```
!pip install kaggle
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.5.13)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle) (2023.5.7)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.27.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.65.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.26.16)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (2.0.1
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.4)
```

```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
! chmod 600 ~/.kaggle/kaggle.json
```

```
mkdir: cannot create directory ‘/root/.kaggle’: File exists
```

```
import os
os.listdir()
```

```
['.config', 'kaggle.json', 'sample_data']
```

```
! kaggle datasets download arjuntejaswi/plant-village
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
100% 329M/329M [00:18<00:00, 18.9MB/s]
```

```
! unzip /content/plant-village.zip
```

```
  inflating: PlantVillage/Tomato_healthy/fda63621-b0eb-4938-8ec4-8afffc81ddd6___GH_HL Leaf 268.JPG
  inflating: PlantVillage/Tomato_healthy/fdbbea63-18e0-401f-8269-64ba005ab53f___RS_HL 0213.JPG
  inflating: PlantVillage/Tomato_healthy/fde29807-8223-4fc5-a06b-7cc93101a6d1___GH_HL Leaf 174.JPG
  inflating: PlantVillage/Tomato_healthy/fe0b76dc-4263-4cda-8840-310ebe72a432___GH_HL Leaf 490.JPG
  inflating: PlantVillage/Tomato_healthy/fe0e2d7f-3aee-4c36-8fe1-661b4eacc994___GH_HL Leaf 507.1.JPG
  inflating: PlantVillage/Tomato_healthy/fe28e4c7-0c35-4f52-984e-0e60f33a2c6e___GH_HL Leaf 198.JPG
  inflating: PlantVillage/Tomato_healthy/fe8f8808-2631-491e-a46b-bd2a1a4958e7___GH_HL Leaf 213.1.JPG
  inflating: PlantVillage/Tomato_healthy/feda8fd2-1d18-443e-a7d9-15bd6bf8ce66___RS_HL 0332.JPG
  inflating: PlantVillage/Tomato_healthy/ff354b62-5981-43d1-8cfe-ac58bc20ca20___GH_HL Leaf 221.JPG
  inflating: PlantVillage/Tomato_healthy/ff774aec-2504-4d11-8a61-2fd74c689a6f___RS_HL 9904.JPG
  inflating: PlantVillage/Tomato_healthy/ff8b36d5-feaf-4d2d-8126-18670a312657___RS_HL 0229.JPG
  inflating: PlantVillage/Tomato_healthy/ffb39943-eabb-42cf-ad09-b17019e46d66___RS_HL 9871.JPG
  inflating: PlantVillage/Tomato_healthy/ffd8aa68-138f-4114-96c7-21eef72e1e13___RS_HL 9881.JPG
```

```
import tensorflow as tf
from tensorflow.keras import models , layers
import matplotlib.pyplot as plt
```

```
# defining the varibles
IMAGE_SIZE = 256
BATCH_SIZE  = 32
CHANNELS = 3
EPOCHS = 50
```

```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "PlantVillage" ,
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
)
```

```
    Found 20638 files belonging to 15 classes.
```

```
dataset.class_names
```

```
    ['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']
```

```
len(dataset)
```

```
    645
```

```
for img_batch , label_batch in dataset.take(1):
  print(img_batch.shape)
  print(label_batch.numpy())
```

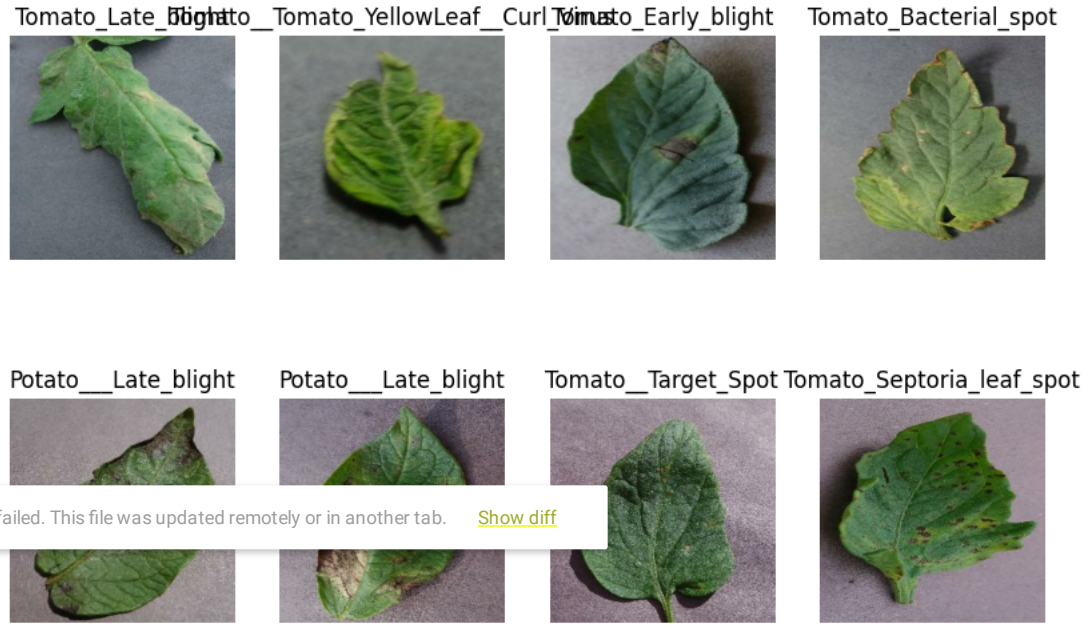
```
    (32, 256, 256, 3)
    [11 11 14  0 14  8  1  8  9  5  0  1  9 11  0 11 14  5 10  9  0 12 14  9
     10 12 13 14  9  0  4  3]
```

```
class_names = dataset.class_names
len(class_names)
```

```
    15
```

```
class_names = dataset.class_names
plt.figure(figsize = (10, 10))
```

```
for img_batch , label_batch in dataset.take(1):
  for i in range(12):
    ax = plt.subplot(3 , 4 , i+1)
    plt.title(class_names[label_batch[i].numpy()])
    plt.imshow(img_batch[i].numpy().astype('uint'))
    plt.axis('off')
```




```
!pip install split-folders
!splitfolders --ratio 0.8 0.1 0.1 -- ./PlantVillage/
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
Copying files: 20639 files [00:05, 3496.49 files/s]
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

tr_datagen = ImageDataGenerator(
    rescale = 1./255 ,
    rotation_range = 10 ,
    horizontal_flip = True ,
)


tr_generator = tr_datagen.flow_from_directory(
    "output/train" ,
    batch_size = BATCH_SIZE ,
    target_size = (IMAGE_SIZE , IMAGE_SIZE) ,

    class_mode = "sparse"
)
```

```
Found 16504 images belonging to 15 classes.
```

```
classes = list[tr_generator.class_indices.keys]
print(classes)
```

```
list[<built-in method keys of dict object at 0x7f3ab07dc900>]
```

```
# for validatio dataset
val_data = ImageDataGenerator(
    rescale = 1. / 255 ,
    horizontal_flip = True ,
    rotation_range = 10


)

val_gen = val_data.flow_from_directory(
    "output/val"  ,
    batch_size = BATCH_SIZE ,
    target_size  = (IMAGE_SIZE , IMAGE_SIZE) ,
    class_mode = "sparse"
)
```

```
 Found 2058 images belonging to 15 classes.
```

```
test_data = ImageDataGenerator(
        rescale=1./255,
        rotation_range=10,
        horizontal_flip=True)

test_gen = test_data.flow_from_directory(
        'output/test',
        target_size=(IMAGE_SIZE,IMAGE_SIZE),
        batch_size=32,
        class_mode="sparse"
)
```



```
no_classes  = len(class_names)
image_input = (IMAGE_SIZE , IMAGE_SIZE , CHANNELS )

model = models.Sequential ([
    layers.InputLayer (input_shape = image_input ) ,
    layers.Conv2D (32 , kernel_size = (3,3) , activation = "relu") ,
    layers.MaxPooling2D((2,2)),

    layers.Conv2D( 64 , (3,3) , activation ="relu"),
    layers.MaxPooling2D((2,2)) ,

    layers.Conv2D( 64 , (3,3) , activation ="relu"),
    layers.MaxPooling2D((2,2)) ,

    layers.Conv2D( 64 , (3,3) , activation ="relu"),
    layers.MaxPooling2D((2,2)) ,

    layers.Conv2D( 64 , (3,3) , activation ="relu"),
    layers.MaxPooling2D((2,2)) ,

    layers.Conv2D( 64 , (3,3) , activation ="relu"),
    layers.MaxPooling2D((2,2)) ,


    layers.Flatten() ,

    layers.Dense(64 , activation = "relu"),
    layers.Dense(no_classes , activation = "softmax" )



])

model.build(input_shape = image_input )


model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 254, 254, 32)      896
```

```
 max_pooling2d_4 (MaxPooling  (None, 127, 127, 32)     0
 2D)

 conv2d_5 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_5 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_6 (Conv2D)           (None, 60, 60, 64)        36928

 max_pooling2d_6 (MaxPooling  (None, 30, 30, 64)       0
 2D)

 conv2d_7 (Conv2D)           (None, 28, 28, 64)        36928

 max_pooling2d_7 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 conv2d_8 (Conv2D)           (None, 12, 12, 64)        36928

 max_pooling2d_8 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_9 (Conv2D)           (None, 4, 4, 64)          36928

 max_pooling2d_9 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 64)                16448

 dense_1 (Dense)             (None, 15)                975
```



```
 Non-trainable params: 0
 _________________________________________________________________
```

```
model.compile(
    optimizer = "adam",
    loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits = False) ,
    metrics = ['accuracy']

)


history = model.fit(
    tr_generator  ,
    steps_per_epoch = 47 ,
    batch_size = BATCH_SIZE ,

    validation_data = val_gen ,
    validation_steps = 7 ,

    epochs = 30,
    verbose = 1
)
```

```
Epoch 2/30
47/47 [==============================] - 30s 631ms/step - loss: 2.4265 - accuracy: 0.1968 - val_loss: 2.3115 - val_accuracy: 0.
Epoch 3/30
47/47 [==============================] - 28s 597ms/step - loss: 2.2549 - accuracy: 0.2600 - val_loss: 2.3230 - val_accuracy: 0.
Epoch 4/30
47/47 [==============================] - 28s 597ms/step - loss: 2.0106 - accuracy: 0.3424 - val_loss: 1.8176 - val_accuracy: 0.
Epoch 5/30
47/47 [==============================] - 40s 855ms/step - loss: 1.8662 - accuracy: 0.3730 - val_loss: 1.7102 - val_accuracy: 0.
Epoch 6/30
47/47 [==============================] - 30s 634ms/step - loss: 1.7372 - accuracy: 0.4275 - val_loss: 1.6153 - val_accuracy: 0.
Epoch 7/30
47/47 [==============================] - 30s 631ms/step - loss: 1.5685 - accuracy: 0.4814 - val_loss: 1.7550 - val_accuracy: 0.
Epoch 8/30
47/47 [==============================] - 29s 628ms/step - loss: 1.5201 - accuracy: 0.4907 - val_loss: 1.4000 - val_accuracy: 0.
Epoch 9/30
47/47 [==============================] - 30s 636ms/step - loss: 1.5321 - accuracy: 0.4967 - val_loss: 1.5219 - val_accuracy: 0.
Epoch 10/30
47/47 [==============================] - 29s 611ms/step - loss: 1.3732 - accuracy: 0.5419 - val_loss: 1.4955 - val_accuracy: 0.
Epoch 11/30
47/47 [==============================] - 38s 814ms/step - loss: 1.3473 - accuracy: 0.5592 - val_loss: 1.3852 - val_accuracy: 0.
```

```
47/47 [==============================] - 32s 689ms/step - loss: 1.2631 - accuracy: 0.5791 - val_loss: 1.1321 - val_accuracy: 0.
Epoch 14/30
47/47 [==============================] - 40s 840ms/step - loss: 1.1293 - accuracy: 0.6090 - val_loss: 1.1770 - val_accuracy: 0.
Epoch 15/30
47/47 [==============================] - 35s 736ms/step - loss: 1.1127 - accuracy: 0.6150 - val_loss: 1.2695 - val_accuracy: 0.
Epoch 16/30
47/47 [==============================] - 33s 703ms/step - loss: 1.0174 - accuracy: 0.6569 - val_loss: 0.9741 - val_accuracy: 0.
Epoch 17/30
47/47 [==============================] - 39s 841ms/step - loss: 1.0668 - accuracy: 0.6463 - val_loss: 1.0915 - val_accuracy: 0.
Epoch 18/30
47/47 [==============================] - 29s 610ms/step - loss: 0.9468 - accuracy: 0.6908 - val_loss: 0.9743 - val_accuracy: 0.
Epoch 19/30
47/47 [==============================] - 29s 618ms/step - loss: 0.8896 - accuracy: 0.7101 - val_loss: 0.9063 - val_accuracy: 0.
Epoch 20/30
47/47 [==============================] - 31s 661ms/step - loss: 0.7794 - accuracy: 0.7360 - val_loss: 0.8459 - val_accuracy: 0.
Epoch 21/30
47/47 [==============================] - 31s 652ms/step - loss: 0.8160 - accuracy: 0.7254 - val_loss: 1.0017 - val_accuracy: 0.
Epoch 22/30
47/47 [==============================] - 29s 606ms/step - loss: 0.7628 - accuracy: 0.7394 - val_loss: 0.6844 - val_accuracy: 0.
Epoch 23/30
47/47 [==============================] - 29s 625ms/step - loss: 0.7919 - accuracy: 0.7181 - val_loss: 0.7594 - val_accuracy: 0.
Epoch 24/30
47/47 [==============================] - 29s 625ms/step - loss: 0.7174 - accuracy: 0.7507 - val_loss: 0.8167 - val_accuracy: 0.
Epoch 25/30
47/47 [==============================] - 31s 670ms/step - loss: 0.7319 - accuracy: 0.7533 - val_loss: 0.6927 - val_accuracy: 0.
Epoch 26/30
47/47 [==============================] - 28s 605ms/step - loss: 0.6383 - accuracy: 0.7839 - val_loss: 0.6106 - val_accuracy: 0.
Epoch 27/30
47/47 [==============================] - 29s 626ms/step - loss: 0.6906 - accuracy: 0.7453 - val_loss: 0.5950 - val_accuracy: 0.
Epoch 28/30
47/47 [==============================] - 29s 624ms/step - loss: 0.6231 - accuracy: 0.7939 - val_loss: 0.9132 - val_accuracy: 0.
Epoch 29/30
47/47 [==============================] - 29s 617ms/step - loss: 0.6693 - accuracy: 0.7773 - val_loss: 0.7364 - val_accuracy: 0.
Epoch 30/30
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
history.params
```

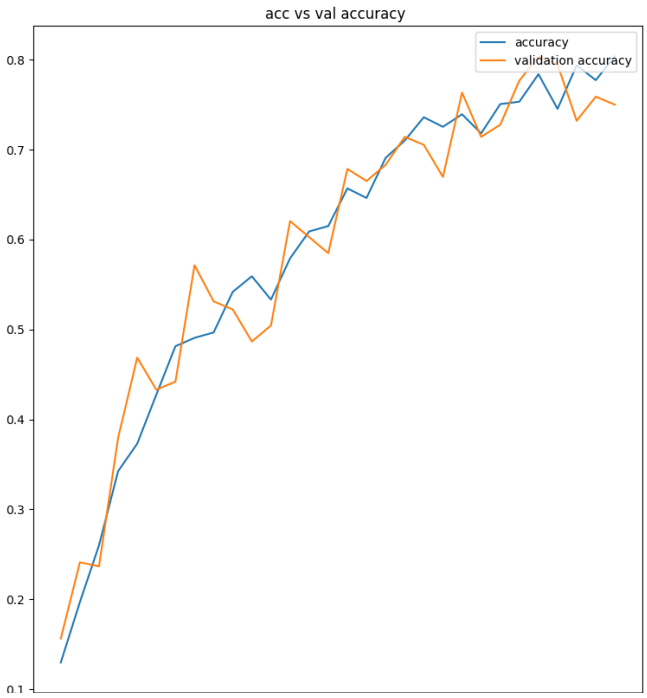
```
{'verbose': 1, 'epochs': 30, 'steps': 47}
```

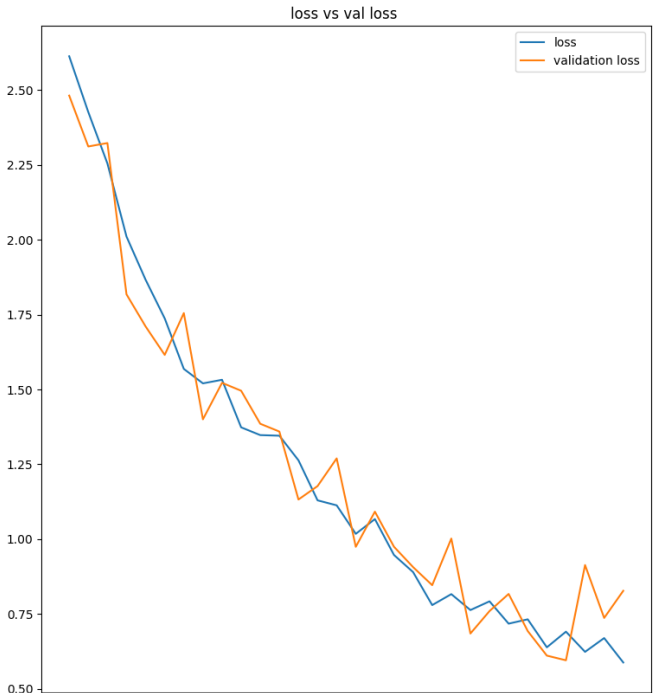
```
history.history.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
fig = plt.figure(figsize = ( 20 , 10 ))

# first cavas

plt.subplot( 1, 2, 1 )
plt.plot(range(30) , acc , label = "accuracy")
plt.plot(range(30) , val_acc , label = "validation accuracy")
plt.title("acc vs val accuracy ")
plt.legend(loc = "upper right")


# second canvas

plt.subplot( 1, 2, 2 )
plt.plot(range(30) , loss , label = "loss")
plt.plot(range(30) , val_loss , label = "validation loss")
plt.title("loss vs val loss ")
plt.legend(loc = "upper right")

plt.show()
```



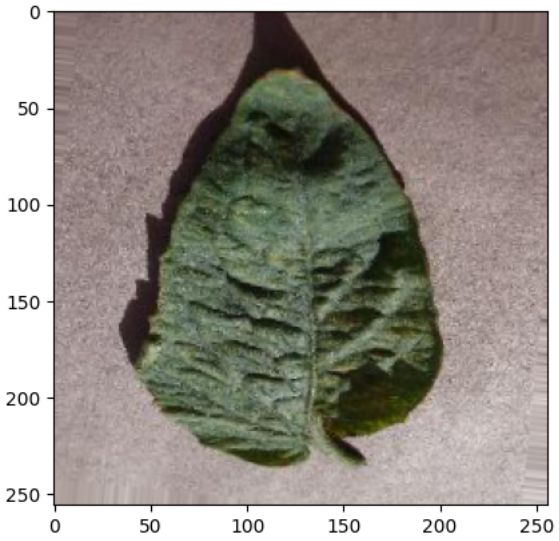



```
import numpy as np

for img_batch , img_label in test_gen:
```



```
  f_label = int (img_label[0])

  print("first image")
  plt.imshow(f_image)
  print(f'label is {class_names[f_label]}')

  # testing on the test batch

  batch_predictions = model.predict(img_batch)

  print("Label predict for first img ", class_names[np.argmax(batch_predictions[0])])

  break
```

```
first image
label is Tomato_Spider_mites_Two_spotted_spider_mite
1/1 [==============================] - 0s 33ms/step
Label predict for first img  Tomato_Spider_mites_Two_spotted_spider_mite
```



```
def prediction (model , image):
  # step 1 - convert kro req size me
  img_array = tf.keras.preprocessing.image.img_to_array(image)
  img_array = tf.expand_dims(img_array  , 0 )

  prediction = model.predict (img_array )
  predicted_class = class_names [np.argmax(prediction[0])]

  confidence = round(100 * (np.max(prediction[0])), 2)
  return predicted_class , confidence


plt.figure ( figsize = (20 ,20 ))
for imgs , labels in test_gen:

  for i in range(12):
    ax = plt.subplot(10 , 2 , i+1)

    plt.imshow(imgs[i])

    pre_class , conf = prediction(model , imgs[i])
    actual =  class_names [int(labels[i])]

    plt.title(f'Predicted {pre_class} , actual {actual} and  confidence is {conf}')


  break
```

```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
```



```
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 19ms/step
```


```
model.save("tom-pot.h5")
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff