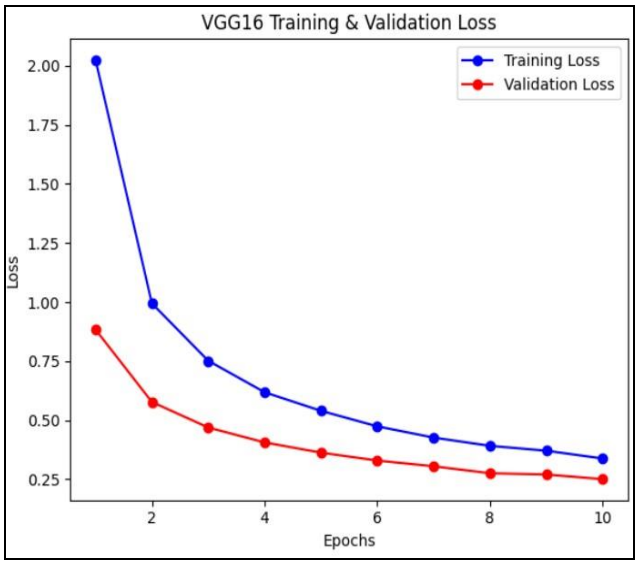

Figure 4: Vgg16 Loss Plot

C. EfficientNetB3

The EfficientNetB3 model was chosen because it scales network depth, width, and resolution in a balanced way. The ImageNet weights were used to start it off, and the PlantVillage dataset was used to fine-tune it.

EfficientNetB3 had the best validation performance of the three architectures and was more efficient for mobile deployment.

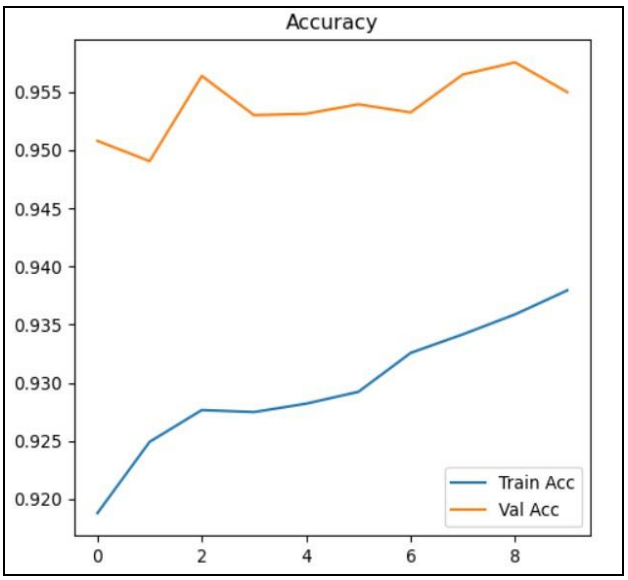

Figure 5: EfficientNetB3 Accuracy Plot

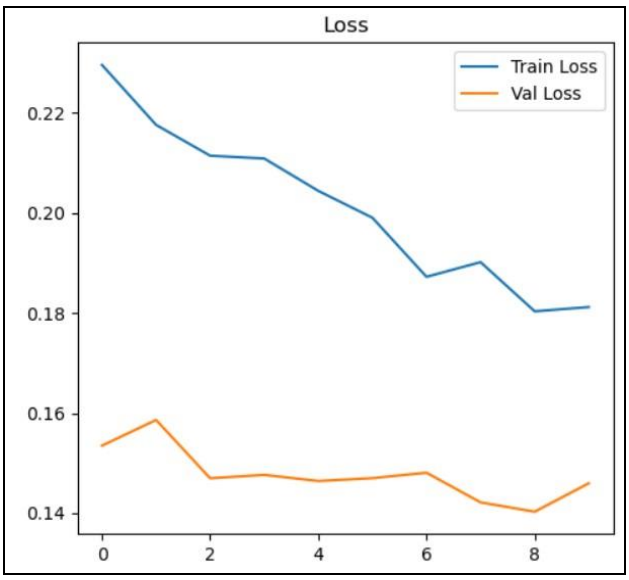

Figure 6: EfficientNetB3 Loss Plot

4. TRAINING PROCEDURE

To make sure that the comparisons were fair, all of the models were trained with the same settings. We used TensorFlow and Keras to train the models.

- Optimizer: Adam
- Learning Rate: 0.001
- Batch Size: 32
- Epochs: 30
- Loss Function: Categorical Cross-Entropy

To improve generalization and stop overfitting, the following methods were used:

- Dropout (0.5)

- EarlyStopping to monitor validation loss
- ModelCheckpoint to save the best-performing model
- ReduceLROnPlateau to adjust learning rate dynamically

These strategies made training more stable and improved the model's overall performance. Among all the models that were tested, EfficientNetB3 did the best.

5. DEPLOYMENT ARCHITECTURE

After training, the last EfficientNetB3 model was changed to TensorFlow Lite (TFLite) format so that it could run quickly on mobile devices. TensorFlow Lite makes models smaller and faster for smartphones.

A Flutter-based mobile app now uses the TFLite model. The classification process happens on the smartphone itself, so it doesn't need to be connected to the internet all the time.
You mostly need the internet to log in and sync your data.

Prediction Process

1. The user either takes a picture of a leaf with the camera or chooses one from the gallery.
2. The picture is made smaller to 224 × 224 pixels.
3. Pixel normalization is used.
4. The embedded TFLite model gets the processed image.
5. The model gives each of the 33 classes a probability score.
6. The class with the highest chance is chosen as the prediction.
7. The name of the disease that is predicted and the confidence score are shown.
8. There are treatment suggestions to help the user.

The prediction runs on the device, which speeds up the response time and makes the system work well even in places where the internet connection is weak or unstable.

The AgriCare app is built on a layered mobile architecture that includes:

- Presentation Layer (Flutter Application)
- Inference Layer on the Device
- Cloud Support Layer (Firebase Services)

This modular structure makes sure that the system can grow and run smoothly.

A. Presentation Layer

Flutter is used to build the frontend so that it works on multiple platforms. There are two types of users in the system:

Module for Farmers:

- Take a picture of a leaf or upload one
- Look at the results of the prediction
- Get treatment recommendations
- Look at the history of predictions • Browse nearby agricultural stores Module for Shopkeepers:
- Keep track of and manage products
- Change the stock
- Keep an eye on sales data

The interface is made to be easy to use and understand for people who live in rural areas.

B. On-Device Inference Layer

The mobile app has the classification model built in as TFLite. All prediction tasks happen on the device itself.

Some benefits are:

- A quicker response to predictions
- Less reliance on being connected to the internet
- Better privacy (pictures aren't sent to servers outside of the network)
- Works well in areas with weak networks

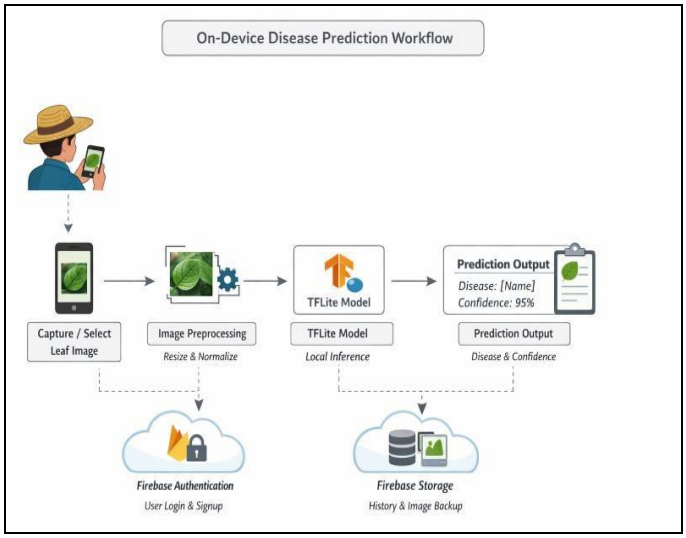

Figure 7: Prediction Workflow

C. Cloud Integration Layer

Firebase services are used to support the backend.

- Firebase Authentication for safe login and managing roles
- Firestore Database to keep track of past predictions
- Firebase Storage for keeping images that have been uploaded

You only need to be connected to the internet to log in and sync your data. Cloud-based processing does not affect how diseases are classified.

6. MOBILE ARCHITECTURE

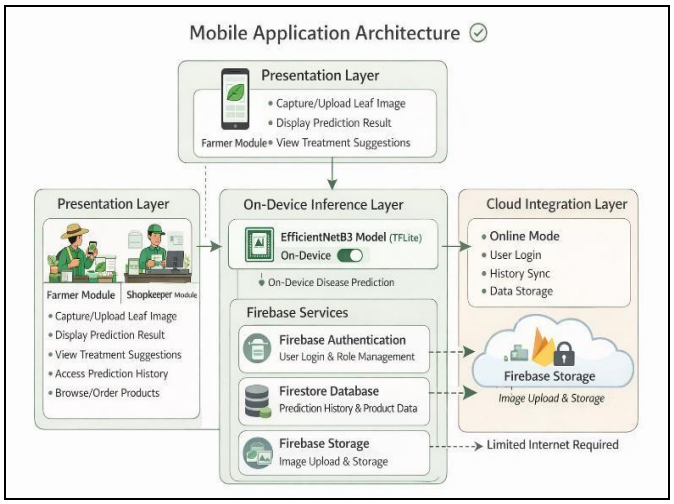

Figure 8: Mobile Architecture

IV. EXPERIMENTAL SETTINGS

This section talks about the experimental setup, the datasets used, the system environment, and the criteria used to judge how well the classification models worked.

1. Experimental Environment

The experimental work was done in a structured development environment that was good for training, validating, and deploying models. We used the TensorFlow and Keras libraries to build the models. These libraries are great for image classification tasks that use convolution.

Python was the main programming language because it has a lot of libraries for working with images and numbers. We used Jupyter Notebook for model development and testing. This program lets you run code interactively, see how well your training is going, and keep an eye on performance.

Training was done on a system that had NVIDIA GPU acceleration whenever possible. This cut down on training time and made the system work better. When GPU support wasn't available, training was done on the CPU. GPU acceleration made it possible for faster convergence and smoother optimization..

After the training phase was over, the final EfficientNetB3 model was exported and changed to TensorFlow Lite (.tflite) format. TensorFlow Lite is designed to work well on mobile devices and lets you run programs quickly while using less memory.

The Flutter framework was used to make the mobile app, which makes sure it works on all platforms and responds quickly to user input. The app had the converted TensorFlow Lite model built right in so that it could classify plant diseases on the smartphone.

Firebase services were added to the backend to handle:

- User authentication

It is important to point out that the classification process happens on the device itself. You only need an internet connection to verify your identity and sync your data. This setup makes sure that it works reliably even in places where the network isn't always available.

2. Evaluation Metrics

Standard multi-class classification metrics were used to see how well the models that were put into place worked. These

metrics give a clear picture of how well predictions work for all 33 disease types.

The evaluation was based on the parts of the confusion matrix:

- TP (True Positive)
- TN (True Negative)
- FP (False Positive)
- FN (False Negative)
- The primary metrics considered are:
- Accuracy
- Precision • Recall
- F1-Score
- Accuracy

Accuracy shows how many samples were correctly classified out of all the samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The results of the three models that were put into action are shown below.

Model	Training Accuracy	Validation Accuracy
Custom CNN	80.8	85.0
VGG16	89.0	92.0
EfficientNetB3	93.5	95.7

EfficientNetB3 had the best validation accuracy of 95.7%, which means it did a better job of classifying than the other models.

Precision

Precision tells us how well the model finds positive cases without making any mistakes.

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

Higher precision values mean that there are fewer false positives.

Table.2.Precision Comparison (%)

- Prediction history storage
- Image storage

Recall

Recall evaluates the model's ability to correctly detect all actual positive cases.

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

Higher recall indicates better detection of disease cases.

Table.3.Recall Comparison (%)

Model	Recall
Custom CNN	78.48%
VGG16	92.39%
EfficientNetB3	94.65%

F1-Score

The F1-Score provides a balanced measure by combining Precision and Recall. It is particularly useful for multi-class classification problems.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

A higher F1-score reflects balanced and consistent performance.

Table.4.F1-Score Comparison (%)

Model	F1-Score
Custom CNN	77.94%
VGG16	92.33%
EfficientNetB3	94.64%

Loss Analysis

Training loss and validation loss were monitored throughout the training process to evaluate convergence behavior and generalization capability.

EfficientNetB3 achieved the lowest validation loss of **0.14**, indicating better optimization and reduced prediction error.

Table.5.Training And Validation Loss

Model	Precision
Custom CNN	81.99%
VGG16	92.71%
EfficientNetB3	94.82%

Model	Training Loss	Validation Loss
Custom CNN	0.42	0.36
VGG16	0.2	0.21
EfficientNetB3	0.18	0.14

Overall Performance Ranking

Based on the evaluation metrics, the overall model comparison is summarized below.

Table.6. Model Performance Summary

Model	Accuracy Rank	Performance
Custom CNN	3	Moderate
VGG16	2	Good
EfficientNetB3	1	Best

This test shows that EfficientNetB3 has the best classification performance while still being suitable for mobile use. The experimental outcomes confirm the efficacy of the proposed AgriCare system for practical plant disease identification.

V. IMPLEMENTATION AND USER INTERFACE

This part talks about how the AgriCare mobile app works in real life and how its user interface is designed. The system was made to make it easy for people to find plant diseases and get help with farming. The Flutter framework was used to build the app so that it would work well and be compatible with other platforms.

The trained classification model was put into the mobile app in TensorFlow Lite format. This makes it possible to predict diseases directly on the user's device. You only need an internet connection to log in, save your prediction history, and use the marketplace features.

The app is made up of modules that handle things like authentication, disease prediction, history management, and store management.

A. Choosing a Role and Logging In

Users can sign up or log in to the app with their email address or Google account. When they log in, users can choose to be either a farmer or a shopkeeper. This structure based on roles lets the system offer different features based on the type of user.

The authentication system makes sure that users can safely access and manage their data.

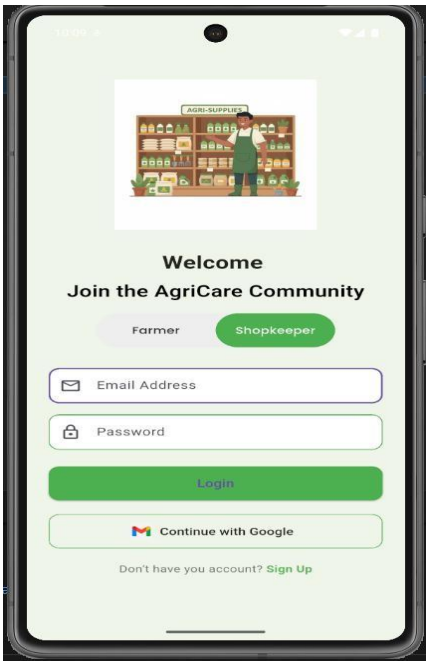

Figure 9: Login and Role Selection Interface

B. Uploading and Analyzing Plant Images

After logging in, farmers are taken to the home dashboard. You can either take a picture of a plant leaf with the device's camera or choose one from the gallery on the dashboard. After you choose an image, the app processes it and sends it to the built-in classification model.

The user interface is meant to be simple and clear so that people from rural areas can use the system without any problems.

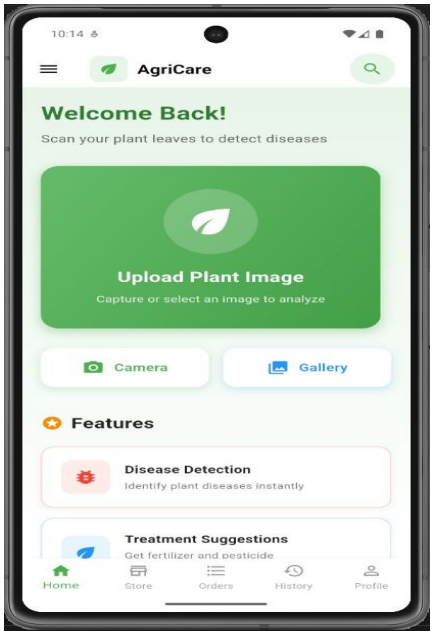

Figure 10: Home Dashboard Interface

C. Results of Disease Prediction

After the app processes the image, it shows the predicted disease name and some related suggestions. The result screen shows:

- Disease name
- Fertilizer recommendations
- Pesticide suggestions
- Organic treatment suggestions

This structured presentation helps people understand how the plant is doing and what they need to do to fix it.

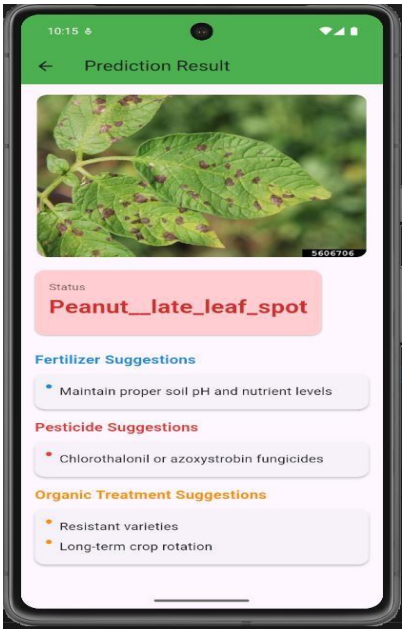

Figure 11: Disease Prediction Result Screen

D. Managing the History of Predictions

The app keeps track of the images that have been analyzed before and the results that went with them. With this feature, users can keep an eye on past predictions and keep an eye on plant health problems that keep coming back. Cloud database services keep prediction records safe and can be accessed whenever there is an internet connection.

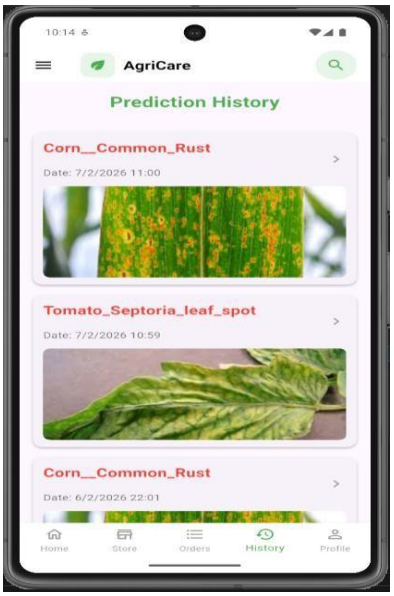

Figure 12: Prediction History Interface

E. Managing the Marketplace and Orders

The app not only helps people find out what diseases they have, but it also has a marketplace module where farmers can look at nearby agricultural stores and order fertilizers or pesticides. Through their own dashboard, shopkeepers can keep track of products, update their inventory, and keep track of orders that are coming in.

This built-in feature makes the system more useful by linking diagnosis results to product availability.

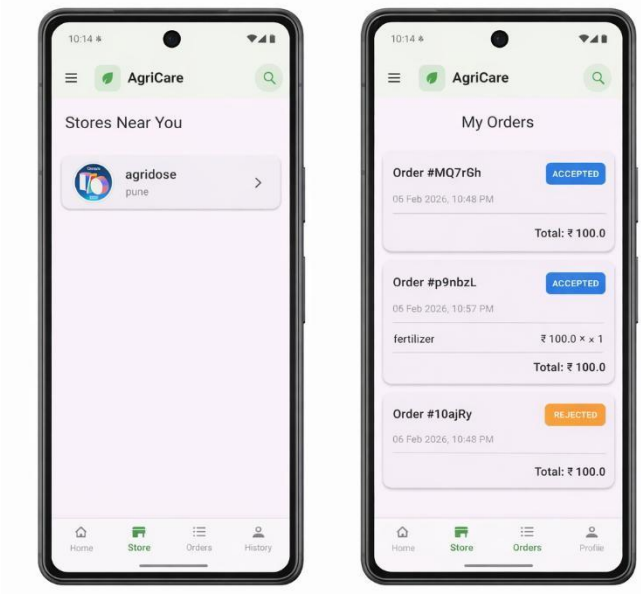

Figure 13: Store and Order Management Interface

VI. RESULTS AND DISCUSSION

This part gives a quick overview of how the three models that were put into use—Custom CNN, VGG16, and EfficientNetB3—compared in terms of performance. The Custom CNN got an 85.0% accuracy rate on the validation set, but its precision and recall values were only average. It was able to correctly classify some disease categories, but not as well as deeper architectures. This model was used as a starting point for the evaluation.

The VGG16 model got better, with a validation accuracy of 92.0% and balanced precision and recall above 92%. The deeper structure made it easier to extract features and classify plants with different diseases more consistently. EfficientNetB3 was the best model overall, with a validation accuracy of 95.7%, a precision of 94.82%, a recall of 94.65%, and an F1-score of 94.64%. The fact that precision and recall are so close to each other shows that the classification results are stable and reliable. The model also worked well in real time after it was added to the mobile app.

Overall, the comparison shows that EfficientNetB3 strikes the best balance between accuracy and ease of use, making it the best model to use in the AgriCare system.

A. Analyzing the Confusion Matrix

To better understand how well the implemented models worked at classifying, confusion matrices were made for Custom CNN, VGG16, and EfficientNetB3. These matrices give a clear picture of how the predicted classes match up with the real disease categories.

The diagonal elements of the Custom CNN model's confusion matrix show that many samples are correctly classified. But there are also some off-diagonal values, which show that some disease categories were incorrectly classified. This happens mostly with diseases that have similar patterns on the leaves' surfaces. The Custom CNN architecture isn't very deep, so it can't capture complex visual features as well as deeper models. This makes it less accurate when it comes to classification.

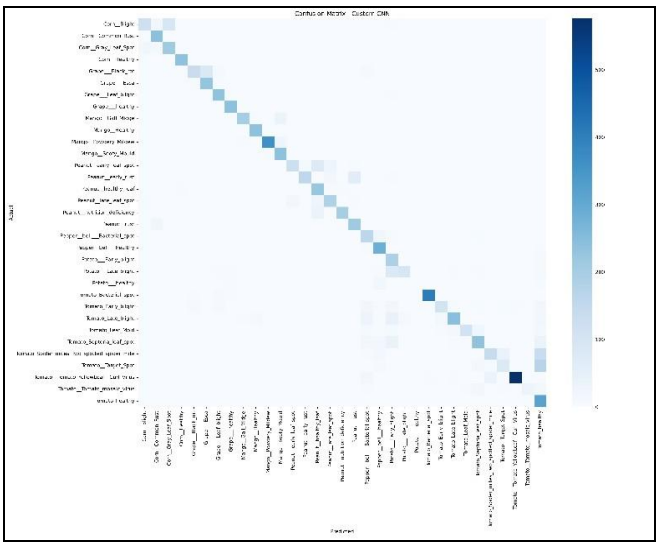

Figure 14: Confusion Matrix of Custom CNN Model

The confusion matrix from the VGG16 model shows that it does a better job of classifying than the Custom CNN model. Most of the predictions are on the diagonal of the matrix, which means that many disease categories are correctly classified.

VGG16's deeper architecture lets the model learn more specific features of images, which cuts down on misclassification. However, there are still a few mistakes among disease classes that look alike.

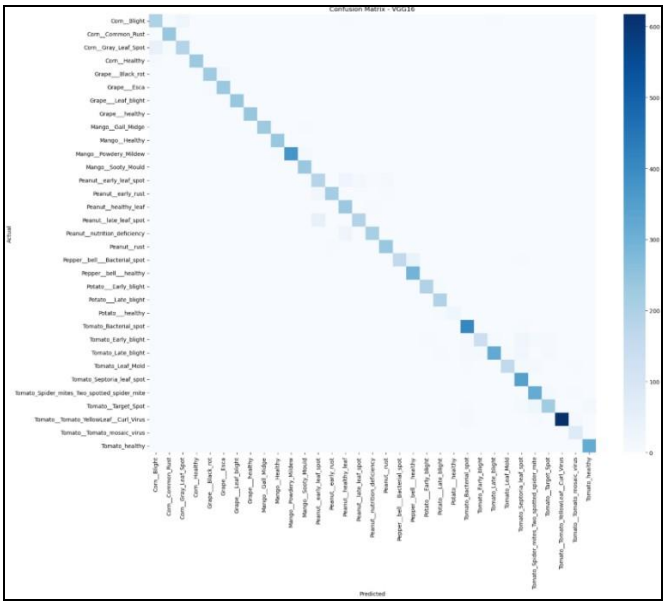

Figure 15: Confusion Matrix of VGG16 Model

The confusion matrix for EfficientNetB3 shows that it is the best at classifying data among the models that were tested. Most of the values are along the diagonal, which means that most plant disease classes are predicted correctly. There are only a few mistakes in the off-diagonal positions.

This result shows that EfficientNetB3 can get better visual features from the leaf images, which makes predictions more accurate and stable across different disease categories.

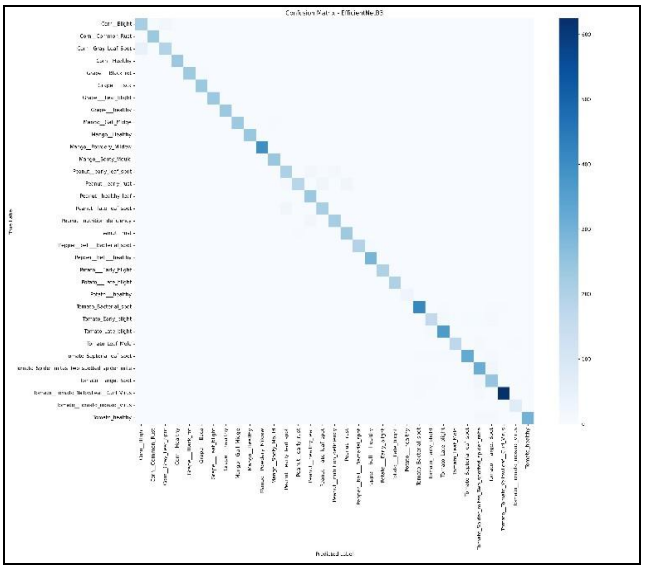

Figure 16: Confusion Matrix of EfficientNetB3 Model

EfficientNetB3 was the best at classifying the architectures that were tested. It had a stronger diagonal dominance in the confusion matrix and made fewer mistakes than Custom CNN and VGG16.

VI. CONCLUSION

This paper introduced AgriCare, a mobile app that helps users find diseases on plant leaves by using image classification techniques. We tried out and tested three different models: a Custom CNN, VGG16, and
EfficientNetB3. The PlantVillage dataset, which has 33 categories of healthy and diseased plants, was used to train and test the models.

The experimental comparison demonstrated that EfficientNetB3 surpassed the other models in validation accuracy, precision, recall, and overall stability. The model had a validation accuracy of 95.7% and high precision and recall values, showing that it could reliably classify multiple disease classes.

We changed the EfficientNetB3 model into TensorFlow Lite format after training it, and it worked well with a Flutterbased mobile app. The prediction process takes place right on the user's device, which speeds up the response time and makes it less dependent on a constant internet connection.
Using cloud services only for storing and verifying data.

The system that was made makes it easy and useful to use smartphones to find out what kind of plant disease it is. The AgriCare app can help farmers manage their crops better and improve farming practices by making it easier to find and treat problems quickly.

REFERENCES

[1] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, "Real-Time Detection of Apple Leaf Diseases Using Deep Learning Based on Improved Convolutional Neural Networks," IEEE Access, vol. 7, pp. 59069–59080, 2019.

[2] R. Zhou, S. Kaneko, F. Tanaka, M. Kayamori, and M. Shimizu, "Disease Detection of Cercospora Leaf Spot in Sugar Beet by Robust Template Matching," Computers and Electronics in Agriculture, vol. 108, pp. 58–70, 2014.

[3] J. G. A. Barbedo and C. V. Godoy, "Automatic Classification of Soybean Diseases Based on Digital Images of Leaf Symptoms," Proc. SBIAGRO, pp. 1–6, 2015.

[4] J. G. A. Barbedo, "A Review on the Main Challenges in Automatic Plant Disease Identification Based on Visible Range Images," Biosystems Engineering, vol. 144, pp. 52–60, 2016.

[5] Z. B. Husin et al., "Feasibility Study on Plant Chili Disease Detection Using Image Processing Techniques," Int. Conf. Intelligent Systems Modelling and Simulation, pp. 291–296, 2012.

[6] D. A. Bashish, M. Braik, and S. B. Ahmad, "A Framework for Detection and Classification of Plant Leaf and Stem Diseases," Int. Conf. Signal and Image Processing, pp. 113–118, 2010.

[7] J. D. Punajari, R. Yakkundimath, and A. S. Byadgi, "Image Processing Based Detection of Fungal Diseases in Plants," Procedia Computer Science, vol. 46, pp. 1802–1808, 2015.

[8] H. Al-Hiary et al., "Fast and Accurate Detection and Classification of Plant Diseases," Int. Journal of Computer Applications, vol. 17, no. 1, pp. 31– 38, 2011.

[9] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," ICLR, 2015.

[10] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," ICML, 2019.