Pollen Grains image classification with VGGnet

The dataset for classification of pollen grains is taken from Kaggle. This dataset includes images of Brazilian Savannah pollens. There are 23 types of pollen in this dataset. Out of them 11 pollen types are included in this project. VGGnet is used for classification of these pollen images.

```
import os
import numpy as np
import pandas as pd
import cv2
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
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
import os
import matplotlib.pyplot as plt


train_path ='/content/seed/'
valid_path = '/content/test_seed/'

len(os.listdir(train_path))
12
```

Although there are 11 folders, it is showing a count of 12. This is due to the presence of .ipynb_checkpoints file.

```
for i in os.listdir(train_path):
    print("No of Images in ",i," category is
",len(os.listdir(os.path.join(train_path,i))))
```

```
No of Images in  arecaceae  category is  12
No of Images in  urachloa  category is  12
No of Images in  dipteryx  category is  12
No of Images in  syagrus  category is  13
No of Images in  eucalipto  category is  12
No of Images in  cecropeia  category is  12
No of Images in  .ipynb_checkpoints  category is  0
No of Images in  matayba  category is  12
No of Images in  protium  category is  12
No of Images in  mabea  category is  12
No of Images in  myrcia  category is  12
No of Images in  anadenanthera  category is  12
```

Thus, We need to delete the .ipynb_checkpoints directory.

```
import shutil
shutil.rmtree('/content/seed/.ipynb_checkpoints')

for i in os.listdir(train_path):
    print("No of Images in ",i," category is
",len(os.listdir(os.path.join(train_path,i))))

No of Images in  arecaceae  category is  12
No of Images in  urachloa  category is  12
No of Images in  dipteryx  category is  12
No of Images in  syagrus  category is  13
No of Images in  eucalipto  category is  12
No of Images in  cecropeia  category is  12
No of Images in  matayba  category is  12
No of Images in  protium  category is  12
No of Images in  mabea  category is  12
No of Images in  myrcia  category is  12
No of Images in  anadenanthera  category is  12

for i in os.listdir(valid_path):
    print("No of Images in ",i," category is
",len(os.listdir(os.path.join(valid_path,i))))

No of Images in  test_arecaceae  category is  4
```

```
No of Images in  test_mabea  category is  4
No of Images in  test_cecropeia  category is  4
No of Images in  test_syagrus  category is  4
No of Images in  test_myrcia  category is  4
No of Images in  test_dipteryx  category is  4
No of Images in  .ipynb_checkpoints  category is  0
No of Images in  test_protium  category is  4
No of Images in  test_matayba  category is  4
No of Images in  test_eucalipto  category is  4
No of Images in  test_anadenanthera  category is  4
No of Images in  test_urachloa  category is  4
```

```
import shutil
shutil.rmtree('/content/test_seed/.ipynb_checkpoints')
```

```
for i in os.listdir(valid_path):
    print("No of Images in ",i," category is
",len(os.listdir(os.path.join(valid_path,i))))
```

```
No of Images in  test_arecaceae  category is  4
No of Images in  test_mabea  category is  4
No of Images in  test_cecropeia  category is  4
No of Images in  test_syagrus  category is  4
No of Images in  test_myrcia  category is  4
No of Images in  test_dipteryx  category is  4
No of Images in  test_protium  category is  4
No of Images in  test_matayba  category is  4
No of Images in  test_eucalipto  category is  4
No of Images in  test_anadenanthera  category is  4
No of Images in  test_urachloa  category is  4
```

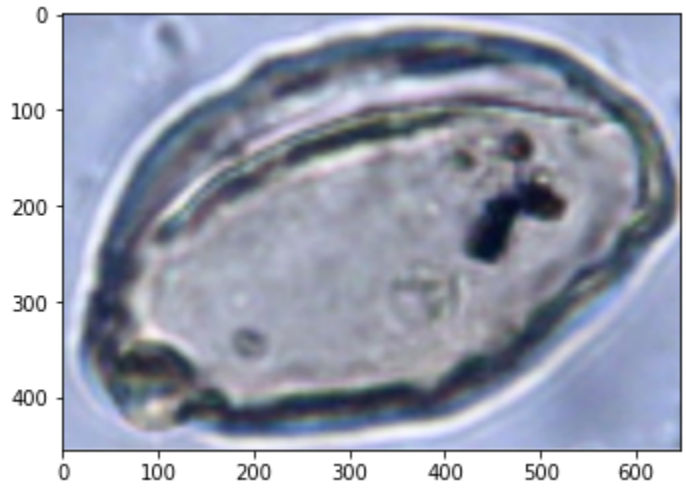
To provide path for jpg files

```
image_files = glob(train_path + '/*/*.jp*g')
valid_image_files = glob(valid_path + '/*/*.jp*g')
```

```
len(image_files)
132
```

```
len(valid_image_files)
44

plt.imshow(image.img_to_array(image.load_img(np.random.choice(image_files)
)).astype('uint8'))
plt.show()
```




```
IMAGE_SIZE = [100, 100]

vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet',
include_top=False)

for layer in vgg16.layers:
    layer.trainable = False

folders = glob('/content/seed/*')
folders
['/content/seed/arecaceae',
 '/content/seed/urachloa',
 '/content/seed/dipteryx',
 '/content/seed/syagrus',
 '/content/seed/eucalipto',
 '/content/seed/cecropeia',
 '/content/seed/matayba',
 '/content/seed/protium',
 '/content/seed/mabea',
 '/content/seed/myrcia',
 '/content/seed/anadenanthera']
```

```
test_folders = glob('/content/test_seed/*')

len(folders)
11
```

i.e 11 types of pollen (classes) are considered.

```
len(test_folders)
11
```

Implementing VGGnet

```
x = Flatten()(vgg16.output)
prediction = Dense(len(folders), activation='softmax')(x)

model = Model(inputs=vgg16.input, outputs=prediction)

model.summary()

Model: "model_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 100, 100, 3)]     0

 block1_conv1 (Conv2D)       (None, 100, 100, 64)      1792

 block1_conv2 (Conv2D)       (None, 100, 100, 64)      36928

 block1_pool (MaxPooling2D)  (None, 50, 50, 64)        0

 block2_conv1 (Conv2D)       (None, 50, 50, 128)       73856

 block2_conv2 (Conv2D)       (None, 50, 50, 128)       147584

 block2_pool (MaxPooling2D)  (None, 25, 25, 128)       0

 block3_conv1 (Conv2D)       (None, 25, 25, 256)       295168
```

```
 block3_conv2 (Conv2D)       (None, 25, 25, 256)       590080

 block3_conv3 (Conv2D)       (None, 25, 25, 256)       590080

 block3_pool (MaxPooling2D)  (None, 12, 12, 256)       0

 block4_conv1 (Conv2D)       (None, 12, 12, 512)       1180160

 block4_conv2 (Conv2D)       (None, 12, 12, 512)       2359808

 block4_conv3 (Conv2D)       (None, 12, 12, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 6, 6, 512)         0

 block5_conv1 (Conv2D)       (None, 6, 6, 512)         2359808

 block5_conv2 (Conv2D)       (None, 6, 6, 512)         2359808

 block5_conv3 (Conv2D)       (None, 6, 6, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 3, 3, 512)         0

 flatten_7 (Flatten)         (None, 4608)              0

 dense_7 (Dense)             (None, 11)                50699

=================================================================
Total params: 14,765,387
Trainable params: 50,699
Non-trainable params: 14,714,688
_________________________________________________________________
```

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

```
from tensorflow.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(train_path,
                                                 target_size = (100, 100),
                                                 batch_size = 32,
                                                 class_mode =
'categorical')
```

Found 132 images belonging to 11 classes.

```
valid_set = test_datagen.flow_from_directory(valid_path,
                                             target_size = (100, 100),
                                             batch_size = 32,
                                             class_mode = 'categorical')
```

Found 44 images belonging to 11 classes.

To View Classes

```
print(valid_set.class_indices)
```

```
{'test_anadenanthera': 0, 'test_arecaceae': 1, 'test_cecropeia': 2,
'test_dipteryx': 3, 'test_eucalipto': 4, 'test_mabea': 5, 'test_matayba':
6, 'test_myrcia': 7, 'test_protium': 8, 'test_syagrus': 9,
'test_urachloa': 10}
```

Training and Testing of VGGnet

```
r = model.fit_generator(
  training_set,
  validation_data=valid_set,
  epochs=25)
```

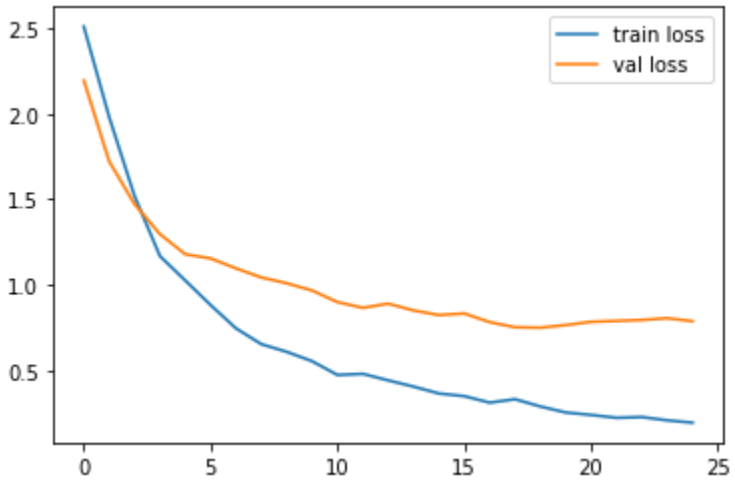
```
Epoch 1/25
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4:
UserWarning: `Model.fit_generator` is deprecated and will be removed in a
future version. Please use `Model.fit`, which supports generators.
  after removing the cwd from sys.path.
5/5 [==============================] - 2s 213ms/step - loss: 2.5082 -
accuracy: 0.1212 - val_loss: 2.1932 - val_accuracy: 0.2500
Epoch 2/25
5/5 [==============================] - 1s 148ms/step - loss: 1.9795 -
accuracy: 0.3788 - val_loss: 1.7203 - val_accuracy: 0.4318
Epoch 3/25
5/5 [==============================] - 1s 152ms/step - loss: 1.5181 -
accuracy: 0.6364 - val_loss: 1.4710 - val_accuracy: 0.5000
Epoch 4/25
5/5 [==============================] - 1s 140ms/step - loss: 1.1679 -
accuracy: 0.7576 - val_loss: 1.2958 - val_accuracy: 0.5909
Epoch 5/25
5/5 [==============================] - 1s 139ms/step - loss: 1.0248 -
accuracy: 0.7803 - val_loss: 1.1781 - val_accuracy: 0.6591
Epoch 6/25
5/5 [==============================] - 1s 171ms/step - loss: 0.8816 -
accuracy: 0.8636 - val_loss: 1.1538 - val_accuracy: 0.6364
Epoch 7/25
5/5 [==============================] - 1s 148ms/step - loss: 0.7453 -
accuracy: 0.8485 - val_loss: 1.0958 - val_accuracy: 0.6591
Epoch 8/25
5/5 [==============================] - 1s 143ms/step - loss: 0.6530 -
accuracy: 0.8636 - val_loss: 1.0425 - val_accuracy: 0.6591
Epoch 9/25
5/5 [==============================] - 1s 142ms/step - loss: 0.6083 -
accuracy: 0.8485 - val_loss: 1.0086 - val_accuracy: 0.6136
Epoch 10/25
5/5 [==============================] - 1s 144ms/step - loss: 0.5535 -
accuracy: 0.9015 - val_loss: 0.9665 - val_accuracy: 0.6591
Epoch 11/25
5/5 [==============================] - 1s 178ms/step - loss: 0.4732 -
accuracy: 0.9167 - val_loss: 0.8986 - val_accuracy: 0.6818
Epoch 12/25
5/5 [==============================] - 1s 152ms/step - loss: 0.4797 -
accuracy: 0.9242 - val_loss: 0.8653 - val_accuracy: 0.6818
```

```
Epoch 13/25
5/5 [==============================] - 1s 143ms/step - loss: 0.4416 -
accuracy: 0.9470 - val_loss: 0.8896 - val_accuracy: 0.6818
Epoch 14/25
5/5 [==============================] - 1s 177ms/step - loss: 0.4052 -
accuracy: 0.9091 - val_loss: 0.8499 - val_accuracy: 0.7500
Epoch 15/25
5/5 [==============================] - 1s 147ms/step - loss: 0.3651 -
accuracy: 0.9545 - val_loss: 0.8232 - val_accuracy: 0.7045
Epoch 16/25
5/5 [==============================] - 1s 149ms/step - loss: 0.3495 -
accuracy: 0.9318 - val_loss: 0.8323 - val_accuracy: 0.7273
Epoch 17/25
5/5 [==============================] - 1s 153ms/step - loss: 0.3116 -
accuracy: 0.9545 - val_loss: 0.7817 - val_accuracy: 0.7500
Epoch 18/25
5/5 [==============================] - 1s 180ms/step - loss: 0.3316 -
accuracy: 0.9394 - val_loss: 0.7519 - val_accuracy: 0.7727
Epoch 19/25
5/5 [==============================] - 1s 146ms/step - loss: 0.2892 -
accuracy: 0.9697 - val_loss: 0.7492 - val_accuracy: 0.7727
Epoch 20/25
5/5 [==============================] - 1s 143ms/step - loss: 0.2541 -
accuracy: 0.9924 - val_loss: 0.7644 - val_accuracy: 0.7500
Epoch 21/25
5/5 [==============================] - 1s 146ms/step - loss: 0.2399 -
accuracy: 0.9697 - val_loss: 0.7837 - val_accuracy: 0.7273
Epoch 22/25
5/5 [==============================] - 1s 142ms/step - loss: 0.2231 -
accuracy: 0.9848 - val_loss: 0.7888 - val_accuracy: 0.7273
Epoch 23/25
5/5 [==============================] - 1s 142ms/step - loss: 0.2274 -
accuracy: 0.9697 - val_loss: 0.7943 - val_accuracy: 0.7273
Epoch 24/25
5/5 [==============================] - 1s 178ms/step - loss: 0.2084 -
accuracy: 0.9848 - val_loss: 0.8045 - val_accuracy: 0.7273
Epoch 25/25
5/5 [==============================] - 1s 150ms/step - loss: 0.1945 -
accuracy: 0.9924 - val_loss: 0.7871 - val_accuracy: 0.7273
```

Got 99% accuracy on training dataset and 72% accuracy on testing dataset with VGGnet

Plotting Loss and Accuracy

```
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
```




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