Name: Tarun Tanmay

Class: MBATech CE

Roll No: N049

Sem: 6

Sap Id: 70471018055

In [ ]:

```
#Experiment 8
#Transfer Learning using VGG16
```

In [1]:

```
from keras.applications.vgg16 import VGG16 #vgg16 applicationn is a pre-trained model that has pre-trained weights
model=VGG16() #VGG16 model can be used for prediction, feature extraction, and fine-tuning
model.summary() #generates what's in the model
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467904/553467096 [==============================] - 4s 0us/step
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (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
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

Preprocessing of VGG16

Preprocessing of training data

DOG Image:

In [ ]:

```
#re-shaping of image to 224*224
from keras.preprocessing.image import load_img
image=load_img('dog.jpg', target_size=(224,224))
#converting image to numpy array
from keras.preprocessing.image import img_to_array
image=img_to_array(image)
```

In [ ]:

```
image.shape
```

Out[ ]:

```
(224, 224, 3)
```

In [ ]:

```
#reshaping arry image to one dimension
image=image.reshape((1,image.shape[0],image.shape[1],image.shape[2]))
```

In [ ]:

```
image.shape
```

Out[ ]:

```
(1, 224, 224, 3)
```

Feature Engineering:

Preprocessing of testing data

In [ ]:

```
#same preprocessing for test image
from keras.applications.vgg16 import preprocess_input
image=preprocess_input(image)
```

In [ ]:

```
image.shape
```

Out[ ]:

```
(1, 224, 224, 3)
```

In [ ]:

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

In [ ]:

```
yhat.shape #one dimension and thousand neurons in output layer
```

Out[ ]:

```
(1, 1000)
```

In [ ]:

```
yhat[0][0]
```

Out[ ]:

```
1.5842575e-07
```

In [ ]:

```
yhat[0][10]
```

Out[ ]:

```
1.0157481e-07
```

In [ ]:

```
from keras.applications.vgg16 import decode_predictions
#converting predictions to labels
label=decode_predictions(yhat)
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
40960/35363 [==============================] - 0s 0us/step
```

In [ ]:

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

In [ ]:

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

```
Doberman (0.34%)
```

The training data Image was of a dog, specifically 'Doberman'

CAR Image:

Preprocessing of training data:

In [ ]:

```
#re-shaping of image to 224*224
from keras.preprocessing.image import load_img
image=load_img('car.jpg', target_size=(224,224))
#converting image to numpy array
from keras.preprocessing.image import img_to_array
image=img_to_array(image)
```

In [ ]:

```
#reshaping arry image to one dimension
image=image.reshape((1,image.shape[0],image.shape[1],image.shape[2]))
```

In [ ]:

```
#same preprocessing for test image
from keras.applications.vgg16 import preprocess_input
image=preprocess_input(image)
```

In [ ]:

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

In [ ]:

```
from keras.applications.vgg16 import decode_predictions
#converting predictions to labels
label2=decode_predictions(yhat)
```

In [ ]:

```
lb2=label2[0][0]
```

In [ ]:

```
print('%s (%.2f%%)' % (lb2[1],lb2[2]%100)) #we get accuracy as 84%
```

```
sports_car (0.84%)
```

The above image was predicted to be of a sports car

BROCCOLI Image:

In [ ]:

```
Preprocessing of training data:
```

In [ ]:

```
#re-shaping of image to 224*224
from keras.preprocessing.image import load_img
image=load_img('broccoli.jpeg', target_size=(224,224))
#converting image to numpy array
from keras.preprocessing.image import img_to_array
image=img_to_array(image)
```

In [ ]:

```
#reshaping arry image to one dimension
image=image.reshape((1,image.shape[0],image.shape[1],image.shape[2]))
```

In [ ]:

```
#same preprocessing for test image
from keras.applications.vgg16 import preprocess_input
image=preprocess_input(image)
```

In [ ]:

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

In [ ]:

```
from keras.applications.vgg16 import decode_predictions
#converting predictions to labels
label3=decode_predictions(yhat)
```

In [ ]:

```
lb3=label3[0][0]
```

In [ ]:

```
print('%s (%.2f%%)' % (lb3[1],lb3[2]%100)) #we get 100% accuracy for broccoli, since it was a very clear i
mage containing only the vegetable
```

```
broccoli (1.00%)
```

We achieve 100% accuracy since the image of broccoli was very clear.

In [ ]:

```
model_1=VGG16() #initialising the VGG16 model
model_1.summary() #checking what's in the VGG16 model
from keras.models import Model
modell_1=Model(inputs=model_1.inputs,outputs=model_1.layers[-2].output)
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_10 (InputLayer)        [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (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
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

model_1 has a column of predictions

In [ ]:

```
modell_1.summary()
```

```
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_10 (InputLayer)        [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (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
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
=================================================================
Total params: 134,260,544
Trainable params: 134,260,544
Non-trainable params: 0
_________________________________________________________________
```

Prediction column has been dropped from modell_1 which was earlier present in model_1

In [ ]:

```
features=modell_1.predict(image)
features.shape #one dimension and 4096 neurons represents 4096 features
```

Out[ ]:

```
(1, 4096)
```

In [ ]:

```
features[0][0:10]
```

Out[ ]:

```
array([0.       , 0.       , 2.0713835, 0.       , 6.9527607, 0.       ,
       0.9079108, 0.       , 0.       , 5.9435163], dtype=float32)
```

In [ ]:

```
#the above features are for broccoli image since it was used lastly
```

In [ ]:

```
model_2=VGG16(include_top=False, input_shape=(300,300,3))
```

In [ ]:

```
from keras.layers import Dense #Dense adds the fully connected layer to the neural network
from keras.layers import Flatten #flatten is the function that converts the pooled feature map to a single column that is passed to the fully connected layer
flat1=Flatten()(model_2.layers[-1].output) #flattening the last layer of model_2
#dense layer is fully connected
class1=Dense(1024,activation='relu')(flat1) #relu is non-linear function which is fast, and gives output equal to input which is more than 0, otherwise output is 0 for negative values
output=Dense(10,activation='softmax')(class1) #softmax actiavtion function maps output in [0,1] range such that sum of all those outputs is 1
model_2=Model(inputs=model_2.inputs,outputs=output)
```

In [ ]:

```
model_2.summary() #checking what layers are in the model
```

```
Model: "model_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_12 (InputLayer)        [(None, 300, 300, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 300, 300, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 300, 300, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 150, 150, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 150, 150, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 150, 150, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 75, 75, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 75, 75, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 75, 75, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 75, 75, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 37, 37, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 37, 37, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 37, 37, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 37, 37, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 18, 18, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
flatten_5 (Flatten)          (None, 41472)             0
_________________________________________________________________
dense_4 (Dense)              (None, 1024)              42468352
_________________________________________________________________
dense_5 (Dense)              (None, 10)                10250
=================================================================
```

```
Total params: 57,193,290
Trainable params: 57,193,290
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
#we use the already trained VGG16 application, and fetch convolutional layers from block 1 and 2
#setting to False so that we shouldn't train the models again
model_3=VGG16(include_top=False, input_shape=(300,300,3))
model_3.get_layer('block1_conv1').trainable=False
model_3.get_layer('block1_conv2').trainable=False
model_3.get_layer('block2_conv1').trainable=False
model_3.get_layer('block2_conv2').trainable=False
model_3.summary() #checking the layers in the model_3
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_14 (InputLayer)        [(None, 300, 300, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 300, 300, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 300, 300, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 150, 150, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 150, 150, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 150, 150, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 75, 75, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 75, 75, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 75, 75, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 75, 75, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 37, 37, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 37, 37, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 37, 37, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 37, 37, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 18, 18, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 9, 9, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,454,528
Non-trainable params: 260,160
_________________________________________________________________
```

CONCLUSION:

1) VGG16 is used as a pre-trained model, for transfer learning.

2) Pre-trained model is used to extract 4096 features, convert number of classes from thousand to ten, and freeze four convolutional layers of the model.

3) Minor changes can be made in the pre-trained model to save the time of training, and to get classes for similar images.