Name - Vishakha Dhonde

Roll no - COBB055

```
!nvidia-smi
```

```
Sat May  6 07:48:02 2023       
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.85.12    Driver Version: 525.85.12    CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   56C    P8    12W /  70W |      0MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
                                                                               
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

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

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

```
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
#from keras.applications.
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
```

```
IMAGE_SIZE = [224, 224]
```

```
train_path = '/content/drive/MyDrive/New Plant Diseases Dataset(Augmented)/train'
valid_path = '/content/drive/MyDrive/New Plant Diseases Dataset(Augmented)/valid'
```

```
vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.
58892288/58889256 [==============================] - 1s 0us/step
```

```
for layer in vgg.layers:
  layer.trainable = False
```

```
folders = glob('/content/drive/MyDrive/New Plant Diseases Dataset(Augmented)/train/*')
```

```
x = Flatten()(vgg.output)
```

```
prediction = Dense(len(folders), activation='softmax')(x)
```

```
model = Model(inputs=vgg.input, outputs=prediction)
```

```
model.summary()
```

```
Model: "model_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
conv_0 (Conv2D)              (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 10)                250890
=================================================================
Total params: 14,965,578
Trainable params: 250,890
Non-trainable params: 14,714,688
_________________________________________________________________
```

```
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
```

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

```
training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/New Plant Diseases Dataset(Augmented)
                                                 target_size = (224, 224),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
Found 3038 images belonging to 10 classes.
```

```
test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/New Plant Diseases Dataset(Augmented)/vali
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
Found 499 images belonging to 10 classes.
```

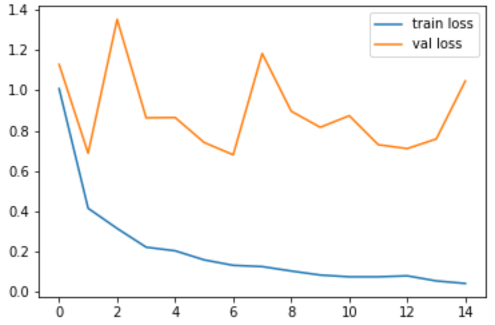
```
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=15,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

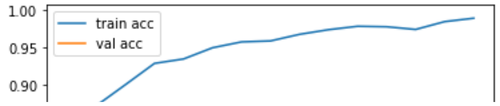
```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecatec
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/15
95/95 [==============================] - 2144s 22s/step - loss: 1.4255 - accuracy: 0.6327 - val_loss: 1.1286 - val_accuracy: 0.6333
Epoch 2/15
95/95 [==============================] - 44s 459ms/step - loss: 0.4205 - accuracy: 0.8731 - val_loss: 0.6876 - val_accuracy: 0.7575
Epoch 3/15
95/95 [==============================] - 44s 458ms/step - loss: 0.3371 - accuracy: 0.8942 - val_loss: 1.3525 - val_accuracy: 0.6533
Epoch 4/15
95/95 [==============================] - 44s 458ms/step - loss: 0.2330 - accuracy: 0.9294 - val_loss: 0.8635 - val_accuracy: 0.7315
Epoch 5/15
95/95 [==============================] - 44s 458ms/step - loss: 0.2175 - accuracy: 0.9279 - val_loss: 0.8649 - val_accuracy: 0.7134
Epoch 6/15
95/95 [==============================] - 44s 459ms/step - loss: 0.1596 - accuracy: 0.9470 - val_loss: 0.7410 - val_accuracy: 0.7695
Epoch 7/15
95/95 [==============================] - 44s 460ms/step - loss: 0.1275 - accuracy: 0.9603 - val_loss: 0.6799 - val_accuracy: 0.7836
Epoch 8/15
95/95 [==============================] - 44s 459ms/step - loss: 0.1210 - accuracy: 0.9622 - val_loss: 1.1831 - val_accuracy: 0.7315
Epoch 9/15
95/95 [==============================] - 43s 457ms/step - loss: 0.1188 - accuracy: 0.9626 - val_loss: 0.8963 - val_accuracy: 0.7455
Epoch 10/15
95/95 [==============================] - 44s 458ms/step - loss: 0.0832 - accuracy: 0.9726 - val_loss: 0.8165 - val_accuracy: 0.7475
Epoch 11/15
95/95 [==============================] - 44s 458ms/step - loss: 0.0786 - accuracy: 0.9750 - val_loss: 0.8740 - val_accuracy: 0.7655
Epoch 12/15
95/95 [==============================] - 44s 458ms/step - loss: 0.0664 - accuracy: 0.9799 - val_loss: 0.7300 - val_accuracy: 0.7896
Epoch 13/15
95/95 [==============================] - 43s 457ms/step - loss: 0.0608 - accuracy: 0.9778 - val_loss: 0.7107 - val_accuracy: 0.7816
Epoch 14/15
95/95 [==============================] - 43s 457ms/step - loss: 0.0644 - accuracy: 0.9817 - val_loss: 0.7588 - val_accuracy: 0.7896
Epoch 15/15
95/95 [==============================] - 44s 458ms/step - loss: 0.0354 - accuracy: 0.9911 - val_loss: 1.0466 - val_accuracy: 0.7315
```

```
import matplotlib.pyplot as plt

# plot the loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```





```
from tensorflow.keras.models import load_model

model.save('model_VGG16.h5')
```

```
layer_names = []

for layer in model.layers[:16]:
  layer_names.append(layer.name)
print(layer_names)
```

```
['input_1', 'block1_conv1', 'block1_conv2', 'block1_pool', 'block2_conv1', 'block2_conv2', 'block2_pool', 'block3_conv1', 'block3_conv2
```

```
model.layers[0].output
```

```
<KerasTensor: shape=(None, 224, 224, 3) dtype=float32 (created by layer 'input_1')>
```