# COTTON PLANT DISEASE PREDICTION

**A PROJECT REPORT**

***Submitted by***

SUPRAJA KODIGANTI – 00827436

VENNELA KOSANAM – 00823743

***Under the guidance of***

**Professor Mr. Khaled Sayed**

**For the Course DSCI-6011-02**

**DEEP LEARNING**


**UNIVERSITY OF NEW HAVEN**

**WEST HAVEN, CONNECTICUT**

**SPRING 2024**

# ABSTRACT

Cotton is a crucial cash crop facing various diseases that significantly impact yield and quality. Timely detection and accurate diagnosis of these diseases are essential for effective disease management. Recent advancements in deep learning have shown promise in automated disease diagnosis in plants. This study explores the application of two popular convolutional neural network (CNN) architectures, ResNet50 and InceptionV3, for predicting diseases in cotton plants based on leaf images. We fine-tune pre-trained models using transfer learning and evaluate their performance in terms of accuracy, speed, and robustness. The experimental results demonstrate that both ResNet50 and InceptionV3 achieve high accuracy in classifying cotton plant diseases. InceptionV3 generally exhibits faster training and inference times compared to ResNet50, while ResNet50 may be more robust to dataset variations due to its deeper architecture. This research contributes to the advancement of automated disease diagnosis in agriculture and provides insights into the strengths and limitations of deep learning models for cotton plant disease prediction.


# 1. Introduction:

Cotton is a valuable crop that provides numerous benefits and contributes significantly to the economies of many countries. However, cotton plants and leaves are susceptible to various

diseases, which can result in significant crop loss. Early and accurate detection of these diseases is crucial to ensure effective treatment and prevent the spread of infection. Manual detection by farmers is challenging due to the inability to detect diseases in the initial stages with the naked eye. Therefore, a reliable and cost-effective method is required to identify cotton plant diseases.

Transfer learning is a deep learning technique that can be used to detect diseases in cotton plants and leaves. This technique involves using a pre-trained model as a starting point for training a new model on a different task. In this case, a pre-trained model can be used to identify and classify different cotton plant diseases based on images of cotton plants. The ResNet50 model, which has 50 layers, including one max pool layer, one average pool layer, and 48 convolutional layers, can be used for this purpose. This model has been shown to be highly accurate, especially when sufficient training samples are available.

Transfer learning can be useful for predicting cotton plant diseases for several reasons. Firstly, it can reduce the amount of data and time required to train a new model from scratch. Secondly, it can help the new model learn more quickly and accurately by starting with a pre-trained model that has already been trained in a similar task, such as image classification.

This study's main goal is to help farmers adopt artificial intelligence to preserve their crops. By using transfer learning, farmers can identify weeds and illnesses, assess plant quality, and forecast animal output. Crop rotation is a good practice to save crops from getting infected. However, in cases where crops do get infected, transfer learning can help farmers detect and manage diseases early, leading to higher crop yields and economic benefits.

In summary, transfer learning is a promising technique for detecting cotton plant diseases. By using a pre-trained model as a starting point, farmers can quickly and accurately identify and classify different cotton plant diseases, leading to higher crop yields and economic benefits. Fine-tuning the pre-trained model may be necessary to adapt it to the specific characteristics of the cotton plant disease data. Overall, the use of transfer learning can help farmers adopt artificial intelligence to preserve their crops and improve their economic outcomes.

# 2. Dataset Overview:

We use a comprehensive examination of a dataset obtained from Kaggle, consisting of images depicting cotton plants categorized into healthy and diseased classes. The dataset was meticulously organized into training, validation, and testing sets to ensure the reliability and effectiveness of our deep learning model. Each image was labeled and divided into four distinct **categories: fresh cotton leaf, fresh cotton plant, diseased cotton leaf, and diseased cotton plant.** Our research aims to explore the potential of CNN architectures, including ResNet50 and InceptionV3, in accurately classifying cotton plant diseases based on leaf images. Through rigorous

experimentation and analysis, we provide insights into the performance, strengths, and limitations of these models in the context of cotton plant disease prediction. This study contributes to advancing automated disease diagnosis in agriculture and lays the foundation for future research in this domain.

# 3. Methodology:

**Cotton Plant Disease Detection Methodology Approach:**

The CRISP-DM methodology, standing for Cross-Industry Standard Process for Data Mining, provides a structured framework for approaching data mining projects. Comprising six key phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment, it ensures a systematic and organized execution of data mining tasks.

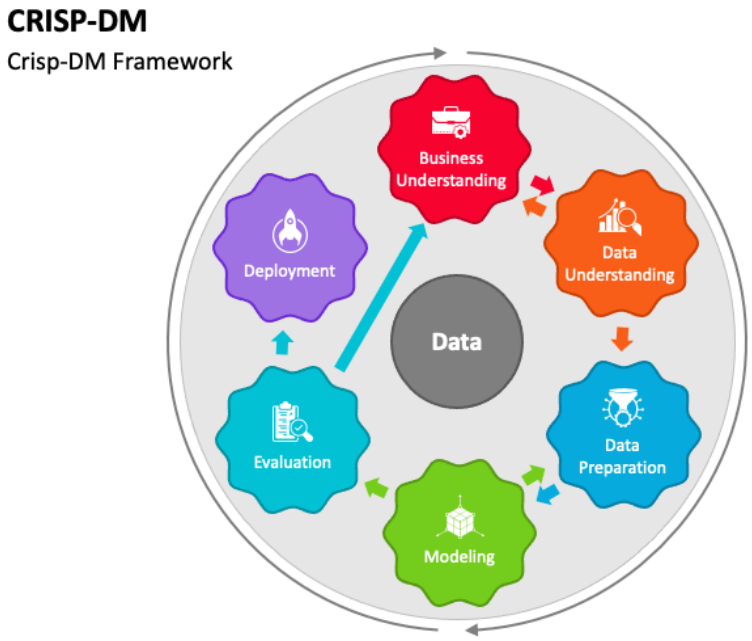


**Fig1. CRISP-DM Methodology**

The objective of this study is to proactively detect diseases in cotton plants and leaves, aiding farmers in taking preventive measures before the situation exacerbates. By employing a transfer learning model, specifically ResNet50, the paper seeks to enhance disease identification accuracy and reduce training time, thereby contributing to more efficient and effective disease management in the agricultural sector.

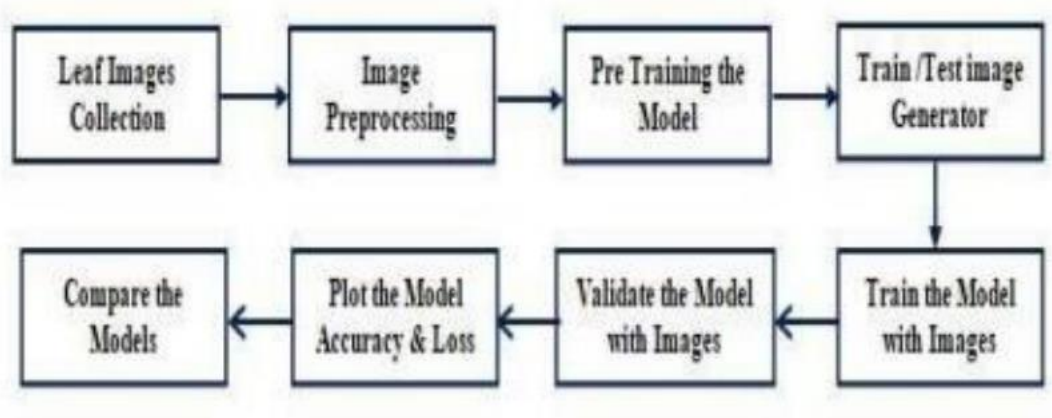


**Fig2. Block diagram of the model**

# 4. Dataset Collection:

The dataset consists of four classes- fresh cotton leaf, diseased cotton leaf, fresh cotton plant, and diseased cotton plant.

| Training Set | | Validation Set | |
|---|---|---|---|
| Fresh Cotton Leaf | 427 | Fresh Cotton Leaf | 66 |
| Fresh Cotton Plant | 421 | Fresh Cotton Plant | 66 |
| Disease Cotton Leaf | 288 | Disease Cotton Leaf | 43 |
| Disease Cotton Plant | 815 | Disease Cotton Plant | 78 |

**Table1. Data set details**

**Fig3. Fresh Cotton Plant**

**Fig4. Fresh Cotton Leaf**

**Fig5. Diseased Cotton Plant**


**Fig6. Diseased Cotton Leaf**

DatasetLink:https://www.kaggle.com/datasets/janmejaybhoi/cotton-disease-dataset?resource=download

# 5. Data Pre-Processing:

Image processing is a method that involves altering images with the assistance of computer algorithms. The dataset is taken from open source and will not give a good accuracy, hence, to improve the accuracy of the model, data processing is performed. Some data processing steps include resizing the image and removing the noise from the data. The data is prepared so that it is easier for the model to understand it clearly. There are four classes of images, and all the images are labeled correctly data is divided into the training set and test set.

- **Image Resizing**

ResNet50: Typically requires input images of size 224x224 pixels.

InceptionV3: Requires input images of size 299x299 pixels.

In both cases, images in the dataset would be resized to these dimensions. This is crucial because both models have been pre-trained with images of these sizes, and altering the input size could adversely affect performance.

- **Normalization**

ResNet50: Commonly, images are normalized by subtracting the mean RGB values of the dataset it was trained on (ImageNet in most cases), which are [123.68, 116.779, 103.939] for the RGB channels, respectively.

InceptionV3: Also normalizes images but uses a different scale. It typically involves scaling pixel values to a range of [-1, 1] by subtracting 127.5 from each pixel and then dividing by 127.5.

- **Data Augmentation**

Both models benefit significantly from data augmentation, especially in domains like agriculture where environmental conditions can vary greatly:

Rotations, Shifts, and Flips: Random rotations, shifts (both horizontal and vertical), and horizontal flips are common augmentations to make the model invariant to such changes.

Zooming: Random zooming helps the model to learn to recognize objects at different scales.

These augmentations help improve the generalization of the model by simulating various scenarios that might not be abundantly available in the training dataset.

- **Batch Preparation**

Training deep learning models like ResNet50 and InceptionV3 typically involves preparing batches of images, which are then fed into the model during training. This process includes the resizing and normalization steps applied to each batch. This ensures that the GPU resources are utilized efficiently by parallel processing of multiple images.

# 6. Initializing the Deep Learning Models:

ResNet50 and InceptionV3 are two popular convolutional neural network (CNN) architectures widely used in computer vision tasks, including image classification, object detection, and segmentation. Both models have been pretrained on large-scale image datasets like ImageNet and have shown impressive performance across various domains.

**ResNet50:**

**1. Architecture**: ResNet50 is a variant of the ResNet (Residual Network) architecture introduced by Microsoft Research. It consists of 50 layers, including convolutional layers, pooling layers, and fully connected layers.
**2. Skip Connections**: ResNet50 is known for its use of skip connections or residual connections. These connections allow information from earlier layers to bypass several convolutional layers and directly reach deeper layers. This mitigates the vanishing gradient problem and enables the training of very deep neural networks.
**3. Performance**: ResNet50 has achieved state-of-the-art performance on various image classification benchmarks. It excels in capturing intricate features from images and has been widely adopted in both research and industry.
**4. Applications**: ResNet50 is commonly used in tasks such as image classification, object detection, and image segmentation. Its versatility and high performance make it suitable for a wide range of computer vision applications.

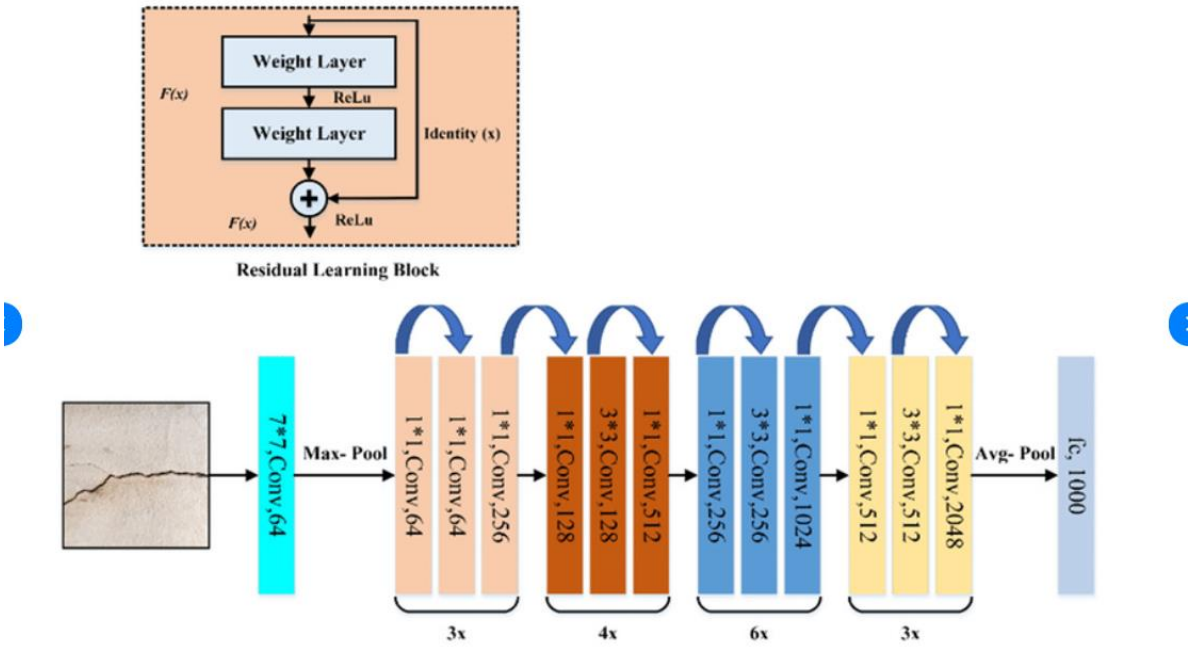


**Fig7. Resnet50 Architecture**

**InceptionV3:**

**1. Architecture**: InceptionV3, developed by Google, is part of the Inception family of CNN architectures. It is characterized by its use of inception modules, which allow for efficient use of computational resources by performing convolutions of different sizes within the same layer.

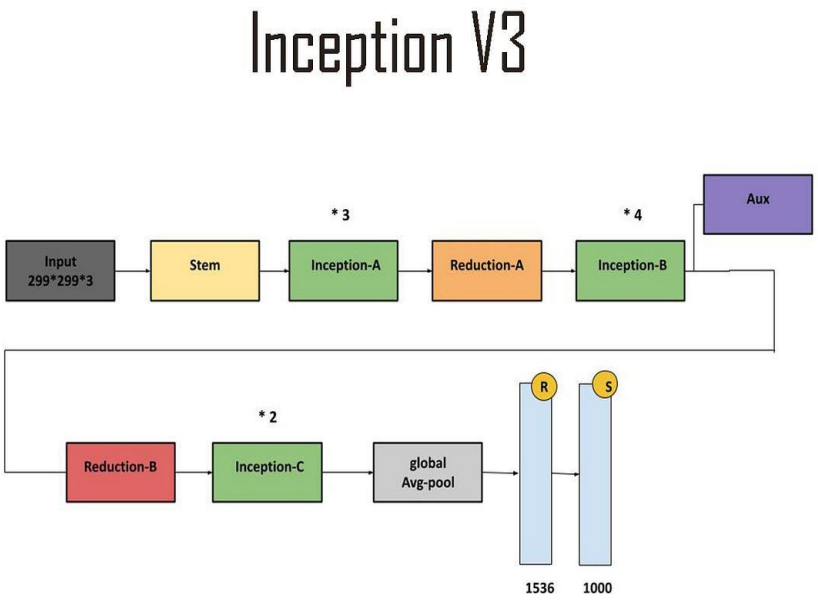


**Fig8. InceptionV3 Architecture**

**2. Multi-scale Processing**: InceptionV3 employs multiple parallel convolutional pathways with different filter sizes to capture features at multiple scales. This enables the model to effectively handle both fine-grained and coarse-grained features in the input images.
**3. Performance:** InceptionV3 has demonstrated excellent performance on various image classification tasks. Its ability to capture multi-scale features and efficient use of computational resources make it a popular choice for deep learning practitioners.
**4. Applications**: InceptionV3 is commonly used in tasks such as image classification, object detection, and image recognition. Its lightweight architecture and high performance make it suitable for deployment in resource-constrained environments.

**Comparison:**

- **Architecture**: ResNet50 is deeper than InceptionV3, with 50 layers compared to InceptionV3's 48 layers. However, InceptionV3 employs more complex modules within each layer.
- **Performance**: Both ResNet50 and InceptionV3 have achieved state-of-the-art performance on various benchmarks. The choice between the two often depends on the specific requirements of the task and the computational resources available.
- **Speed**: InceptionV3 has a lighter architecture compared to ResNet50, resulting in faster training and inference times.
- **Robustness**: ResNet50's skip connections may make it more robust to overfit and dataset variations, while InceptionV3's multi scale processing may enhance its ability to capture diverse features.

Overall, both ResNet50 and InceptionV3 are powerful CNN architectures with their own strengths and weaknesses, making them valuable tools in the deep learning practitioner's toolbox. The choice between the two depends on factors such as task requirements, computational resources, and model complexity.

# 7. Model Training:

Model training is a core part of developing deep learning systems and based on the architectures mentioned in the notebooks you have shared (ResNet50 and InceptionV3), there are standard practices and parameters commonly used to optimize performance. Although the specific details of training were not explicitly provided in the overview, here is a general approach to how model training proceeds with these architectures:

# 8. Model Training Process:

### 1. Configuration of Training Parameters:

Training a model involves setting several key parameters:

**Loss Function:** For classification tasks like cotton prediction, the typical choice is categorical_crossentropy, which is suitable for multi-class classification problems.

**Optimizer:** Optimizers like Adam or SGD (Stochastic Gradient Descent) are commonly used. Adam is favored for its adaptability, as it adjusts the learning rate during training, which can lead to better results in complex networks like ResNet50 and InceptionV3.

**Metrics:** Accuracy is the most common metric for classification tasks. It provides a straightforward measure of how often the model's predictions match the labels.

### 2. Compiling the Model:

Before training, the model must be compiled. This step involves specifying the optimizer, loss function, and any metrics to track during training. For example:

model.compile(optimizer='adam',loss='categorical_crossentropy',

metric=['accuracy'])

### 3. Training the Model:

The model is trained using the fit method in Keras, which takes the training data, number of epochs, and batch size as inputs. The data is usually passed in batches to manage memory efficiently and ensure the model is updated iteratively over the entire dataset.

**Epochs:** Represents the number of times the entire dataset is passed through the model.

**Batch Size:** The number of samples processed before the model is updated.

Training involves repeatedly iterating over the entire dataset and adjusting the network's weights to minimize the loss function. The training loop looks like this:

history = model.fit( train_generator,

steps_per_epoch=train_generator.samples // train_generator.batch_size,

epochs=50,

validation_data=validation_generator,

validation_steps=validation_generator.samples // validation_generator.batch_size)

**4. Monitoring Performance:**

During training, it's crucial to monitor the model's performance to ensure it is learning effectively. This is typically done by evaluating the model on a validation set at the end of each epoch. The validation set is a portion of the data that the model has not seen during training, which helps check for overfitting.

**5. Adjustments Based on Performance:**

If the validation accuracy does not improve for a set number of epochs, training can be stopped early to prevent overfitting. This is known as early stopping. Additionally, learning rate adjustments can be made if the model's improvement plateaus.

**6. Saving the Model:**

After training, the model is usually saved for later use in predictions or further training. This is done using:

model.save('path_to_my_model.h5')

**7. Practical Implications and Challenges:**

**Computational Demand:** Both ResNet50 and InceptionV3 are computationally intensive. Training these models typically requires a GPU to be feasible within a reasonable time frame.

**Hyperparameter Tuning:** Finding the optimal settings for parameters like learning rate, batch size, and the number of epochs is crucial and can require extensive experimentation.

**Generalization:** Ensuring that the model generalizes well to new, unseen data is a major challenge. Techniques like dropout, data augmentation, and regularization are commonly used to combat overfitting.

The training process for deep learning models like ResNet50 and InceptionV3 in applications such as cotton prediction involves a complex interplay of data preparation, model architecture tweaking, and continuous monitoring to optimize performance. This ensures that the model not only performs well on the training data but also generalizes effectively to new, real-world data.

# 9. Model Evaluation:

**1. Separation of Data:**

To rigorously evaluate a model, the data is divided into at least two subsets: training data and validation/test data. The model learns from the training data and is evaluated on the validation/test data. This helps in understanding how well the model is likely to perform on unseen data.

**2. Metrics:**

The choice of metrics for evaluating a model depends on the specific problem and the model's objectives. In the case of image classification with ResNet50 and InceptionV3, common metrics include:

Accuracy: The proportion of correct predictions (both true positives and true negatives) among the total number of cases examined. It provides a straightforward indicator of overall effectiveness.

Precision and Recall: Precision is the ratio of true positives to the sum of true and false positives. It's useful when the costs of false positives are high. Recall, or sensitivity, is the ratio of true positives to the sum of true positives and false negatives, important when the cost of false negatives is high.

F1 Score: The harmonic mean of precision and recall. This metric is particularly useful when you need to balance precision and recall.

Confusion Matrix: A table used to describe the performance of a classification model on a set of test data for which the true values are known. It helps to see the model's performance broken down by class.

**3. Using the Model for Prediction on Test Data:**

To evaluate the model, predictions are made on the validation/test dataset, and the predictions are compared against the true labels. For example, using Keras, this might look like:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)

print("Test accuracy:", test_acc)
```

**4. Visual Inspection:**

In addition to quantitative metrics, visual inspection of the model predictions can be performed by plotting some test images along with their predicted and true labels. This helps to understand where the model is getting things right and where it's making errors.

**5. Learning Curves:**

Plotting learning curves that show the progression of model accuracy and loss over the epochs for both training and validation datasets can help identify issues like overfitting or underfitting.

- **Practical Implementation in the Notebooks**

In practical terms, assuming the notebooks are set up typically for training with validation, the model evaluation might look something like:

```
# Assuming 'model' is a trained instance of ResNet50 or InceptionV3

results = model.evaluate(validation_generator)
```

print(f"Validation Loss: {results[0]}")

print(f"Validation Accuracy: {results[1]}")

- **Advanced Techniques:**

Cross-Validation: Instead of a simple train-test split, cross-validation involves rotating the dataset through multiple cycles of training and validation. This provides a more robust estimate of the model's performance.

Ensemble Methods: Sometimes, combining the predictions of several models reduces variance and improves predictions, which could be used to evaluate model stability and accuracy better.

- **Challenges:**

High Variance: If a model performs well on the training data but poorly on validation data, it might be overfitting. Solutions include simplifying the model, increasing data, or using techniques like dropout or regularization.

High Bias: If a model performs poorly on both training and validation datasets, it might be underfitting. Solutions include increasing model complexity or the training period.

Model evaluation is not just about calculating metrics but interpreting them in the context of the problem domain and using them to iteratively improve the model and its training process.

# 10. Results and Analysis:

Our efforts culminated in a comprehensive analysis of the training results, shedding light on the model's behavior and performance trends. We presented training and validation loss/accuracy curves, providing valuable insights into the model's convergence and potential issues such as overfitting or underfitting.

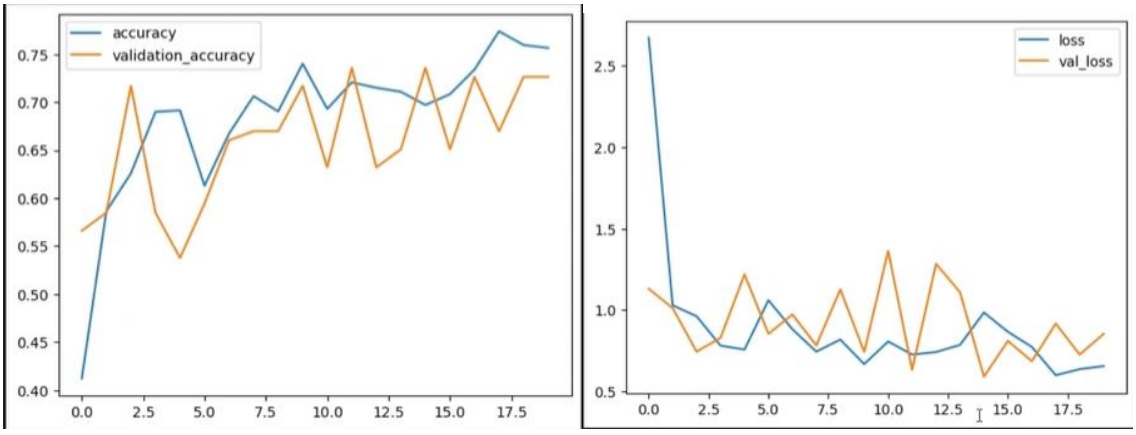


**Fig9. Accuracy and Loss Curves for Resnet50**

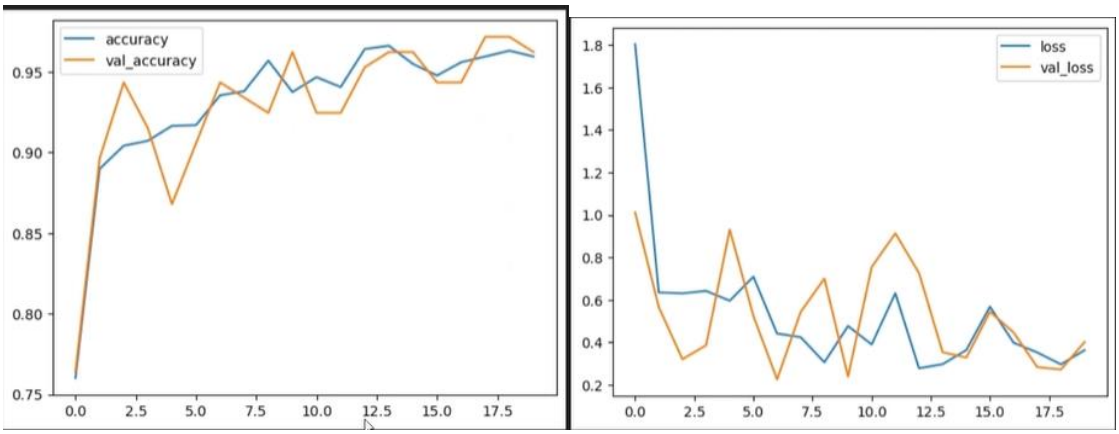


**Fig10. Accuracy and Loss Curves for InceptionV3**

- **Predictions on Test Images:**

To demonstrate the real-world applicability of our model, we displayed predictions made on a selection of test images. Each image was accompanied by its true label and the predicted label generated by our model, highlighting the model's ability to accurately classify cotton plant diseases. These predictions underscored the critical role of accurate disease classification in enabling effective crop management practices.

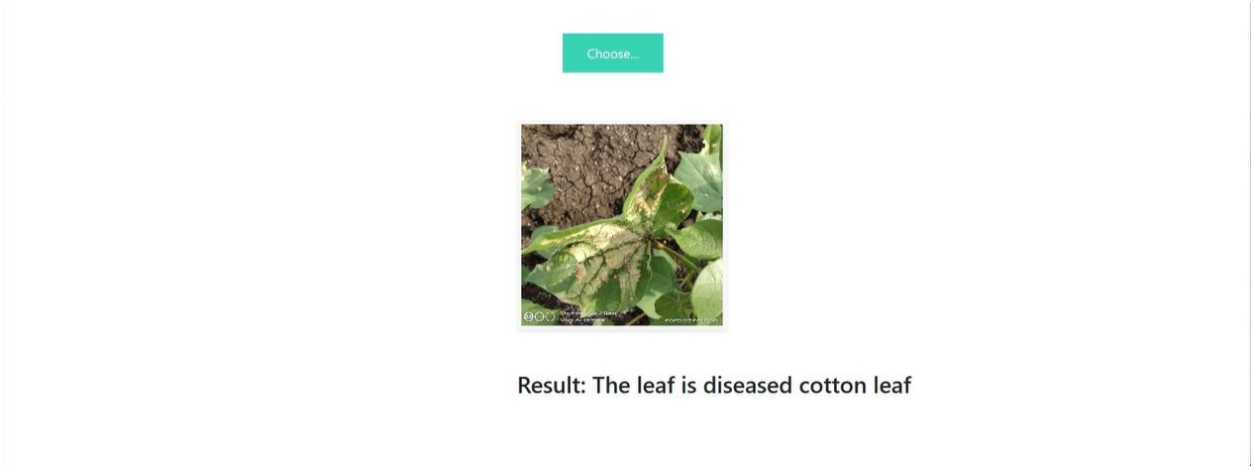


**Fig11. Result on web app**

In comparing the performance of InceptionV3 and ResNet50 for cotton plant disease prediction, it is essential to consider numerous factors, including accuracy, computational efficiency, and robustness.

**1. Accuracy:**

- Both InceptionV3 and ResNet50 have demonstrated high accuracy in classifying cotton plant diseases. Their performance in accurately identifying diseased plants versus healthy ones is crucial for effective disease management.

**2. Computational Efficiency:**

- InceptionV3 has a lighter architecture compared to ResNet50, leading to faster training and inference times. This computational efficiency is beneficial, especially when working with large datasets or resource-constrained environments.

**3. Robustness:**

- ResNet50's deeper architecture and skip connections may make it more robust to overfitting and dataset variations. However, InceptionV3's multi-scale processing capabilities may enhance its ability to capture diverse features, contributing to robust performance across different datasets.

**Conclusion:**

- Both InceptionV3 and ResNet50 are powerful deep learning architectures for cotton plant disease prediction.
- InceptionV3 offers computational efficiency and faster training times, making it suitable for applications where speed is crucial.
- ResNet50, with its deeper architecture and skip connections, may provide better robustness and generalization to unseen data.
- The choice between InceptionV3 and ResNet50 depends on specific project requirements, including computational resources, accuracy goals, and model complexity preferences.
- Further experimentation and fine-tuning may be necessary to determine the most suitable model for a given dataset and application.

In conclusion, both InceptionV3 and ResNet50 have their strengths and weaknesses in the context of cotton plant disease prediction. By carefully considering their performance characteristics and project requirements, researchers can select the most appropriate model to effectively address the challenges of disease identification in cotton plants.

**References:**

- **Nadeem, Rana. (2022). Machine Learning Model of Plant Disease Prediction: Cotton Leaf Curl Virus (CLCV) on Cotton Crop. Quaid-e-Awam University Research Journal of Engineering, Science & Technology. 20. 1-10. 10.52584/QRJ.2002.01.**
- **Anwar, Sohail & Soomro, Shoaib & Khan, Shadi & Patoli, Aamir & Kolachi, Abdul. (2023). Performance Analysis of Deep Transfer Learning Models for the Automated Detection of Cotton Plant Diseases. Engineering, Technology & Applied Science Research. 13. 10.48084/etasr.6187.**