Assignment 4

```
NAME = "Minhajul Abedin"
ID = "18301224"
COLLABORATORS_ID = ["", ""]
```

In this assignment you will have to create the VGG-19 network from scratch, using the keras functional API and explore its usage for different tasks. (The Functional API). [Going through the *Setup* and *Introduction* sections of the previous tutorial will suffice if you haven't attended the live demo sessions.]

Read the VGG paper: Very Deep Convolutional Networks for Large-Scale Image Recognition - Karen Simonyan, Andrew Zisserman and complete the following tasks

Task 1

Complete the following block to create a VGG19 network suitable for the ILSVRC classification task in keras and print the network's summary. (You don't have to train the network.)

[Hint: **Section 2 CONVNET CONFIGURATIONS** of the paper contains necessary information about the network architecture of the VGG network. (Column E of Table 1 is the VGG19 network architecture.)]

```
# Import necessary libraries
import tensorflow as tf


inputs = tf.keras.Input(shape=(224, 224, 3), name='input_1')

# block-1
x = tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu', name = 'block1_conv1')(inputs)
x=tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu', name = 'block1_conv2')(x)
block_1_output = tf.keras.layers.MaxPooling2D(2, name = 'block1_pool')(x)

# block-2
x = tf.keras.layers.Conv2D(128, 3, padding='same', activation='relu', name = 'block2_conv1')(block_1
x=tf.keras.layers.Conv2D(128, 3, padding='same', activation='relu', name = 'block2_conv2')(x)
block_2_output = tf.keras.layers.MaxPooling2D(2, name = 'block2_pool')(x)

# block-3
x = tf.keras.layers.Conv2D(256, 3, padding='same', activation='relu',  name = 'block3_conv1')(block_
x=tf.keras.layers.Conv2D(256, 3, padding='same', activation='relu', name = 'block3_conv2')(x)
x=tf.keras.layers.Conv2D(256, 3, padding='same', activation='relu', name = 'block3_conv3')(x)
x=tf.keras.layers.Conv2D(256, 3, padding='same', activation='relu', name = 'block3_conv4')(x)
block_3_output = tf.keras.layers.MaxPooling2D(2, name = 'block3_pool')(x)

# block-4
x = tf.keras.layers.Conv2D(512, 3, padding='same', activation='relu', name = 'block4_conv1')(block_3
x=tf.keras.layers.Conv2D(512, 3, padding='same', activation='relu', name = 'block4_conv2')(x)
```

```
x=tf.keras.layers.Conv2D(512, 3, padding='same', activation='relu', name = 'block4_conv3')(x)
x=tf.keras.layers.Conv2D(512, 3, padding='same', activation='relu', name = 'block4_conv4')(x)
block_4_output = tf.keras.layers.MaxPooling2D(2, name = 'block4_pool')(x)

# block-5
x = tf.keras.layers.Conv2D(512, 3, padding='same', activation='relu', name = 'block5_conv1')(block_4
x=tf.keras.layers.Conv2D(512, 3, padding='same', activation='relu', name = 'block5_conv2')(x)
x=tf.keras.layers.Conv2D(512, 3, padding='same', activation='relu', name = 'block5_conv3')(x)
x=tf.keras.layers.Conv2D(512, 3, padding='same', activation='relu', name = 'block5_conv4')(x)
block_5_output = tf.keras.layers.MaxPooling2D(2, name = 'block5_pool')(x)

flatten = tf.keras.layers.Flatten(name='flatten')(block_5_output)
fc1 = tf.keras.layers.Dense(4096, activation="relu", name='fc1')(flatten)
fc2 = tf.keras.layers.Dense(4096, activation="relu", name='fc2')(fc1)

outputs = tf.keras.layers.Dense(1000, activation="softmax", name='predictions')(fc2)

model = tf.keras.Model(inputs=inputs, outputs=outputs, name="vgg19")

model.summary()
```

```
Model: "vgg19"
_________________________________________________________________
 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_conv4 (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_conv4 (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_conv4 (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: 143,667,240
```

Your summary should match the following:

```
Model: "vgg19"
_________________________________________________________________
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_conv4 (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_conv4 (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_conv4 (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: 143,667,240
Trainable params: 143,667,240
Non-trainable params: 0
_________________________________________________________________
```

Task 2

What percentage of total parameters are in the fully connected layers?

Answer: Total parameters $= 143667240$
Total parameters in fully connected layers: (fc1 + fc2 + predictions)

$$
\begin{aligned}
&= 102764544 + 16781312 + 409700 \\
&= 123642856
\end{aligned}
$$

Percentage of total parameters are in the fully connected layers:

$$
\begin{aligned}
&= \left(\frac{123642856}{143667240} \times 100 \right) \% \\
&= 86.0619\% \approx 86.06\%
\end{aligned}
$$

In the whole architecture, 86.06% of the parameters are from fully connected layers.

Task 3

The VGG19 network from task 1 contains a Dense (FC) output layer with 1000 units for the ILSVRC classification task. How could you modify the current output layer so that it is capable of classifying the CIFAR-100 dataset, which has a total number of 100 classes?

Answer: For the ILSVRC classification task, we keep 1000 as units in output layer as the dataset include 1000 classes. Here, units is the first parameter of Dense method which determine dimensionality of output space. The code is:

```
outputs = tf.keras.layers.Dense(1000, activation="softmax", name='predictions')(fc2)
```

Now, for CIFAR-100 dataest we have only 100 classes. Therefore, in the output layer there will be 100 units for 100 classes. We can do that by only changing the first parameter (units) value from 1000 to 100. The code will be:

```
outputs = tf.keras.layers.Dense(100, activation="softmax", name='predictions')(fc2)
```

Task 4

The VGG19 network from task 1 was designed for the ILSVRC classification challenge, but as we know there was a seperate challenge for image localisation. How could you modify the current ouput layer so that is suitable for the ILSVRC localisation task?

[Hint: **Section A LOCALISATION** contains information about the localisation task.]

Answer: In the localisation task, a bounding box identifies the objects in the picture. Moreover, a bounding box will be represnted by its center coordinates(x, y), height and width. Thus, the output layer should be a 4-D vector which store the center coordinates, width, and height of the bounding box. There are two option available, one is single-class regression (SCR) where the bounding box prediction is shared across all classes. Another one is per-class regression (PCR) which is class specipic. In SCR the output layer will be 4-D. In contrast, for PCR 4000-D (4 x 1000) as there are 1000 classes in ILSVRC dataset.

In case of SCR, the output layer will be:

```
outputs = tf.keras.layers.Dense(4, activation="softmax", name='predictions')(fc2)
```

I will choose PCR as according to the authors of the paper it has outperformed the SCR. The output layer will be:

```
outputs = tf.keras.layers.Dense(4000, activation="softmax", name='predictions')(fc2)
```

Task 5

The VGG19 network from task 1 was designed for the ILSVRC classification challenge. Explain how the authors proposed this deep network as a feature extractor for other computer vision tasks. [Hint: **B GENERALISATION OF VERY DEEP FEATURES** contains information about using the VGG network as a feature extractor.]

Answer: The authors utialize their proposed deep network ConvNets which is pre trained on ILSVRC as a feature extractor on other smaller datasets where training large model from scratch is not feasible due to over-fitting because deep image represantations that learnt on ILSVRC genralise well to other datasets. To make this happen, they remove the last fully connected layer or output layer from the pre-trained ConvNet on ILSVRC. Instead of this output layer, they use 4096-D activations of the penultimate layer as image features.The image descriptor that results is L2-normalized and paired with a linear SVM classifier that has been trained on the target dataset. Also, pre-trained ConvNet weights are kept constant for simplicity while no fine-tuning is performed.

The aggregation of features gathered from vaious places and sizes and carried out in a similiar manner to the ILSVRC evaluation procedure. Then, the network densely applied over the scaled image plane. After that, they performed global average pooling that produce 4096-D image descriptor which then averaged with the descriptor of horizontally flipped image. However, as evalutaion over multi-scale is beneficial, they extract feature over several scale. Finally, the resulting multi-scale features can be either stacked or pooled across scale where stacking teaches a future classifier how to mix picture statistics appropriately across a variety of scales.

--THE END--