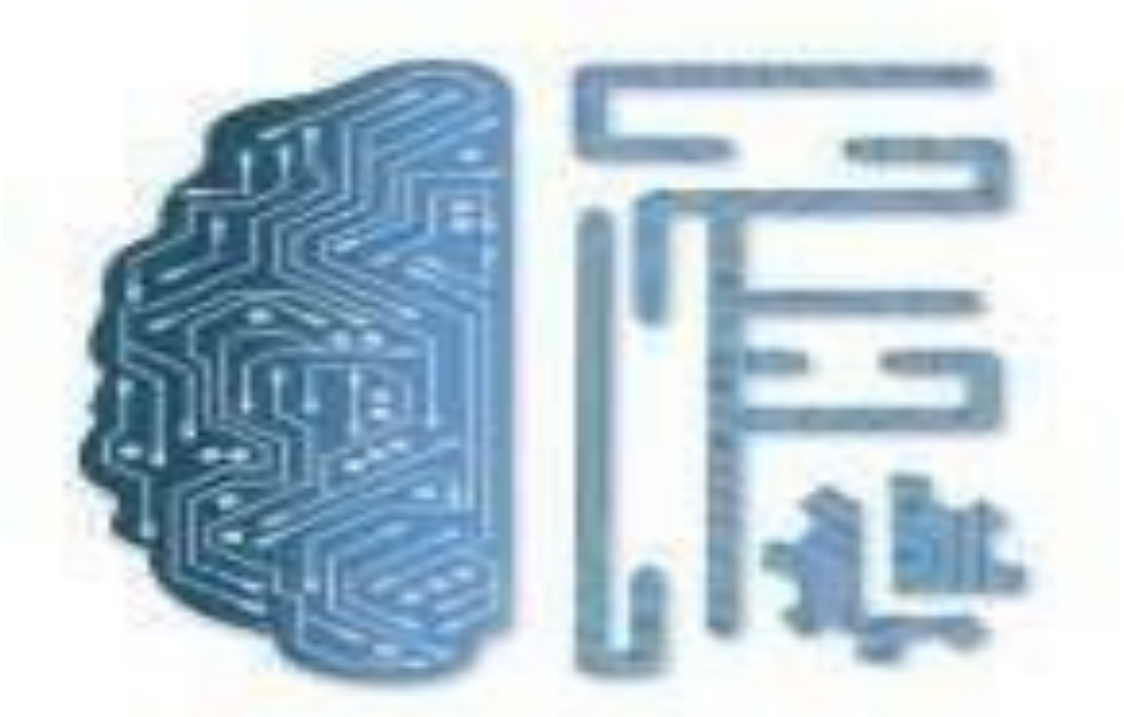
FEYNN LABS CONSULTING SERVICES



Efficient Disease Detection of Paddy Crop using CNN

Contributors:

- Prabhala v n s l Tejaswi
- Arnav Samal
- Harshal Avinash Taware

Feasibility:

Similar to plant disease detection, utilizing ML and DL techniques, such as Convolutional Neural Networks (CNNs), for predicting paddy crop yield requires ample labeled data and robust hardware setups. Access to online databases with images of paddy crops at different growth stages and environmental conditions is crucial. Additionally, deploying these models necessitates powerful computers with GPUs, preferably on cloud platforms for real-time prediction. Therefore, to make paddy crop prediction with ML and DL feasible, there's a need for adequate technology, extensive datasets, and suitable hardware and software infrastructure.

Viability:

For ML and DL-based paddy crop prediction systems to be viable, they must accurately forecast crop yields across various environmental factors and cultivation practices. This accuracy is pivotal for farmers to make informed decisions regarding crop management, resource allocation, and harvest planning. The system should be scalable to accommodate different agricultural regions and adaptable to changing climate conditions and farming techniques. Moreover, user-friendly interfaces, including intuitive dashboards and mobile applications, are essential for farmers to easily access and interpret prediction results, enhancing the system's viability and utility.

Monetization:

There are several monetization strategies for paddy crop prediction services. One approach is to offer subscription-based access, where farmers pay a regular fee to utilize the prediction system, granting them access to yield forecasts and agronomic recommendations. Alternatively, a pay-per-use model can be implemented, charging farmers for each prediction analysis conducted. Premium features and personalized support can be offered for an additional fee, providing advanced forecasting capabilities and tailored advice. Furthermore, monetization can extend beyond direct farmer subscriptions by leveraging collected data to offer insights and analytics to agricultural researchers, businesses, and government agencies interested in understanding paddy crop trends and optimizing production practices.

1.Problem Statement:

Efficient paddy crop prediction is essential for maximizing yield and ensuring food security. However, traditional methods often lack accuracy and timeliness, leading to potential crop losses. The goal is to develop an intelligent paddy crop prediction system that leverages Convolutional Neural Networks (CNNs) to accurately forecast crop yield based on various input factors such as environmental conditions, historical data, and crop growth stages.

2.Customer Needs Assessment:

The agricultural sector faces numerous challenges, including the need for accurate crop yield predictions to optimize resource allocation and maximize productivity. Farmers, agricultural enterprises, and governmental organizations require a reliable tool to enhance decision-making processes and mitigate risks associated with crop management. By addressing these needs, the initiative aims to improve agricultural sustainability, profitability, and resilience in paddy cultivation regions globally.

3.Target Specifications and Characterization:

The end-users of the system include small and large-scale paddy farmers, agricultural businesses, and relevant stakeholders in regions with significant paddy cultivation. The system prioritizes accuracy, scalability, and ease of integration with existing agricultural technologies. User-friendly interfaces and mobile applications enable accessibility and seamless interaction with the prediction system. Collaboration with agricultural services, research institutions, and governments enhances the system's effectiveness and adoption. Data privacy, security, and regulatory compliance are paramount, along with cost-effective pricing models tailored to diverse user segments. Continuous improvement through feedback mechanisms ensures the system remains adaptive and responsive to evolving user needs and technological advancements.

4.Benchmarking:

- ✓ **PlantVillage's Nuru:**

  Nuru, developed by PlantVillage, utilizes AI to diagnose agricultural diseases, achieving an accuracy rate of 90%. Its scalability enables processing of large datasets efficiently, with positive user feedback on its accessibility and interface.

- ✓ **Microsoft's FarmBeats:**

  FarmBeats combines IoT and AI for precision agriculture, including disease prediction. It boasts quick data processing, smooth integration with farm technologies, and cost-effectiveness by reducing manual monitoring.

- ✓ **IBM's Watson Decision Platform for Agriculture:**

IBM's platform employs AI to analyze diverse data sources for disease prediction, demonstrating high sensitivity and specificity. It prioritizes user feedback for continuous improvement and emphasizes data security and privacy.

5.Applicable Patents:

- ✓ Image Recognition and Disease Detection:
- ✓ US Patent No. 10,952,943: Describes a system for plant disease detection using deep learning.
- ✓ **Data Integration and Prediction:**

  US Patent No. 10,609,105: Covers an intelligent agricultural disease prediction system integrating various data sources.
- ✓ AI-powered Recommendations and Decision Support:
- ✓ US Patent No. 10,802,202: Focuses on a system for crop disease diagnosis and treatment recommendation using AI.

6.Applicable Regulations:

An intelligent crop disease prediction system's regulatory framework is determined by a number of aspects, such as:

1. Location: Each country and region has considerably different regulations.
2. Data Collection and Use: How we gather, store, and use data will be subject to laws governing data privacy, security, and ownership. Examine the principles of the
3. General Data Protection Regulation (GDPR) and the Personal Data Protection Bill in India.
4. Algorithms and AI openness: Certain areas are putting laws into place pertaining to algorithmic fairness and openness, especially when it comes to AI utilized in decision-making.
5. Pesticides and Chemicals: If system recommends specific treatments, regulationsrelated to pesticide or chemical usage may apply.

6. Product Classification: Depending on system's features and intended use, it may be classified as agricultural software, a decision support tool, or even a diagnostic device, leading to different regulatory requirements.

7.Applicable Constraints

Here are some applicable constraints to consider when developing an intelligent crop disease prediction system:

1. *Data Availability and Quality:*

Volume: Large amounts of diverse data are needed to train and validate machine learning models effectively.

Accuracy: Data must be accurate and reliable for predictions to be accurate.

Representativeness: Data should cover a wide range of crop varieties, regions, and environmental conditions to ensure generalizability.

Accessibility: Obtaining and integrating data from various sources (e.g., farms, weather stations, research institutions) can be challenging due to privacy concerns, ownership issues, and compatibility problems.

2. *Technical Limitations:*

Computational Resources: Training and running complex machine learning models require significant processing power and memory.

Connectivity: Reliable internet access is often needed for real-time data collection, model updates, and access to cloud-based services.

Interoperability: Integrating the system with existing agricultural equipment and software systems can pose challenges due to compatibility issues.

3. *Expertise and Adoption:*

Technical Expertise: Farmers and agricultural professionals may need training to use andinterpret the system's results effectively.

Trust and Acceptance: Building trust in AI-based systems can be a challenge, as farmers maybe hesitant to rely on unfamiliar technologies.

Adaptability to Local Practices: Systems need to be adaptable to different farming practices, cropping systems, and cultural contexts to ensure widespread adoption.

4. *Ethical Considerations:*

Data Bias: Algorithms can perpetuate biases if trained on biased data, leading to unfair ordiscriminatory outcomes for certain groups of farmers or regions.

5. *Cost and Maintenance:*

Development and Deployment: Developing and deploying the system can be costly, potentially limiting accessibility for smaller farms or resource-constrained regions.

Ongoing Maintenance: Ongoing maintenance, updates, and technical support are necessary toensure the system's accuracy and reliability over time.

6. *Regulatory Compliance*

Data Privacy and Security: Systems must comply with regulations governing data collection,storage, and usage, as discussed earlier.

Algorithm Transparency: In some regions, regulations may require explanations for AI-baseddecisions, ensuring fairness and accountability.

8.Enterprise Blueprint

For efficient paddy crop prediction using CNN, our business model will be centered around providing farmers with an advanced tool for accurate forecasting and proactive disease management. We'll target paddy farmers in both small and large countries, focusing on regions where paddy cultivation is significant. Collaborations with government institutions, research centers, and agricultural services specialized in paddy farming will ensure tailored solutions. Revenue will be diversified through a registration model, supplemented by advanced data management services for additional fees. Distribution will be facilitated through online platforms and mobile applications optimized for paddy

farmers, supported by comprehensive customer support and training programs. Continuous research and development efforts, alongside partnerships with agricultural experts, will drive the improvement of prediction models. Key resources include expert data scientists and streamlined operational processes. Investments will be allocated to research, operations, and marketing, with a focus on sustainability and scalability. Partnerships with agricultural equipment suppliers and government agencies will enhance integration and knowledge sharing. Environmental considerations will be prioritized in technology development to promote sustainable farming practices among customers.

9.Concept Generation:

The concept for our Intelligent Crop Disease Prediction system revolves around leveraging advanced machine learning algorithms to empower farmers with early detection capabilities for crop diseases. By integrating image analysis, environmental data, and historical disease patterns, the system aims to predict and identify potential issues, allowing farmers to implementtimely interventions. This concept addresses the critical need for proactive disease managementin agriculture, ultimately minimizing crop losses and optimizing yields. The user-friendly interface and accessibility through mobile applications and web platforms ensure seamless adoption by farmers, while ongoing research and collaboration with agricultural experts contribute to continuous improvement and innovation in disease prediction models.
Thisconcept aligns with the broader goals of precision agriculture, fostering sustainability and resilience in global food production.

10.Concept Development:

To start this process, we will use advanced machine learning techniques to develop robust cropdisease prediction models. Once the model is complete, we upload it to the Flask server for seamless deployment and instant prediction. Our method is accurate and effective in detectingcrop diseases. To facilitate the use of this useful tool, we will provide users with an easy-to-useinterface using the GitHub distribution. Through the platform, farmers interact with a simple model to ensure disease control. This solution not only simplifies the estimation process, but also helps permaculture practices. Through continuous improvement and innovation, our goalis to provide farmers with reliable and measurable tools to improve crop health, minimize losses, and adapt to change in precision agriculture.
10.Final Product Prototype:

In our final product prototype, we have seamlessly integrated a cutting-edge crop disease prediction model into a user-friendly system, featuring both frontend and backend components.The frontend is designed with an intuitive interface accessible through web and mobile applications. Farmers can easily input data, such as images or environmental parameters, initiating the predictive analysis process.
The backend, powered by Flask, efficiently processes the input data using our advanced machine learning model. This backend system ensures real-time predictions with a high degreeof accuracy. We've prioritized scalability, enabling the system to handle a diverse range of cropsand expand its geographic coverage.

Through GitHub deployment, our prototype becomes readily accessible to farmers worldwide.The frontend allows for easy interaction, displaying comprehensive insights into predicted cropdiseases. This holistic solution not only streamlines disease management but also aligns with sustainable and resilient agricultural practices. As we iterate on user feedback and advancements in technology, our final product prototype represents a pivotal step towards revolutionizing precision agriculture.

11.Product Details:

Our Intelligent Crop Disease Prediction System is a comprehensive solution designed to empower farmers with advanced technology for early disease detection and proactive crop management. Here are the key details of the product:

Technology Stack:
Backend: Developed using Flask, our backend seamlessly integrates a state-of-the-art machine learning model for accurate crop disease prediction.

Frontend: The user-friendly interface is accessible through web and mobileapplications, providing an intuitive experience for farmers.

Machine Learning Model:
Utilizes advanced algorithms trained on diverse datasets, including images, environmental factors, and historical disease patterns, ensuring high accuracy in diseaseprediction.
User Input: Farmers can input data such as images or environmental parameters through thefrontend, initiating the prediction process.

The backend processes input data in real-time, delivering prompt and accurate predictions to farmers.
Deployment: Deployed using GitHub for easy accessibility, enabling farmers worldwideto benefit from the system.

MARKET SEGMRNTATION ANALYSIS

```
from PIL import Image
import glob
```

```
files = glob.glob('C:/Users/red/Downloads/archive/model/Labelled/*/*')
```

```
files_reshape = list(map(lambda x: x.replace('/Labelled\\', '/Resized\\'), files))
```

```
basewidth = 300
for file, file_save in zip(files, files_reshape):
    img = Image.open(file)
    wpercent = (basewidth/float(img.size[0]))
    hsize = int((float(img.size[1])*float(wpercent)))
    img = img.resize((basewidth,hsize), Image.ANTIALIAS)
    img.save(file_save)
```

To do this, a mask is created using the following steps:

```
from skimage.morphology import binary_closing, binary_opening, erosion
```

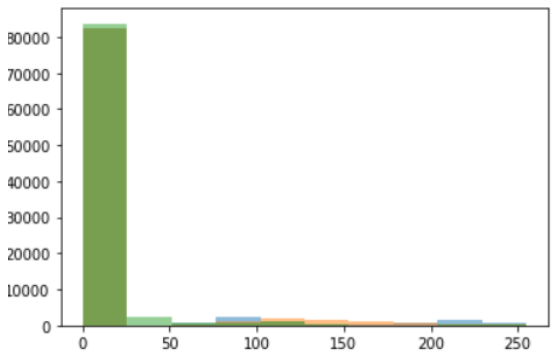
```
files_bgremoved = list(map(lambda x: x.replace('/Labelled\\', '/BGRemoved\\'), files))
```

```
selem = np.zeros((25, 25))

ci,cj=12, 12
cr=13

# Create index arrays to z
I,J=np.meshgrid(np.arange(selem.shape[0]),np.arange(selem.shape[1]))

# calculate distance of all points to centre
dist=np.sqrt((I-ci)**2+(J-cj)**2)

# Assign value of 1 to those points where dist<cr:
selem[np.where(dist<=cr)]=1
```

```
%matplotlib inline
import numpy as np
from scipy import ndimage

# fig, ax = plt.subplots(20,2, figsize=(10,80))
idx = 0
for file, file_save in zip(files, files_bgremoved):
    bg_frac = 0
    thres = 220
    img = Image.open(file)
    im_arr = np.array(img)
#     ax[idx, 0].imshow(im_arr)
    R = im_arr[:, :, 0]
    G = im_arr[:, :, 1]
    B = im_arr[:, :, 2]
    while bg_frac < 0.6:
        bg_mask = ((R>thres) | (B>thres))# & (G < 100)
        bg_frac = bg_mask.sum()/len(bg_mask.flatten())
        thres -= 5
    # we use opening first since our mask is reversed (the foreground and background are reversed here)
    bg_mask = binary_closing(erosion(binary_opening(bg_mask, selem), np.ones((3, 3))), np.ones((5,5)))

    #Get biggest blob
    label, num_label = ndimage.label(~bg_mask)
    size = np.bincount(label.ravel())
    biggest_label = size[1:].argmax() + 1
    bg_mask = label == biggest_label

    im_arr[~bg_mask, 0] = 255
    im_arr[~bg_mask, 1] = 255
    im_arr[~bg_mask, 2] = 255

    img = Image.fromarray(im_arr)
    img.save(file_save)
    idx+=1
```

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.hist(R.flatten(), alpha=0.5)
plt.hist(G.flatten(), alpha=0.5)
plt.hist(B.flatten(), alpha=0.5)
```

```
(array([83779.,  2307.,   865.,   781.,  1153.,   383.,   118.,    89.,
         148.,   377.]),
 array([  0. ,  25.5,  51. ,  76.5, 102. , 127.5, 153. , 178.5, 204. ,
        229.5, 255. ]),
 <a list of 10 Patch objects>)
```



```
import os
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import models
from tensorflow.keras import layers
from tensorflow.keras import optimizers
import matplotlib.pyplot as plt
%matplotlib inline

base_dir = ('C:/Users/red/Downloads/archive/model/BGRemoved')
os.makedirs(base_dir, exist_ok=True)
```

```
datagen = ImageDataGenerator(
    rotation_range=30, width_shift_range=0.15,
    height_shift_range=0.15, shear_range=0.15,
    zoom_range=0.2,horizontal_flip=True,
    fill_mode="nearest", validation_split=0.3)
batch_size = 32

train_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training')

val_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')
```

```
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau

checkpoint = ModelCheckpoint('VGG16.h5', verbose=1, monitor='val_accuracy', save_best_only=True, mode='auto')
```

```
datagen = ImageDataGenerator(
    rotation_range=30, width_shift_range=0.15,
    height_shift_range=0.15, shear_range=0.15,
    zoom_range=0.2,horizontal_flip=True,
    fill_mode="nearest")
# datagen = ImageDataGenerator(rescale=1./255)

def extract_features(trainorval, sample_count):
    features = np.zeros(shape=(sample_count, 7, 7, 512))
    labels = np.zeros(shape=(sample_count, 4))
    if trainorval=="training":
        generator = train_generator
    else:
        generator = val_generator
    i = 0
    for inputs_batch, labels_batch in generator:
        features_batch = conv_base.predict(preprocess_input(inputs_batch))
        try:
            features[i * batch_size : (i + 1) * batch_size] = features_batch
            labels[i * batch_size : (i + 1) * batch_size] = labels_batch
        except ValueError:
            break
        i += 1
        if i * batch_size >= sample_count:
            break
    return features, labels
```

```
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input

conv_base = VGG16(weights='imagenet',
                  include_top=False,
                  input_shape=(224, 224, 3))
```

```
conv_base.trainable = True
for layer in conv_base.layers:
    layer.trainable = True
```

```
train_features, train_labels = extract_features('training', 2319)
validation_features, validation_labels = extract_features('validation', 991)

train_features = np.reshape(train_features, (2319, 7 * 7 * 512))
validation_features = np.reshape(validation_features, (991, 7 * 7 * 512))
```

```
model = models.Sequential()
model.add(layers.Dense(256, activation='relu', input_dim=7 * 7 * 512))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.2))  #Removing 50% of the weights!
model.add(layers.Dense(4, activation='softmax'))
```

```
from tensorflow.keras.optimizers import Adam

INIT_LR = 1e-1
EPOCHS = 30

opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)

model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint('VGG16.h5', verbose=1, monitor='val_acc', save_best_only=True, mode='auto')

history = model.fit(train_features, train_labels,
                    epochs=30,
                    batch_size=batch_size,
                    validation_data=(validation_features, validation_labels),
                    callbacks=[checkpoint])
```

```
Train on 2319 samples, validate on 991 samples
WARNING:tensorflow:From C:\Users\Justin\Anaconda3\envs\tf-gpu\lib\site-packages\tensorflow\python\ops\math_grad.py:1250: ad
d_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future ve
rsion.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Using TensorFlow backend.
Epoch 1/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3477 - acc: 0.5844
Epoch 00001: val_acc improved from -inf to 0.58426, saving model to VGG16.h5
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3451 - acc: 0.5846 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 2/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3831 - acc: 0.5849
Epoch 00002: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 3/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3933 - acc: 0.5843
Epoch 00003: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 4/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3967 - acc: 0.5841
Epoch 00004: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 5/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3933 - acc: 0.5843
Epoch 00005: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 6/30
```

```
2304/2319 [============================>.] - ETA: 0s - loss: 6.3979 - acc: 0.5840
Epoch 00006: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 7/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4001 - acc: 0.5838
Epoch 00007: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 8/30
2304/2319 [============================>.] - ETA: 0s - loss: 6.4013 - acc: 0.5838
Epoch 00008: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 9/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4102 - acc: 0.5832
Epoch 00009: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 10/30
2304/2319 [============================>.] - ETA: 0s - loss: 6.3913 - acc: 0.5844
Epoch 00010: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 11/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4034 - acc: 0.5836
Epoch 00011: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 12/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4034 - acc: 0.5836
Epoch 00012: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
-
Epoch 13/30
2304/2319 [============================>.] - ETA: 0s - loss: 6.3979 - acc: 0.5840
Epoch 00013: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 14/30
2304/2319 [============================>.] - ETA: 0s - loss: 6.3946 - acc: 0.5842
Epoch 00014: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 15/30
2304/2319 [============================>.] - ETA: 0s - loss: 6.4013 - acc: 0.5838
Epoch 00015: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 16/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3899 - acc: 0.5845
Epoch 00016: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 17/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3967 - acc: 0.5841
Epoch 00017: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 18/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4068 - acc: 0.5834- ETA: 1s - loss: 6.58
Epoch 00018: val_acc did not improve from 0.58426
2319/2319 [==============================] - 2s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 19/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4001 - acc: 0.5838
```

```
Epoch 20/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4136 - acc: 0.5830
Epoch 00020: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 21/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4034 - acc: 0.5836
Epoch 00021: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 22/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4001 - acc: 0.5838
Epoch 00022: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 23/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3899 - acc: 0.5845
Epoch 00023: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 24/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4034 - acc: 0.5836
Epoch 00024: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 25/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3933 - acc: 0.5843
Epoch 00025: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 26/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3933 - acc: 0.5843
Epoch 00026: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
```

```
Epoch 27/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3933 - acc: 0.5843
Epoch 00027: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 28/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.3899 - acc: 0.5845
Epoch 00028: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 29/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4001 - acc: 0.5838
Epoch 00029: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
Epoch 30/30
2272/2319 [============================>.] - ETA: 0s - loss: 6.4034 - acc: 0.5836
Epoch 00030: val_acc did not improve from 0.58426
2319/2319 [==============================] - 3s 1ms/sample - loss: 6.3964 - acc: 0.5841 - val_loss: 6.3937 - val_acc: 0.584
3
```

]:

```
datagen = ImageDataGenerator(
    rotation_range=30, width_shift_range=0.15,
    height_shift_range=0.15, shear_range=0.15,
    zoom_range=0.2,horizontal_flip=True,
    fill_mode="nearest", validation_split=0.3)
batch_size = 32

train_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training')

val_generator = datagen.flow_from_directory(
    base_dir,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation')
```

```
Found 2319 images belonging to 4 classes.
Found 991 images belonging to 4 classes.
```

[ ]:
```
conv_base.trainable = True

set_trainable = False
for layer in conv_base.layers:
    if layer.name == 'block4_conv1':
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False
```

[ ]:
```
model = models.Sequential()
model.add(conv_base)
model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu', input_dim=7 * 7 * 512))
# model.add(layers.Dropout(0.2))
model.add(layers.Dense(128, activation='relu'))
# model.add(layers.Dropout(0.2))  #Removing 50% of the weights!
model.add(layers.Dense(4, activation='softmax'))
```

```
def extract_features19(trainorval, sample_count):
    features = np.zeros(shape=(sample_count, 7, 7, 512))
    labels = np.zeros(shape=(sample_count, 4))
    if trainorval=="training":
        generator = train_generator
    else:
        generator = val_generator
    i = 0
    for inputs_batch, labels_batch in generator:
        features_batch = conv_base19.predict(preprocess_input(inputs_batch))
        try:
            features[i * batch_size : (i + 1) * batch_size] = features_batch
            labels[i * batch_size : (i + 1) * batch_size] = labels_batch
        except ValueError:
            break
        if i==0:
            print("one down")
        i += 1
        if i * batch_size >= sample_count:
            break
    return features, labels
```

```
from tensorflow.keras.applications.vgg19 import VGG19, preprocess_input

conv_base19 = VGG19(weights='imagenet',
                  include_top=False,
                  input_shape=(224, 224, 3))
```

```
train_features19, train_labels19 = extract_features19('training', 2319)
validation_features19, validation_labels19 = extract_features19('validation', 991)

train_features19 = np.reshape(train_features19, (2319, 7 * 7 * 512))
validation_features19 = np.reshape(validation_features19, (991, 7 * 7 * 512))
```

:
```
model = models.Sequential()
model.add(layers.Dense(256, activation='relu', input_dim=7 * 7 * 512))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dropout(0.5))  #Removing 50% of the weights!
model.add(layers.Dense(4, activation='softmax'))
```

```
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
```

```
resnet_base = ResNet50(weights='imagenet',
                       include_top=False,
                       input_shape=(224, 224, 3))
```

```
def extract_features_resnet(trainorval, sample_count):
    features = np.zeros(shape=(sample_count, 7, 7, 2048))
    labels = np.zeros(shape=(sample_count, 4))
    if trainorval=="training":
        generator = train_generator
    else:
        generator = val_generator
    i = 0
    for inputs_batch, labels_batch in generator:
        features_batch = resnet_base.predict(preprocess_input(inputs_batch))
        try:
            features[i * batch_size : (i + 1) * batch_size] = features_batch
            labels[i * batch_size : (i + 1) * batch_size] = labels_batch
        except ValueError:
            break
        i += 1
        if i * batch_size >= sample_count:
            break
    return features, labels
```

```
train_features_res, train_labels_res = extract_features_resnet('training', 2319)
validation_features_res, validation_labels_res = extract_features_resnet('validation', 991)

train_features_res = np.reshape(train_features_res, (2319, 7 * 7 * 2048))
validation_features_res = np.reshape(validation_features_res, (991, 7 * 7 * 2048))
```

```
import numpy as np
import pickle
import cv2
from os import listdir
from sklearn.preprocessing import LabelBinarizer
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array, load_img
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

```
EPOCHS = 70
INIT_LR = 1e-3
BS = 16
default_image_size = tuple((256, 256))
image_size = 0
directory_root = 'C:/Users/red/Downloads/archive/model//BGRemoved'
width=256
height=256
depth=3
```

```
def convert_image_to_array(image_dir):
    try:
        image = cv2.imread(image_dir)
        if image is not None :
            image = cv2.resize(image, default_image_size)
            return img_to_array(image)
        else :
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None
```

```
image_list, label_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)
    for directory in root_dir :
        # remove .DS_Store from list
        if directory == ".DS_Store" :
            root_dir.remove(directory)

    for disease_folder in root_dir :
        plant_disease_image_list = listdir(f"{directory_root}/{disease_folder}")

        for image in plant_disease_image_list:
            image_directory = f"{directory_root}/{disease_folder}/{image}"
            if image_directory.endswith(".jpg") == True or image_directory.endswith(".JPG") == True:
                image_list.append(convert_image_to_array(image_directory))
                label_list.append(disease_folder)
    print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")
```

```
[INFO] Loading images ...
Error : [Errno 2] No such file or directory: 'C:/Users/red/Downloads/archive/model//BGRemoved'
```

```
image_size = len(image_list)
print(image_size)
```

```
0
```

```
label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)
pickle.dump(label_binarizer,open('label_transform.pkl', 'wb'))
n_classes = len(label_binarizer.classes_)
```

```
print(label_binarizer.classes_)
```

```
['BrownSpot' 'Healthy' 'Hispa' 'LeafBlast']
```

]:
```
np_image_list = np.array(image_list, dtype=np.float16) / 225.0
```

]:
```
print("[INFO] Spliting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_size=0.30, random_state = 42)
```

```
[INFO] Spliting data to train, test
```

]:
```
aug = ImageDataGenerator(
    rotation_range=30, width_shift_range=0.15,
    height_shift_range=0.15, shear_range=0.15,
    zoom_range=0.2,horizontal_flip=True,
    fill_mode="nearest")
```

]:
```
model = Sequential()
inputShape = (height, width, depth)
chanDim = -1
if K.image_data_format() == "channels_first":
    inputShape = (depth, height, width)
    chanDim = 1
model.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))
```

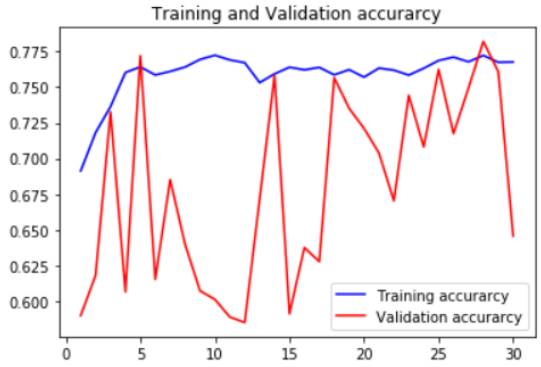
```
model.summary()
```

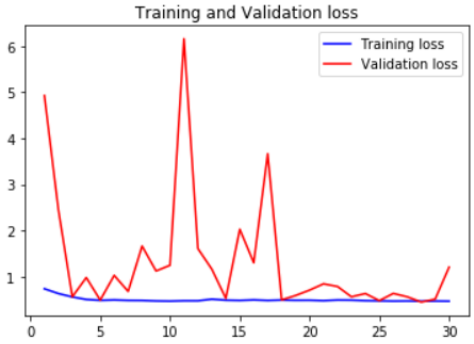
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 256, 256, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 256, 256, 32)      0
_________________________________________________________________
batch_normalization_1 (Batch (None, 256, 256, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 85, 85, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 85, 85, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 85, 85, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 85, 85, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 85, 85, 64)        256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 85, 85, 64)        36928
_________________________________________________________________
activation_3 (Activation)    (None, 85, 85, 64)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 85, 85, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 42, 42, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 42, 42, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 42, 42, 128)       73856
_________________________________________________________________
activation_4 (Activation)    (None, 42, 42, 128)       0
_________________________________________________________________
batch_normalization_4 (Batch (None, 42, 42, 128)       512
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 42, 42, 128)       147584
_________________________________________________________________
activation_5 (Activation)    (None, 42, 42, 128)       0
_________________________________________________________________
batch_normalization_5 (Batch (None, 42, 42, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 21, 21, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 21, 21, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 56448)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               14450944
_________________________________________________________________
activation_6 (Activation)    (None, 256)               0
_________________________________________________________________
batch_normalization_6 (Batch (None, 256)               1024
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 1028
_________________________________________________________________
activation_7 (Activation)    (None, 4)                 0
=================================================================
Total params: 14,732,420
Trainable params: 14,731,076
Non-trainable params: 1,344
_________________________________________________________________
```

```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
# distribution
model.compile(loss="binary_crossentropy", optimizer=opt,metrics=["accuracy"])
# train the network
print("[INFO] training network...")
```

```
[INFO] training network...
```

]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
#Train and validation accuracy
plt.plot(epochs, acc, 'b', label='Training accurarcy')
plt.plot(epochs, val_acc, 'r', label='Validation accurarcy')
plt.title('Training and Validation accurarcy')
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
```





In [ ]:
```
# model = model.load('best_model_5conv2dense.h5')
from keras.models import load_model
from keras.models import model_from_json

json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
loaded_model.compile(loss="binary_crossentropy", optimizer=opt,metrics=["accuracy"])

# load weights into new model
loaded_model.load_weights("best_model_5conv2dense_woutBG.h5")
```

In [ ]:
```
print("[INFO] Calculating model accuracy")
scores = loaded_model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")
```

```
[INFO] Calculating model accuracy
993/993 [==============================] - 5s 5ms/step
Test Accuracy: 78.19738388061523
```

```
scores_brownspot = loaded_model.evaluate(x_test[y_test[:,0]==1], y_test[y_test[:,0]==1])
print(f"Test Accuracy: {scores_brownspot[1]*100}")
```

```
179/179 [==============================] - 1s 7ms/step
Test Accuracy: 85.33519506454468
```

```
scores_healthy = loaded_model.evaluate(x_test[y_test[:,1]==1], y_test[y_test[:,1]==1])
print(f"Test Accuracy: {scores_healthy[1]*100}")
```

```
432/432 [==============================] - 2s 5ms/step
Test Accuracy: 92.1875
```

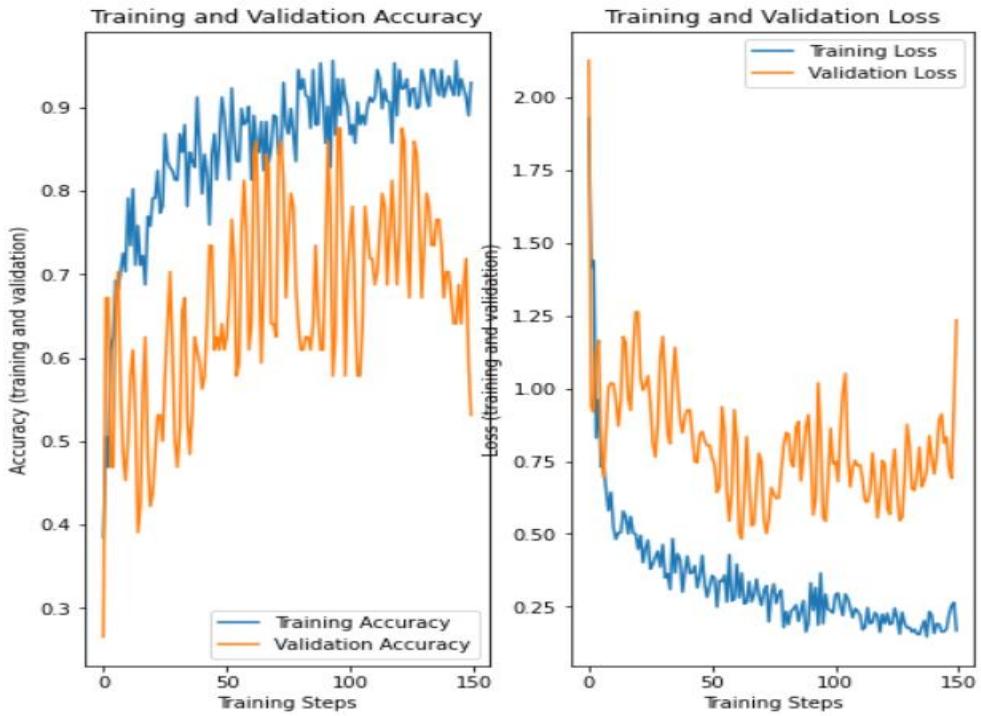
```
scores_hispa = loaded_model.evaluate(x_test[y_test[:,2]==1], y_test[y_test[:,2]==1])
print(f"Test Accuracy: {scores_hispa[1]*100}")
```

```
148/148 [==============================] - 1s 7ms/step
Test Accuracy: 54.39189076423645
```

```
scores_leafblast = loaded_model.evaluate(x_test[y_test[:,3]==1], y_test[y_test[:,3]==1])
print(f"Test Accuracy: {scores_leafblast[1]*100}")
```

```
234/234 [==============================] - 1s 6ms/step
Test Accuracy: 61.965811252593994
```

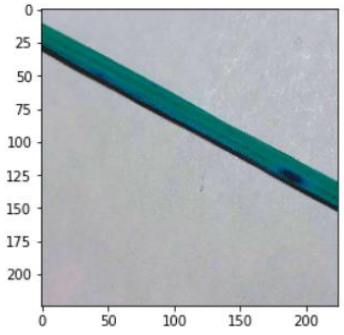
```
import matplotlib.pylab as plt
import numpy as np

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(EPOCHS)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.ylabel("Accuracy (training and validation)")
plt.xlabel("Training Steps")

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.ylabel("Loss (training and validation)")
plt.xlabel("Training Steps")
plt.show()
```



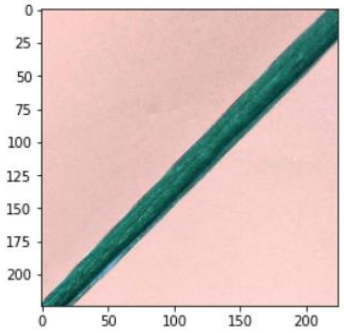

```
# Import OpenCV
import cv2

# Utility
import itertools
import random
from collections import Counter
from glob import iglob


def load_image(filename):
    img = cv2.imread(os.path.join(data_dir, validation_dir, filename))
    img = cv2.resize(img, (IMAGE_SIZE[0], IMAGE_SIZE[1]) )
    img = img /255

    return img


def predict(image):
    probabilities = model.predict(np.asarray([img]))[0]
    class_idx = np.argmax(probabilities)

    return {classes[class_idx]: probabilities[class_idx]}
```

```
for idx, filename in enumerate(random.sample(validation_generator.filenames, 5)):
    print("SOURCE: class: %s, file: %s" % (os.path.split(filename)[0], filename))

    img = load_image(filename)
    prediction = predict(img)
    print("PREDICTED: class: %s, Accuracy: %f" % (list(prediction.keys())[0], list(prediction.values())[0]))
    plt.imshow(img)
    plt.figure(idx)
    plt.show()
```

```
SOURCE: class: LeafBlast, file: LeafBlast/IMG_3106.jpg
PREDICTED: class: Hispa, Accuracy: 0.894776
```

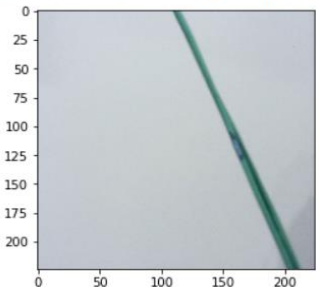


```
<Figure size 432x288 with 0 Axes>
SOURCE: class: Hispa, file: Hispa/IMG_20190419_131606.jpg
PREDICTED: class: Hispa, Accuracy: 0.956794
```

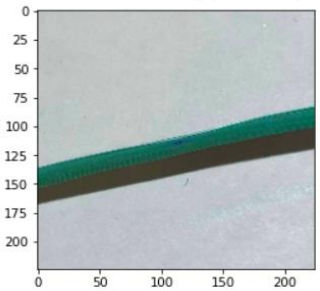


```
SOURCE: class: LeafBlast, file: LeafBlast/IMG_4868.jpg
PREDICTED: class: Hispa, Accuracy: 0.952164
```


```
SOURCE: class: LeafBlast, file: LeafBlast/IMG_4868.jpg
PREDICTED: class: Hispa, Accuracy: 0.952164
```

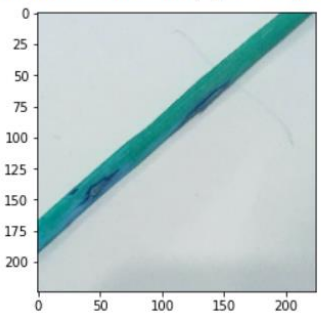



```
<Figure size 432x288 with 0 Axes>
SOURCE: class: LeafBlast, file: LeafBlast/IMG_3205.jpg
PREDICTED: class: Hispa, Accuracy: 0.978016
```



```
<Figure size 432x288 with 0 Axes>
SOURCE: class: LeafBlast, file: LeafBlast/IMG_5491.jpg
PREDICTED: class: Hispa, Accuracy: 0.895210
```


```
PREDICTED: class: Hispa, Accuracy: 0.895210
```

```
<Figure size 432x288 with 0 Axes>
```

Business Model for Disease Detection of Paddy Crop Using CNN

Introduction:

The agriculture sector faces numerous challenges, including crop diseases that can significantly impact yield and profitability. Traditional methods of disease detection often rely on manual inspection, which can be time-consuming and subjective. AI-powered disease detection models, particularly those based on convolutional neural networks (CNNs), offer a more efficient and accurate alternative.

In this part of the report, we will look at the business model suggested for the idea presented earlier. There are many business models available but we have chosen the 'Subscription Business Model' which is the one suited for our idea.

What is a Subscription Model?

Subscription business models are based on the idea of selling a product or service to receive monthly or yearly recurring subscription revenue. They focus on customer retention over customer acquisition. In essence, subscription business models focus on the way revenue is made so that a single customer pays multiple payments for prolonged access to a good or service instead of a large upfront one-time price.

Advantages of the Subscription Model

1. Predictable Revenue Streams: Subscriptions provide a steady stream of recurring revenue, facilitating financial planning and business growth.
2. Customer Retention: Subscription models encourage long-term relationships with customers, leading to higher retention rates and lifetime value.
3. Scalability: As the subscriber base grows, revenue scales proportionally without significant increases in operational costs.
4. Value-added Services: Subscribers can access additional services such as regular updates, personalized recommendations, and customer support.

Key Components of the Subscription Model

1. Tiered Pricing: Offer multiple subscription tiers with varying levels of features and support to cater to different customer segments.
2. Billing Cycle: Provide flexibility in billing cycles (monthly, quarterly, annually) to accommodate varying customer preferences.
3. Subscription Management Platform: Implement a robust platform for managing subscriptions, billing, payments, and customer interactions.
4. Value Proposition: Communicate the value proposition of the subscription offering, emphasizing the benefits of AI-powered disease detection for improving crop health and productivity.

Market Analysis

1. Identifying the Market: Our product is an AI model for disease detection in paddy crops. Therefore, the market we'll be launching into is the agriculture industry, specifically targeting farmers, agronomists, agricultural companies, and related stakeholders involved in paddy cultivation.

2. Collecting Data/Statistics Regarding the Market:
a. Global Agriculture Market Size: According to Statista, the global agriculture market was valued at over $8.7 trillion in 2020 and is projected to reach around $11 trillion by 2026.
b. Crop Protection Market: The global crop protection market size was valued at over $70 billion in 2020, according to Grand View Research.
c. Paddy Cultivation Statistics: Data on global paddy cultivation area, yield, and production can be sourced from organizations like the Food and Agriculture Organization (FAO) of the United Nations or national agricultural departments.
3. Forecasts/Predictions on the Market:
a. Regression Models or Time Series Forecasting: Since we're focusing on the agriculture market and paddy cultivation, you can use historical data on factors like crop yields, weather patterns, disease outbreaks, and agricultural technology adoption rates to perform forecasts. You can employ regression models or time series forecasting techniques like ARIMA (AutoRegressive Integrated Moving Average) or SARIMA (Seasonal ARIMA) to predict future trends in paddy cultivation and the demand for disease detection solutions.
b. Example Forecast: Using historical data on paddy crop diseases, agricultural technology adoption rates, and other relevant variables, you could build a regression model to predict the adoption rate of AI-based disease detection solutions in the paddy cultivation sector over the next few years. This forecast can help you estimate the potential market size and demand for your product/service

Forecasting and Financial Modeling

Utilize historical data and market trends to forecast subscription revenue, customer acquisition, and churn rates. Develop financial models to project revenue growth, profitability, and returnon investment (ROI) over time. Consider factors such as market penetration, pricing elasticity, and customer lifetime value (CLV) in your forecasts.

Marketing and Promotion Strategies

Deploy targeted marketing campaigns to raise awareness and drive adoption of the subscription offering among farmers and agricultural stakeholders. Leverage digital channels, industry partnerships, and educational content to showcase the value proposition of AI-based disease detection and highlight the benefits of subscribing.
Scalability & Future Plans
1. Continuously monitor market dynamics and customer feedback to adapt the subscription offering and pricing strategy accordingly.

2. Invest in ongoing product development and innovation to enhance the effectiveness and reliability of the disease detection AI model.
3. Foster partnerships with agricultural organizations, input suppliers, and agritech startups to expand market reach and accelerate customer acquisition.
4. Prioritize customer success and satisfaction through responsive support, training resources, and proactive engagement initiatives.

12.Financial Equation:

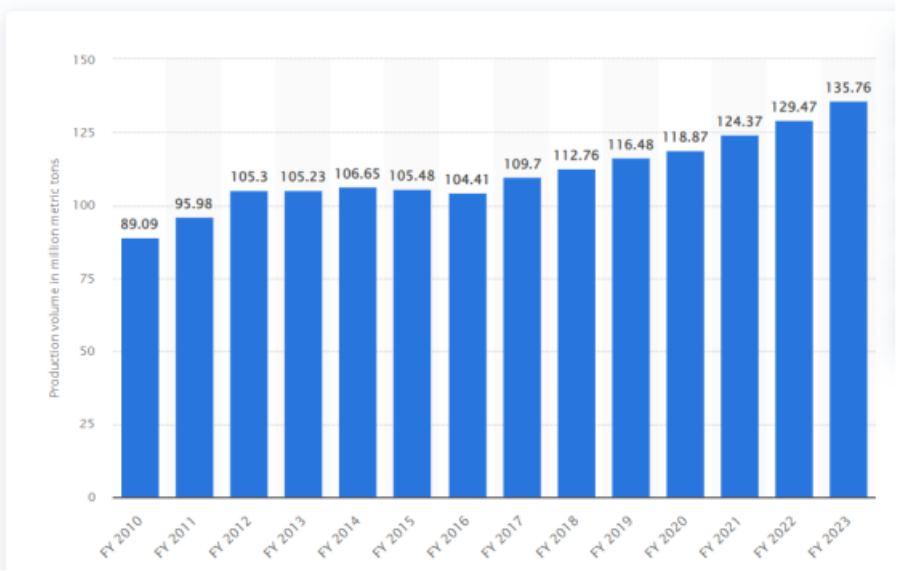
Y = mX(t) - c
Y = Total profit
m = cost of subscription * no. of subscribers
cost of subscription = area of land with paddy crops(in acers) * 0.2 * X(t)
c = app development cost and maintainence cost
Where, X(t) = Average farm income per Acre with respect to time
0.2 => 20% of the subscriber's income from harvest
Total cost for app and maintainance = $ 50,000 to $ 1,00,000
Total profit = (cost of subscription*no. of subscribers) – app development cost

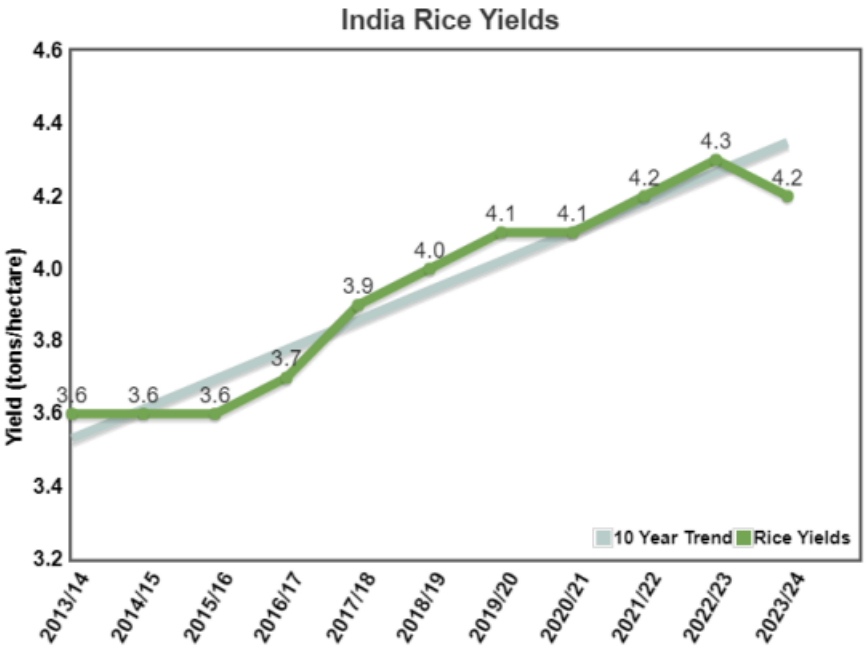
Table No. 20: Arrivals of paddy/rice in markets of major producing states in India during 1998-99 to 2000-01.

(000' Quintals)

S. N.	Name of important states	PADDY			RICE		
		1998-99	1999-2000	2000-01	1998-99	1999-2000	2000-01
1	Andhra Pradesh(47 Markets)	17277.6	16186.7	17764.3	14288.9	12433.9	14297.6
2	Bihar (37 Markets)	429.9	467.5	411.7	1107.7	1350.3	1127.2
3	Gujarat (46 Markets)	1133.8	1326.4	872.6	988.9	873.2	738.8
4	Haryana (23 Markets)	6414.7	5350.5	5408.6	4421.8	3581.2	3602.8
5	Karnataka (48 Markets)	2084.5	2006.4	2700.3	1645.1	1674.8	2399.8
6	Kerala (4 Markets)	233.7	236	198.3	932.6	879.8	730.4
7	Madhya Pradesh(37 Markets)	1756	2152.2	1425.9	1135.1	1414.3	956.7
8	Maharashtra (76 Markets)	464.7	412.1	344.3	656.7	763.1	944.2
9	Orissa (15 Markets)	973.6	973.6	973.6	1281.2	1281.2	1281.2
10	Punjab (38 Markets)	25175.2	17939	18124.1	17491.8	12898.7	11066.4
11	Tamil Nadu (46 Markets)	12470.9	13741.7	8216.5	8645.1	10710.1	6601.2
12	Uttar Pradesh (135 Markets)	19856.8	21274.4	19688	18561.6	18319.9	17668.8
13	West Bengal (39 Markets)	2547.3	2649.4	3314.7	6745.1	6749.6	6650.9
	Total (591 Markets)	90818.7	84715.9	79442.9	77901.6	72930.1	68066.0

Source : Department of Agriculture and Cooperation, New Delhi





Market Year	Area (1000 Ha)	Milled Production (1000 Tons)	Rough Production (1000 Tons)	Yield (T/Ha)
2013/2014	44,136	106,646	159,985	3.6
2014/2015	44,110	105,482	158,239	3.6
2015/2016	43,499	104,408	156,628	3.6
2016/2017	43,994	109,698	164,563	3.7
2017/2018	43,774	112,758	169,154	3.9
2018/2019	44,156	116,484	174,743	4.0
2019/2020	43,662	118,870	178,323	4.1
2020/2021	45,769	124,368	186,571	4.1
2021/2022	46,279	129,471	194,226	4.2
2022/2023	47,832	135,755	203,653	4.3
2023/2024	48,000	134,000	201,020	4.2

13. Conclusion:

Implementing a subscription business model for disease detection in paddy crops presents a compelling opportunity to provide ongoing value to farmers while generating recurring revenue for the business. By leveraging the scalability, predictability, and customer-centricity of the subscription model, we can establish a sustainable and impactful solution for addressing crop diseases and promoting agricultural productivity.

14.References

Patents: **https://patents.google.com**