Double-click (or enter) to edit

Name : Nidhi Singh

Reg. No. : 22MAI0015

DEEP LEARNING LAB

DIGITAL ASSIGNMENT - III

COURCE CODE : MCSE603P

LAB_7.1 :

https://colab.research.google.com/drive/116F_J29WAs225R3kp0zVOfBDebxENMzJ?usp=share_link

LAB_7.2 :

https://colab.research.google.com/drive/1CIsGz6qG9IOKrxbhl8vtkTb73w_6DtCP?usp=share_link

LAB_8.1 :

https://colab.research.google.com/drive/1f3DyMQ1bZQEhfNp1sOPDWllmqPt9hida?usp=share_link

LAB_8.2 :

https://colab.research.google.com/drive/1cdfZxYwB_HA75ULIhx-pcA5HbXc29gnP?usp=share_link DRIVE LINK :-

Primer to Transfer Learning

VGG-16 is a convolutional neural network that is 16 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224. You can use classify to classify new images using the VGG-16 network.

Example: Prediction of Elephant VGG16

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input, decode_predictions
import numpy as np
```

```
model = VGG16()
```

```
    Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
    553467096/553467096 [==============================] - 20s 0us/step
```

```
img_path = 'elephant.jfif'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
```

```
features = model.predict(x)
```

```
print('Predicted:', decode_predictions(features, top=3)[0])
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 [==============================] - 0s 0us/step
Predicted: [('n01871265', 'tusker', 0.6306532), ('n02504458', 'African_elephant', 0.3506527), ('n02504013', 'Indian_elephant', 0.0186923
```

Example: Prediction ResNet

```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np
```

```
model = ResNet50(weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels.h5
102967424/102967424 [==============================] - 1s 0us/step
```

```
img_path = 'elephant.jfif'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
```

```
preds = model.predict(x)
```

```
1/1 [==============================] - 1s 1s/step
```

decode the results into a list of tuples (class, description, probability)

(one such list for each sample in the batch)

```
print('Predicted:', decode_predictions(preds, top=3)[0])
```

```
Predicted: [('n02504458', 'African_elephant', 0.44517902), ('n01871265', 'tusker', 0.41670927), ('n02504013', 'Indian_elephant', 0.13809
```

Example Prediction of Dog

example of using a pre-trained model as a classifier

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

```
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.applications.vgg16 import decode_predictions
from tensorflow.keras.applications.vgg16 import VGG16
import numpy as np
```

load an image from file

```
img = image.load_img('dog.jfif', target_size=(224, 224))
```

convert the image pixels to a numpy array

```
img = image.img_to_array(img)
x = np.expand_dims(img, axis=0)
```

prepare the image for the VGG model

```
image = preprocess_input(x)
```

load the model

```
model = VGG16()
```

predict the probability across all output classes

```
yhat = model.predict(image)
```

```
    1/1 [==============================] - 1s 1s/step
```

convert the probabilities to class labels

```
label = decode_predictions(yhat)
```

retrieve the most likely result, e.g. highest probability

```
label = label[0][0]
```

print the classification

```
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
    papillon (49.71%)
```

```
print('Predicted:', decode_predictions(yhat, top=3)[0])
```

```
    Predicted: [('n02086910', 'papillon', 0.4971199), ('n02085620', 'Chihuahua', 0.110318586), ('n02098286', 'West_Highland_white_terrier',
```

Exercise : Try the predicition with the following models(i) Xception (ii)Inceptionv3 (iii) VGG19 (iv) MobileNet (V) InceptionResnetV2

(i) Xception

```
from tensorflow.keras.applications.xception import preprocess_input, decode_predictions, Xception
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
import numpy as np
```

```
image_path = '/content/dog.jfif'
# load an image from file
img = image.load_img(image_path, target_size=(299, 299))
# convert the image pixels to a numpy array
```

```
img = image.img_to_array(img)
x = np.expand_dims(img, axis=0)
# prepare the image for the Xception model
image = preprocess_input(x)
```

```
model_3 = Xception()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels.
91884032/91884032 [==============================] - 0s 0us/step
```

```
# predict the probability across all output classes
m1 = model_3.predict(image)
# convert the probabilities to class labels
label = decode_predictions(m1)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
1/1 [==============================] - 2s 2s/step
white_wolf (6.49%)
```

```
print('Predicted:', decode_predictions(m1, top=3)[0])
```

```
Predicted: [('n02114548', 'white_wolf', 0.06494567), ('n02113186', 'Cardigan', 0.04624985), ('n02109961', 'Eskimo_dog', 0.034882713)]
```

II. Inceptionv3

```
from tensorflow.keras.applications.inception_v3 import preprocess_input, decode_predictions, InceptionV3
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
import numpy as np
```

```
image_path = '/content/dog.jfif'
# load an image from file
img = image.load_img(image_path, target_size=(299, 299))
# convert the image pixels to a numpy array
img = image.img_to_array(img)
x = np.expand_dims(img, axis=0)
# prepare the image for the Inceptionv3 model
image = preprocess_input(x)
```

```
# load the model
model_4 = InceptionV3()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_
96112376/96112376 [==============================] - 1s 0us/step
```

```
# predict the probability across all output classes
m2 = model_4.predict(image)
# convert the probabilities to class labels
label = decode_predictions(m2)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f7cd9a9a7a0> trigg
1/1 [==============================] - 3s 3s/step
white_wolf (24.09%)
```

```
print('Predicted:', decode_predictions(m2, top=3)[0])
```

```
Predicted: [('n02114548', 'white_wolf', 0.24094132), ('n02111889', 'Samoyed', 0.07823342), ('n02134084', 'ice_bear', 0.05429261)]
```

III. VGG19

```
from tensorflow.keras.applications.vgg19 import preprocess_input, decode_predictions, VGG19
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
import numpy as np


image_path = '/content/dog.jfif'
# load an image from file
img = image.load_img(image_path, target_size=(224, 224))
# convert the image pixels to a numpy array
img = image.img_to_array(img)
x = np.expand_dims(img, axis=0)
# prepare the image for the VGG19 model
image = preprocess_input(x)


# load the model
model_5 = VGG19()


# predict the probability across all output classes
m3 = model_5.predict(image)
# convert the probabilities to class labels
label = decode_predictions(m3)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7f7ce9ac6680> trigg
1/1 [==============================] - 1s 886ms/step
papillon (58.66%)
```

```
print('Predicted:', decode_predictions(m3, top=3)[0])
```

```
Predicted: [('n02086910', 'papillon', 0.5865521), ('n02098286', 'West_Highland_white_terrier', 0.21367186), ('n02085620', 'Chihuahua', 0
```

IV. MobileNet

```
from tensorflow.keras.applications.mobilenet import preprocess_input, decode_predictions, MobileNet
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
import numpy as np


image_path = '/content/dog.jfif'
# load an image from file
img = image.load_img(image_path, target_size=(224, 224))
# convert the image pixels to a numpy array
img = image.img_to_array(img)
x = np.expand_dims(img, axis=0)
# prepare the image for the MobileNet model
image = preprocess_input(x)


# load the model
model_6 = MobileNet()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobilenet_1_0_224_tf.h5
17225924/17225924 [==============================] - 0s 0us/step
```

```
# predict the probability across all output classes
m4 = model_6.predict(image)
# convert the probabilities to class labels
label = decode_predictions(m4)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))
```

```
    1/1 [==============================] - 1s 520ms/step
```

```
print('Predicted:', decode_predictions(m4, top=3)[0])

    Predicted: [('n02113186', 'Cardigan', 0.19658937), ('n02113023', 'Pembroke', 0.13743022), ('n02104029', 'kuvasz', 0.070074245)]
```

V. InceptionResnetV2

```
from tensorflow.keras.applications.vgg19 import preprocess_input, decode_predictions, VGG19
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
import numpy as np


image_path = '/content/dog.jfif'
# load an image from file
img = image.load_img(image_path, target_size=(224, 224))
# convert the image pixels to a numpy array
img = image.img_to_array(img)
x = np.expand_dims(img, axis=0)
# prepare the image for the MobileNet model
image = preprocess_input(x)


# load the model
model_7 = MobileNet()


# predict the probability across all output classes
m5 = model_7.predict(image)
# convert the probabilities to class labels
label = decode_predictions(m5)
# retrieve the most likely result, e.g. highest probability
label = label[0][0]
# print the classification
print('%s (%.2f%%)' % (label[1], label[2]*100))

    1/1 [==============================] - 1s 522ms/step
    envelope (51.76%)
```

```
print('Predicted:', decode_predictions(m5, top=3)[0])

    Predicted: [('n03291819', 'envelope', 0.51764065), ('n10565667', 'scuba_diver', 0.07861161), ('n04209239', 'shower_curtain', 0.05133261)
```