Import neccessary packages

In [1]:
```
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
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

```
Using TensorFlow backend.
```

In [2]:
```
# example of loading an image with the Keras API
from keras.preprocessing.image import load_img
```

In [3]:
```
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 [4]:
```
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 [5]:
```
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_folde
                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 ...
[INFO] Processing Pepper__bell___Bacterial_spot ...
[INFO] Processing Potato___healthy ...
[INFO] Processing Tomato_Leaf_Mold ...
[INFO] Processing Tomato__Tomato_YellowLeaf__Curl_Virus ...
[INFO] Processing Tomato_Bacterial_spot ...
[INFO] Processing Tomato_Septoria_leaf_spot ...
[INFO] Processing Tomato_healthy ...
[INFO] Processing Tomato_Spider_mites_Two_spotted_spider_mite ...
[INFO] Processing Tomato_Early_blight ...
[INFO] Processing Tomato__Target_Spot ...
[INFO] Processing Pepper__bell___healthy ...
[INFO] Processing Potato___Late_blight ...
[INFO] Processing Tomato_Late_blight ...
[INFO] Processing Potato___Early_blight ...
[INFO] Processing Tomato__Tomato_mosaic_virus ...
[INFO] Image loading completed
```

Get Size of Processed Image

In [6]:
```
image_size = len(image_list)
```

Transform Image Labels uisng Scikit Learn (http://scikit-learn.org/)'s LabelBinarizer

In [7]:
```
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 [8]:
```
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 [9]:
```
np_image_list = np.array(image_list, dtype=np.float16) / 225.0
```

In [10]:
```
print("[INFO] Spliting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_si
```

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

In [11]:
```
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 [12]:

```
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 [13]:
```
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 [14]:
```
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 [15]:
```
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
    )
```

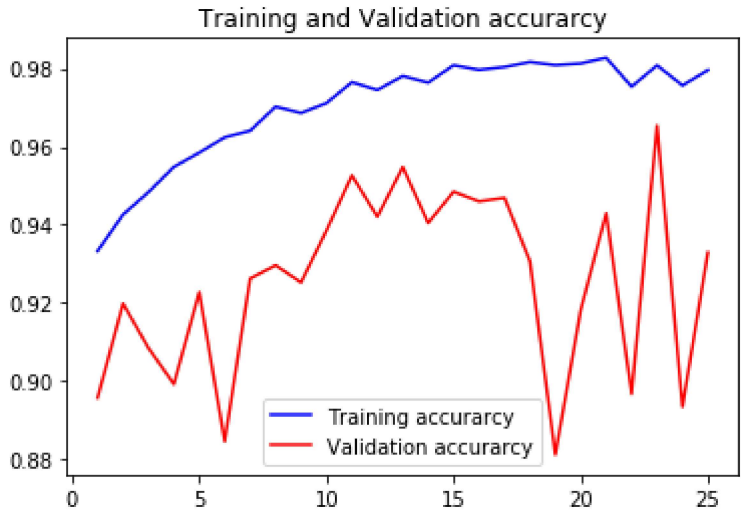
```
Epoch 1/25
73/73 [==============================] - 33s 447ms/step - loss: 0.2150 - acc: 0.9332 -
val_loss: 0.9762 - val_acc: 0.8957
Epoch 2/25
73/73 [==============================] - 28s 389ms/step - loss: 0.1737 - acc: 0.9426 -
val_loss: 0.4727 - val_acc: 0.9198
Epoch 3/25
73/73 [==============================] - 29s 392ms/step - loss: 0.1532 - acc: 0.9483 -
val_loss: 0.5002 - val_acc: 0.9084
Epoch 4/25
73/73 [==============================] - 29s 398ms/step - loss: 0.1310 - acc: 0.9548 -
val_loss: 0.6854 - val_acc: 0.8992
Epoch 5/25
73/73 [==============================] - 29s 401ms/step - loss: 0.1175 - acc: 0.9584 -
val_loss: 0.3923 - val_acc: 0.9227
Epoch 6/25
73/73 [==============================] - 29s 404ms/step - loss: 0.1033 - acc: 0.9624 -
val_loss: 1.2521 - val_acc: 0.8845
Epoch 7/25
73/73 [==============================] - 29s 396ms/step - loss: 0.0976 - acc: 0.9641 -
val_loss: 0.4309 - val_acc: 0.9262
Epoch 8/25
73/73 [==============================] - 29s 395ms/step - loss: 0.0816 - acc: 0.9702 -
val_loss: 0.3631 - val_acc: 0.9296
Epoch 9/25
73/73 [==============================] - 29s 399ms/step - loss: 0.0887 - acc: 0.9686 -
val_loss: 0.4080 - val_acc: 0.9252
Epoch 10/25
73/73 [==============================] - 29s 398ms/step - loss: 0.0804 - acc: 0.9711 -
val_loss: 0.2780 - val_acc: 0.9385
Epoch 11/25
73/73 [==============================] - 30s 407ms/step - loss: 0.0653 - acc: 0.9765 -
val_loss: 0.1558 - val_acc: 0.9526
Epoch 12/25
73/73 [==============================] - 30s 414ms/step - loss: 0.0685 - acc: 0.9745 -
val_loss: 0.2453 - val_acc: 0.9420
Epoch 13/25
73/73 [==============================] - 29s 402ms/step - loss: 0.0605 - acc: 0.9781 -
val_loss: 0.1460 - val_acc: 0.9548
Epoch 14/25
73/73 [==============================] - 29s 400ms/step - loss: 0.0667 - acc: 0.9764 -
val_loss: 0.3001 - val_acc: 0.9404
Epoch 15/25
73/73 [==============================] - 29s 400ms/step - loss: 0.0509 - acc: 0.9808 -
val_loss: 0.2357 - val_acc: 0.9484
Epoch 16/25
73/73 [==============================] - 29s 396ms/step - loss: 0.0547 - acc: 0.9797 -
val_loss: 0.1928 - val_acc: 0.9460
Epoch 17/25
73/73 [==============================] - 29s 397ms/step - loss: 0.0499 - acc: 0.9804 -
val_loss: 0.1855 - val_acc: 0.9469
Epoch 18/25
73/73 [==============================] - 29s 401ms/step - loss: 0.0513 - acc: 0.9817 -
val_loss: 0.2931 - val_acc: 0.9306
Epoch 19/25
73/73 [==============================] - 29s 402ms/step - loss: 0.0536 - acc: 0.9808 -
val_loss: 1.1409 - val_acc: 0.8811
Epoch 20/25
73/73 [==============================] - 29s 398ms/step - loss: 0.0481 - acc: 0.9813 -
val_loss: 0.5076 - val_acc: 0.9182
Epoch 21/25
```

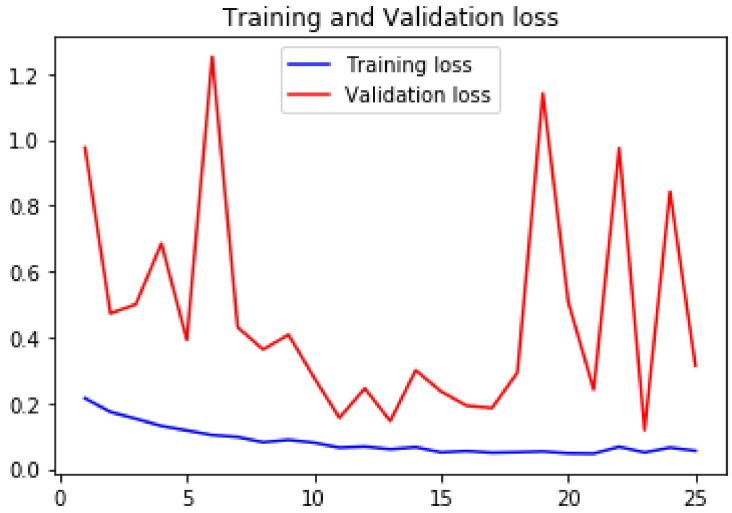
```
73/73 [==============================] - 29s 401ms/step - loss: 0.0472 - acc: 0.9828 -
val_loss: 0.2416 - val_acc: 0.9429
Epoch 22/25
73/73 [==============================] - 29s 400ms/step - loss: 0.0679 - acc: 0.9753 -
val_loss: 0.9751 - val_acc: 0.8967
Epoch 23/25
73/73 [==============================] - 29s 396ms/step - loss: 0.0506 - acc: 0.9808 -
val_loss: 0.1184 - val_acc: 0.9654
Epoch 24/25
73/73 [==============================] - 29s 402ms/step - loss: 0.0653 - acc: 0.9757 -
val_loss: 0.8421 - val_acc: 0.8934
Epoch 25/25
73/73 [==============================] - 29s 397ms/step - loss: 0.0557 - acc: 0.9796 -
val_loss: 0.3141 - val_acc: 0.9329
```

Plot the train and val curve

In [16]:
```
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()
```





Model Accuracy

In [17]:
```
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: 93.28821303477343
```

Save model using Pickle

In [18]:
```
# save the model to disk
print("[INFO] Saving model...")
pickle.dump(model,open('cnn_model.pkl', 'wb'))
```

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

In [19]:
```
loaded_model = pickle.load(open('cnn_model.pkl', 'rb'))
```

In [20]:
```
image_dir="/kaggle/input/plantvillage/PlantVillage/Tomato_healthy/00bce074-967b-4d50-967
im1 = load_img(image_dir)

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 [21]:
```
result=model.predict(npp_image)
print(result)
```

```
[[4.33317704e-15 1.41996352e-16 2.59286953e-10 1.73510052e-19
  2.06448882e-14 4.19498485e-18 1.84960648e-19 1.13825245e-14
  2.12339646e-16 3.88705144e-11 1.63137889e-15 5.11029760e-12
  3.82474260e-16 1.58414418e-16 1.00000000e+00]]
```

In [22]:
```
itemindex = np.where(result==np.max(result))
print("probability:"+str(np.max(result))+"\n"+label_binarizer.classes_[itemindex[1][0]])
```

```
probability:1.0
Tomato_healthy
```