Import neccessary packages

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
! kaggle datasets download emmarex/plantdisease
! unzip plantdisease.zip
```

In [17]:
```
import numpy as np
import pickle
import cv2
from os import listdir
from sklearn.preprocessing import LabelBinarizer
import keras
import tensorflow as tf
from tensorflow.keras.layers import *
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers 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 tensorflow.keras.optimizers import Adam
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from IPython.display import Image
```

In [1]:
```
%tensorflow_version 1.x
```

```
TensorFlow 1.x selected.
```

In [3]:
```
print(tf.__version__)
```

```
1.15.2
```

In [4]:
```
EPOCHS = 25
INIT_LR = 1e-3
BS = 32
default_image_size = tuple((256, 256))
image_size = 0
directory_root = '../input/plantvillage/'
width=256
height=256
depth=3
```

Function to convert images to array

In [5]:
```
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
```

Fetch images from directory

In [10]:
```
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 plant_folder in root_dir :
        plant_disease_folder_list = listdir(f"{directory_root}/{plant_folder}")

        for disease_folder in plant_disease_folder_list :
            # remove .DS_Store from list
            if disease_folder == ".DS_Store" :
                plant_disease_folder_list.remove(disease_folder)

        for plant_disease_folder in plant_disease_folder_list:
            print(f"[INFO] Processing {plant_disease_folder} ...")
            plant_disease_image_list = listdir(f"{directory_root}/{plant_folder}/{plant

            for single_plant_disease_image in plant_disease_image_list :
                if single_plant_disease_image == ".DS_Store" :
                    plant_disease_image_list.remove(single_plant_disease_image)

            for image in plant_disease_image_list[:200]:
                image_directory = f"{directory_root}/{plant_folder}/{plant_disease_fold
                if image_directory.endswith(".jpg") == True or image_directory.endswith
                    image_list.append(convert_image_to_array(image_directory))
                    label_list.append(plant_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: '../input/plantvillage/'
```

Get Size of Processed Image

In [ ]:
```
image_size = len(image_list)
```

Transform Image Labels uisng Scikit LabelBinarizer

In [ ]:
```
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 the classes

In [ ]:
```
print(label_binarizer.classes_)
```

```
['Pepper__bell___Bacterial_spot' 'Pepper__bell___healthy'
 'Potato___Early_blight' 'Potato___Late_blight' 'Potato___healthy'
 'Tomato_Bacterial_spot' 'Tomato_Early_blight' 'Tomato_Late_blight'
 'Tomato_Leaf_Mold' 'Tomato_Septoria_leaf_spot'
 'Tomato_Spider_mites_Two_spotted_spider_mite' 'Tomato__Target_Spot'
 'Tomato__Tomato_YellowLeaf__Curl_Virus' 'Tomato__Tomato_mosaic_virus'
 'Tomato_healthy']
```

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

In [ ]:
```
print("[INFO] Spliting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_s
```

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

In [ ]:
```
aug = ImageDataGenerator(
    rotation_range=25, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2,
    zoom_range=0.2,horizontal_flip=True,
    fill_mode="nearest")
```

In [ ]:
```
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(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))
```

Model Summary

In [ ]:

```
model.summary()
```

```
_________________________________________________________________
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, 1024)              57803776
_________________________________________________________________
activation_6 (Activation)    (None, 1024)              0
_________________________________________________________________
batch_normalization_6 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 15)                15375
_________________________________________________________________
activation_7 (Activation)    (None, 15)                0
=================================================================
Total params: 58,102,671
Trainable params: 58,099,791
Non-trainable params: 2,880
_________________________________________________________________
```

In [ ]:

```
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...
```

In [ ]:

```
history = model.fit_generator(
    aug.flow(x_train, y_train, batch_size=BS),
    validation_data=(x_test, y_test),
    steps_per_epoch=len(x_train) // BS,
    epochs=EPOCHS, verbose=1
    )
```

```
Epoch 1/25
73/73 [==============================] - 43s 583ms/step - loss: 0.2012 - acc: 0.9365 - v
al_loss: 0.5047 - val_acc: 0.9022
Epoch 2/25
73/73 [==============================] - 35s 479ms/step - loss: 0.1920 - acc: 0.9378 - v
al_loss: 0.8381 - val_acc: 0.8955
Epoch 3/25
73/73 [==============================] - 36s 491ms/step - loss: 0.1617 - acc: 0.9469 - v
al_loss: 0.9227 - val_acc: 0.8950
Epoch 4/25
73/73 [==============================] - 36s 490ms/step - loss: 0.1249 - acc: 0.9569 - v
al_loss: 0.9686 - val_acc: 0.8845
Epoch 5/25
73/73 [==============================] - 34s 470ms/step - loss: 0.1052 - acc: 0.9611 - v
al_loss: 0.2461 - val_acc: 0.9453
Epoch 6/25
73/73 [==============================] - 36s 487ms/step - loss: 0.0884 - acc: 0.9680 - v
al_loss: 0.1789 - val_acc: 0.9500
Epoch 7/25
73/73 [==============================] - 34s 468ms/step - loss: 0.0798 - acc: 0.9707 - v
al_loss: 0.1902 - val_acc: 0.9558
```

```
Epoch 8/25
73/73 [==============================] - 35s 483ms/step - loss: 0.0752 - acc: 0.9729 - v
al_loss: 0.5176 - val_acc: 0.9218
Epoch 9/25
73/73 [==============================] - 35s 476ms/step - loss: 0.0848 - acc: 0.9698 - v
al_loss: 0.3263 - val_acc: 0.9277
Epoch 10/25
73/73 [==============================] - 35s 482ms/step - loss: 0.0737 - acc: 0.9736 - v
al_loss: 0.2105 - val_acc: 0.9495
Epoch 11/25
73/73 [==============================] - 35s 481ms/step - loss: 0.0699 - acc: 0.9746 - v
al_loss: 0.7867 - val_acc: 0.8999
Epoch 12/25
73/73 [==============================] - 35s 473ms/step - loss: 0.0598 - acc: 0.9772 - v
al_loss: 0.1873 - val_acc: 0.9516
Epoch 13/25
73/73 [==============================] - 35s 478ms/step - loss: 0.0604 - acc: 0.9791 - v
al_loss: 0.3801 - val_acc: 0.9381
Epoch 14/25
73/73 [==============================] - 34s 468ms/step - loss: 0.0526 - acc: 0.9800 - v
al_loss: 0.5882 - val_acc: 0.9135
Epoch 15/25
73/73 [==============================] - 36s 487ms/step - loss: 0.0512 - acc: 0.9810 - v
al_loss: 0.3619 - val_acc: 0.9357
Epoch 16/25
73/73 [==============================] - 35s 482ms/step - loss: 0.0520 - acc: 0.9808 - v
al_loss: 0.1112 - val_acc: 0.9642
Epoch 17/25
73/73 [==============================] - 34s 472ms/step - loss: 0.0491 - acc: 0.9820 - v
al_loss: 0.6471 - val_acc: 0.9195
Epoch 18/25
73/73 [==============================] - 35s 484ms/step - loss: 0.0478 - acc: 0.9824 - v
al_loss: 0.1848 - val_acc: 0.9582
Epoch 19/25
73/73 [==============================] - 34s 469ms/step - loss: 0.0443 - acc: 0.9843 - v
al_loss: 0.1428 - val_acc: 0.9602
Epoch 20/25
73/73 [==============================] - 35s 483ms/step - loss: 0.0455 - acc: 0.9832 - v
al_loss: 0.0754 - val_acc: 0.9755
Epoch 21/25
73/73 [==============================] - 35s 476ms/step - loss: 0.0384 - acc: 0.9857 - v
al_loss: 0.5299 - val_acc: 0.9233
Epoch 22/25
73/73 [==============================] - 35s 481ms/step - loss: 0.0463 - acc: 0.9842 - v
al_loss: 0.0925 - val_acc: 0.9701
Epoch 23/25
73/73 [==============================] - 35s 481ms/step - loss: 0.0440 - acc: 0.9844 - v
al_loss: 1.2281 - val_acc: 0.8855
Epoch 24/25
73/73 [==============================] - 34s 472ms/step - loss: 0.0396 - acc: 0.9847 - v
al_loss: 0.3955 - val_acc: 0.9337
Epoch 25/25
73/73 [==============================] - 35s 483ms/step - loss: 0.0394 - acc: 0.9860 - v
al_loss: 0.1300 - val_acc: 0.9645
```

Plot the train and val curve

In [ ]:

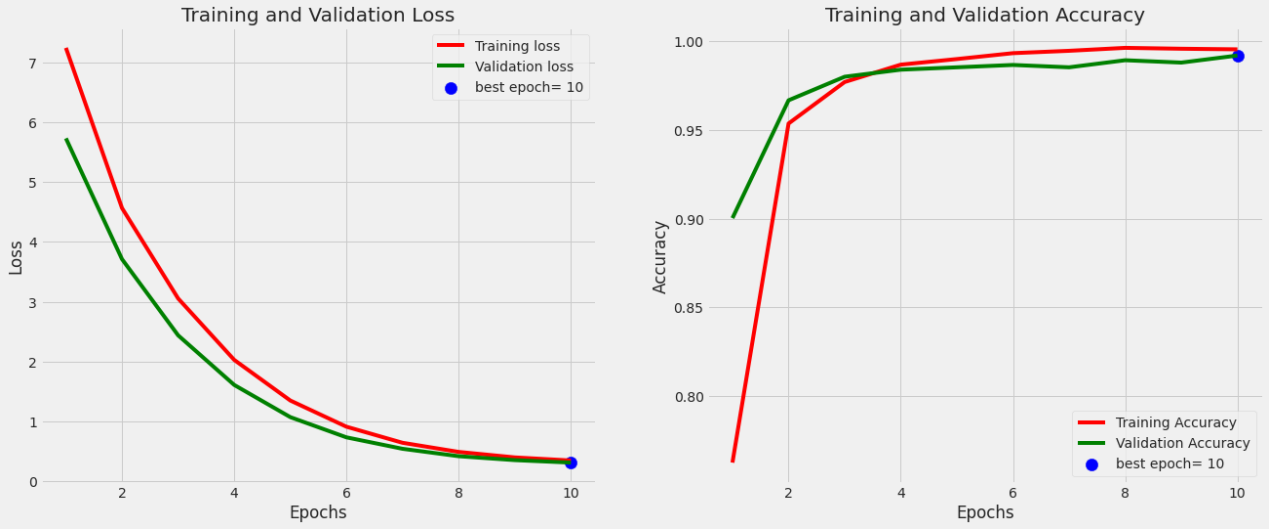
```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
```

```
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 [18]:

Out[18]:



Model Accuracy

In [ ]:

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

```
[INFO] Calculating model accuracy
591/591 [==============================] - 1s 2ms/step
Test Accuracy: 96.44670223221561
```

Save model using Pickle

In [ ]:

```
# save the model to disk
print("[INFO] Saving model...")
pickle.dump(model,open('cnn_model.pkl', 'wb'))
```

```
[INFO] Saving model...
```

In [6]:

```
loaded_model = pickle.load(open('/content/drive/My Drive/Colab Notebooks/cnn_model.pkl'
```

```
WARNING:tensorflow:From /tensorflow-1.15.2/python3.7/tensorflow_core/python/ops/resource
```

```
_variable_ops.py:1630: calling BaseResourceVariable.__init__ (from tensorflow.python.op
s.resource_variable_ops) with constraint is deprecated and will be removed in a future v
ersion.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From /tensorflow-1.15.2/python3.7/keras/backend/tensorflow_backend.p
y:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /tensorflow-1.15.2/python3.7/tensorflow_core/python/ops/math_gra
d.py:1424: where (from tensorflow.python.ops.array_ops) is deprecated and will be remove
d in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /tensorflow-1.15.2/python3.7/keras/backend/tensorflow_backend.p
y:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variab
les instead.
```

In [7]:

```
image_dir="/content/drive/My Drive/Colab Notebooks/sample_inputs/potato_healthy.JPG"

im=convert_image_to_array(image_dir)
np_image_li = np.array(im, dtype=np.float16) / 225.0
npp_image = np.expand_dims(np_image_li, axis=0)
```

In [8]:

```
result=loaded_model.predict(npp_image)
print(result)
```

```
[[6.0540270e-03 2.7965103e-05 8.7222460e-09 4.1387661e-08 9.9292141e-01
  1.7614186e-11 2.3912577e-08 1.5094177e-10 1.2893961e-12 2.2474073e-06
  2.1706064e-11 5.7275906e-07 6.3949135e-10 9.9365879e-04 1.8972030e-10]]
```

In [15]:

```
label_binarizer = ['Pepper__bell___Bacterial_spot','Pepper__bell___healthy',
 'Potato___Early_blight','Potato___Late_blight','Potato___healthy',
 'Tomato_Bacterial_spot','Tomato_Early_blight','Tomato_Late_blight',
 'Tomato_Leaf_Mold','Tomato_Septoria_leaf_spot',
 'Tomato_Spider_mites_Two_spotted_spider_mite','Tomato__Target_Spot',
 'Tomato__Tomato_YellowLeaf__Curl_Virus','Tomato__Tomato_mosaic_virus',
 'Tomato_healthy']
```

In [16]:

```
itemindex = np.where(result==np.max(result))
print("probability:"+str(np.max(result))+"\n"+label_binarizer[itemindex[1][0]])
```

```
probability:0.9929214
Potato___healthy
```

In [ ]: