Mounting Dataset From Drive.

```
from google.colab import drive
drive.mount('/content/drive')
```

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

1.Import Libraries.

```
# Import Libraries
import warnings
warnings.filterwarnings("ignore")

import os
import glob
import matplotlib.pyplot as plt
import tensorflow as tf
# Keras API
import keras
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten
from keras.layers import Conv2D,MaxPooling2D,Activation,AveragePooling2D,BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
```

2.Load Data into Train and Test Variables.

```
# My data is in google drive.
train_dir ="drive/My Drive/train_set/"
test_dir="drive/My Drive/test_data/"
```

3.Function To count Images In Each Data Set.

```
# function to get count of images
def get_files(directory):
  if not os.path.exists(directory):
```

```
    return 0
  count=0
  for current_path,dirs,files in os.walk(directory):
    for dr in dirs:
      count+= len(glob.glob(os.path.join(current_path,dr+"/*")))
  return count

train_samples =get_files(train_dir)
num_classes=len(glob.glob(train_dir+"/*"))
test_samples=get_files(test_dir) # For testing i took only few samples from unseen data. we c
print(num_classes,"Classes")
print(train_samples,"Train images")
print(test_samples,"Test images")
```

```
10 Classes
12820 Train images
3202 Test images
```

```
# Preprocessing data.
train_datagen=ImageDataGenerator(rescale=1./255,
                                 shear_range=0.2,
                                 zoom_range=0.2,
                                 validation_split=0.2, # validation split 20%.
                                 horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)


# set height and width and color of input image.
img_width,img_height =256,256
input_shape=(img_width,img_height,3)
batch_size =32

train_generator =train_datagen.flow_from_directory(train_dir,
                                                   target_size=(img_width,img_height),
                                                   batch_size=batch_size)
test_generator=test_datagen.flow_from_directory(test_dir,shuffle=True,
                                                   target_size=(img_width,img_height),
                                                   batch_size=batch_size)
```

```
Found 12820 images belonging to 10 classes.
Found 3201 images belonging to 10 classes.
```

```
# The name of the 12 diseases.
train_generator.class_indices
```

```
{'Tomato_Bacterial_spot': 0,
 'Tomato_Early_blight': 1,
 'Tomato_Late_blight': 2,
 'Tomato_Leaf_Mold': 3,
 'Tomato_Septoria_leaf_spot': 4,
 'Tomato_Spider_mites_Two_spotted_spider_mite': 5,
```

```
 'Tomato__Target_Spot': 6,
 'Tomato__Tomato_YellowLeaf__Curl_Virus': 7,
 'Tomato__Tomato_mosaic_virus': 8,
 'Tomato_healthy': 9}
```

4.CNN Parameter Building.

```
# CNN building.
model = Sequential()
model.add(Conv2D(32, (5, 5),input_shape=input_shape,activation='relu'))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Conv2D(32, (3, 3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(128,activation='relu'))
model.add(Dense(num_classes,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 252, 252, 32)      2432

 max_pooling2d (MaxPooling2D  (None, 84, 84, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 82, 82, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 41, 41, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 39, 39, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 19, 19, 64)       0
 2D)

 flatten (Flatten)           (None, 23104)             0

 dense (Dense)               (None, 512)               11829760

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 128)               65664

 dense_2 (Dense)             (None, 10)                1290
```

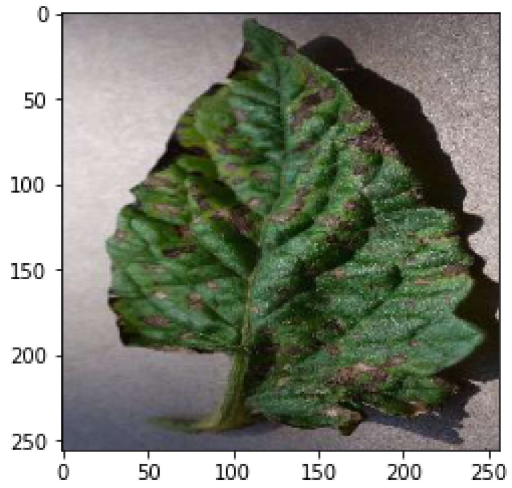
```
=================================================================
Total params: 11,926,890
Trainable params: 11,926,890
Non-trainable params: 0
_________________________________________________________________
```

```
model_layers = [ layer.name for layer in model.layers]
print('layer name : ',model_layers)
```

```
layer name :  ['conv2d', 'max_pooling2d', 'conv2d_1', 'max_pooling2d_1', 'conv2d_2', 'ma
```

```
# Take one image to visualize it's changes after every layer
from keras.preprocessing import image
import numpy as np
img1 = image.load_img('/content/drive/My Drive/train_set/Tomato_Early_blight/TomatoEarlybligh
plt.imshow(img1);
```

```
#preprocess image
img1 = image.load_img('/content/drive/My Drive/train_set/Tomato_Early_blight/TomatoEarlybligh
img = image.img_to_array(img1)
img = img/255
img = np.expand_dims(img, axis=0)
```




```
# Visualizing output after every layer.
from keras.models import Model
conv2d_output = Model(inputs=model.input, outputs=model.get_layer('conv2d').output)
max_pooling2d_output = Model(inputs=model.input,outputs=model.get_layer('max_pooling2d').outp
conv2d_1_output = Model(inputs=model.input,outputs=model.get_layer('conv2d_1').output)
max_pooling2d_1_output = Model(inputs=model.input,outputs=model.get_layer('max_pooling2d_1').
conv2d_2_output = Model(inputs=model.input,outputs=model.get_layer('conv2d_2').output)
max_pooling2d_2_output = Model(inputs=model.input,outputs=model.get_layer('max_pooling2d_2').
flatten_output = Model(inputs=model.input,outputs=model.get_layer('flatten').output)
conv2d_features = conv2d_output.predict(img)
max_pooling2d_features = max_pooling2d_output.predict(img)
```

```
conv2d_1_features = conv2d_1_output.predict(img)
max_pooling2d_1_features = max_pooling2d_1_output.predict(img)
conv2d_2_features = conv2d_2_output.predict(img)
max_pooling2d_2_features = max_pooling2d_2_output.predict(img)
flatten_features = flatten_output.predict(img)
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<l
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<l
```

5.Visualizing The Image After Every Layer.

↳ *6 cells hidden*

6.Training The Model.

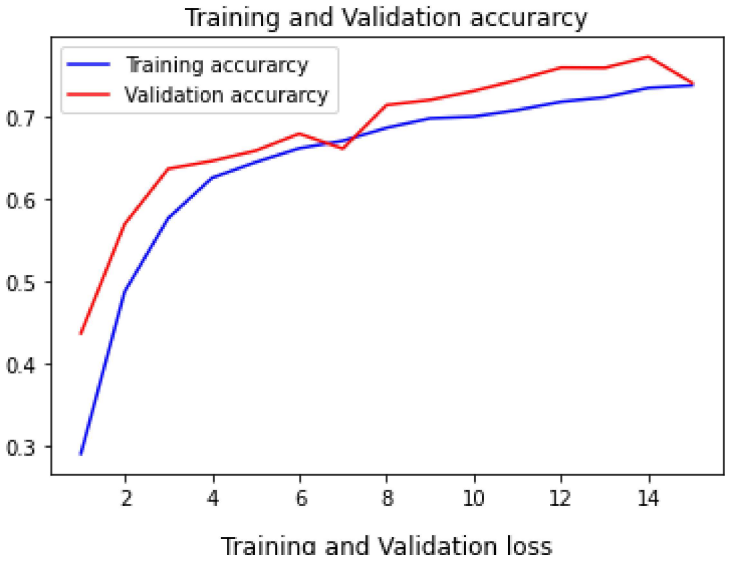
```
# validation data.
validation_generator = train_datagen.flow_from_directory(
    train_dir, # same directory as training data
    target_size=(img_height, img_width),
    batch_size=batch_size)


# Model building to get trained with parameters.
opt=tf.keras.optimizers.Adam(lr=0.001)
model.compile(optimizer=opt,loss='categorical_crossentropy',metrics=['accuracy'])
train=model.fit_generator(train_generator,

                          epochs=15,
                          steps_per_epoch=train_generator.samples // batch_size,
                          validation_data=validation_generator,
                          validation_steps= validation_generator.samples// batch_size,verbose
```

```
400/400 [==============================] - 1473s 4s/step - loss: 1.4787 - accuracy: 0.48
Epoch 3/15
400/400 [==============================] - 1490s 4s/step - loss: 1.2241 - accuracy: 0.57
Epoch 4/15
400/400 [==============================] - 1471s 4s/step - loss: 1.0944 - accuracy: 0.62
Epoch 5/15
400/400 [==============================] - 1459s 4s/step - loss: 1.0429 - accuracy: 0.64
Epoch 6/15
400/400 [==============================] - 1458s 4s/step - loss: 0.9853 - accuracy: 0.66
Epoch 7/15
400/400 [==============================] - 1452s 4s/step - loss: 0.9529 - accuracy: 0.67
Epoch 8/15
400/400 [==============================] - 1517s 4s/step - loss: 0.9144 - accuracy: 0.68
Epoch 9/15
```

```
400/400 [==============================] - 1456s 4s/step - loss: 0.8756 - accuracy: 0.69
Epoch 10/15
400/400 [==============================] - 1456s 4s/step - loss: 0.8714 - accuracy: 0.69
Epoch 11/15
400/400 [==============================] - 1448s 4s/step - loss: 0.8382 - accuracy: 0.70
Epoch 12/15
400/400 [==============================] - 1439s 4s/step - loss: 0.8071 - accuracy: 0.71
Epoch 13/15
400/400 [==============================] - 1437s 4s/step - loss: 0.7954 - accuracy: 0.72
Epoch 14/15
400/400 [==============================] - 1440s 4s/step - loss: 0.7683 - accuracy: 0.73
Epoch 15/15
400/400 [==============================] - 1440s 4s/step - loss: 0.7571 - accuracy: 0.73
```

7.Plot For Accuracy And Losses.

```
accuracy = train.history['accuracy']
val_acc = train.history['val_accuracy']
loss = train.history['loss']
val_loss = train.history['val_loss']
epochs = range(1, len(accuracy) + 1)
#Train and validation accuracy
plt.plot(epochs, accuracy, '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()
```



8. Evaluate model using unseen data.

```
score,accuracy =model.evaluate(test_generator,verbose=1)
print("Test score is {}".format(score))
print("Test accuracy is {}".format(accuracy))
```

```
101/101 [==============================] - 833s 8s/step - loss: 0.7878 - accuracy: 0.723
Test score is 0.7877618670463562
Test accuracy is 0.7238363027572632
```

9.Saving Model.

```
# Save entire model with optimizer, architecture, weights and training configuration.
from keras.models import load_model
model.save('crop.h5')
```

```
# Save model weights.
from keras.models import load_model
model.save_weights('crop_weights.h5')
```

```
# Get classes of model trained on
classes = train_generator.class_indices
classes
```

```
{'Tomato_Bacterial_spot': 0,
 'Tomato_Early_blight': 1,
 'Tomato_Late_blight': 2,
```

```
        'Tomato_Leaf_Mold': 3,
        'Tomato_Septoria_leaf_spot': 4,
        'Tomato_Spider_mites_Two_spotted_spider_mite': 5,
        'Tomato__Target_Spot': 6,
        'Tomato__Tomato_YellowLeaf__Curl_Virus': 7,
        'Tomato__Tomato_mosaic_virus': 8,
        'Tomato_healthy': 9}
```

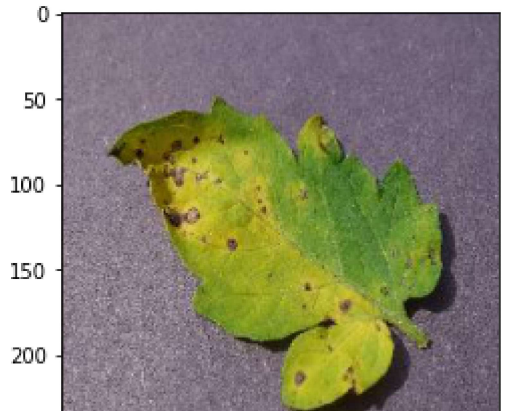
10.Load Model.

```
# Loading model and predict.
from keras.models import load_model
model=load_model('crop.h5')

Classes = ["Potato___Early_blight","Potato___Late_blight","Potato___healthy","Tomato___Bacter
```

11.Time For Predictions.

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

# Pre-Processing test data same as train data.
img_width=256
img_height=256
#model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

from keras.preprocessing import image

def prepare(img_path):
    img = image.load_img(img_path, target_size=(256, 256))
    x = image.img_to_array(img)
    x = x/255
    return np.expand_dims(x, axis=0)


result = model.predict([prepare('/content/drive/My Drive/test_data/Tomato_Septoria_leaf_spot/
disease=image.load_img('/content/drive/My Drive/test_data/Tomato_Septoria_leaf_spot/septorial
plt.imshow(disease)
y=np.argmax(result,axis=1)
print (Classes[int(y+3)])
#print (Classes[int(result)])
```

```
Tomato___Septoria_leaf_spot
```



12.Convert Model To "tflite format.".

- This conversion is done because to make our model interpertable with App.
- tflite is tensorflowlite made for mobile versions.

```
import tensorflow as tf
converter = tf.lite.TFLiteConverter.from_keras_model_file('crop.h5')
tfmodel = converter.convert()
open ("output.tflite" , "wb") .write(tfmodel)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-53-297e3c4914b1> in <module>()
      1 import tensorflow as tf
----> 2 converter = tf.lite.TFLiteConverter.from_keras_model_file('crop.h5')
      3 tfmodel = converter.convert()
      4 open ("output.tflite" , "wb") .write(tfmodel)

AttributeError: type object 'TFLiteConverterV2' has no attribute
'from_keras_model_file'
```

SEARCH STACK OVERFLOW

✓ 0s completed at 2:21 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.