Introduction

Potatoes, one of the world's most crucial food crops, are an integral part of the global diet and agricultural economy. However, potato crops are vulnerable to a wide array of diseases, particularly those affecting the leaves, which can lead to substantial yield losses and reduced crop quality. Addressing these challenges effectively is critical for ensuring food security and the economic stability of millions of farmers worldwide. Traditionally, identifying and diagnosing potato leaf diseases has relied heavily on manual inspection by experts, a process that is both time-consuming and prone to human error. The advent of modern technologies, particularly in the field of artificial intelligence (AI) and deep learning, offers new avenues to tackle these issues with greater accuracy and efficiency.

1.1 Background & Context

Potato (Solanum tuberosum) is one of the staple crops globally, contributing significantly to the world's food supply. However, potato cultivation faces challenges, including the susceptibility of potato plants to various diseases. Timely identification and management of these diseases are crucial to ensuring a healthy potato yield and sustaining agricultural productivity. Potato diseases, caused by bacteria, fungi, viruses, and other pathogens, manifest through distinctive symptoms on the leaves, stems, and tubers. Traditional methods of disease identification often rely on visual inspection by farmers and agricultural experts. However, human inspection is prone to errors, and the delay in disease detection can lead to substantial crop losses.

1.2 Purpose

The primary purpose of the Potato Disease Classification System is to leverage advanced machine learning techniques, specifically Convolutional Neural Networks (CNNs), to address the challenges associated with manual identification of potato diseases in agricultural settings. The content serves the following purposes:

- Early diseases detection
- Accurate Classification
- User Friendly Classification
- Timely Inventions
- Sustainable Agriculture
- Stakeholder Empowerment

1.3 Functional Features:

- **Image Uploading: -** Users should be able to upload images of potato plants, including leaves, stems, and any affected parts, through a user-friendly interface.
- **Image Preprocessing: -** The model should perform preprocessing on the uploaded images so that they can be used as input to the CNN model.
- **Convolution neural network Implementation: -** Integrate a CNN-based model capable of classifying potato diseases based on visual symptoms present in the input images.
- **Multi class classification: -** Support multi-class classification to identify and differentiate between various types of potato diseases commonly found in agricultural settings.
- **Accuracy Matrices**: Provide accuracy metrics for the trained model to quantify its performance.
- **Real time Result: -** Provide real-time or near-real-time feedback on disease classification, enabling users to make swift decisions regarding crop management.

1.4 Significance of project:

The significance of Potato Disease Classification model extends beyond its technical capabilities. Its user-friendly interface empowers farmers, irrespective of their technical expertise, fostering widespread adoption. The real-time feedback on disease classification aids farmers in making informed decisions promptly, optimizing overall crop health and yield.

The Potato Disease Classification System, incorporating Convolutional Neural Networks (CNNs), is a pivotal advancement in modern agriculture. By facilitating the early and accurate identification of diseases in potato crops, this system revolutionizes crop management practices. Its precision allows for targeted interventions, minimizing the need for indiscriminate pesticide application and promoting sustainable agricultural methods.

1.5 Organization of report:

This report is structured to provide a systematic presentation of the research conducted on potato diseases classification using Convolutional Neural Networks (CNN). The sections are organized to offer a clear progression of the study, starting with an introduction to set the context, followed by a literature review, dataset description, methodology, results, discussion, and conclusion. This structured approach aims to facilitate a comprehensive understanding of the application of CNN in the context of plant disease identification.

- **Convolutional Neural Networks in Image Classification**
  - Basics of CNNs
  - Previous work on plant disease classification using CNNs
- **Review of Related Studies**
  - Summary of key papers, methodologies, and findings relevant to potato leaf disease detection

- **Current Limitations in Existing Methods**
  - Issues with accuracy, scalability, and accessibility

- **Definition of the Problem**
  - Accurate and efficient classification of potato leaf diseases
- **Specific Research Questions**
  - How can CNNs improve the accuracy of disease detection?
  - What pre-processing techniques enhance model performance?
  - How to create a robust dataset for training the model?

- **Objectives**
  - Develop a CNN model for classifying potato leaf diseases
  - Create a high-quality, annotated dataset of potato leaf images
  - Evaluate the model's performance
  - Implement the model in a user-friendly application

- **Data Collection**
  - Sources of potato leaf images
  - Annotation process
- **Data Pre-processing**
  - Image enhancement and augmentation techniques
- **Model Development**
  - Architecture of the CNN
  - Training and validation procedures

2.Related Work

2.1 Literature Survey

[1] This paper explores the use of CNNs combined with Gabor filters for plant disease detection. The integration of Gabor filters helps in enhancing the texture features, improving the overall accuracy of the CNN model. The authors demonstrate significant improvements in disease classification performance, highlighting the potential of combining traditional image processing techniques with deep learning. [2] This study proposes a 9-layer deep CNN model specifically designed for the automatic identification of plant leaf diseases. The model achieves high accuracy by learning discriminative features from the leaf images. The paper provides a detailed analysis of the model's architecture and training process, demonstrating its effectiveness in accurately classifying various plant diseases.[3] This paper investigates the use of transfer learning for plant disease detection. By leveraging pre-trained CNN models, the authors fine-tune the networks for the specific task of plant disease classification. The study shows that transfer learning significantly reduces the need for large training datasets while maintaining high accuracy, making it a practical solution for agricultural applications.[4] The authors present a real-time plant disease detection system using CNNs. The model is capable of processing leaf images and identifying diseases in real-time, providing immediate feedback to farmers. The paper highlights the practical implementation of the model, including the development of a user-friendly interface for real-world application.

[5] This review paper provides a comprehensive overview of deep learning applications in agriculture, with a particular focus on plant disease detection. The authors discuss various deep learning models, including CNNs, and their effectiveness in different agricultural tasks. The review highlights the challenges and opportunities in deploying deep learning models in agricultural environments. [6] This study develops a CNN-based model for classifying crop diseases. The model is trained on a large dataset of crop images and achieves high classification accuracy. The paper emphasizes the importance of data augmentation and pre-processing in improving model performance. The authors also compare their model with traditional machine learning techniques, demonstrating the superior performance of CNNs.

2.2 Gap Identified

Identifying gaps in the literature or existing solutions involves recognizing areas where current research or implementations may fall short or leave room for improvement. Some potential gaps include:

- Limited Dataset
- Labeling Challenges
- Transferability
- Real- Time Processing
- Interpretability
- Integration with the Farming Practices

3. Problem Statement and Objectives

3.1 Problem Statement

" Project statement is to classify whether a plant is healthy or not by using an image of a potato leaf. It is a multi-class classification problem. Following are the three classes: • Late Blight- Potato leaves are more deteriorated. • Early Blight- Potato leaves are in the early stage of disease. • Healthy- Leaves are healthy"

3.2 Objectives

- Diagnose potato disease using leaf pictures that we are going to do through advanced machine learning technique.
- Detect early disease in potato.
- Implement an automated disease detection system.
- Improve accuracy in identifying potato diseases.

3.3 Scope

The system will analyze images of potato plants to identify and classify various diseases. It aims to assist farmers and researchers in early detection and management of potato diseases.:

- **Image recognition expertise**: CNNs excel at extracting features from images, making them ideal for analyzing visual patterns of diseased and healthy potato plants.
- **High accuracy:** Studies have shown that CNNs can achieve high accuracy in potato disease classification, often surpassing traditional methods like manual identification or spectral analysis.
- **Non-invasive and efficient:** Analyzing images is a non-invasive approach, eliminating the need for physical samples or destructive testing. Additionally, CNNs can process large datasets quickly, enabling efficient disease detection at scale.
- **Adaptability to diverse diseases:** CNNs can be trained on datasets containing various potato diseases, allowing them to identify a broad spectrum of threats.

4. Overall Description

4.1 Product Perspective

1. Precision Agriculture App for Farmers:

- Mobile app: Integrates with a smartphone camera to capture potato plant images.
- Real-time analysis: CNN model on the cloud analyses images and identifies diseases instantly.
- Disease reports: Provides detailed reports on identified diseases, including severity level and recommended treatment options.
- Field mapping: Geotag images to create field maps highlighting disease hotspots for targeted management.
- Weather updates and alerts: Informs farmers about upcoming weather conditions that could Favor disease outbreaks

2. Disease Monitoring and Prediction System for Agribusinesses:

- Web-based platform: Allows large-scale farmers and agricultural cooperatives to upload potato field images.
- Advanced analytics: Employs CNNs to analyse images and predict disease outbreaks based on historical data and weather patterns.
- Early warning system: Sends alerts to farmers when disease conditions are favourable, enabling preventative measures.
- Yield forecasting: Estimates potential crop losses based on disease prevalence, informing insurance companies and market analysis.
- Data and insights: Provides aggregated disease data and trends to government agencies and research institutions.

4.2 Product Function

- **Web-Based Platform:**
    - Upload large batches of potato field images captured by drones or automated systems.
    - Advanced analytics using CNNs to detect disease outbreaks, predict future infections, and estimate potential yield losses.
    - Early warning system to alert farmers about impending disease threats.
    - Data-driven insights on disease prevalence, geographic spread, and impact on production costs.
    - Traceability and certification modules for verifying disease-free potatoes throughout the supply chain.

4.3 User Characteristics:

Farmers:

- **Technical proficiency**: Range from those with basic smartphone skills to those comfortable with computers and apps.
- **Farm size:** Include small-scale family farms, large commercial operations, and cooperatives.
- **Geographic locations:** Reside in diverse regions with varying access to technology and internet connectivity.
- **Primary concerns:**
    - Accurate and timely disease identification
    - Ease of use and accessibility
    - Cost-effectiveness of solutions
    - Practical guidance on disease management
    - Impact on yields and farm profitability

4.3 Hardware & Software Requirement:

- **Hardware Requirement:**

1. **Server Infrastructure:**

   Multi-core processors (e.g., Intel Xeon or AMD EPYC) for parallel processing.
   Sufficient RAM (e.g., 16 GB or more) for efficient data processing and analysis.
   Adequate storage capacity (e.g., SSDs) for storing email data, machine learningmodels, and threat intelligence feeds.

2. **Network Infrastructure:**

   High-speed and reliable network connectivity to ensure seamless communication between components.

3. **Scalability Considerations:**

   Architecture designed for scalability to accommodate varying workloads and organizational sizes.

- **Software Requirement:**

  - Programming language: Python (most common due to extensive libraries and frameworks for deep learning).
  - Deep learning framework: TensorFlow, PyTorch, Keras, or others.
  - Image processing libraries: OpenCV, Pillow, scikit-image.
  - Data manipulation libraries: NumPy, pandas.
  - Machine learning libraries: scikit-learn (for preprocessing, evaluation).

5.System Design

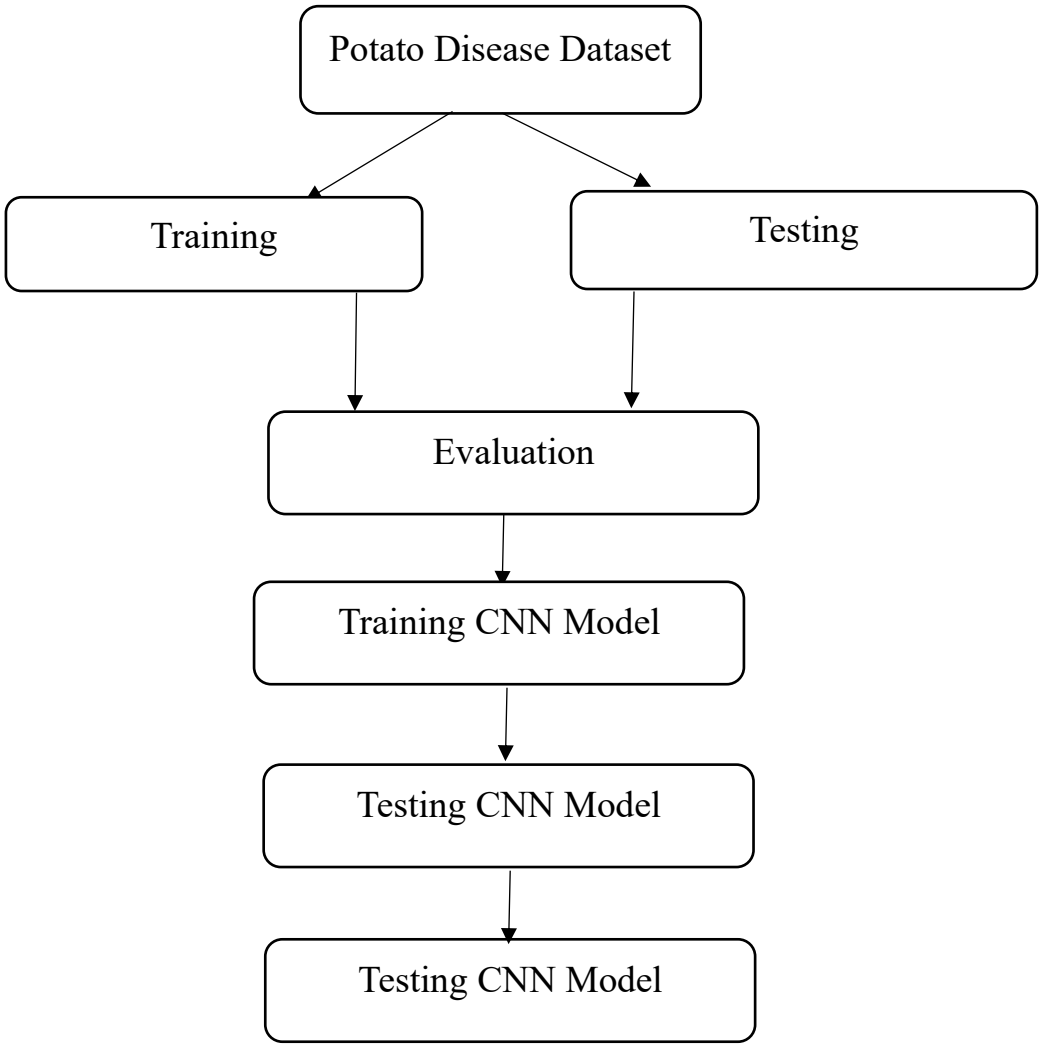
5.1 UML Diagram

5.1.1 Proposed System

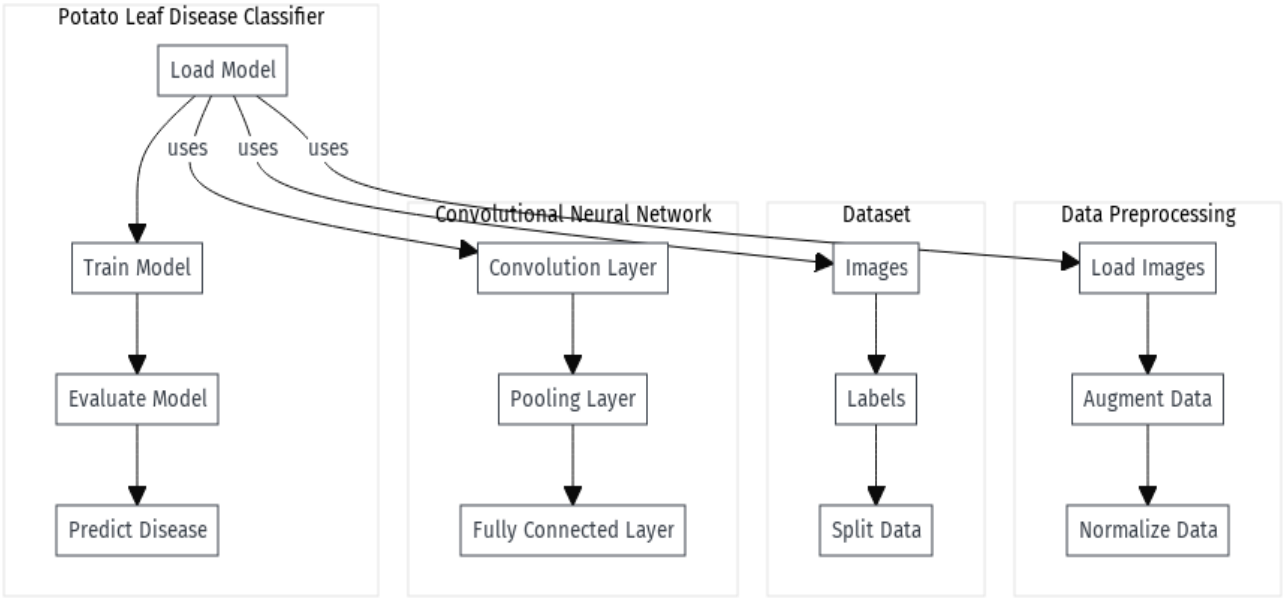
1. Data Acquisition and Preprocessing:
    - Image data collection:
    - Image preprocessing:
2. CNN Model Design and Training:
    - Model architecture selection:
    - Training setup:
3. Inference and Disease Classification:
    - Deploy the trained CNN model:
    - Image processing and prediction:
    - Visualization and recommendations:
4. System Monitoring and Improvement:
    - Performance monitoring:
    - Feedback mechanisms:
    - Continuous learning and updates:

5.1.2 Block Diagram

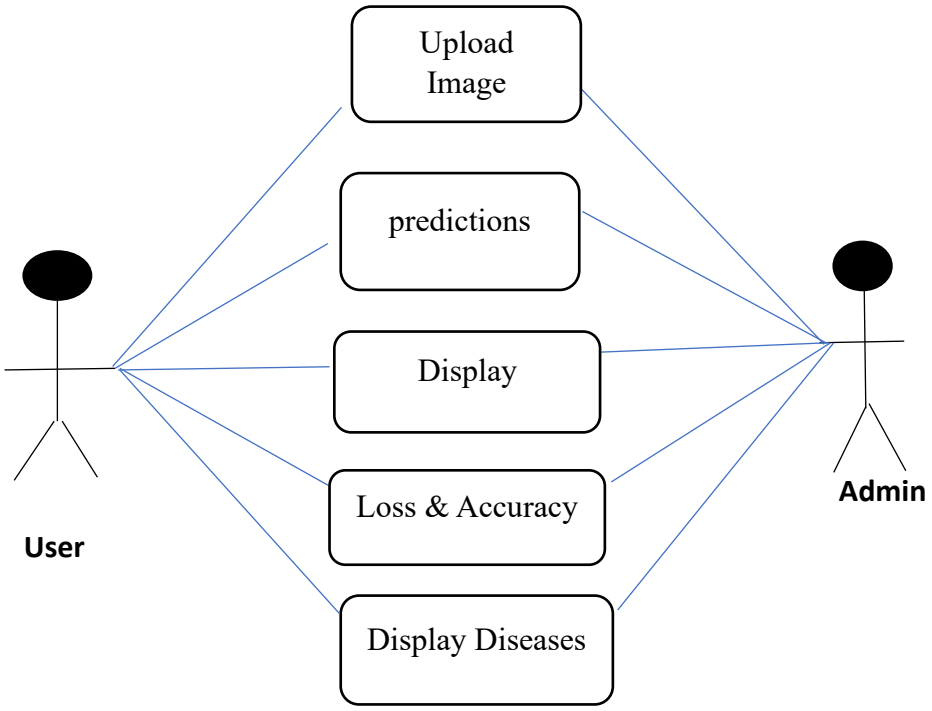
`



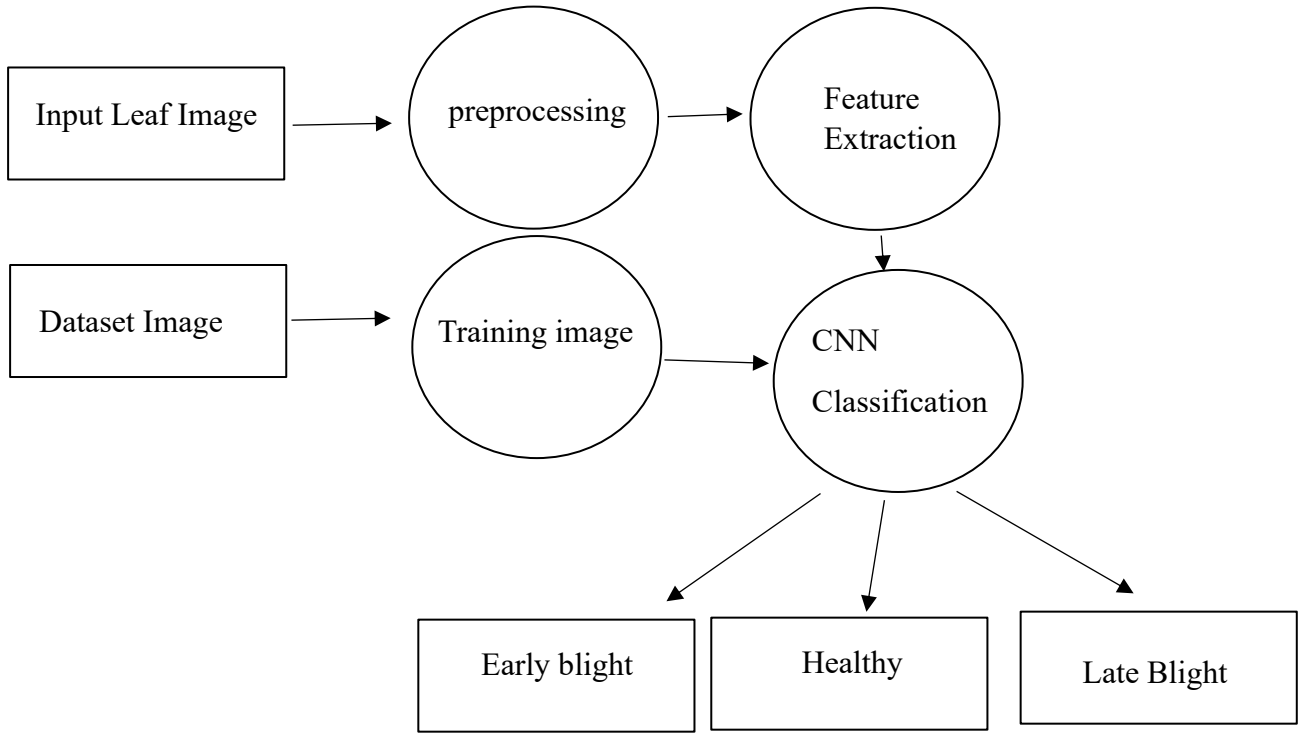
5.1.3 Component Diagram



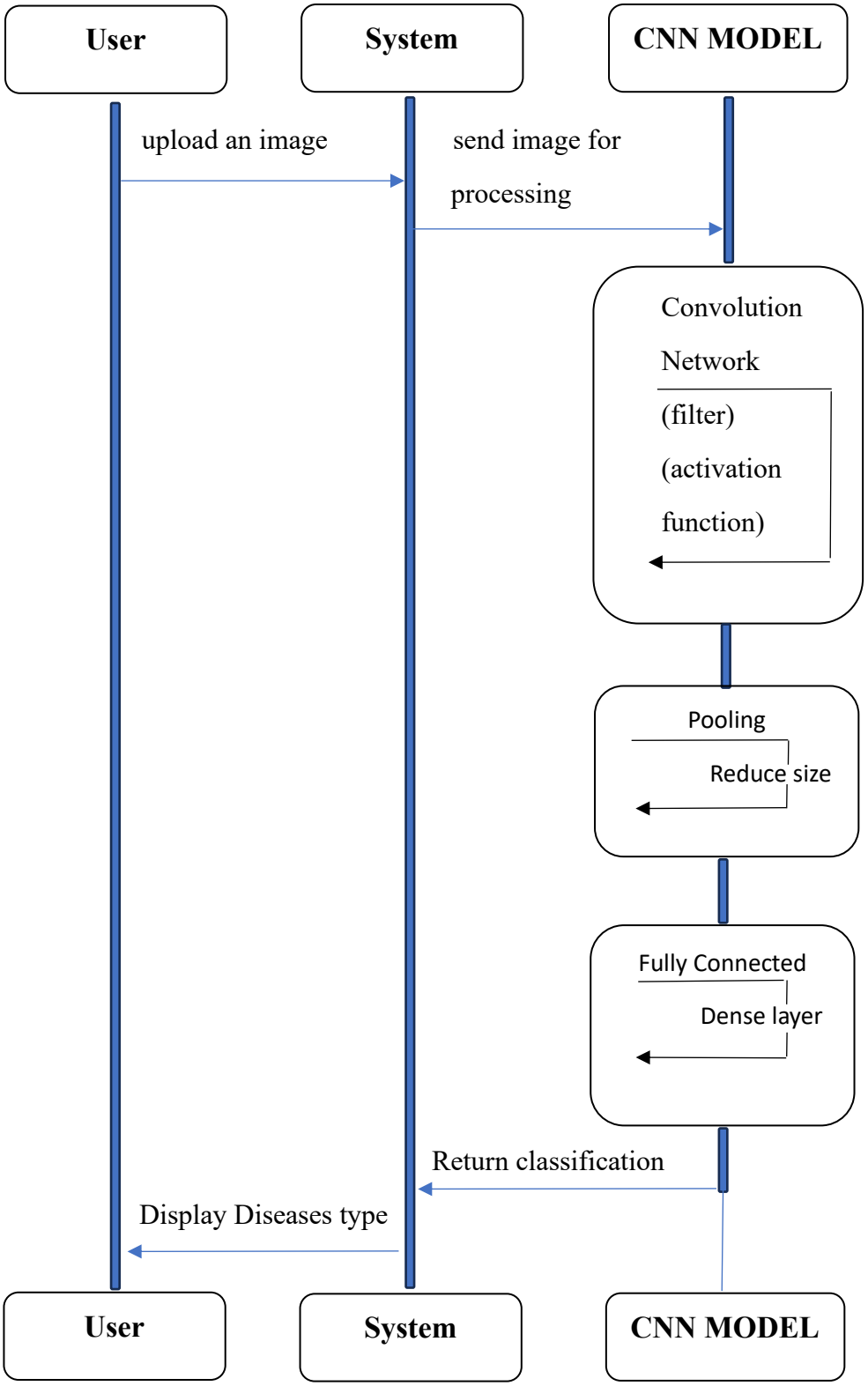
5.1.4 Use Case Diagram



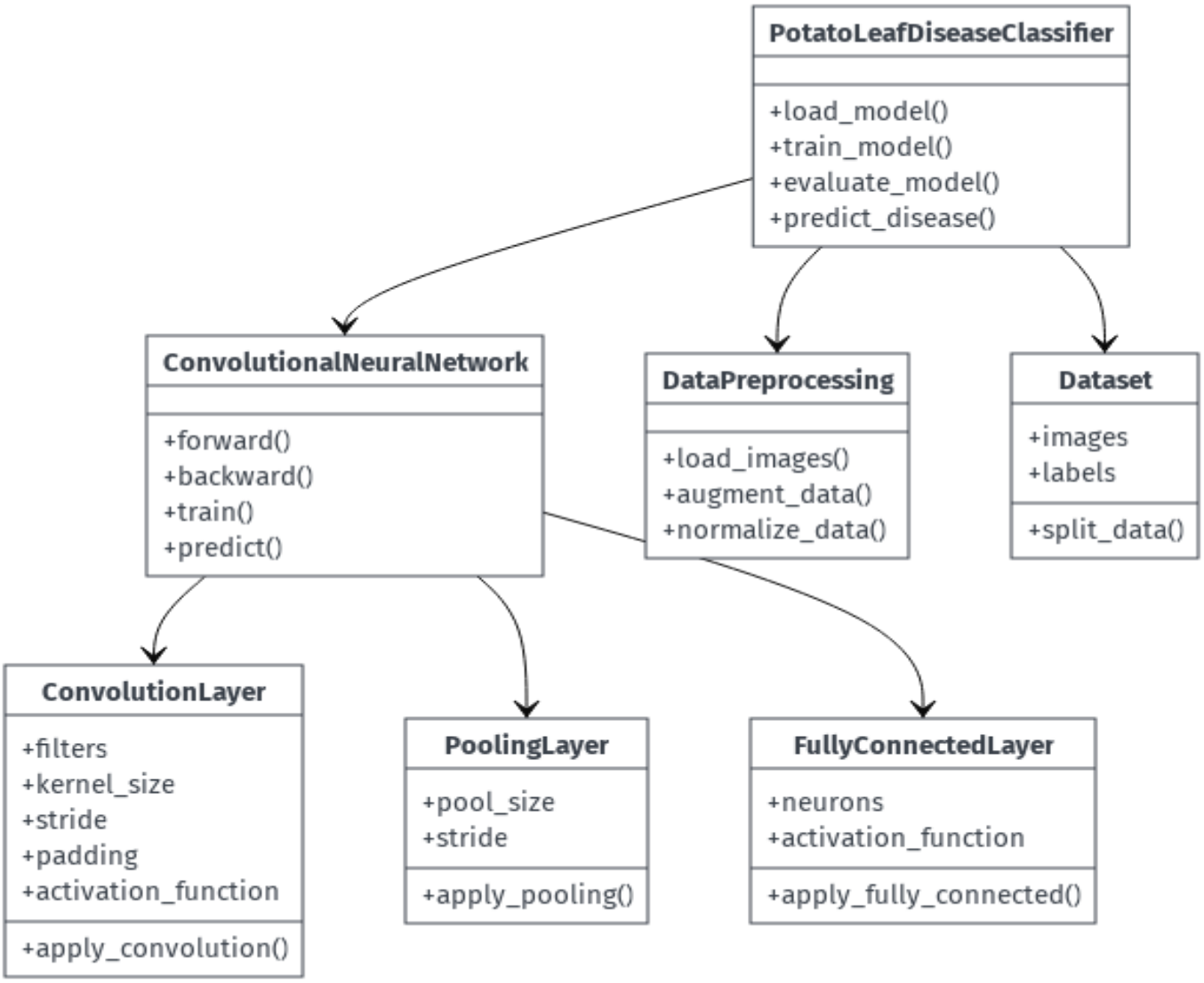
5.1.5 Data Flow Diagram



5.1.6 Sequence Diagram



5.1.7 Class Diagram



6. Implementation Details

6.1 Project Modules

1. **Data Collection And preprocessing**
    - **Data Acquisition:** Collect a diverse and representative dataset of potato plant images, including healthy plants and various diseased conditions.
    - **Data Annotation:** Manually label each image with the corresponding disease type or status.
    - **Data Augmentation:** Increase the diversity of the dataset by applying transformations like rotation, scaling, and flipping to generate additional training samples.
2. **Model Architecture:**
    - **Convolutional Layers:** Design the architecture with convolutional layers to capture spatial hierarchies and features from the input images.
    - **Pooling Layers**: Use pooling layers to down sample spatial dimensions and reduce computational complexity.
    - **Fully Connected Layers:** Connect the output of convolutional layers to fully connected layers for classification.
    - **Activation Functions:** Integrate activation functions (e.g., ReLU) to introduce non-linearity.
3. **Training:**
    - **Loss Function:** Define an appropriate loss function, such as categorical cross-entropy, to measure the difference between predicted and actual classes.
    - **Training Parameters:** Set parameters like learning rate, batch size, and epochs for model training.
4. **Validation and Testing:**
    - **Validation Set:** Split the dataset into training and validation sets to monitor the model's performance during training.
    - **Testing Set:** Use a separate testing set to evaluate the final model's generalization performance.

6.2 General Installation Steps

- **Install Required Libraries:**

  !pip install Tensorflow

  !pip install numpy

  !pip install matplotlib

  !pip install pandas

- **Import The Libraries:**

  import tensorflow as tf

  from tensorflow.keras import layers,models

  from tensorflow.keras.layers import Dense

  import matplotlib.pyplot as plt

  import numpy as np

  import pandas as pd

- **Dataset Preparation:**

  Download a dataset of potato plant images labeled with disease categories. You may find such datasets on platforms like Kaggle. But we clicked Pictures And added to Our dataset.

- **Data Preprocessing:**

  Write a script to preprocess the dataset, including resizing images, normalizing pixel values, and splitting itinto training, validation, and testing sets.

- **Model Development:**

  Create a Python script to define and train the CNN model using TensorFlow and Keras.

  Design the model architecture, compile it with an appropriate optimizer and loss function, and train it on the prepared dataset.

- **Save the Trained Model:**

  Save the trained model weights and architecture

7. Testing And Validation

7.1 Testing:

7.1.1 Training:

We will train our 80% Model . and in remaining 20% we will 10% validation &10% Testing.

```
#80% ==>training
#20% ==> 10% validation , 10% test
train_size = 0.8
len(dataset)*train_size
```

```
36.800000000000004
```

7.1.2 Accuracy Testing:

Evaluate overall accuracy on a balanced test dataset containing an equal number of healthy and diseased potato plant images.

```
[ ]  score = model.evaluate(test_ds)
```

```
6/6 [==============================] - 3s 19ms/step - loss: 0.0000e+00 - accuracy: 1.0000
```

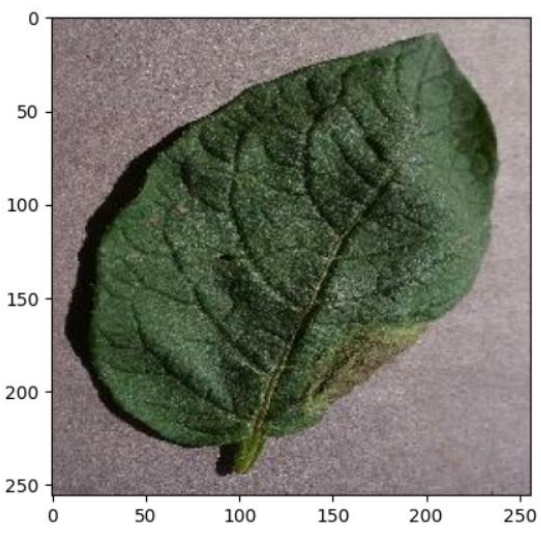
```
[ ]  score
```

```
[0.0, 1.0]
```

7.1.3 Imbalanced Dataset Testing:

Use a test dataset with imbalanced classes (e.g., more healthy plants than diseased ones) to check if the model handles class imbalances effectively.

```
for image_batch , label_batch in dataset.take(1):
  plt.imshow(image_batch[0].numpy().astype("uint8"))
```



7.1.4 Generalization Testing:

Evaluate the model on a set of potato plant images not seen during training to assess its generalization to new, unseen data.

```
input_shape= (BATCH_SIZE , IMAGE_SIZE, IMAGE_SIZE ,chanels)
n_classes = 3
model  = models.Sequential([
    resize_and_rescale,
    data_augmentation,
    layers.Conv2D(32,kernel_size=(3,3), activation ='relu',input_shape=input_shape),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size=(3,3), activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size=(3,3), activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size=(3,3), activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size=(3,3), activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size=(3,3), activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64, activation = 'relu'),
    layers.Dense(n_classes, activation = 'softmax'),
])
model.build(input_shape=input_shape)
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_3 (Sequential)   (20, 256, 256, 3)         0

 sequential_4 (Sequential)   (20, 256, 256, 3)         0

 conv2d_6 (Conv2D)           (20, 254, 254, 32)        896

 max_pooling2d_6 (MaxPoolin  (20, 127, 127, 32)        0
 g2D)

 conv2d_7 (Conv2D)           (20, 125, 125, 64)        18496

 max_pooling2d_7 (MaxPoolin  (20, 62, 62, 64)          0
 g2D)

 conv2d_8 (Conv2D)           (20, 60, 60, 64)          36928

 max_pooling2d_8 (MaxPoolin  (20, 30, 30, 64)          0
 g2D)

 conv2d_9 (Conv2D)           (20, 28, 28, 64)          36928

 max_pooling2d_9 (MaxPoolin  (20, 14, 14, 64)          0
 g2D)

 conv2d_10 (Conv2D)          (20, 12, 12, 64)          36928

 max_pooling2d_10 (MaxPooli  (20, 6, 6, 64)            0
 ng2D)

 conv2d_11 (Conv2D)          (20, 4, 4, 64)            36928

 flatten_1 (Flatten)         (20, 256)                 0

 dense_2 (Dense)             (20, 64)                  16448

 dense_3 (Dense)             (20, 3)                   195

=================================================================
Total params: 183747 (717.76 KB)
Trainable params: 183747 (717.76 KB)
Non-trainable params: 0 (0.00 Byte)
```

7.5. Test Cases

1. **Dataset Loading:**
   Test Case: Verify that the dataset is loaded correctly from the specified source.

   Expected Result: The dataset loads without errors, and the correct number of images and labels are present.

   Actual output: Dataset Loads without errors.

   Test Result: Pass

2. **Data Preprocessing:**

   Test Case: Ensure images are resized to the required dimensions (e.g., 256x256).

   Expected Result: Images are uniformly resized without distortion.

   Actual output: Images are uniformly resized.

   Test result: **Pass**

3. **Data Augmentation:**

   Test Case: Validate that data augmentation techniques (e.g., rotation, flipping) are applied correctly.

   Expected Result: Augmented images show the expected transformations.

   Actual output: Augmented images show the expected transformations

   Test result: **Pass**

4. **Data Classification:**

   Test Case: Check weather data is classified or not into 3 categories (i.e., Early blight, Late Blight, Healthy)

   Expected Result: Data classifies into 3 categories (i.e., Early blight, Late Blight, Healthy).

   Actual output: Data classifies into 3 categories (i.e., Early blight, Late Blight, Healthy).

   Test result: **Pass**

5. **Data Manipulation:**

   Test Case: Convert image into matrix format.

   Expected Result: Convert image into matrix format

   Actual output: Images Converted image into matrix format.

   Test result: **Pass**

6. **Model Compilation:**

   Test Case: Check if the model compiles successfully with the specified optimizer, loss function, and metrics.

   Expected Result: The model compiles without errors.

   Actual Output: The model compiled without errors.

   Test Result: Pass

7. **Model Layers:**

   Test Case: Ensure the CNN model has the correct number and types of layers (e.g., convolutional, pooling, fully connected).

   Expected Result: The model architecture matches the design specifications.

   Actual output: The model architecture matches the design specifications.

   Test Result: Pass

8. **Training Execution:**

   Test Case: Verify that the model trains without errors for a specified number of epochs.

   Expected Result: The training process completes successfully.

   Actual output: The training process completes successfully.

   Test Result: Pass.

9. **Overfitting Detection:**

   Test Case: Check if the model shows signs of overfitting (e.g., training accuracy much higher than validation accuracy).

   Expected Result: The training and validation accuracies should be reasonably close, indicating a balanced model.

   Actual output: The training and validation accuracies should be reasonably close, indicating a balanced model.

   Test Result: Pass

10. **Accuracy Evaluation:**

    Test Case: Validate the model's accuracy on the test dataset.

    Expected Result: The accuracy meets its threshold (e.g., 80%).

    Actual output: The accuracy meets or exceeds a predefined threshold (e.g., 82%)

    Test Result: Pass.

11. Precision and Recall:

Test Case: Calculate the precision and recall for each class.

Expected Result: Precision and recall should be high, indicating that the model is correctly identifying positive cases and minimizing false negatives.

Actual Result: Precision and recall should be high, indicating that the model is correctly identifying positive cases and minimizing false negatives.

Test Result: Pass.

12. F1 Score:

Test Case: Compute the F1 score for the model.

Expected Result: The F1 score should be high, balancing both precision and recall.

Actual Result: The F1 score should be high, balancing both precision and recall.

Test Result: Pass.

8. Result Analysis and Conclusion

8.1 Result & Snapshot of Work Done:

- **Classes, length & Shape Of Dataset**

```
class_names = dataset.class_names
class_names
```

```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy', 'models']
```

```
len(dataset)
```

```
46
```

```
for image_batch , label_batch in dataset.take(1):
  print(image_batch.shape)
  print(label_batch.numpy())
```

```
(20, 256, 256, 3)
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

- **Created Our Image In 3D Array Form**

```
for image_batch , label_batch in dataset.take(1):
  print(image_batch[0].numpy())
```

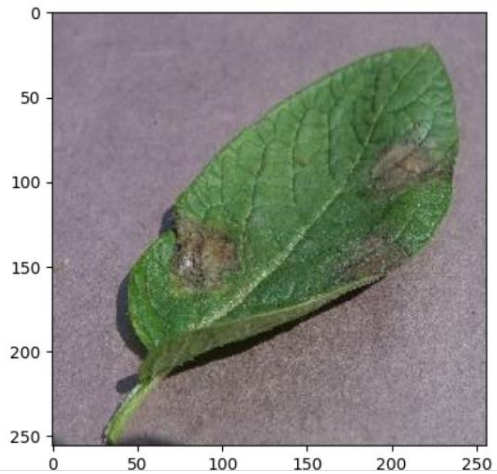
```
[[[130. 119. 125.]
  [138. 127. 133.]
  [130. 119. 125.]
  ...
  [169. 159. 167.]
  [175. 165. 173.]
  [180. 170. 178.]]

 [[134. 123. 129.]
  [140. 129. 135.]
  [126. 115. 121.]
  ...
  [164. 154. 162.]
  [165. 155. 163.]
  [170. 160. 168.]]

 [[134. 123. 129.]
  [142. 131. 137.]
  [125. 114. 120.]
  ...
  [170. 160. 168.]
  [171. 161. 169.]
  [175. 165. 173.]]
```

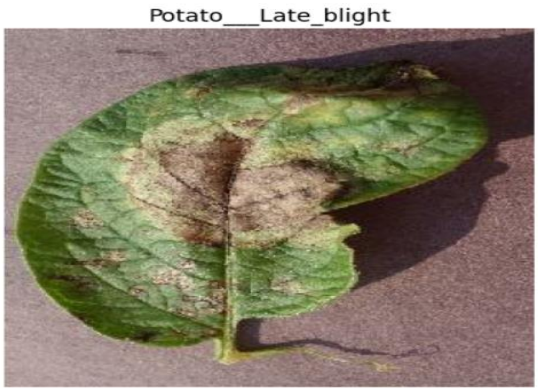
- **Image Show**

```
for image_batch , label_batch in dataset.take(1):
  plt.imshow(image_batch[0].numpy().astype("uint8"))
```



- **Show Image With There Type**

```
for image_batch , label_batch in dataset.take(1):
  for i in range (12):
    plt.imshow(image_batch[0].numpy().astype("uint8"))
    plt.title(class_names[label_batch[0]])
    plt.axis("off")
```



- **Model Building**

```
input_shape= (BATCH_SIZE , IMAGE_SIZE, IMAGE_SIZE ,chanels)
n_classes = 3
model  = models.Sequential([
    resize_and_rescale,
    data_augmentation,
    layers.Conv2D(32,kernel_size=(3,3), activation ='relu',input_shape=input_shape),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size=(3,3), activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size=(3,3), activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size=(3,3), activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size=(3,3), activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64,kernel_size=(3,3), activation ='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64, activation = 'relu'),
    layers.Dense(n_classes, activation = 'softmax'),
])
model.build(input_shape=input_shape)
```

- **History Of epochs:**

```
[ ]  history.params
```

```
{'verbose': 1, 'epochs': 30, 'steps': 36}
```

```
[ ]  history.history.keys()
```

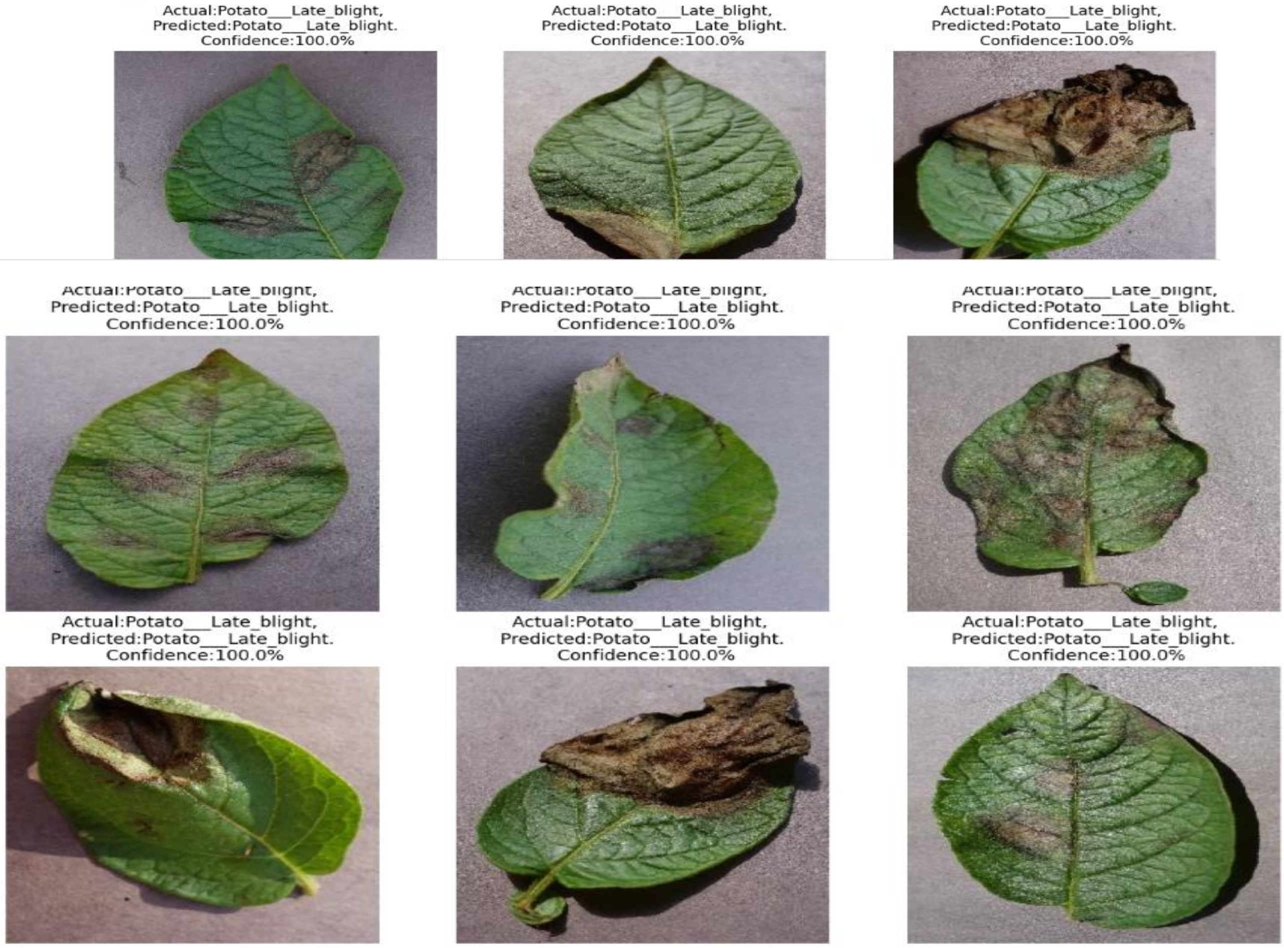
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

Final Result:

```
plt.figure(figsize=(15,15))
for image,labels in test_ds.take(1):
    for i in range(9):
      ax = plt.subplot(3,3 , i+1)
      plt.imshow(image[i].numpy().astype("uint8"))
      predicted_class , confidence = predict(model , image[i].numpy())

      actual_class = class_names[labels[i]]
      plt.title(f"Actual:{actual_class}, \n Predicted:{predicted_class}. \n Confidence:{
      plt.axis("off")
```

```
1/1 [==============================] - 0s 382ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
```



8.4 Conclusion:

In this project, with the help of deep learning techniques and convolution neural network classification-based approach is proposed to detect the late blight, early blight and healthy leaf images of potato plant. We found that CNN is the best way to perform this type of classification object. This model gains 100% of validation accuracy. We think this type of project will play a vital role in our agriculture sector. Most of the farmers of the village in India are not literate and they don't know about the disease properly.

8.5 Future Scope

In future, our aim is to create an android application that can detect the disease of every types of crop and can provide the proper solution for those diseases of the crop. And also, by increasing our database, we will able to get better accuracy. By building an android app we will continue the development process. And we will create a system where the farmers of India can easily get instant service and advice on their problem by detecting the disease. The whole project can be put up on the internet and user can simply sit at home and use the system to detect the disease and spray the required disinfectant.

9. References

Plant Disease Detection Using Deep Learning and Convolutional Neural Networks

- Authors: Piyush Dhiman, Priya Kalra, et al.
- Year: 2020
- DOI: 10.1109/Confluence47617.2020.9058032

Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network

- Authors: Sundararajan Sellamuthu, Marimuthu Palanisamy, et al.
- Year: 2019
- DOI: 10.1016/j.biosystemseng.2019.01.002

Transfer Learning for Image-Based Plant Disease Detection

- Authors: Sharada P. Mohanty, David P. Hughes, et al.
- Year: 2016
- DOI: 10.3389/fpls.2016.01419

Image-Based Plant Disease Detection Using Deep Learning

- Authors: A. Durmuş, E. Güneş, and M. Kırcı
- Year: 2017
- DOI: 10.1016/j.procs.2017.04.031

Deep Learning Approaches for Agricultural Applications

- Authors: A. Kamilaris, F.X. Prenafeta-Boldú
- Year: 2018
- DOI: 10.1016/j.compag.2018.05.018

Convolutional Neural Networks for Crop Disease Detection

- Authors: Ramcharan, Abhinav, et al.
- Year: 2017
- DOI: 10.1016/j.compag.2017.09.017