*# โค้ดจากข้อ 3.2*

*# import library*

*import* cv2

*import* numpy *as* np

*from* keras.applications.vgg16 *import* VGG16

*from* keras.applications.vgg16 *import* preprocess\_input

*from* keras.preprocessing.image *import* load\_img

*from* keras.preprocessing.image *import* img\_to\_array

*from* keras.models *import* Model

*from* matplotlib *import* pyplot *as* plt

*from* numpy *import* expand\_dims

*from* scipy *import* signal

*# โหลด VGG16 Model*

model = VGG16()

*# สรุปรวมโครงสร้างและจำนวนพารามิเตอร์ของโมเดล VGG16*

model.summary()

*# ดึง array ของ weight และ bias ของ VGG16 ใน Layer 1*

kernels, biases = model.layers[1].get\_weights()

*# ดึงโครงสร้างการทำงานทั้งหมดของ VGG16 Model ใน Layer 1 มาแสดง*

model.layers[1].get\_config()

*# อ่านไฟล์ภาพที่ต้องการนำเข้า Model และปรับขนาดของภาพให้ตรงกับขนาด Input ของโมเดล VGG16*

img = load\_img(*f*'bird.jpg', target\_size=(224, 224))

original = img

img\_mean = [123.68, 116.779, 103.939] *#BGR*

*# แปลงภาพเป็น NumPy array*

img\_array = np.array(img)

*# ดึงค่าสีแต่ละ channel ของ img ออกมา*

imgB = img\_array[:, :, 0]

imgG = img\_array[:, :, 1]

imgR = img\_array[:, :, 2]

*# นำมาคำนวณค่า img - img\_mean ของแต่ละ R,G,B*

B = imgB - img\_mean[0]

G = imgG - img\_mean[1]

R = imgR - img\_mean[2]

*# รวมแต่ละ channel กลับเป็นรูปเหมือนเดิม*

new\_img = cv2.merge([B, G, R])

*# เพิ่ม dimension*

img4d = expand\_dims(new\_img, axis=0)

img4d.shape

fig, axes = plt.subplots(2, 2, figsize=(10, 10))

*# แสดงภาพ Original*

axes[0,0].imshow(original)

axes[0,0].set\_title('Original Image')

*# แสดงภาพ new image*

axes[0,1].imshow(new\_img)

axes[0,1].set\_title('New Image')

*# แสดงภาพ new image*

axes[1,0].imshow(new\_img)

axes[1,0].set\_title('New Image')

*# แสดงภาพ new image*

axes[1,1].imshow(new\_img)

axes[1,1].set\_title('New Image')

*# ข้อ 3.3*

*# สร้างมาเก็บผลลัพธ์หลังจากทำ convolution*

img\_result = np.zeros((224, 224,3))

*for* i *in* range(65):

    image\_sum = np.zeros((i,224, 224))

*def* relu(x):

*return*(np.maximum(0, x))

*# ลูปเพื่อทำ convolution และ ReLU activation สำหรับแต่ละช่องสี*

*for* i *in* range(64):

*# Convolution และสร้างผลลัพธ์สำหรับแต่ละช่องสี*

    img\_result[: , :, 0] = signal.convolve2d( new\_img[ : , : , 0], kernels[: , : , 0, i] , mode='same',boundary='fill', fillvalue=0)

    img\_result[: , :, 1] = signal.convolve2d( new\_img[ : , : , 1], kernels[: , : , 1, i] , mode='same',boundary='fill', fillvalue=0)

    img\_result[: , :, 2] = signal.convolve2d( new\_img[ : , : , 2], kernels[: , : , 2, i] , mode='same',boundary='fill', fillvalue=0)

*# สร้าง image\_sum และใช้ ReLU activation*

    image\_sum[i,:,:] = img\_result[ : , : ,0] + img\_result[ : , : , 1] + img\_result[ : , : , 2]

    image\_sum[i,:,:] = relu(image\_sum[i,:,:])

*# รวมผลลัพธ์จากแต่ละช่องสีเพื่อสร้างภาพสีสุดท้าย*

imgRGB\_sum = cv2.merge([img\_result[: , :, 0], img\_result[: , :, 1], img\_result[: , :, 2]])

*# แสดงผลลัพธ์ด้วย subplot*

fig, axs = plt.subplots(8,8, figsize=(15, 15), facecolor='w', edgecolor='k')

fig.subplots\_adjust(hspace = .5, wspace=.001)

axs = axs.ravel()

*for* i *in* range(64):

    axs[i].imshow(image\_sum[i,:,:],cmap='viridis')

    axs[i].set\_title(i)