Department of Electrical and Computer Engineering
North South University

CSE 498R
Project Report

AI-driven Plant Disease Diagnosis for Sustainable Agriculture

Section: 27, 07, 27,12
Team Number: 19
Group Members

Obantika Roy Anti	**ID# 1921383042**
Fabiha Tazri Okita	**ID# 1922086042**
Sadia Sultana	**ID# 1921218642**
Md. Samin Ahmed	**ID# 1911998642**

Faculty Advisor:

DR. DIHAN MD. NURUDDIN HASAN
Assistant Professor

PhD, Dept. of Electrical and Computer Engineering, National University of Singapore B. Sc. in Electrical & Electronic Engineering, Bangladesh University of Engineering and Technology

Submission Date:
30 October,2023

Department of Electrical and Computer Engineering

North South University
Summer 2023

Table of Contents

1. Introduction

Agriculture is vital for sustaining our growing global population. However, plant diseases pose a significant threat to crop yields and food security. This abstract presents an innovative approach to address the challenge of AI-driven plant disease diagnosis for pepper bell plant leaf, by offering a sustainable solution for the agricultural sector. Recent developments in artificial intelligence, notably in computer vision and machine learning, have made it possible to create reliable and effective systems for the early identification and diagnosis of plant diseases. These systems use image recognition algorithms to swiftly and accurately identify disease symptoms in plants, enabling prompt intervention and treatment.

Our work makes use of a sizable dataset pepper bell plant leaf images based on healthy and spotted. On the basis of this dataset, deep learning models, such as convolutional neural networks, are trained to identify image segmentation and object detection methods may also be utilized to identify specific regions of interest in the leaf images. This paper explores the key components and methodologies that involve data collection, preprocessing, and the exploration of various deep learning techniques like transfer learning and data- augmentation to build and train an accurate disease detection model. The results of our tests indicate that the proposed AI-driven approach for pepper bell plant leaf disease diagnosis in sustainable agriculture is highly effective. Our models exhibit a remarkable ability to differentiate between pepper bell plant's leaf affected by diseases and those that are not, achieving exceptional levels of accuracy, sensitivity, and specificity. This research has the potential to significantly influence the field of agriculture and sustainable farming. Our method empowers farmers and agricultural professionals to swiftly and precisely identify plant diseases, leading to plant disease detection. This, in turn, can contribute to healthier crops, increased yields, and a more sustainable agricultural system.

In conclusion, our AI-driven plant disease diagnosis method for pepper bell plant and other plant has the capacity to revolutionize agricultural practices, ensuring healthier crops and more efficient resource utilization. This research contributes to the advancement of sustainable agriculture and underscores the potential for mitigating crop diseases and enhancing food security.

2. Problem Statement

Agriculture is a cornerstone of human civilization, providing food, fibers, and raw materials essential for our survival and prosperity. However, the agricultural sector faces numerous challenges, and one of the most pressing is the impact of plant diseases on crop health and yield. Plant diseases can have detrimental effects on crop yields, and identifying them early is critical for effective management. Traditional methods rely on visual inspection, which is time-consuming and may not always yield accurate results. Farmers often struggle with diagnosing plant diseases, and a misdiagnosis can lead to the improper use of pesticides or other treatments, which can be costly and harmful to the environment. Our goal is to create a system that assists farmers in quickly and accurately diagnosing pepper bell plant leaf diseases to ensure sustainable agriculture

practices. We considered various solutions, including manual inspection and traditional diagnostic methods, but these can be labor-intensive and prone to human error. Therefore, we are focusing on an AI-driven pepper bell disease diagnosis system for commercial implementation. Farmers can use their smartphones or dedicated devices to capture images of the affected plants. This can be done in the field or the greenhouse.

Pepper Bell, a popular and versatile vegetable, are cultivated in various regions globally. They are susceptible to a range of diseases and pests that can severely impact crop yield and quality. Timely and accurate disease detection is essential to minimize the economic losses faced by farmers and ensure a sustainable supply of pepper bell. Traditional methods of disease identification are often time-consuming and labor-intensive, making them inefficient for addressing this agricultural challenge. To safeguard pepper bell crops and support sustainable agriculture, there is a need for a technological solution that leverages deep learning to detect diseases in pepper bells efficiently.

Pepper bell plants are susceptible to a variety of diseases, which can lead to significant crop losses and economic damage. Early detection and treatment of diseases is essential for minimizing crop losses. However, traditional methods of disease detection, such as visual inspection by farmers, are often inaccurate and time-consuming. The delay in disease detection can lead to the unchecked spread of diseases, impacting crop yield, quality, and the livelihood of farmers. Skilled personnel and extensive data collection are required for accurate disease identification, making this process expensive and often inaccessible for small-scale farmers. The objective of this project is to develop a deep learning-based system for the early and accurate detection of diseases in pepper bell plants. The system should analyze images of pepper bell plants and identify disease symptoms, enabling timely intervention and improved crop management. The proposed solution involves the development of a deep learning model trained on a diverse dataset of pepper bell plant images. This model will be integrated into a user-friendly system for field application. Creating a comprehensive dataset of pepper bell plant images, covering various growth, varieties, and disease states, to train the deep learning model effectively. Ensuring high accuracy and reliability in disease detection, even when faced with variations in lighting, environmental conditions, and plant morphology.

Designing a system that can be used in the field by farmers, considering factors like data collection, image processing, and model deployment. Developing a system that can analyze images in real-time, providing immediate disease detection results to guide agricultural practices. Many pepper bell farmers have limited technical expertise and may not be able to use complex disease detection systems. There is a need for a disease detection solution that is easy to use and accessible to all farmers.

3. Proposed Solution

The proposed autonomous system's goal is to create a complete AI-driven system for plant disease that is pepper bell plant disease diagnosis in sustainable agriculture is to safeguard crops, secure food supplies, and elevate the overall quality of life for individuals reliant on agriculture. Detection of plant diseases is crucial for effective management and the prevention of yield losses. The proposed system can assist in adapting to these challenges by offering disease warnings and data-driven insights for optimized resource utilization. The motivation behind AI-driven plant disease diagnosis for pepper bell plant in sustainable agriculture is deeply rooted in the pursuit of food security, environmental sustainability, and the well-being of farmers and consumers. The proposed solution involves the development of a deep learning model trained on a diverse dataset of pepper bell plant images. This model will be integrated into a user-friendly system for field application.

Pepper bell plant disease detection using deep learning offers a number of benefits, including:

- **Detection of disease:** Deep learning models can be trained to detect pepper bell plant leaf diseases. This allows farmers to take corrective action quickly, preventing the disease from spreading and causing significant damage to the crop.
- **Checking accuracy:** Deep learning models can achieve very high accuracy in detecting pepper bell plant diseases, even in complex and challenging conditions. This is because they are able to learn subtle patterns and features in images that are difficult for humans to identify.
- **Scalability:** Deep learning models can be easily scaled to handle large volumes of data, making them ideal for use in large-scale agricultural operations.
- **Affordability:** The cost of deep learning technology has decreased significantly in recent years, making it more accessible to farmers of all sizes.
- **Speed:** Deep learning models can process images quickly, enabling real-time disease

detection.

The proposed solution for bell pepper disease detection using deep learning involves the following steps:

1. Data collection: A large and diverse dataset of pepper bell images, both healthy and diseased, needs to be collected. The dataset should be representative of the different types of diseases that can affect pepper bells, and should also include images of the different parts of the plant, such as the leaves, stems, and fruits.
2. Data preprocessing: The collected images need to be preprocessed to ensure that they are consistent in size, format, and quality. This may involve resizing the images, converting them to grayscale, and normalizing the pixel values.

a. **Data Cleaning:** Remove any irrelevant or low-quality images from the dataset.
b. **Data Augmentation:** Enhance the dataset by applying techniques like rotation, flipping, and resizing to increase its diversity and improve model generalization.

3. Model training: A deep learning model, such as neural networks need to be trained on the preprocessed dataset. The model will learn the following steps to identify the different types of diseases in the images.

a. Split the dataset into training, validation, and testing sets.
b. Train the chosen AI model on the training set using the annotated images.
c. Utilize transfer learning if a pre-trained model is available to improve training efficiency and performance.

4. **Model Evaluation:**
    a. Assess the model's performance using various evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrix on the test dataset.
    b. Perform cross-validation if needed to ensure the model's robustness.
5. **Inference:**

    a. Use the trained model to make predictions on the input images.
    b. Post-process the model's output to provide information on the presence of diseases.

6. **Feedback Loop:**
    a. Continuously collect feedback from users and experts to improve the model's accuracy and usability.
    b. Periodically retrain the model with new data to keep it up-to-date with evolving disease strains and conditions.

Overall, deep learning has the potential to revolutionize the way that pepper bell plant diseases are detected and managed. By enabling early detection and accurate diagnosis, deep learning models can help farmers to reduce crop losses and improve yields.

Here are some specific examples of how pepper bell plant disease detection using deep learning can be used to benefit farmers:

- A farmer can use a deep learning model to inspect their pepper bell plants on a daily basis, even if they have a large field. This would help them to identify any diseased plants, before the disease has a chance to spread.
- A farmer can use a deep learning model to get a second opinion on a diagnosis that they have made. This would help them to be more confident in their treatment decisions.

- A farmer can use a deep learning model to track the spread of a disease over time. This information can be used to develop more effective disease management strategies.

Deep learning is a rapidly developing field, and new applications for this technology are being discovered all the time. It is likely that deep learning will play an even greater role in pepper bell plant disease detection and management in the future.

Our system diagram is followed by following workflow:

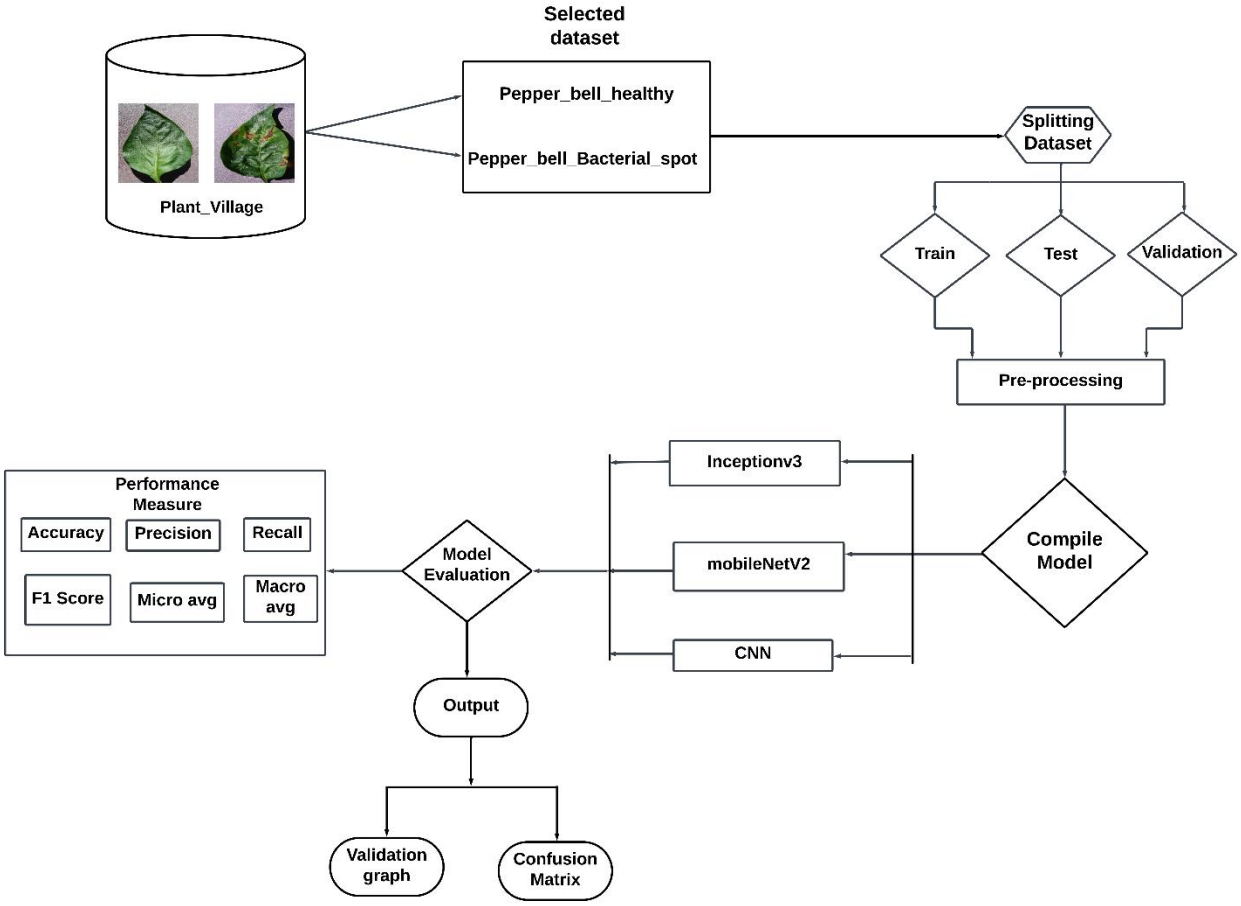


Figure: Flowchart of pepper bell disease detection working process

4. Related Work:

Litterature Review 01:

Title: Leaf Disease Classification in Bell Pepper Plant using VGGNet

This paper is written by: Pranajit Kumar Das, published on ResearchGate on March,2023 can be summarized like this way: -

Dataset: A total of 2475 images, with 1478 healthy images and 997 images with bacterial disease spots, are used for training, validation, and testing purposes.

Methodology: Steps that are followed to classify bacterial spot leaves in bell peppers are as follows:
Image collection, spliting images into training, testing and valiation, then images for training and validation set are used to train and validiate the model and then tested with testing set of images. Thus the work classifies the images into healthy and diseased.

Applied Models and Accuracy: The pre-trained models VGG16 and VGG19 are trained using the training images of the bell pepper plants collected from the Plant Village dataset. The model VGG16 gave 97% accuracy and the model VGG19 gave 96% accuracy.

Remarks: In this paper, transfer learning based VGGNet is applied to classify leaf-based disease classification in bell pepper plants. However, the limitation of having a free dataset is that only two classes in the PlantVillage dataset such as diseased (bacterial spots) class and healthy class are available.

Literature Review 02:

Title: Bell Pepper Leaf Disease Classification Using CNN

Methodology: This paper offers analysis of current plant-based disease detection systems. A number of simulation methodologies for neurons and layers were employed in this study, which used a CNN supplied with a bell pepper plant picture dataset. The suggested method seeks to establish an automated system for vision-based illness identification and classification of specific disorders of the pepper bell leaf by analyzing GA quality using CNN. In this article, CNN was used to identify bacterial spots on bell pepper leaves. (CNNs) are one of the most important types of neural networks for image identification and classification. CNNs are extensively utilized in applications such as object detection, identification, and so on. CNN recognizes things by taking an input picture, processing it, and categorizing it.

Result: The image was analyzed using Keras and the Tensor Flow library. In this study, several hundred leaves were used for testing. The images are taken in a well-lit environment. The algorithm employs neural networks to determine whether a leaf is healthy or infected. The test accuracy was 96.78%.

Remarks:

1. We are thinking of use CNN, InceptionV3, MobileNetV2 algorithms for our deep learning approach
2. Our expected result is 90 to 95% accuracy.

Literature Review 03:

Title: Pepper leaf disease recognition based on enhanced lightweight convolutional neural networks

This paper is written by: Min Dai 1 *, Wenjing Sun1 , Md Mehedi Hassan Dorjoy1 , Hong Miao1,2, Xin Zhang3

Methodology-

- In this study, A novel deep convolutional neural network model, referred to as google net-EL, has been introduced for the precise and efficient detection of multiple diseases affecting pepper plants.
- The paper presents an improved lightweight CNN model designed to accurately identify diseases on pepper leaves. The experiment, they utilized a total of 9183 images, which included 5669 original images of pepper leaf diseases and 3514 preprocessed images with various image augmentations.
- The proposed model offers notable advantages in terms of disease recognition accuracy and computational performance when deployed on resource-constrained computing platforms.

- The experiments were conducted on a Windows desktop system with 16GB of memory, an Intel Core i7-9750H CPU running at GHz, and an NVIDIA GeForce GTX 1650 GPU. To ensure that other parameters did not interfere with the model performance test, we initially trained GoogLeNet-EL using different learning rates, specifically 0.01, 0.001, and 0.0001. After 100 epochs, the final accuracies achieved were 63.68%, 97.87%, and 94.72%, respectively. As a result, a uniform learning rate of 0.001 was chosen for subsequent testing.

Result:

- This paper introduces a convolutional neural network model, known as GoogLeNet-EL, specifically designed for accurate and easily transferable identification of pepper leaf diseases.
- By optimizing the network depth and width of the Inception module, this model achieves a significant reduction in memory requirements, reducing them by 52.31% and 86.69% when compared to GoogLeNet based on Inception-V1 and Inception-V3, respectively.
- Experimental findings indicate that the selection of the LeakyReLU activation function, batch normalization algorithm, and SGDM optimizer provides the best performance for the GoogLeNet-EL model, resulting in a superior model fitting effect.
- The introduction of spatial pyramid pooling enhances the model's feature learning capabilities, boosting the recognition accuracy to 97.87%, a notable improvement of over 6% compared to Google Net.

- In real-world scenarios, a comparison with mainstream models such as AlexNet, ResNet-50, and MobileNet-V2 demonstrates that the proposed model offers significant advantages. It exhibits a 61.49%, 41.78%, and 23.81% reduction in average testing time, respectively.
- Furthermore, the accuracy, recall, and F1 values of the model are approximately 99%, which significantly surpasses those of other network models. These results underscore the enhanced lightweight model's remarkable advantages in terms of recognition accuracy and computational performance for diagnosing pepper leaf diseases, particularly on resource-constrained computing platforms. This enhancement makes it well-suited for large-scale deployment in pepper plantations.

Literature review : 04

Title: Recent advances on postharvest technologies of bell pepper: A review Quazeem Omotoyosi Tiamiyu a , Segun Emmanuel Adebayo a,* , Nimat Ibrahim b

Bell peppers (Capsicum annuum L.) are economically vital crops grown in tropical and sub-tropical regions. However, their susceptibility to diseases and limited shelf life result in substantial postharvest losses, estimated at 40% annually. Traditional chemical fumigation, while effective against fungal infections, has associated environmental and health concerns. This review delves into emerging non-chemical strategies for mitigating postharvest losses in bell peppers, outlining their mechanisms, benefits, limitations, and recommendations for practical application.

Traditional Chemical Fumigation:

1. The use of chemicals for disease control.
2. Efficient against fungal infections.
3. Environmental and health risks limit its use.

Biological Methods:

1. Employing beneficial microorganisms for disease management.
2. Environmentally friendly and promising for loss reduction.

Botanical Methods:

1. Utilizing natural compounds, such as essential oils, for disease control.
2. Safer alternative to chemicals, with proven efficacy.

Mechanisms and Benefits:

1. Methods work by inhibiting disease growth, enabling early intervention, and reducing losses. They also minimize environmental and health risks.

Current Limitations:

1. Biological and botanical methods may require further research and standardization. Non-destructive technologies and AI might be cost-prohibitive for small-scale farmers.

Recommendations for Potential Applications:

1. Suggested actions for future use, including increased research for biological and botanical methods, raising awareness, and developing cost-effective solutions for small-scale farmers.

5. Technical Approach

In recent years, advancements in computer vision and deep learning have provided a transformative solution to the challenges posed by traditional crop disease detection methods. Deep learning, a subset of artificial intelligence, leverages neural networks to analyze images and identify patterns, making it well-suited for the task of disease detection in pepper bell plants. The system can learn to recognize disease symptoms, even in complex and varying environmental conditions. The technical approach for disease detection in pepper bell using deep learning is rooted in the following key components:

1. Image Dataset Collection:

The foundation of our approach involves the creation of a comprehensive and diverse dataset of pepper bell plant images. This dataset encompasses various growth stages, pepper bell varieties, and disease states. The dataset's diversity is crucial for training the deep learning model to accurately recognize a wide range of disease symptoms.

2. Deep Learning Model Development:

We will develop a deep convolutional neural network such as- InceptionV3, mobilenetV2 and CNN architecture tailored for disease detection in pepper bells. These models will undergo training on the image dataset to learn the visual cues and patterns associated with various crop diseases. The deep learning model will be optimized to achieve high accuracy and robustness, even when faced with environmental variations.

3. Real-Time Analysis and Inference:

Our system aims to provide real-time disease detection capabilities, which are vital for on-the-spot decision-making by farmers. The deep learning model, once trained, will be deployed to analyze images in real-time, enabling immediate disease detection results to guide agricultural practices.

To apply models on our applied dataset supportive libraries has been imported. Those libraries are-

1. os: The "os" library in Python provides a way to interact with the operating system. It allows you to perform various tasks related to file and directory manipulation, such as creating, deleting, renaming, or checking the existence of files and directories. It also provides functions to work with paths, environment variables, and execute system commands.
2. cv2: "cv2" refers to OpenCV, which stands for Open-Source Computer Vision Library. OpenCV is a popular computer vision library that provides a wide range of functions and algorithms for image and video processing. It offers tools for image manipulation, feature detection, object recognition, and various other computer vision tasks
3. numpy: NumPy is a fundamental library for numerical computing in Python. It provides a powerful N-dimensional array object that enables efficient manipulation of large, multi-

dimensional arrays and matrices. It is extensively used in scientific computing, data analysis, and machine learning applications.

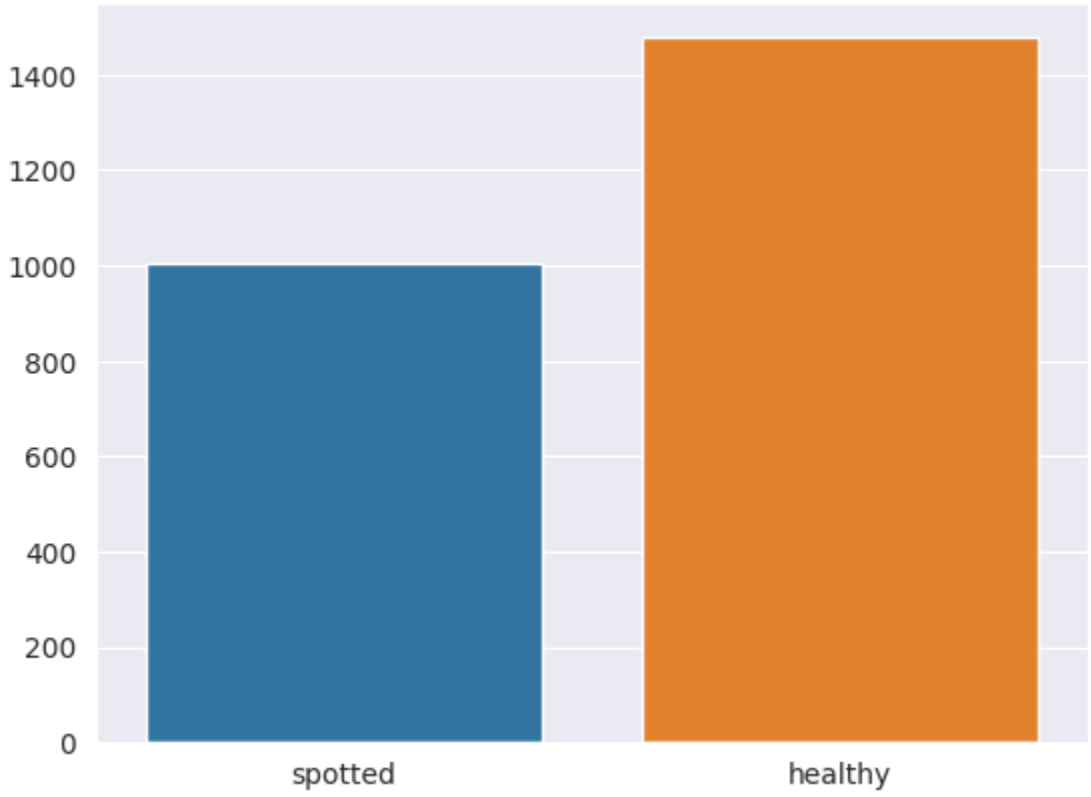
4. sklearn model selection: The "sklearn" library, also known as scikit-learn, is a popular machine learning library in Python. It provides a comprehensive set of tools for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, and model selection. The "model_selection" module within scikit-learn offers functions and classes for model selection and evaluation, such as splitting datasets into training and test sets, cross-validation, hyperparameter tuning, and performance metrics computation.

5. keras: Keras is a high-level deep learning library that runs on top of other deep learning frameworks, such as TensorFlow or Theano. It provides a user-friendly and intuitive interface for building and training neural networks. It offers a wide range of pre-processing and data augmentation utilities, as well as support for GPU acceleration. Keras simplifies the process of designing and training deep learning models and has gained popularity for its ease of use and flexibility.

6. tensorflow: TensorFlow is an open-source library for machine learning and numerical computation developed by the Google Brain team. It is one of the most popular and widely used frameworks for building and deploying machine learning models. TensorFlow provides a comprehensive ecosystem of tools, libraries, and resources that aid in the development of various AI applications.

```
import matplotlib.pyplot as plt
import numpy as np
import os
import shutil
from tensorflow import keras
import seaborn as sns
import random
from keras.models import load_model
from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import cv2
import pandas as pd
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sb
import tensorflow as tf
from sklearn.metrics import classification_report,confusion_matrix
import seaborn as sb
```

Data Augmentation and Preprocessing:
Total images = 2
Total number of classes = 2
Total spotted images = 1004
Total healthy images: 1478



For data augmentation we have used some methods like-

Rescale: Rescaling is an augmentation method that involves adjusting the scale or intensity of an image. It is commonly used to normalize the pixel values within a specific range, such as rescaling the pixel values from the original range of 0-255 to a new range of 0-1.

Zoom: Zoom augmentation involves magnifying or shrinking a portion of an image, giving the appearance of zooming in or out. This augmentation technique can be useful for simulating different viewpoints or capturing details at different scales. It helps to introduce variability and increase the robustness of a machine learning model by training it on images with varying zoom levels.

Flip: Flip augmentation involves flipping an image horizontally or vertically. Horizontal flipping involves reversing the order of pixels in each row, while vertical flipping reverses the order of rows in the image. Flipping is a common augmentation technique used to create variations of an image, especially when the orientation of the object is not crucial. It helps in training models that need to be invariant to horizontal or vertical flips, increasing the generalization capability of the model.

ImageDataGenerator(): Keras ImageDataGenerator allows augment images in real-time while model is still training. Any random transformations on each training image as it is passed to the model. Main methods used in this utility are:

- rotation_range: Degree range for random rotations.
- width_shift_range and height_shift_range: Fraction of total width or height by which the image can be shifted.
- shear_range: Shear angle in radians.
- zoom_range: Range for random zoom.
- horizontal_flip and vertical_flip: Boolean indicating whether to perform random horizontal and vertical flips.
  - ImageDataGenerator().flow()
  - flow_from_directory()

A total of 2475 images are used for training, validation, and testing purposes, with 1478 healthy images and 997 images with bacterial disease spots. We split the dataset into train, test, and validation. The ratio of splitting is 80%, 10% and 10%.

We used three models for pepper bell disease detection.
Those are-

InceptionV3:

1.InceptionV3 is a CNN architecture developed by Google.

2. It utilizes inception modules, which consist of multiple parallel convolutional layers with different filter sizes.

3.These modules allow the network to capture features at various scales and resolutions.

4.InceptionV3 excels in image recognition tasks and is known for its accuracy and efficiency.

Here is the code of applying InceptionV3 by importing libraries in the Google Colab:

```
from keras.applications import InceptionV3
from keras.models import Sequential
from keras.layers import GlobalAveragePooling2D, Dense, Dropout, BatchNormalization
from keras.callbacks import ModelCheckpoint

keras.backend.clear_session()

# Load pre-trained InceptionV3 model
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(256, 256, 3))

# Add custom top layers
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(2, activation='softmax'))

# Set the first layers to non-trainable (optional)
for layer in model.layers[0].layers:
    layer.trainable = False

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Define a ModelCheckpoint callback
checkpoint = ModelCheckpoint('best_weights.h5', save_best_only=True, save_weights_only=True,
monitor='val_loss', mode='min', verbose=1)

# Train the model with the callback

inception = model.fit(train_ds, epochs= 100,steps_per_epoch = int(round(1032/32)),
 validation_data=val_ds,validation_steps = int(round(128/32)), callbacks=[checkpoint])

# Load the best weights
model.load_weights('best_weights.h5')

# Summary of the model
model.summary()
```

MobileNetV2:

1.MobileNetV2 is a CNN architecture optimized for mobile and embedded devices.
2.It employs depth wise separable convolutions to reduce computational complexity.

3. This model strikes a balance between accuracy and model size, making it suitable for resource-constrained environments.

4. MobileNetV2 is used for real-time image processing on mobile devices, including image classification and object detection tasks.

Here is the code of applying MobileNetV2 by importing libraries in the Google Colab:

```
checkpoint = tf.keras.callbacks.ModelCheckpoint('best_weights_MobileNetV2.h5', save_best_only=True, save_weights_only=True, monitor='val_accuracy', mode='max', verbose=1)
```

```
from keras.applications import MobileNetV2
from keras.models import Sequential
from keras.layers import GlobalAveragePooling2D, Dense, Dropout, BatchNormalization
keras.backend.clear_session()
# Load pre-trained MobileNetV2 model
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(256, 256, 3))

# Add custom top layers
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(2, activation='softmax'))

# Set the first layers to non-trainable (optional)
for layer in model.layers[0].layers:
    layer.trainable = False

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Summary of the model
model.summary()
```

Convolutional Neural Network (CNN):

CNN is a class of deep neural networks designed for image processing tasks.

1.It uses convolutional layers to automatically extract features from input images.
2.Pooling layers reduce spatial dimensions while retaining important information.
3.Activation functions introduce non-linearity for capturing complex relationships.
4.Fully connected layers make predictions based on learned features.
5.Widely used in image classification, object detection, and image segmentation.

Here is the code of applying CNN Model by importing libraries in the Google Colab:

```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense,Dropout
from keras.layers import BatchNormalization

# Initialising the CNN
classifier = Sequential()
# Step 1 - Adding Convolution layer
classifier.add(Conv2D(32, (3, 3), input_shape = (256,256, 3), activation = 'relu'))

# Step 2 - Adding MaxPooling layers
classifier.add(MaxPooling2D(pool_size = (2, 2)))
# Adding a second convolutional layer
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Step 3 - Flattening
classifier.add(Flatten())

# Step 4 - Full connection
classifier.add(Dense(units = 512, activation = 'relu'))
classifier.add(BatchNormalization()),
classifier.add(Dense(256,activation='relu')),
classifier.add(Dropout(0.25)),
classifier.add(Dense(units = 2, activation = 'softmax'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

So, these are the codes of our applied three models.

In summary, CNNs are the foundation of deep learning for image-related tasks. InceptionV3 is a powerful image recognition architecture with multiple inception modules, while MobileNetV2 is designed for efficiency and is suitable for deployment on mobile and embedded devices.

For each model, we used the following codes to find accuracy, confusion matrix, heatmap and test score:

The code for getting the score of accuracy-

```
score=model.evaluate(test_ds)
print("Loss:",score[0],"Accuracy:",score[1])
```

The code for getting the validation graph –

```
plot_train_history(model_info)
```

The code for getting the getting the value of confusion matrix-

```
pred= np.round(model.predict(test_ds, verbose=1))
test_labels=test_ds.labels
test_pred_labels=[]
for i in range(len(pred)):
  test_pred_labels.append(np.argmax(pred[i]))
conf_matrix= confusion_matrix(test_pred_labels,test_labels)
print (conf_matrix)
```

The code for getting the heatmap of Confusion Matrix-

```
sb.heatmap(conf_matrix,cmap='Purples',
annot=True,xticklabels=['spotted','healthy'],yticklabels=['spotted','healthy'],linewidths=1,
            linecolor='green').plot()
plt.show()
```

The code for getting the test report of having accuracy, precision ,recall, macro avg, micro avg-

```
test_report = classification_report(test_ds.labels,test_pred_labels, target_names=['spotted','healthy'],
output_dict=True)
test_df = pd.DataFrame(test_report).transpose()
test_df
```

6. Technology Platform

Inception V3:

The Inception V3 architecture is made up of a number of modules, each of which consists of a combination of convolution, pooling, and activation layers. The modules are arranged in a hierarchical fashion, with each module extracting features at a higher level of abstraction.

The following is a brief description of each of the main components of the Inception V3 architecture:

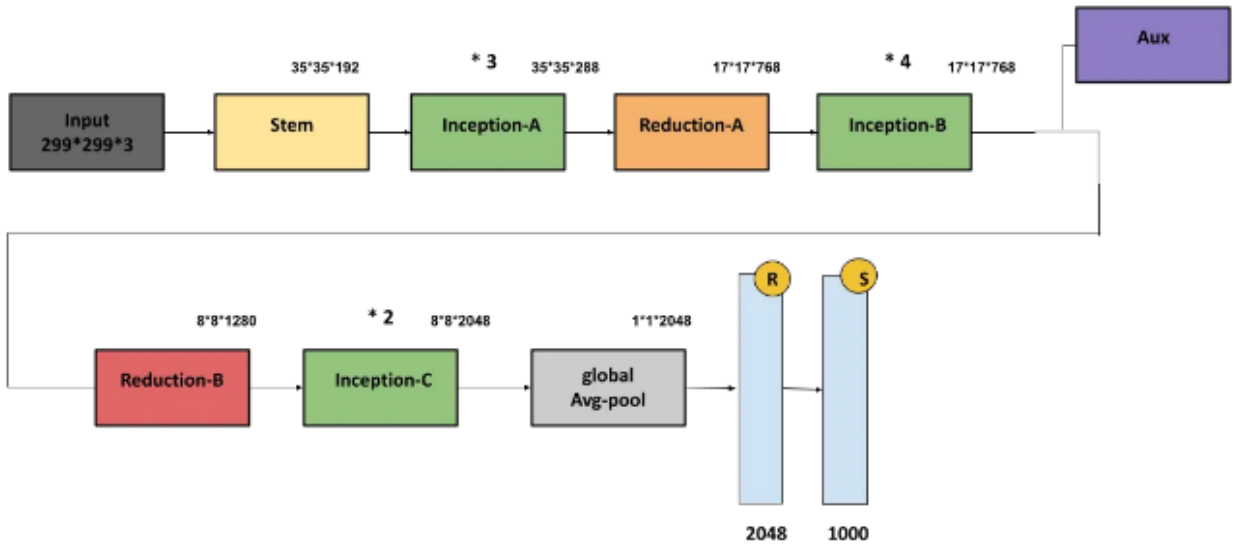
Stem Block: This block is responsible for reducing the input image size to 32x32 pixels.
Inception Blocks: These blocks are the core of the Inception V3 architecture and are responsible for extracting features from the image.
Reduction Blocks: These blocks are used to reduce the spatial dimensions of the feature maps, which helps to reduce the computational complexity of the network.

Average Pooling Layer: This layer is used to convert the feature maps into a single vector.
Fully Connected Layer: This layer is used to classify the image into one of the target categories.

The Inception V3 architecture is a very powerful and accurate CNN architecture. It has been shown to achieve state-of-the-art results on a number of image classification benchmarks.

The steps are-

- The input image is passed through the Stem Block, which reduces the image size to 32x32 pixels.
- The image is then passed through a series of Inception Blocks, which extract features from the image at different levels of abstraction.
- After each Inception Block, there is a Reduction Block, which reduces the spatial dimensions of the feature maps.
- After the final Inception Block, there is an Average Pooling Layer, which converts the feature maps into a single vector.
- Finally, the vector is passed through a Fully Connected Layer, which classifies the image into one of the target categories.



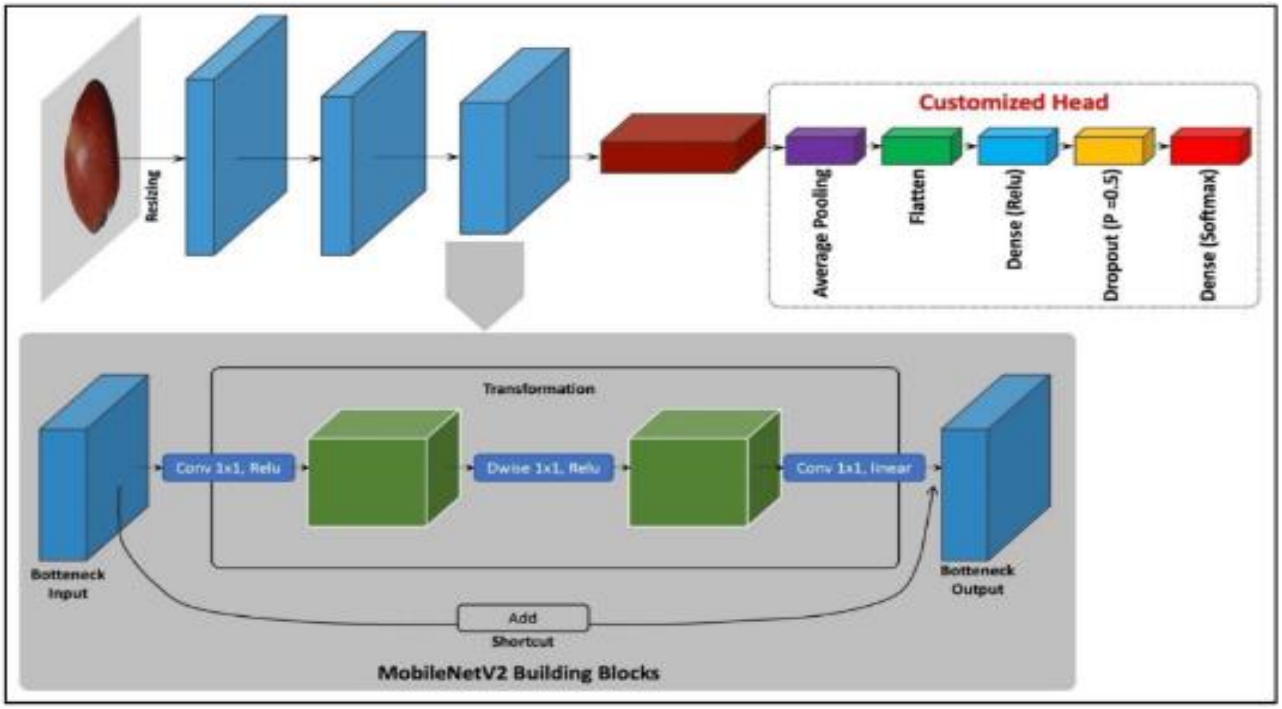
Inception V3

MobileNetV2:

MobileNetV2 is a modified version of the MobileNetV2 architecture that has been specifically designed for image classification tasks.

MobileNetV2 makes use of a number of architectural innovations, including:

Customized head: MobileNetV2 replaces the final fully connected layer of the MobileNetV2 architecture with a customized head that is tailored to the specific image classification task at hand.



Transfer learning: MobileNetV2 is typically pre-trained on a large dataset of general-purpose images, such as ImageNet. This allows the model to learn a good set of general-purpose features, which can then be fine-tuned for the specific image classification task at hand.

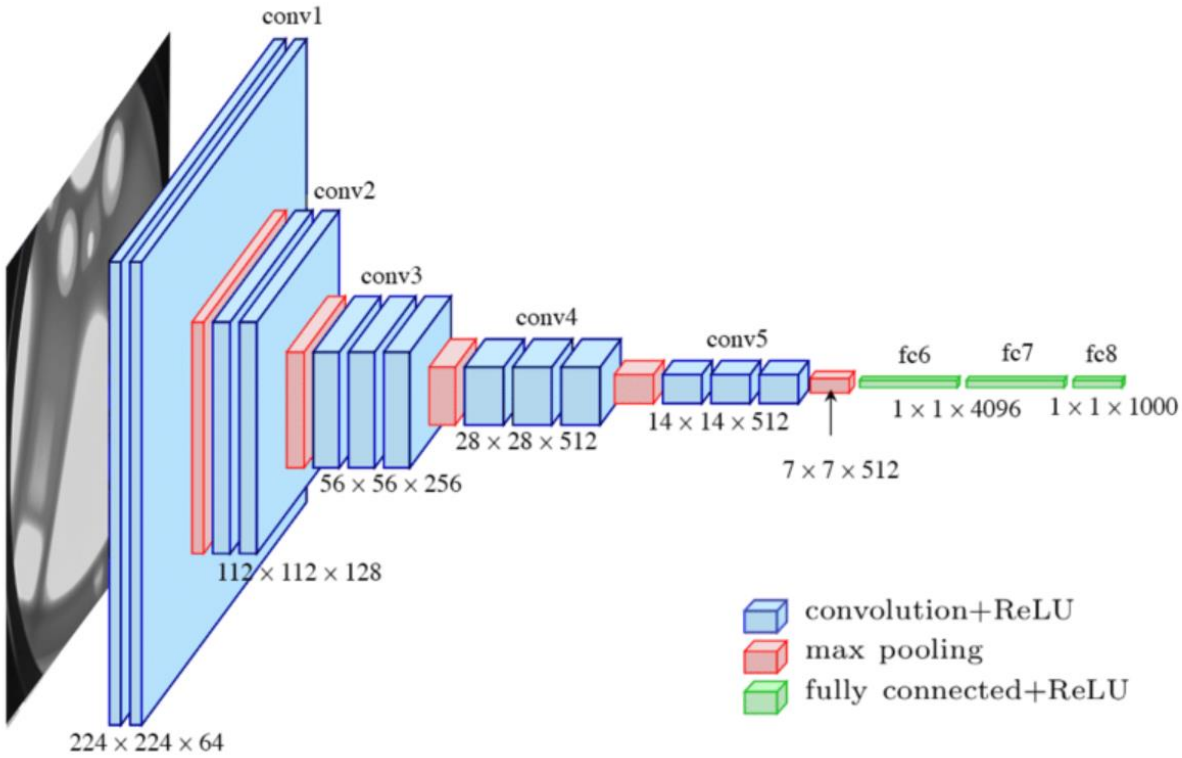
The MobileNetV2 architecture is made up of a number of stacked inverted residual blocks. The first few blocks extract low-level features, such as edges and corners. The later blocks extract higher-level features, such as objects and scenes.

The output of the final inverted residual block is fed into a customized head, which consists of a number of fully connected layers. The customized head is trained to classify the image into one of the target categories.

The following is a brief description of each of the main components of the MobileNetV2 architecture:

- **Butteneck Input:** This layer reduces the dimensionality of the input image.
- **MobileNetV2 Building Blocks:** These blocks are the core of the MobileNetV2 architecture and are responsible for extracting features from the image.
- **Add:** This layer combines the output of the MobileNetV2 Building Blocks with the input image.
- **Average Pooling:** This layer converts the feature maps into a single vector.

- **Dense (Relu):** This layer is a fully connected layer that uses the rectified linear unit (ReLU) activation function.
- **Flatten:** This layer converts the 2D feature maps into a 1D vector.
- **Dropout (P=0.5):** This layer randomly drops out 50% of the neurons in the network, which helps to prevent overfitting.
- **Dense (Softmax):** This layer is a fully connected layer that uses the softmax activation function to classify the image into one of the target categories.



CNN Model:

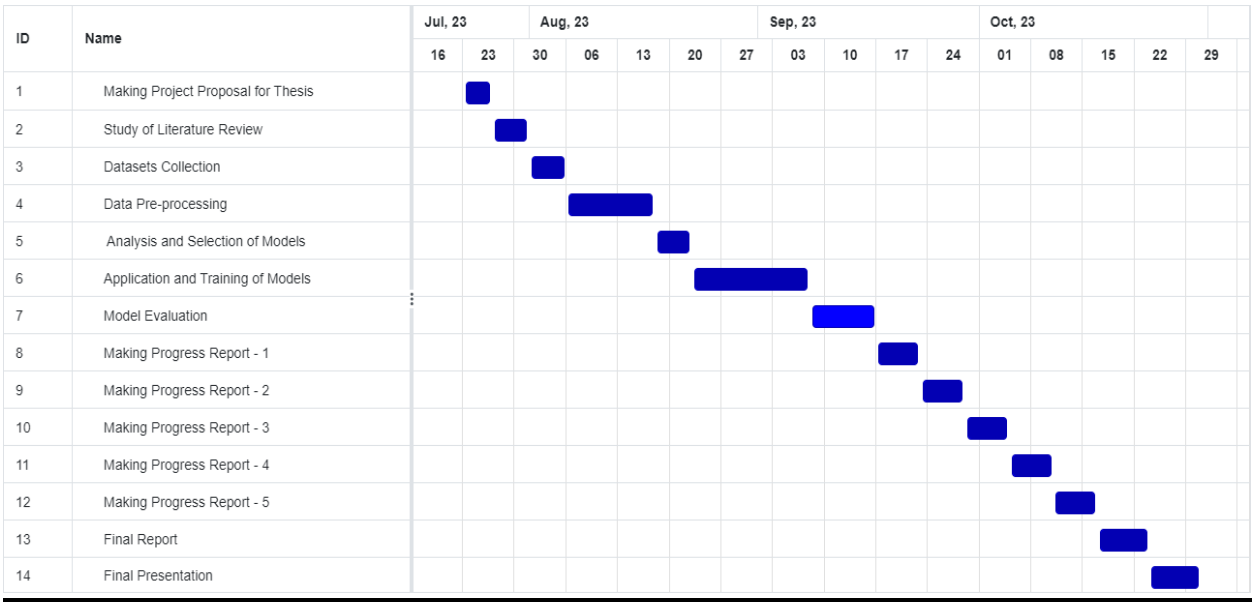
CNN architecture in deep learning is a type of neural network architecture that is specifically designed for image processing and classification tasks. CNNs are trained using a supervised learning approach. This means that the network is given a set of labeled training images, and it learns to predict the correct label for each image.

CNNs are made up of a series of layers, each of which performs a specific operation on the input image. The most common layers in CNNs are:

- **Convolutional layers:** These layers apply a set of filters to the input image, each of which detects a specific feature, such as edges, corners, or textures.

- **Pooling Layer (Subsampling or Max-Pooling):** Pooling layers are produced by convolution layers. They reduce the spatial dimensions of the feature maps, which helps reduce computational complexity and retains the most important information. Max-pooling, for instance, retains the maximum value within a small region of the feature map.
- **Fully connected layers:** These layers are used to classify the output of the pooling layers into one of the target categories.
- **Activation Function (ReLU):** After each convolution operation, an activation function, usually ReLU (Rectified Linear Unit), is applied element-wise. ReLU introduces non-linearity to the network, allowing it to learn complex relationships in the data.

7. Timeline:

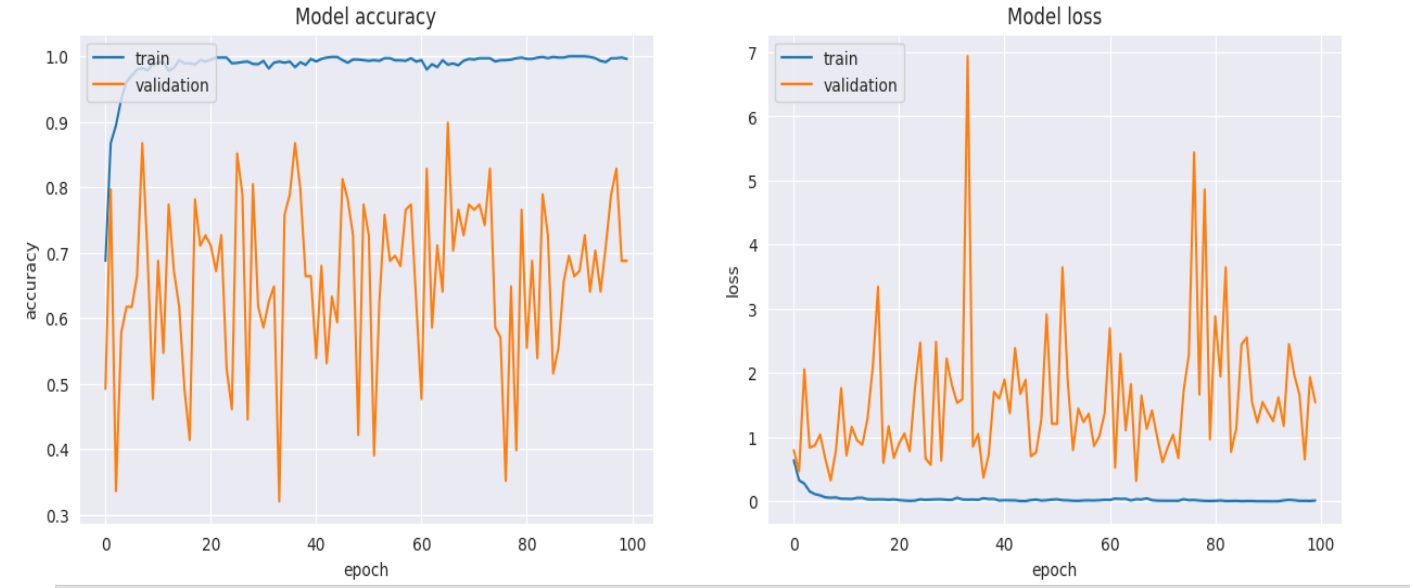


The project began on July 30, 2023, and is expected to be completed on October 29, 2023. We started our work by making a project proposal for our thesis. This stage involves developing the overall project plan, including the research questions, methods, and timeline. In the second phase our work involved in reviewing relevant research on the topic of the thesis. Then we collect dataset for our project. Till this work, it did not take much time. In data preprocessing it took longer time than other work. Because we had to rescale, zoom etc. in this part. Then we selected our models to apply. Application and training model part took a large amount of time. For better accuracy we had the change epochs, and it is time consuming. Then we did our model evaluation part and the rest of the time we spent making progress report and finally completed our final report on time.

8. Description of the Project output and discussion

InceptionV3:

Validation Graph:



Validation Graph of InceptionV3

The code for getting the validation graph –

```
plot_train_history(inception)
```

The graph shows the average values of the model accuracy and model loss over time. The x-axis represents the epoch, which is a full pass through the training data. The y-axis represents the accuracy and loss values.

The model accuracy graph shows that the model accuracy on both the training and validation sets increases over time. The training accuracy increases more quickly than the validation accuracy, which is a sign of overfitting. However, the validation accuracy also increases, which means that the model is learning to generalize to new data.

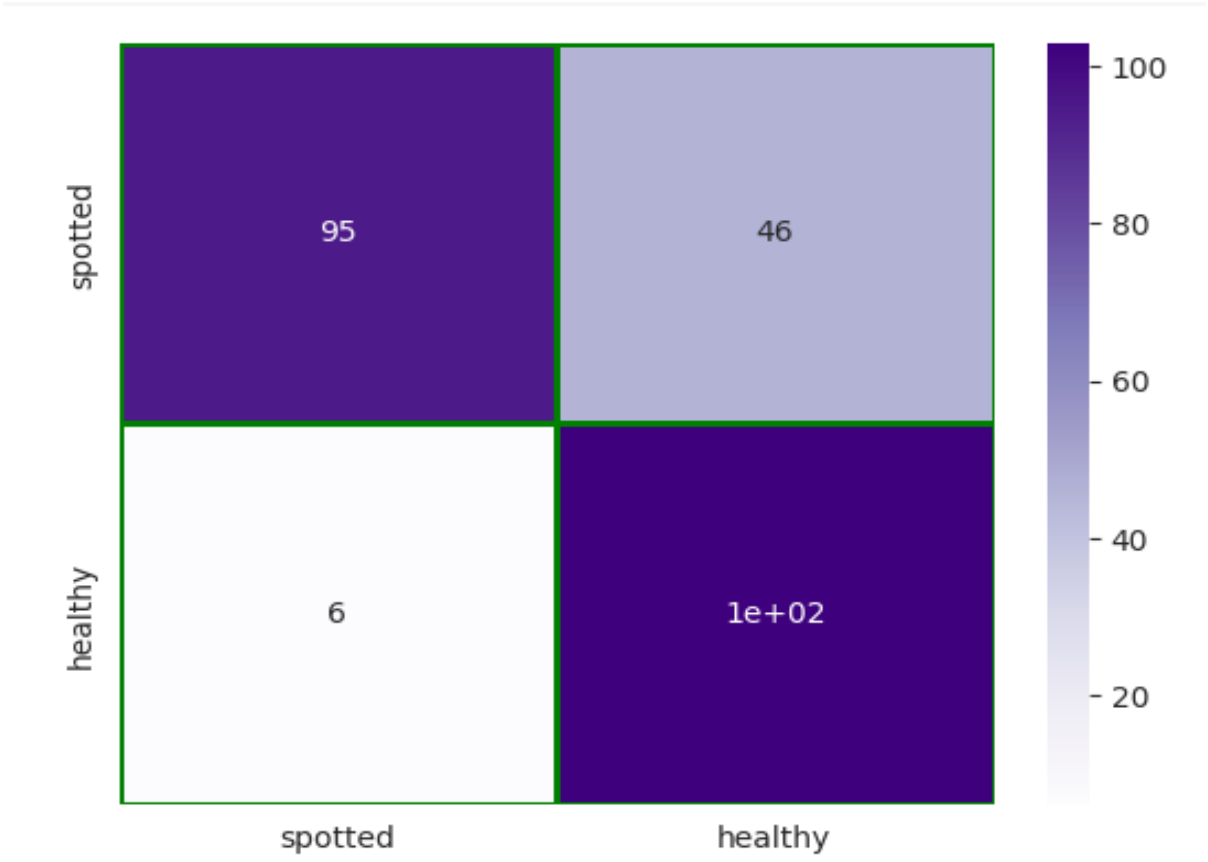
The model loss graph shows that the model loss on both the training and validation sets decreases over time. The training loss decreases more quickly than the validation loss, which is also a sign of overfitting. However, the validation loss also decreases, which means that the model is learning to make better predictions.

Some additional observations about the graph:

- The training accuracy and loss curves are close together, which suggests that the model is overfitting to some extent.
- The validation accuracy and loss curves are starting to diverge, which could be a sign of overfitting.
- The model accuracy on the validation set is still relatively high, at around 0.9.
- The model loss on the validation set is still relatively low, at around 0.3.

Overall, the graph suggests that the model is learning to perform well. However, it is important to monitor the validation accuracy and loss during training to ensure that the model is generalizing to new data.

Heatmap:



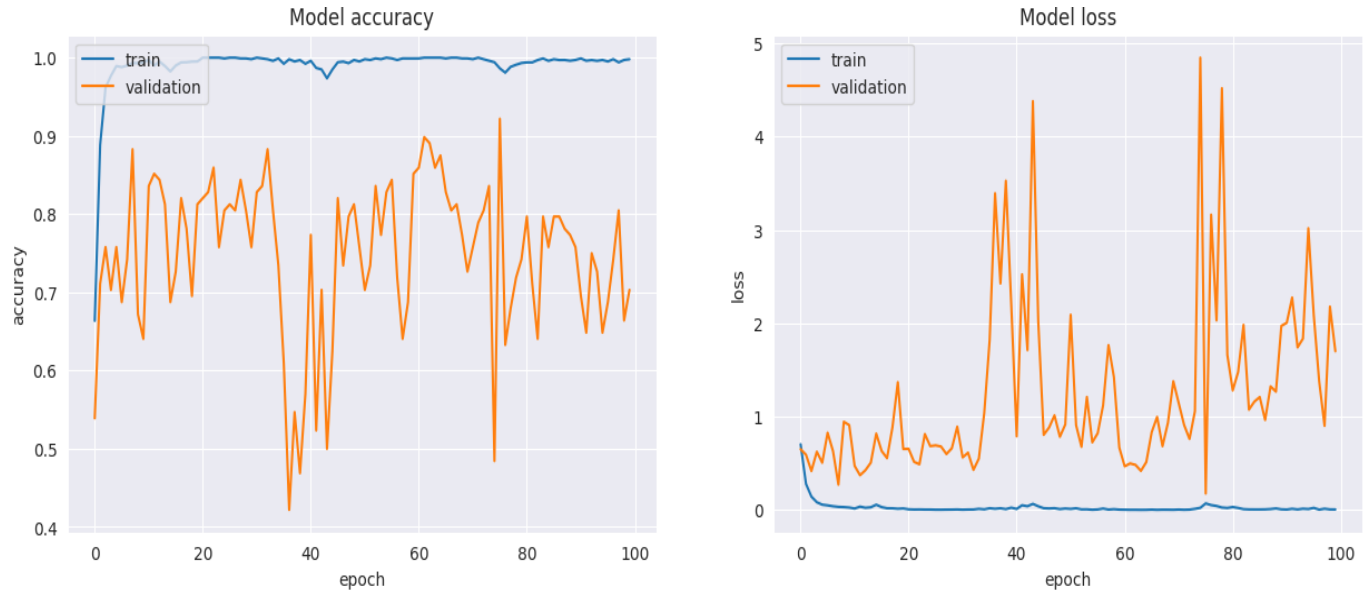
Test-Score:

	precision	recall	f1-score	support
spotted	0.673759	0.940594	0.785124	101.000
healthy	0.944954	0.691275	0.798450	149.000
accuracy	0.792000	0.792000	0.792000	0.792
macro avg	0.809356	0.815935	0.791787	250.000
weighted avg	0.835391	0.792000	0.793066	250.000

These results indicate that the classifier is performing well on both the healthy and spotted classes. It has a high precision and recall for the spotted class, and a high F1-score for both classes.

MobileNetV2:

Validation Graph:



Validation Graph of MobileNetV2

The image shows two graphs, one for model accuracy and one for model loss, over time (in epochs). The graphs show that both the accuracy and loss on the training set improve over time, but the accuracy and loss on the validation set initially improve and then plateau. This suggests that the model is overfitting to the training data.

Here is a more detailed description of each graph:

Model accuracy graph:

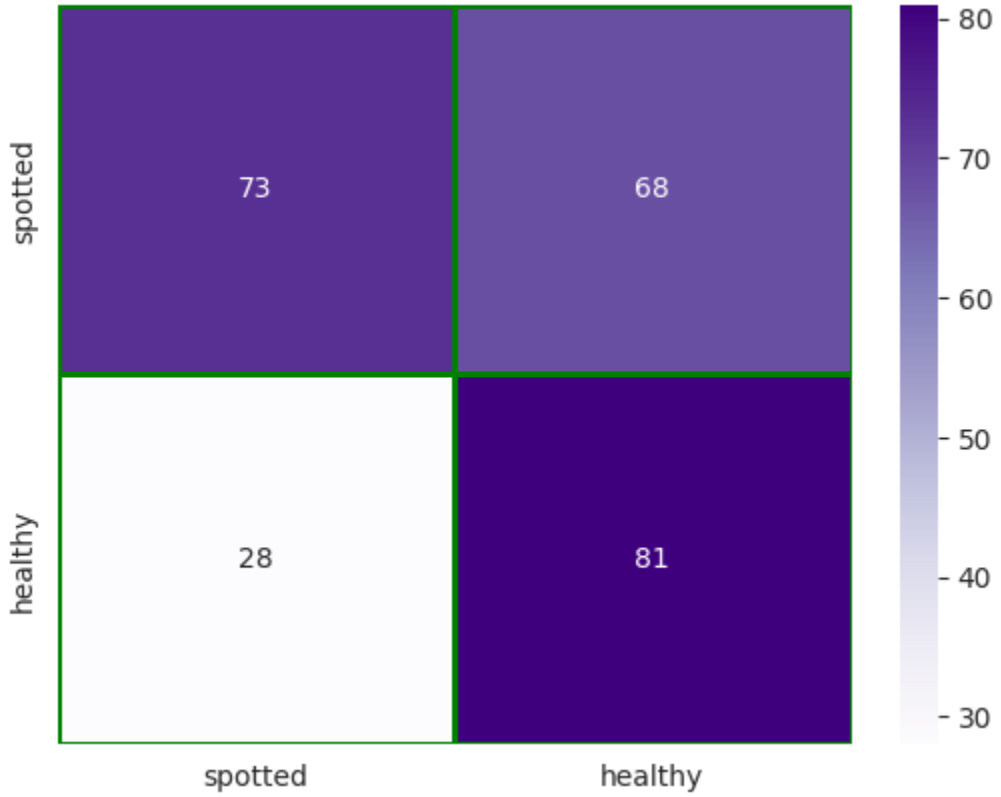
- The training accuracy curve starts at around 0.5 and increases to around 0.9 over the course of 100 epochs.
- The validation accuracy curve starts at around 0.7 and increases to around 0.85 over the course of 50 epochs. After 50 epochs, the validation accuracy curve plateaus.

Model loss graph:

- The training loss curve starts at around 1.0 and decreases to around 0.2 over the course of 100 epochs.
- The validation loss curve starts at around 0.5 and decreases to around 0.3 over the course of 50 epochs. After 50 epochs, the validation loss curve plateaus.

Overall, the graphs suggest that the model is learning to perform well on the training data. However, the model is also overfitting to the training data, as evidenced by the plateauing of the validation accuracy and loss curves.

Heatmap:



Test-Score:

	precision	recall	f1-score	support
spotted	0.517730	0.722772	0.603306	101.000
healthy	0.743119	0.543624	0.627907	149.000
accuracy	0.616000	0.616000	0.616000	0.616
macro avg	0.630425	0.633198	0.615606	250.000
weighted avg	0.652062	0.616000	0.617968	250.000

The result indicate that the classifier is performing moderately well on both the healthy and spotted classes. It has a higher precision and recall for the spotted class than for the healthy class.

CNN Model:

Validation Graph:

Validation Graph of CNN Model

The image shows two graphs, one for model accuracy and one for model loss, over time (in epochs). The graphs show that both the accuracy and loss on the training set improve over time, but the accuracy and loss on the validation set initially improve and then plateau. This suggests that the model is overfitting to the training data.

Model accuracy graph:

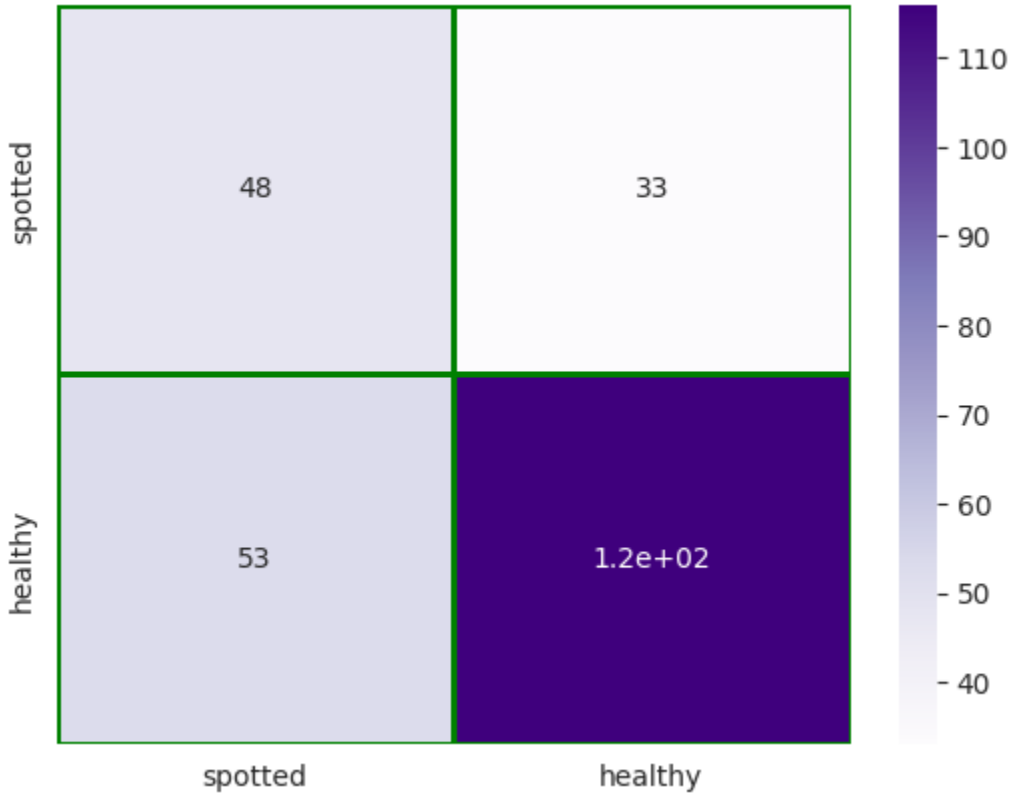
- The training accuracy curve starts at around 0.6 and increases to around 0.95 over the course of 100 epochs.
- The validation accuracy curve starts at around 0.8 and increases to around 0.9 over the course of 50 epochs. After 50 epochs, the validation accuracy curve plateaus.

Model loss graph:

- The training loss curve starts at around 1.0 and decreases to around 0.2 over the course of 100 epochs.
- The validation loss curve starts at around 0.5 and decreases to around 0.3 over the course of 50 epochs. After 50 epochs, the validation loss curve plateaus.

Overall, the graphs suggest that the model is also learning to perform well on the training data. However, the model is also overfitting to the training data, as evidenced by the plateauing of the validation accuracy and loss curves.

Heatmap:



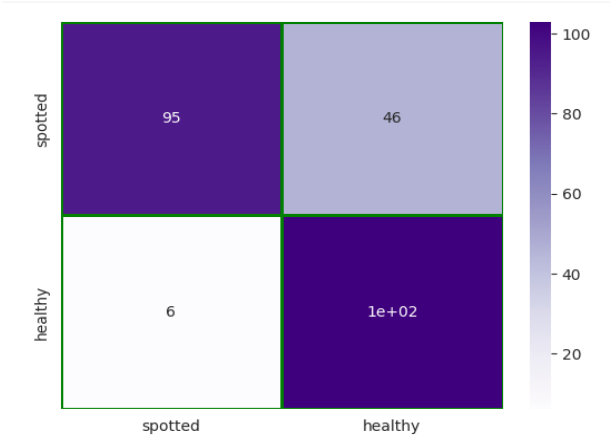
Test-Score:

	precision	recall	f1-score	support
spotted	0.592593	0.475248	0.527473	101.000
healthy	0.686391	0.778523	0.729560	149.000
accuracy	0.656000	0.656000	0.656000	0.656
macro avg	0.639492	0.626886	0.628516	250.000
weighted avg	0.648496	0.656000	0.647917	250.000

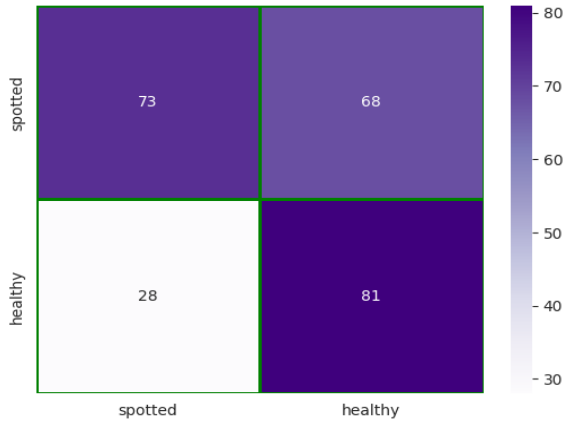
The result indicate that the classifier is performing moderately well on both the healthy and spotted classes. It has a higher precision and recall for the spotted class than for the healthy class.

Comparative analysis of Confusion Matrix:

Here, InceptionV3 is showing best performance for both spotted and healthy region. But mobileNetV2 is better in both spotted and healthy region. And CNN is only good in healthy region.

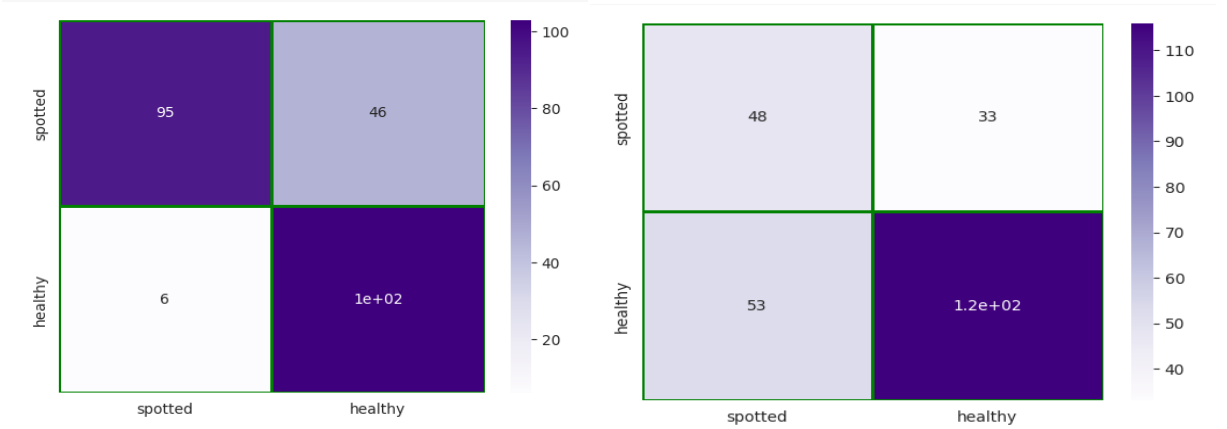


InceptionV3



MobileNetV2

Compared with InceptionV3, MobileNetV2 is not performing well. InceptionV3 is able to detect both healthy and spotted images in a good quantity as the cells are diagonally marked in deep blue (true positive). But detecting True positive results of MobileNetV2 is not as good as InceptionV3. There are so many False positive and False negative results in MobileNetV2 architecture. It is also shown in accuracy as InceptionV3's accuracy is 79% and MobileNetV2's accuracy is 61%.



InceptionV3 CNN

Compared with InceptionV3, CNN cannot standby. In the confusion matrix of CNN, True positive results for healthy images. But giving False positive and False negative results in a large extent. InceptionV3 is able to detect both healthy and spotted images in a good quantity as the cells are diagonally marked in deep blue (true positive).It is also shown in accuracy as InceptionV3's accuracy is 79% and CNN's accuracy is 65%.

Benefits **of using InceptionV3**

1. Able to detect both glaucoma and normal images very well.
2. Have highest accuracy
3. Precision and recall are also satisfactory

More benefits are-

- InceptionV3 is relatively efficient, means that it can be trained and used on large datasets without requiring a lot of computational resources.
- InceptionV3 can be used for a variety of image classification tasks, such as classifying objects, scenes, and faces.
- InceptionV3 has been used to develop models that can accurately classify cancer cells and other medical conditions.
- InceptionV3 has been used to develop models that can accurately identify objects on the

We have chosen this solution but this is not perfect. we have some limitations also.

Limitation of the system:

The architectures that we used in our system tend to be computationally intensive, especially InceptionV3, which has a relatively large number of parameters. Running these models may require significant computational power and memory, making them less suitable for deployment on resource-constrained devices.

In our system, we got accuracy less than 85% after applying three models. Thus, accuracy should be tested with other datasets to get more accurate value. Precision, f1 score and recall also should be more updated.
We have applied only three models. But in future we should use more models on new datasets to get more satisfactory result.
Our system lacks an extensive and diverse dataset, the models may not perform optimally, and overfitting could become a concern.
MobileNetV2 is designed to be lightweight, making it a better choice for mobile and embedded applications. However, InceptionV3 and some other CNNs may have larger model sizes, which can be a limitation when deploying the model to our devices with limited storage.
In our system, on edge devices or in real-time applications, inference time can be a limiting factor. InceptionV3, for instance, may have longer inference times due to its complexity. MobileNetV2, on the other hand, is optimized for faster inference, but it might not be as accurate as InceptionV3. MobileNetV2 is excellent for mobile and embedded scenarios, but InceptionV3 may be more accurate for certain image recognition tasks. CNNs can be highly versatile, but selecting the right architecture for our project is critical.
Pretrained models may require periodic updates as they become outdated or as new versions are released. Keeping these models up to date can be essential for maintaining optimal performance in our project.
Codes and techniques need to be more updated and run time should be faster.

9. Lesson learnt

Algorithm Selection is Crucial: The choice of deep learning algorithms plays a critical role in the success of disease classification systems. Careful consideration of algorithms and architectures can significantly impact the performance of the system.

Data Quality is Essential: The accuracy of a deep learning model heavily relies on the quality and diversity of the dataset used for training. High-quality, well-labeled data is a cornerstone for the success of any machine learning project.

Set Realistic Expectations: It's essential to set realistic expectations for the accuracy of the disease classification system. Setting realistic benchmarks for success is crucial in managing project expectations.

Automation in Agriculture: The development of automated vision-based disease identification systems holds great potential in agriculture. The application of deep learning in agriculture can have far-reaching benefits, from increasing crop yields to reducing the need for manual plant inspection.

Continual Improvement: The study represents a significant step in the direction of automated plant disease identification, but there is always room for improvement. Deep learning models should be seen as evolving systems that can be refined and improved over time.

In summary, the paper on pepper bell leaf disease classification using a deep learning approach provides valuable insights into the potential of using neural networks for plant disease detection. The lessons learned from this study emphasize the importance of algorithm selection, data quality, realistic expectations, the potential for automation in agriculture, and the need for continual improvement in this field.

10. Contribution of individual team member:

Following shows the approximate contributions in percentage of each member of our team while working on this project:

	Planning	**Dataset collection and analysis**	**Preprocessing and Augmentation**	**Model implementation**	**Output analysis**	**Report**
Obantika	30%	30%	30%	30%	30%	30%
Okita	30%	25%	25%	25%	30%	30%
Sadia	20%	20%	20%	20%	20%	20%
Samin	20%	25%	25%	25%	20%	20%
Total	100%	100%	100%	100%	100%	100%

11. Conclusion:

In conclusion, we presented an analysis of current plant-based disease detection systems, with a focus on the vision-based identification and classification of specific disorders of bell pepper leaves. By employing Neural Networks, we aimed to create an automated system for the identification of bacterial spots on bell pepper leaves. Our methodology involved the use of various simulation techniques for neurons and layers, utilizing various models trained on a dataset of bell pepper leaf images and tested with 10% images of the dataset. The results of our study demonstrated the effectiveness of our approach. High accuracy rate indicates the potential of deep learning techniques, specifically InceptionV3, for the automated detection and classification of diseases in bell pepper plants. Furthermore, we considered the use of other deep learning

algorithms like CNN and MobileNetV2 in our work, which does not convey more accuracy than InceptionV3. expected result of achieving accuracy between 75% to 80% was not only met but we got satisfactory accuracy, affirming the viability of deep learning in the field of plant disease classification. This work opens the door to practical applications in agriculture, where early disease detection can significantly contribute to crop management and yield optimization.

This research offers a promising avenue for the development of automated systems for disease detection in bell pepper plants, showcasing the potential for deep learning to play a pivotal role in modern agriculture. The high accuracy achieved in our experiments and the consideration of more advanced algorithms paves the way for further exploration and implementation of these technologies in real-world scenarios, benefiting both farmers and the agricultural industry.

12. References

https://www.researchgate.net/figure/The-architecture-of-Inception-V3-model_fig5_349717475

https://www.researchgate.net/publication/370209339_Leaf_Disease_Classification_in_Bell_Pepper_Plant_using_VGGNet

https://www.frontiersin.org/articles/10.3389/fpls.2023.1230886/full

https://www.researchgate.net/publication/343751436_Bell_Pepper_Leaf_Disease_Classification_Using_CNN

https://www.sciencedirect.com/science/article/pii/S2405844023025094