



**AMITY UNIVERSITY**  
— **UTTAR PRADESH** —

**Project Title:**

Detection and Prediction of Disease in Potato Leaf

**Submitted By:**

Arnav Kumar Singh (A2305222293)

**Under The Guidance Of:**

Dr. Garima Aggarwal

**AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY**  
**AMITY UNIVERSITY NOIDA UTTAR PRADESH**

## **DECLARATION**

I, Dinaaksh Aulakh, student of BTech (2022-2026), hereby declare that the document titled “Detection and Prediction of Disease in Potato Leaf ” that is submitted to Amity Institute School of Engineering and Technology, Amity University Noida, in partial fulfillment requirement for the award of degree of Bachelors in Technology, has not previously been the basis for the award of any diploma, degree or different comparable identify or reputation. In addition, I declare that the file has been solely written and no part of the record is copied from any resource without being duly stated. If anything is determined to be plagiarized above, I’m accountable and university rules and regulations may be used against me.

## **CERTIFICATE**

On the basis of declaration submitted by Dinaaksh Aulakh, student of B.Tech. Computer Science and Engineering, I hereby certify that the In-house project titled “Detection and Prediction of Disease in Potato Leaf”, submitted to Department of Computer Science & Engineering, Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science & Engineering, is an original contribution with existing knowledge and faithful record of work carried out by him under my guidance and supervision.

Dr. Garima Aggarwal  
Associate Professor  
ASET, AUUP  
NOIDA

## **ACKNOWLEDGEMENT**

Presentation, inspiration and motivation are always important to the success of any venture. I would like to thank Prof (Dr) Sanjeev Thakur, Head of Department-CSE, and Amity University for giving me the opportunity to undertake this project. I express my gratitude and appreciation for the advice and constant support of Dr. Garima Aggarwal, who provided me with valuable guidance throughout the course, created and helped me in molding the structure for this presentation as it is shown.

Arnav Kumar Singh(A2305222293)

Date :01/07/2024

## **Abstract**

The project aims to develop a robust and accurate system to predict potato leaf diseases using deep learning and transfer learning algorithms. It helps in tackling the critical issue of early and accurate detection of diseases, which is a major threat to agricultural productivity and can lead to significant economic loss. Therefore, this topic holds high significance towards growth of agricultural economy and prevention can alleviate the ails of struggling farmers.

The application utilizes TensorFlow, a well renowned deep learning library, to train and predict using the convolutional neural network (CNN) model on the dataset of potato leaves. The model is fine-tuned using transfer learning to make it more accurate and adaptable for real time usage. Transfer learning allows for reuse in other use cases and pre-training increases the accuracy and adaptability of the model.

Successful implementation of such a model can lead to increased yield and better growth practices thus reducing the need to rely on chemical treatment for the crop. By providing efficient and user-friendly tools for detecting diseases, the project aims to revolutionize the agriculture sector not only to improve yields but reduce the economic losses as well.

**Keywords: TensorFlow, Deep Learning, Convolution Neural Networks.**

# Table of Contents

<b>1. Declaration.....</b>	<b>I</b>
<b>2. Certificate.....</b>	<b>II</b>
<b>3. Acknowledgement .....</b>	<b>III</b>
<b>4. Abstract.....</b>	<b>IV</b>
<b>5. Introduction.....</b>	<b>1</b>
1.1 Problem Statemen.....	1
1.2 Objective.....	1
1.3 Significance.....	2
<b>6. Literature Review.....</b>	<b>3</b>
<b>7. Materials and Methodology.....</b>	<b>4</b>
3.1 Prerequisites.....	4
3.2 Data Collection.....	4
3.3 Preprocessing.....	4
3.4 Model Architecture.....	5
3.5 Model Training.....	6
3.6 Model Evaluation.....	10
3.7 Backend Server.....	11
3.8 Frontend Application.....	12
<b>8. Conclusion .....</b>	<b>13</b>
<b>9. References.....</b>	<b>14</b>

## List of Figures

Fig.3.3.1	Preprocessing of training dataset	1
Fig.3.4.1	Proposed model architecture	2
Fig.3.4.2	Model Summary	3
Fig.3.5	Training process of each epoch	4
Fig.3.6.1	Accuracy and loss graphs	7
Fig.3.6.2	Test dataset for the model	7
Fig.3.6.3	Confusion matrix of the model	7
Fig.3.7.1	Importing model and initializing classes	9
Fig.3.7.2	Postman POST function	10
Fig.3.8.1	Axios library function	10
Fig.3.8.2	Website hosted on cloud	11
Fig.3.8.3	Drag and drop feature on React website	11
Fig.3.8.4	Visual representation of prediction	11

# 1. Introduction

Since potatoes are an essential crop for the consumption, leaf diseases pose a threat to their quality and yield. Conventional detection techniques depend on skilled individuals visually inspecting samples, which is an arduous task and prone to mistakes. Convolutional Neural Networks (CNNs) are an advancement in machine learning that opened new possibilities for precise and effective illness detection. Complex features in images can be automatically learnt and classified by CNNs. Thus, our aim is to develop a transfer learning model that predicts potato leaf diseases using a CNN-based method.

## 1.1 Problem Statement:

Diseases pose a quiet threat to potatoes, which are a staple diet for millions of people. Farmers all throughout the nation suffer greatly from a lack of early detection of these diseases, which results in an incredible loss of ₹2,865 crore each year. This means that a staggering 12% of the 1.7 billion kg of potatoes produced in India are essentially wasted.

It is essential to identify diseases early and accurately in order to reduce losses and guarantee food security. Farmers can be empowered to: by implementing new technologies and techniques.

Use cutting-edge instruments or visual inspection methods to detect diseases early.  
Take proactive steps to reduce the likelihood of disease outbreaks.  
Use pesticides more efficiently by focusing on certain diseases at the appropriate time.  
We can safeguard potato harvests, boost farmer income, and guarantee a steady supply of this essential food source for our country by tackling these issues.

## 1.2 Objective:

Potato disease prediction and detection application is a revolutionary platform designed specifically for farmers in India, making them empowered and self-dependent. It helps in early detection and curation of diseases in potato leaves, improving both quality and quantity of harvest. It reduces revenue loss and improves the overall profit.

The project uses deep learning to forecast illnesses of the potato leaf. The software is incredibly accurate because it was created using a TensorFlow framework and makes use of transfer learning techniques. The program uses Convolutional Neural Network (CNN) trained and validated on a dataset of healthy and diseased potato leaf photos. Moreover, the program makes use of FastAPI as its backend framework. The program makes use of FastAPI as its backend framework to manage picture uploads and other processing requests. This guarantees seamless operation and effective handling of user requests, including uploads of images.

The potato disease prediction and detection application empowers Indian farmers to become proactive crop guards by fusing state-of-the-art technology with a user-centric design.



## **1.3 Significance**

- 1.3.1 Enhanced ability of farmers to make decisions: Farmers might possibly save cash and maximize crop output by making educated judgments regarding treatment tactics through early and precise disease diagnosis.
- 1.3.2 Decreased dependence on chemical intervention: Farmers can tailor their treatment strategies by recognizing certain illnesses, which may lead to a decrease in the needless application of pesticides and fungicides.
- 1.3.3 A more sustainable agricultural ecosystem can be achieved by reducing reliance on chemical controls.
- 1.3.4 Possibility of broader application: This method's effectiveness in identifying potato leaf illnesses opens the door for comparable applications in the detection of diseases in other crops.
- 1.3.5 Preventing extensive crop loss can be achieved by cost-effective disease management, which can ultimately reduce the economic burden on farmers.

## 2. Literature Review

Multiple research papers and models have been published to provide solutions to agricultural cases that require attention. Different methodologies can be seen being used to find the most efficient and accurate models for potato disease detection. Here we compile a list of all the previous work done in this field.

Author	Dataset	Algorithm Used	Prediction
Tarik et al., 2021 (3)	Self-recorded	CNN	99%
Md Nishad et al., 2022 (4)	PlantVillage,	K-means clustering	97%
Lee et al., 2021 (5)	PlantVillage	CNN	99.53%
Kulendu et al., 2022 (6)	PlantVillage	VGG16, VGG19, MobileNet and	97.89%
Tiwari et al., 2020 (7)	Private	VGG19	97.8%
Samer I.Mohamed, 2020 (8)	PlantVillage	CNN	98.2%

Table 1: Summary of Recent Papers for Potato Leaf Disease Detection and Prevention

Work has also been done for detecting diseases in other crops as well through, primarily by, Zhou et al., (9) where they introduced a hybrid deep learning model using DenseNet121 and Deep CNN. This was done to reduce training time while mitigating training loss. The model was trained on a tomato leaf dataset containing 9 classes from AI Challenger and succeeded with an accuracy of 95%.

### Unique value proposition

- Utilizes a Deep learning Convolutional Neural Network (CNN) with six convolutional layers of 64 filters each.
- Includes max pooling layers to effectively capture precise features and remove spatial dimensions.
- Utilizes pre-trained networks, taking advantage from extensive information acquired from large datasets.
- The approach of CNN and transfer learning enables in feature extraction and classification capabilities of the model.
- Utilizes the PlantVillage dataset, which is renowned for its vast collection of agricultural photos, hence augmenting the efficacy of the model.

## 3. Materials and Methodology

### 3.1 Prerequisites

**3.1.1 Git & GitHub:** Version control system (VCS) used for tracking the changes in code, to ensure easy collaboration and rollback to previous versions. GitHub is a cloud-based repository to store project's code securely.

**3.1.2 Anaconda:** It is a software that provides a pre-configured environments with essential libraries like TensorFlow, NumPy, Pandas and Matplotlib. This makes the setup easy and ensures compatibility for deep learning project.

**3.1.3 Jupyter Notebook:** A coding environment that is interactive for creating and evaluating applications. With its user-friendly interface, it enables you to write Python code, visualize data and test out various disease prediction methods. their harvests.

**3.1.4 VS Code:** It is a code editor with features like code completion, and integration with Jupyter Notebooks. This improves coding efficiency and offers debugging features for potato disease prediction application.

### 3.2 Data collection

The dataset used in the model has been taken from Kaggle. It contains images in Jpeg format of potato leaves. The dataset has been classified into three classes namely. “Potato\_\_Early\_blight”, “Potato\_\_Late\_blight”, “Potato\_Healthy”, and a total of 2152 files.

The complete dataset has been divided into 68 batches, with each batch containing 32 files as it improves efficiency and generalization of data. Input dimension of each file **(32,256, 256,3)**

### 3.3 Preprocessing

We tested multiple methods of data augmentation for normalization and preprocessing. This included resizing the image, changing the RGB value, whitening, shadow, zoom, shear, etc. However, due to the comparatively smaller size of the dataset, too much data augmentation gave negative returns. Therefore, we decided to apply only resize, random flip and random rotation.

```
augmentation = tf.keras.Sequential([
    layers.RandomFlip( "horizontal_and_vertical"),
    layers.RandomRotation( 0.2),
])
resize_image = tf.keras.Sequential([
    layers.Resizing(image,image),
    layers.Rescaling(1.0/255)
])
```

Fig 3.3.1: Preprocessing of training dataset

### 3.4 Model Architecture:

The CNN model for potato disease detection consists of 6 convolutional layers, 6 pooling layers and 1 one flattening layer. Each convolutional layer 64 filters with 3x3 kernels and 1 ReLU activation, which helps in capturing spatial hierarchy in each data. Each convolutional layer is followed by a pooling layer with pool size of 2x2 to reduce spatial dimensions and computational complexity while retaining important features. After convolutional layers and pooling layers, the outcome is flattened into a one-dimensional vector to prepare it for the dense layers. The dense layer consists of 64 filters with one ReLU activation, which adds non-linearity and enables the network to learn complex patterns.

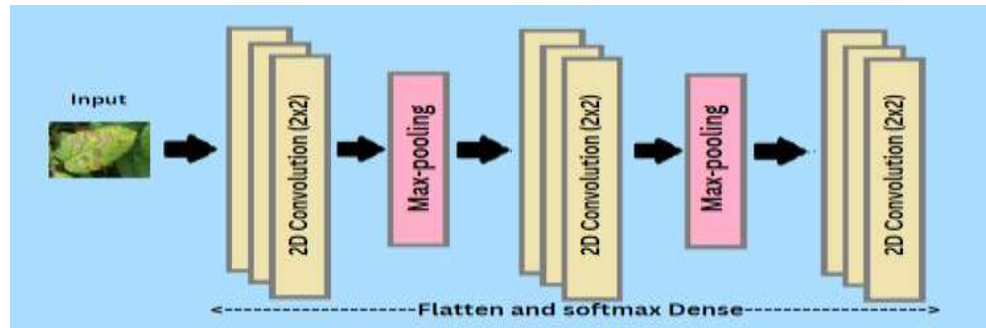


Fig 3.4.1: Proposed model architecture

The model is compiled by Adam optimizer, which is known for its efficiency and adaptive learning rate features, and sparse categorical cross entropy, which is suitable for multiple-class classification tasks. Performance metrics are used to evaluate the model during training phases, ensuring that it accurately classifies the images. This well-structured architecture enables the model to learn from the data and make effective predictions on unseen data.

Model: "sequential_9"		
Layer (type)	Output Shape	Param #
sequential_3 (Sequential)	(68, 256, 256, 3)	0
sequential_4 (Sequential)	(68, 256, 256, 3)	0
conv2d_30 (Conv2D)	(68, 254, 254, 64)	1,792
max_pooling2d_30 (MaxPooling2D)	(68, 127, 127, 64)	0
conv2d_31 (Conv2D)	(68, 125, 125, 64)	36,928
max_pooling2d_31 (MaxPooling2D)	(68, 62, 62, 64)	0
conv2d_32 (Conv2D)	(68, 60, 60, 64)	36,928
max_pooling2d_32 (MaxPooling2D)	(68, 30, 30, 64)	0

Fig 3.4.2: Model Summary

Figure 3.4.2 provides information about how the layering of the model is implemented and the output shape for each layer along with its parameters.

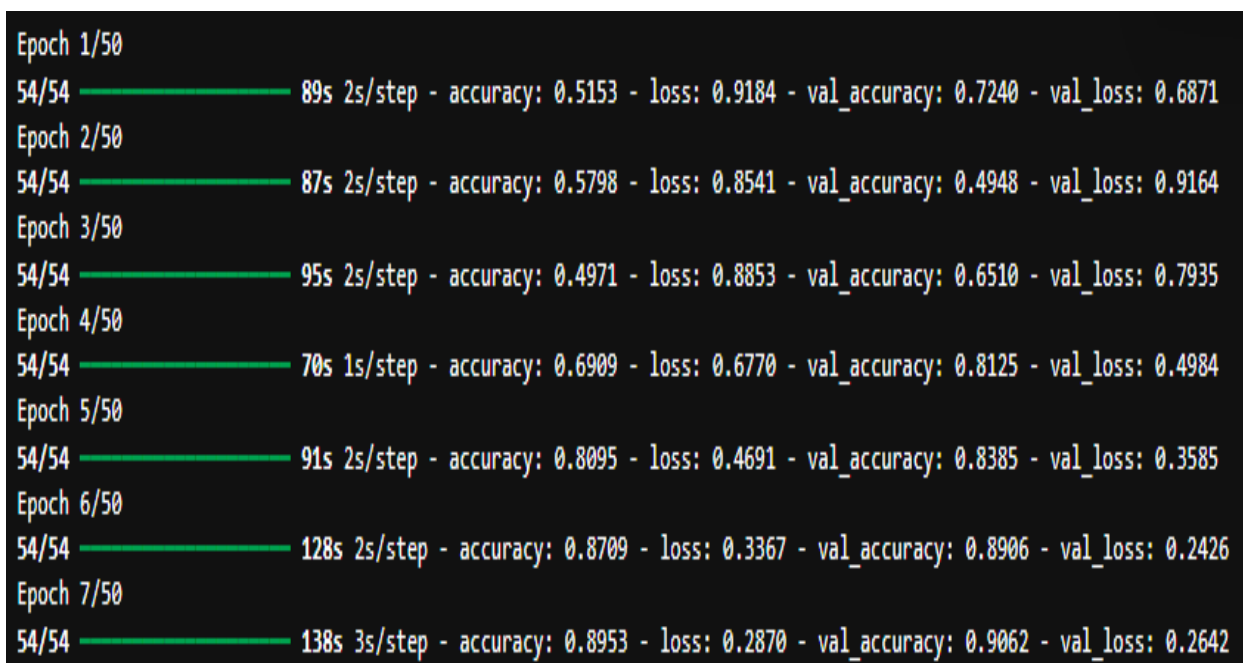
### 3.5 Model training:

The dataset has been divided into an 80-20 split, with 80% of data being used for training the model and the other 20% is utilized for validation and testing.

Following parameters have been used for training the model:

```
Train_d, //80% of total dataset
epochs = 50, // number of times the model will run through the Train_d
batch_size = 68,
verbose = 1,
validation_data = validation_d, // 10% of total dataset
```

Verbose output is set to 1 for detailed feedback during the training process.



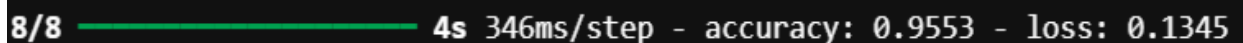
```
Epoch 1/50
54/54 ————— 89s 2s/step - accuracy: 0.5153 - loss: 0.9184 - val_accuracy: 0.7240 - val_loss: 0.6871
Epoch 2/50
54/54 ————— 87s 2s/step - accuracy: 0.5798 - loss: 0.8541 - val_accuracy: 0.4948 - val_loss: 0.9164
Epoch 3/50
54/54 ————— 95s 2s/step - accuracy: 0.4971 - loss: 0.8853 - val_accuracy: 0.6510 - val_loss: 0.7935
Epoch 4/50
54/54 ————— 70s 1s/step - accuracy: 0.6909 - loss: 0.6770 - val_accuracy: 0.8125 - val_loss: 0.4984
Epoch 5/50
54/54 ————— 91s 2s/step - accuracy: 0.8095 - loss: 0.4691 - val_accuracy: 0.8385 - val_loss: 0.3585
Epoch 6/50
54/54 ————— 128s 2s/step - accuracy: 0.8709 - loss: 0.3367 - val_accuracy: 0.8906 - val_loss: 0.2426
Epoch 7/50
54/54 ————— 138s 3s/step - accuracy: 0.8953 - loss: 0.2870 - val_accuracy: 0.9062 - val_loss: 0.2642
```

Fig 3.5: Training process of each epoch

After each epoch the model is validated based on the validation data as it helps in monitoring model's performance and prevent overfitting. Weights of the model are adjusted iteratively to minimize the loss function.

### 3.6 Model Evaluation

Once the model is trained and validated on the designated dataset, the model is then tested based on the 10% dataset to calculate the actual accuracy and loss.



```
8/8 ————— 4s 346ms/step - accuracy: 0.9553 - loss: 0.1345
```

We then use subplots to provide the summary of training and validation loss and accuracy graphs through the matplotlib library

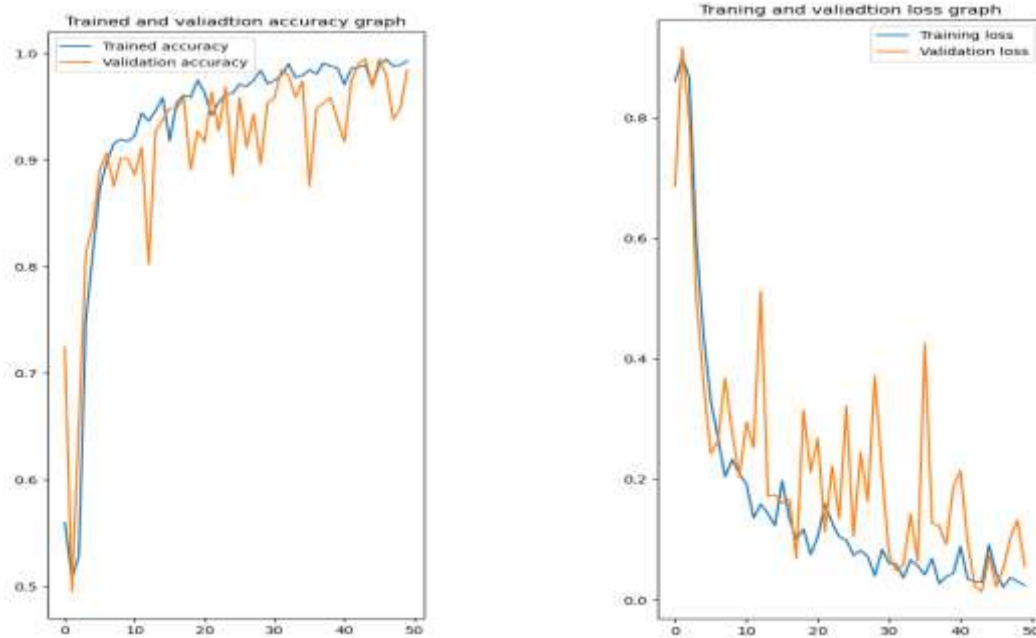


Fig 3.6.1: Accuracy and loss graphs

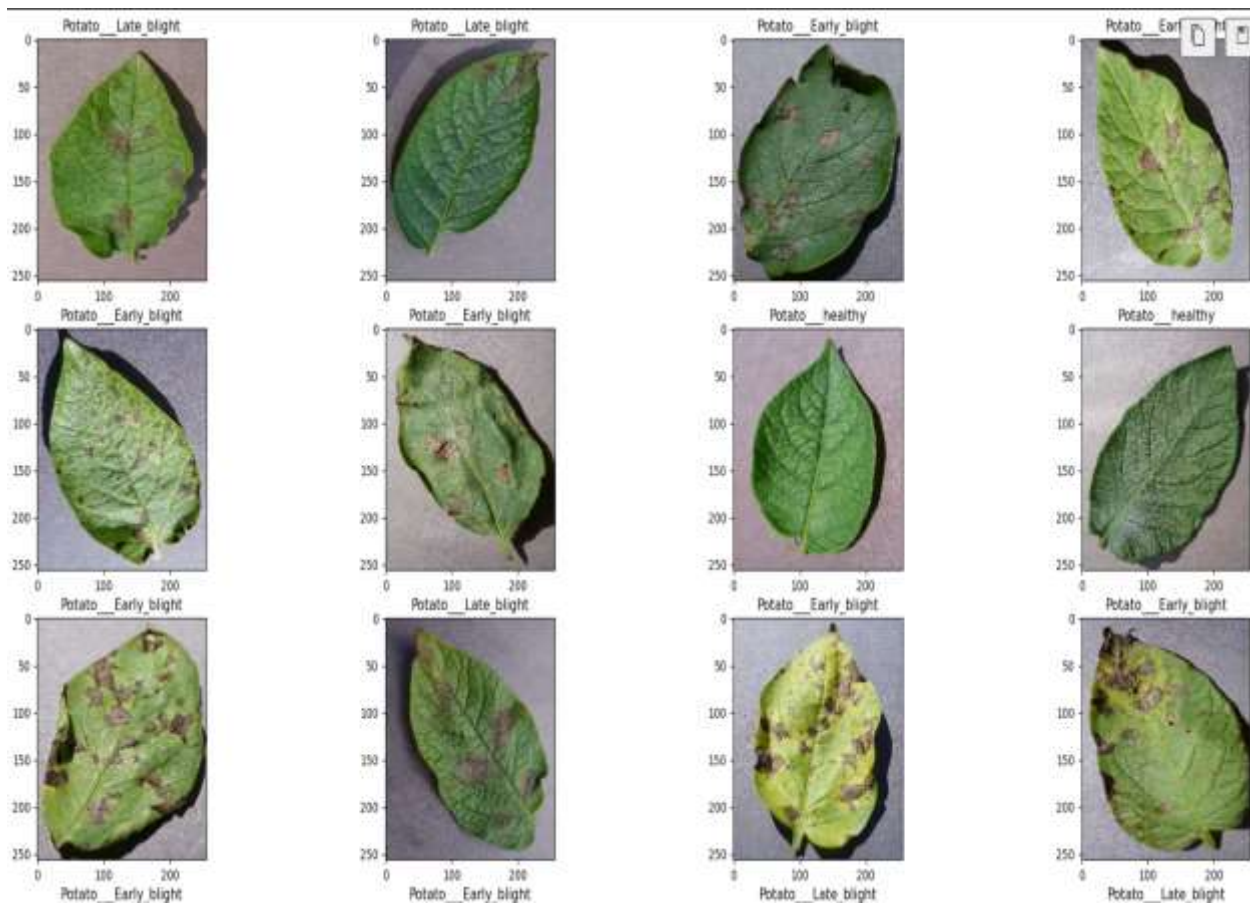


Fig 3.6.2: Test dataset for the mode

On plotting the confusion matrix, we observe that both false negatives (FN) and false positive (FP) values are near zero while the diagonal of the matrix is producing high accuracy.

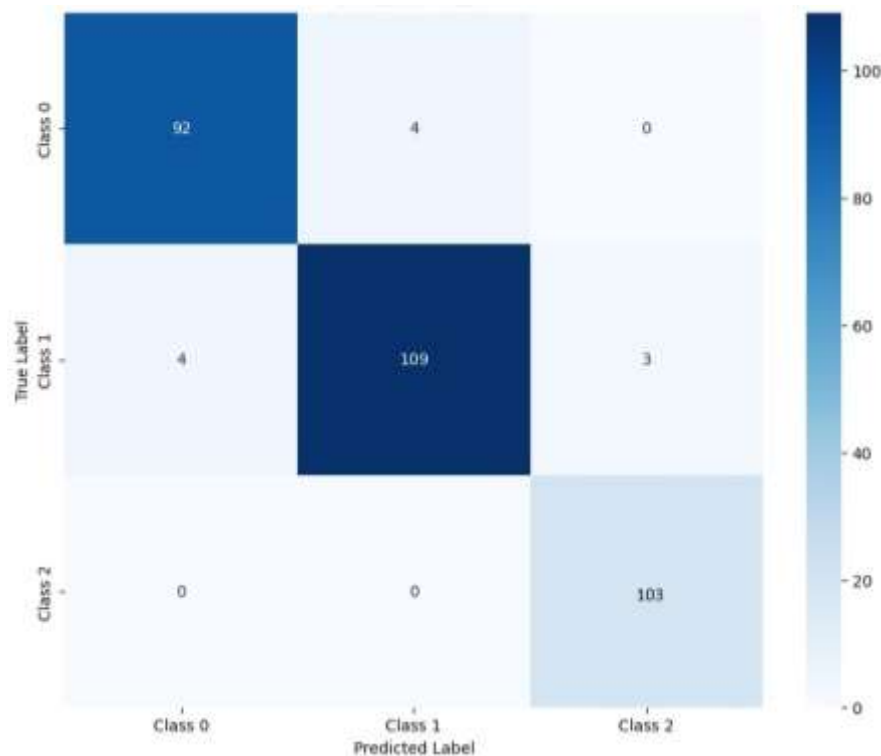


Fig 3.6.3: Confusion matrix of the model

The matrix has three classes (Class 0, Class 1, Class 2) both as true labels and predicted labels. The values in the matrix represent the number of instances for each combination of true and predicted classes.

#### 3.6.1 Class 0 Analysis:

True Positive (TP): 92 instances of Class 0 were correctly predicted as Class 0.

False Negative (FN): 4 instances of Class 0 were incorrectly predicted as Class 1.

False Positive (FP): 0 instances of other classes were incorrectly predicted as Class 0.

True Negative (TN): All other instances (excluding the 92 TP and 4 FN) were correctly identified as not Class 0

#### 3.6.2 Class 1 Analysis:

True Positive (TP): 109 instances of Class 1 were correctly predicted as Class 1.

False Negative (FN): 4 instances of Class 1 were incorrectly predicted as Class 0, and 3 instances as Class 2.

False Positive (FP): 4 instances of Class 0 were incorrectly predicted as Class 1.

True Negative (TN): All other instances (excluding the 109 TP, 4 FN, and 3 FN) were correctly identified as not Class 1.

### 3.6.3 Class 2 Analysis:

\True Positive (TP): 103 instances of Class 2 were correctly predicted as Class 2.

False Negative (FN): No instances of Class 2 were incorrectly predicted as Class 0 or Class 1.

False Positive (FP): 3 instances of Class 1 were incorrectly predicted as Class 2.

True Negative (TN): All other instances (excluding the 103 TP and 3 FP) were correctly identified as not Class 2.

Accuracy is the proportion of total instances that were correctly classified. Precision for each class is given by the ratio of true positive predictions to the total predicted positives for that class.

Precision (Class 0) =  $92 / (92 + 0) = 1.0$  (or 100%)

Precision (Class 1) =  $109 / (109 + 4 + 3) \approx 0.937$  (or 93.7%)

Precision (Class 2) =  $103 / (103 + 0) = 1.0$  (or 100%)

Recall for each class is given by the ratio of true positive predictions to the total actual positives for that class.

Recall (Class 0) =  $92 / (92 + 4) \approx 0.958$  (or 95.8%)

Recall (Class 1) =  $109 / (109 + 4 + 3) \approx 0.937$  (or 93.7%)

Recall (Class 2) =  $103 / (103 + 0) = 1.0$  (or 100%)

F1-Score for each class is the harmonic mean of precision and recall.

F1-Score (Class 0)  $\approx 0.978$  (or 97.8%)

F1-Score (Class 1)  $\approx 0.937$  (or 93.7%)

F1-Score (Class 2) = 1.0 (or 100%)

Overall, the classifier appears to perform very well, especially for Classes 0 and 2, with very high precision, recall, and F1-scores. Class 1 also has strong performance, though there is a slight drop in precision and recall compared to the other two classes.

Throughout all three classes, the classifier performs exceptionally well, attaining high recall, F1-scores, and accuracy. Class 0 and Class 2 in particular both exhibit perfect precision (100%) and strong recall (Class 0: 95.8%, Class 2: 100%). Though slightly lower than the other classes, Class 1 still performs well, with a recall of 93.7% and precision of roughly 93.7%. The model's overall accuracy is high, suggesting that it is useful in correctly identifying most cases. The model's reliability is further demonstrated by the harmonic mean of precision and recall (F1-scores), where Class 0 is 97.8%, Class 1 is 93.7%, and Class 2 is 100%. This research demonstrates how reliable and effective the classifier is in correctly predicting the true labels.



### 3.7 Backend Server

The model then needs to be initialized on a backend server such that whenever a consumer inputs an image of potato leaf, that image can then be sent to the model for detection. This process is implemented through FastAPI in Python.

```
model_path = "new_model.keras"
DL_model = tf.keras.models.load_model(model_path, compile=False)

class_names = ["Potato__Early_blight", "Potato__Late_blight", "Potato__healthy"]
```

Fig 3.7.1: Importing model and initializing classes

After saving and storing the model in a “.keras” format we initialize it into our FastAPI server and declaring our landmark variables as shown in Fig 3.7.1 . This allows us to access the model whenever the prediction function is called.

A POST function is defined to send the file data (image in our case) which the model then reads and gives the necessary predicted outcome:

```
@app.post("/prediction")
async def prediction(
    file: UploadFile = File(...)
):
    content = read_file_as_image(await file.read())
    img_batch = np.expand_dims(content, 0)
    predict = DL_model.predict(img_batch)
    class_predicted = class_names[np.argmax(predict[0])]
    most_accurate = np.max(predict[0])

    return {
        'class': class_predicted,
        'confidence': float(most_accurate)
    }
```

The server is then hosted on a cloud hosting service (free tier) on render (<https://render.com>). An API testing platform like Postman can be used to test the degree of functionality of the server:

```
1  [ ]
2  {
3    "class": "Potato__Late_blight",
4    "confidence": 0.9871373176574707
}
```

Fig 3.7.2: Postman POST function

In figure 3.7.2, we tested whether the server is handling requests correctly and providing the prediction output. The server gave the class of the detected leaf as well as the model's confidence in its analysis.

### 3.8 Frontend Application

A ReactJS website is built using NodeJS and npm. The home.js file contains the code of the graphical interface for the website.

The axios library is used in a 'sendFile' function for sending HTTP requests to the backend server, i.e., when the consumer uploads the image for detection, the 'sendFile' function will pick up the request and forward it to the model for processing. If the function call is successful, the terminal issues a status code of '200:OK'. The code for uploading the image through the axios function is shown in Fig 3.8.1

```
const sendFile = useCallback(async () => {
  if (image) {
    let formData = new FormData();
    formData.append("file", selectedFile);
    try {
      let res = await axios({
        method: "post",
        url: process.env.REACT_APP_API_URL,
        data: formData,
      });
      if (res.status === 200) {
        setData(res.data);
      }
    } catch (error) {
      console.error("Error uploading file:", error);
    } finally {
      setIsloading(false);
    }
  }
}
```

Fig.3.8.1: Axios library function

This is done by declaring the API URL in the code:

```
REACT_APP_API_URL=https://potato-zxei.onrender.com/prediction
```

This URL is then declared in the 'sendFile' function and calls on the backend server for model detection. The ReactJS website is hosted on another cloud hosting service namely, Vercel (free tier, <https://vercel.com/>). The website is built according to import requirements and then hosted in a specific domain for use. This domain is then added into the "Origins" variable which handles all the domains that the model server can access in the FastAPI server file to maintain interconnectivity between the backend server and the frontend website.

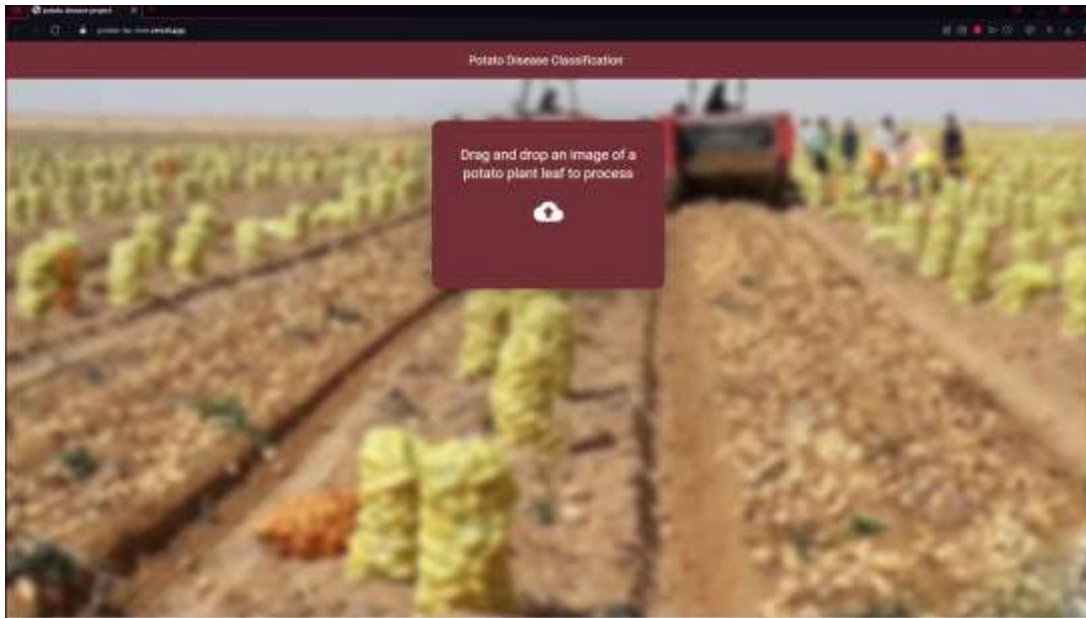


Fig.3.8.2: Website hosted on cloud



Fig.3.8.3: Drag and drop feature on React website

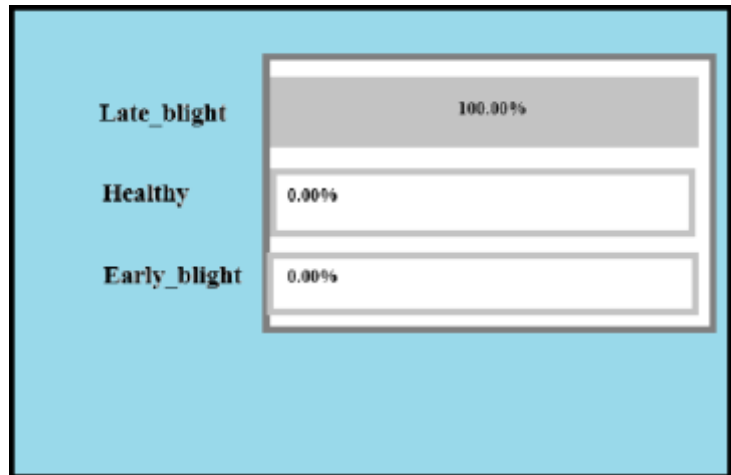


Fig.3.8.4 Visual representation of prediction

Figure 3.8.2 shows the model running on a cloud server where the user inputs the image of potato leaf, and the request is sent to the FastAPI server which is then processed by the model to produce an output prediction. Figure 3.8.3 shows the output produced on the website and the user can then prompt to reset their query. To provide a better understanding, Figure 3.8.4 provides a bar graph representation of the model's behaviour towards the image. In this case since the confidence was 100%, the model did not detect any false positives.

## 4. Conclusion

The aim of this project was to provide a simple and straightforward approach for farmers and field laborers to be able to identify and be aware of crop degradation on their potato plant. This will help not only prevent monetary loss but also avoid time consumption in a field of work where there are already ample tasks at hand.

The project developed a ReactJS website with a minimalist GUI that provides effectiveness and simplicity. Users can upload an image of a potato leaf and the CNN model provides predictions with high accuracy. This early detection process helps eliminate the need for damage control and focuses on disease prevention by bringing any concern to light that may harm production of the crop.

The scope of this project can be increased to a larger extent including creating a mobile application for the same so that the user can click photos in real time within the application and get an analysis of the disease. Moreover, we can also expand the target crop from just a potato leaf to other vegetables as well but not limited to tomatoes, bell peppers, etc. This will help provide more effectiveness to the project and mitigate crop damage in other fields.

## References

1. *In High-Performance Web Apps with FastAPI: The Asynchronous Web Framework Based on Modern Python*. **Torrey, L., & Shavlik, J.** 2023.
2. *An introduction to convolutional neural networks*. **O'shea, K., & Nash, R.** 2015.
3. **Tarik, Marjanul and Akter, Sadia and Mamun, Abdullah and Sattar, Abdus.** *Potato Disease Detection Using Machine Learning*. 2021-02.
4. *Predicting and Classifying Potato Leaf Disease using K-means Segmentation Techniques and Deep Learning Networks*. **Nishad, Md. Ashiqur Rahaman.** 2022, Procedia Computer Science. 1877-0509.
5. *High Efficiency Disease Detection for Potato Leaf with Convolutional Neural Network*. **Lee, .** s.l. : SN Computer Science, 2021. 2661-8907.
6. *Physiological and Molecular Plant Pathology*. **Chakraborty, Kulendu Kashyap.** 2022. 0885-5765.
7. *Potato Leaf Diseases Detection Using Deep Learning*. **Tiwari, D., M. Ashish, S. Bhardwaj.** 2020.
8. *Potato Leaf Disease Diagnosis and Detection System Based on Convolution Neural Network*. **Mohamed, Samer I.** 2020.
9. *Tomato Leaf Disease Identification by Restructured Deep Residual Dense Network*. **C. Zhou, S. Zhou.** 2021. 2169-3536.