Image Processing : Activity 4 Reports

4.1 VGG16 Model Parameters

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing.image import img_to_array
from numpy import expand_dims
from scipy import signal
```

```
#1
#Read image file
ori_img = cv2.imread("yae.jpg")
img_RGB = cv2.cvtColor(ori_img, cv2.COLOR_BGR2RGB)
img = cv2.resize(img_RGB, (224, 224))
```

```
#2
#Load VGG16 model from tensorflow.keras
model = VGG16()
# model detail
model.summary()
```

```
#3
#retrieve kernel weights from the 1st Convolutional layer
kernels, biases = model.layers[1].get_weights()
# View CNN layer 1 architecture
model.layers[1].get_config()
```

```
#4
# Preprocess Image using keras and numpy
# convert the image to an array
img = img_to_array(img)
# expand dimensions so that it represents a single 'sample'
# -> reshape 3D(H,W,Ch) image to 4D image (sample,H,W,Ch)
img = expand_dims(img, axis=0)
# prepare the image (e.g. scale pixel values for the vgg)
img_ready = preprocess_input(img)
```

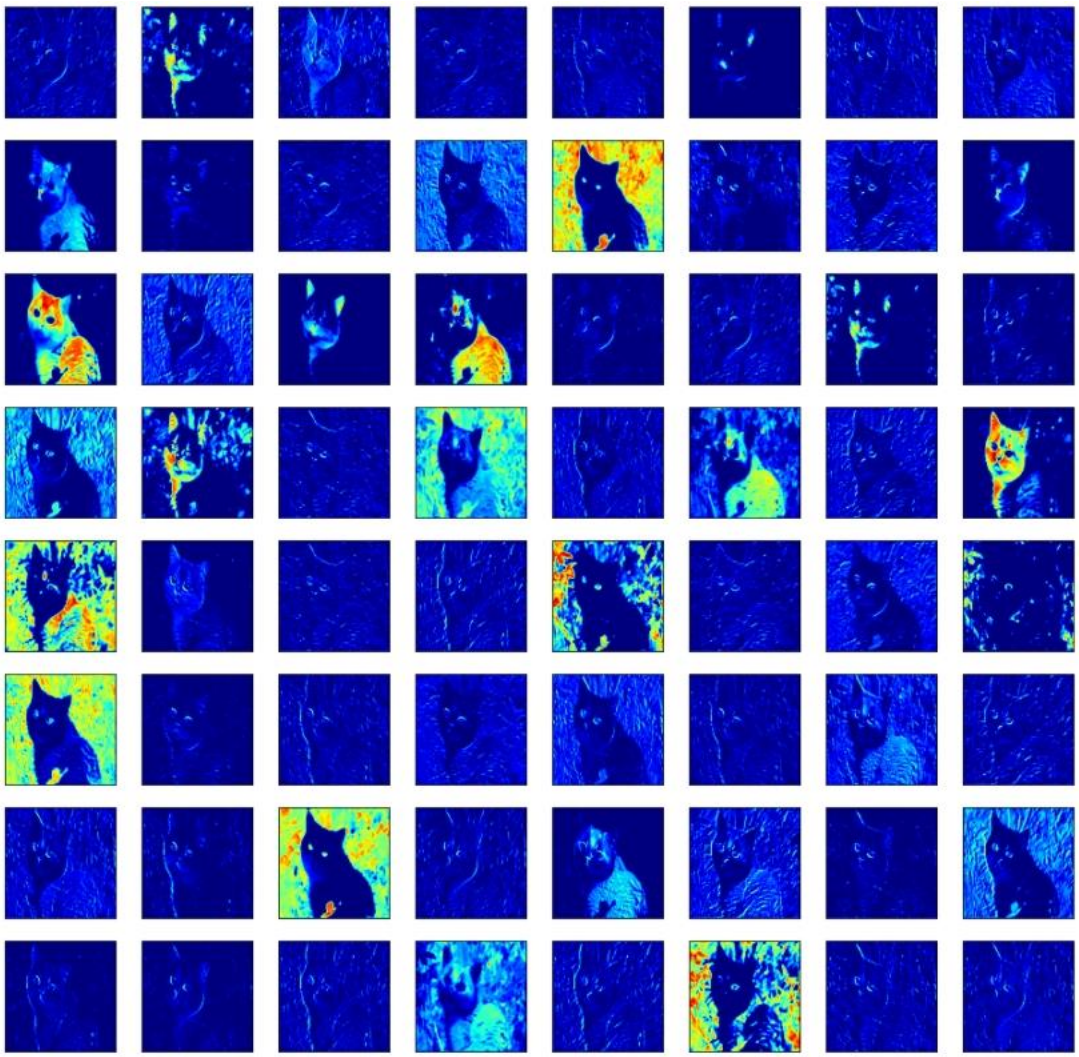
```
#5
# Extract Model CNN Layer 1
model = Model(inputs=model.inputs, outputs=model.layers[1].output)
model.summary()
```

```
#6
# Extract Results from CNN Layer 1 called feature map (shape = (sample = 1, 224, 224, n_filters) )
# CNN Layer 1 -> n_filters = 64
feature_maps = model.predict(img_ready)
```

```
#7
# Display images of feature_maps
plt.figure(figsize=(16,16))
i = 8
j = 8
index = 1
for e in range(i):
    for e in range(j):
        ax = plt.subplot(i, j, index)
        ax.set_xticks([])
        ax.set_yticks([])
        plt.imshow(feature_maps[0, :, :, index-1],cmap="jet")
        index += 1
```

สำหรับข้อ 4.1 จะเป็นการใช้โค้ดตัวอย่างของอาจารย์มาทำการทดลอง โดยจะแสดงผลบน plot ขนาด 8x8 และแสดงผลด้วย cmap = “jet” เพื่อให้ได้สีตามตัวอย่าง

Output :



4.2 Prepare input image from scratch

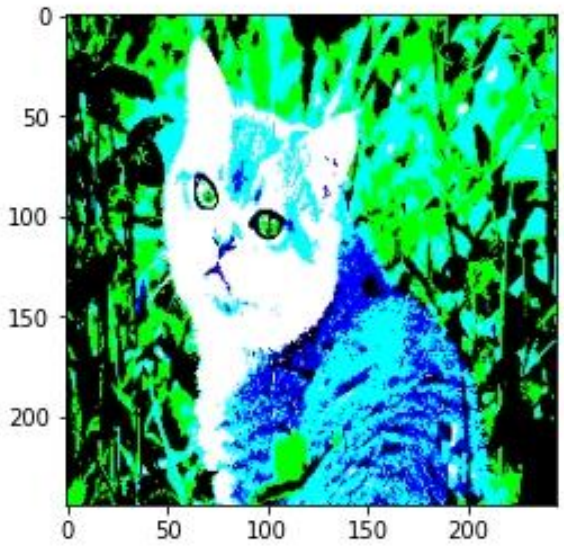
```
img2 = cv2.imread("./cat.jfif")
img2 = cv2.resize(img2,(244,244))

print(img2.shape)

img_mean = [123.68, 116.779, 103.939]
img2 = img2.astype(float)
print(img2.shape)
img2[:,:,0] -= img_mean[0]
img2[:,:,1] -= img_mean[1]
img2[:,:,2] -= img_mean[2]
plt.imshow(img2)
img2 = expand_dims(img2, axis=0)
```

การ Prepare input image from scratch จะเริ่มด้วยการ Resize ภาพให้เป็นขนาด 244x244 แล้วนำค่า mean ของ chanel สี 3 chanel มา subtract จากรูปที่เป็น input ซึ่งจะทำการเปลี่ยน type ของ image input ให้เป็น float จาก uint8 เพื่อจะได้ทำมา subtract กับ float ได้ หลังจากนั้นนำมาแสดงผล ก่อนจะแปลง 3d image เป็น 4d image เพื่อนำไปใช้ทดลองต่อ

Output :



4.3 Conv2D()

```
#3
#if val in Image_sum <0 then val = 0
def relu(img):
    height,weigth = img.shape
    for i in range(height):
        for j in range(weigth):
            if img[i][j] < 0:
                img[i][j] = 0
    return img
#4
Img_result = np.copy(img2[0,:,:,:])
fig, ax = plt.subplots(8, 8, figsize=(16,16))
fig.patch.set_facecolor('white')
col = 0
row = 0
for j in range(64):
    for i in range(3):
        Img_result[: , :, i] = signal.convolve2d(img2[ 0,: , : , i], kernels[: , : , i, j] ,
                                                 mode='same',boundary='fill', fillvalue=0)
    Image_sum = Img_result[ : , : , 0] + Img_result[ : , : , 1] + Img_result[ : , : , 2]
    Image_sum = relu(Image_sum)
    if(col==8):
        row += 1
        col = 0
    ax[row,col].imshow(Image_sum,cmap="jet")
    col +=1
```

การ Convolution จะใช้ฟังก์ชัน convolve2d() ต้องทำในทุก color channel และ ลูปเปลี่ยน kernel ไปด้วยเพื่อให้ได้ภาพครบทุกชุดฟิลเตอร์ โดยกำหนด mode = same เนื่องจากต้องการให้ภาพมีขนาดเท่ากับภาพ input ต้องกำหนดให้ fillvalue = 0 ด้วย จากนั้นก็นำผลของแต่ละ channel มาบวกกัน ได้เป็นฟิลเตอร์ 1 ชุด และนำไป Activation function เข้าฟังก์ชัน relu() เพราะหากมีค่าผลรวมที่น้อยกว่า 0 จะให้มีค่าเท่ากับ 0 จากนั้นแสดงผลลัพธ์เป็นรูปทั้ง 64 รูป โดยกำหนดให้ cmap = “jet”

Output :