ณสิต ผลัญชัย 65010273

วิวัตร เตชะโกศล 65011001

Lab 5.1

```
### START CODE HERE ###

vgg16 = models.vgg16(pretrained=True)
vgg16.eval()

### END CODE HERE ###
```

ทำการโหลด model vgg16 แล้วปรับโหมดเป็น evaluation mode

```
### START CODE HERE ###

vgg16.features

### END CODE HERE ###
```

ทำการแสดงผล Layers ทั้งหมดของ model vgg16 ด้วยคำสั่ง vgg16.features โดยมีทั้งหมด 30 layers

```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): ReLU(inplace=True)
  (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (3): ReLU(inplace=True)
  (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (6): ReLU(inplace=True)
  (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (8): ReLU(inplace=True)
  (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (11): ReLU(inplace=True)
  (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (13): ReLU(inplace=True)
  (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (15): ReLU(inplace=True)
  (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (18): ReLU(inplace=True)
  (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (20): ReLU(inplace=True)
  (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (22): ReLU(inplace=True)
  (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
...
  (27): ReLU(inplace=True)
```

```
### START CODE HERE ###

first_conv = vgg16.features[0]
first_relu = vgg16.features[1]

### END CODE HERE ###
```

จาก vgg16.features ด้านบนจะเห็นได้ว่า Layer ที่1 เป็น convolution และ Layer ที่2 เป็น ReLU จึงทำการดึง vgg16.features ใน index ที่ 0, 1 มาเก็บไว้ใน first_conv และ first_relu ตามลำดับ

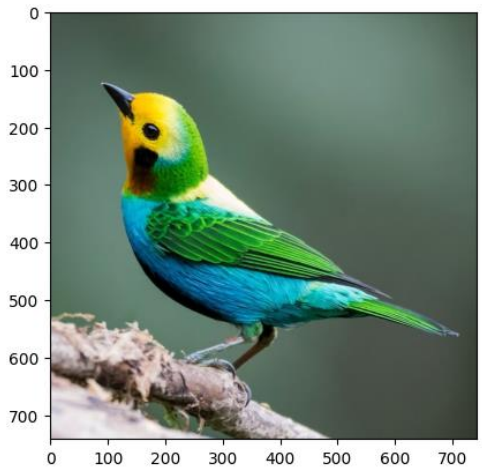
```
### START CODE HERE ###
print("Bias :", vgg16.features[0].bias)
### END CODE HERE ###
```

ทำการแสดงผล Biases ของ Layer ที่1 ได้ผลลัพท์ดังรูปด้านล่าง

```
Bias : Parameter containing:
tensor([ 0.4034,  0.3778,  0.4644, -0.3228,  0.3940, -0.3953,  0.3951, -0.5496,
         0.2693, -0.7602, -0.3508,  0.2334, -1.3239, -0.1694,  0.3938, -0.1026,
         0.0460, -0.6995,  0.1549,  0.5628,  0.3011,  0.3425,  0.1073,  0.4651,
         0.1295,  0.0788, -0.0492, -0.5638,  0.1465, -0.3890, -0.0715,  0.0649,
         0.2768,  0.3279,  0.5682, -1.2640, -0.8368, -0.9485,  0.1358,  0.2727,
         0.1841, -0.5325,  0.3507, -0.0827, -1.0248, -0.6912, -0.7711,  0.2612,
         0.4033, -0.4802, -0.3066,  0.5807, -1.3325,  0.4844, -0.8160,  0.2386,
         0.2300,  0.4979,  0.5553,  0.5230, -0.2182,  0.0117, -0.5516,  0.2108],
       requires_grad=True)
```

```
### START CODE HERE ###

image = (cv2.cvtColor(cv2.imread(r'C:\Users\Nickv\Documents\ImageProcessing\Week5\asset\bird.jpg'), cv2.COLOR_BGR2RGB))
plt.imshow(image)
plt.show()

### END CODE HERE ###
```

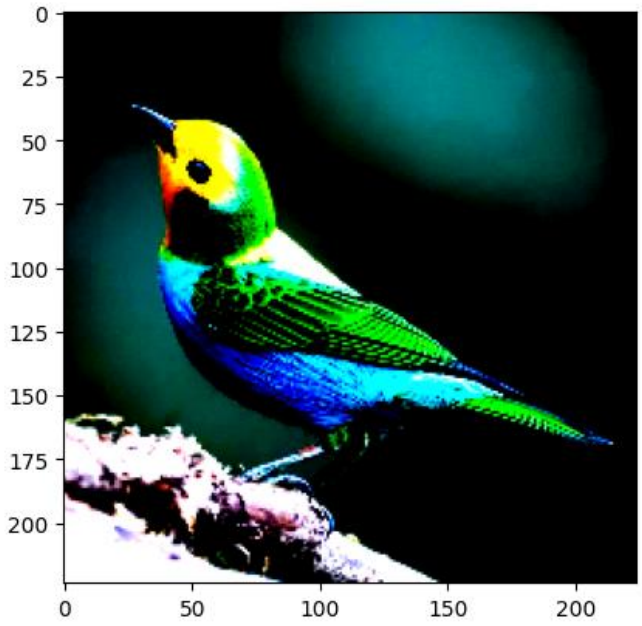
ทำการโหลดรูปเข้ามาและเปลี่ยนสีจาก BGR เป็น RGB แล้วแสดงผล




```
### START CODE HERE ###

image = cv2.resize(image, (224, 224))
image = image.astype(np.float32) / 255.0
mean = np.array([0.485, 0.456, 0.406])
std = np.array([0.229, 0.224, 0.225])
normalized_image = (image - mean) / std
normalized_image = np.clip(normalized_image, 0, 1)

plt.imshow(normalized_image)
plt.show()

### END CODE HERE ###
```

ทำการเตรียมรูปให้พร้อมสำหรับนำไป convolution ด้วยการ resize ขนาดรูปเป็น (224, 224) และ normalized ด้วยค่า mean, std of Imagenet จากเว็ป https://pytorch.org/vision/main/models/generated/torchvision.models.vgg16.html

โดยใช้ mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225] จากนั้นทำให้ range ของ image array อยู่ใน range 0 – 1 แล้วแสดงผลรูปที่ผ่านการ normalize ได้ภาพด้านล่าง




```
### START CODE HERE ###

torch_image = torch.from_numpy(normalized_image).permute(2, 0, 1).unsqueeze(0).float()
torch_image.shape

### END CODE HERE ###
```

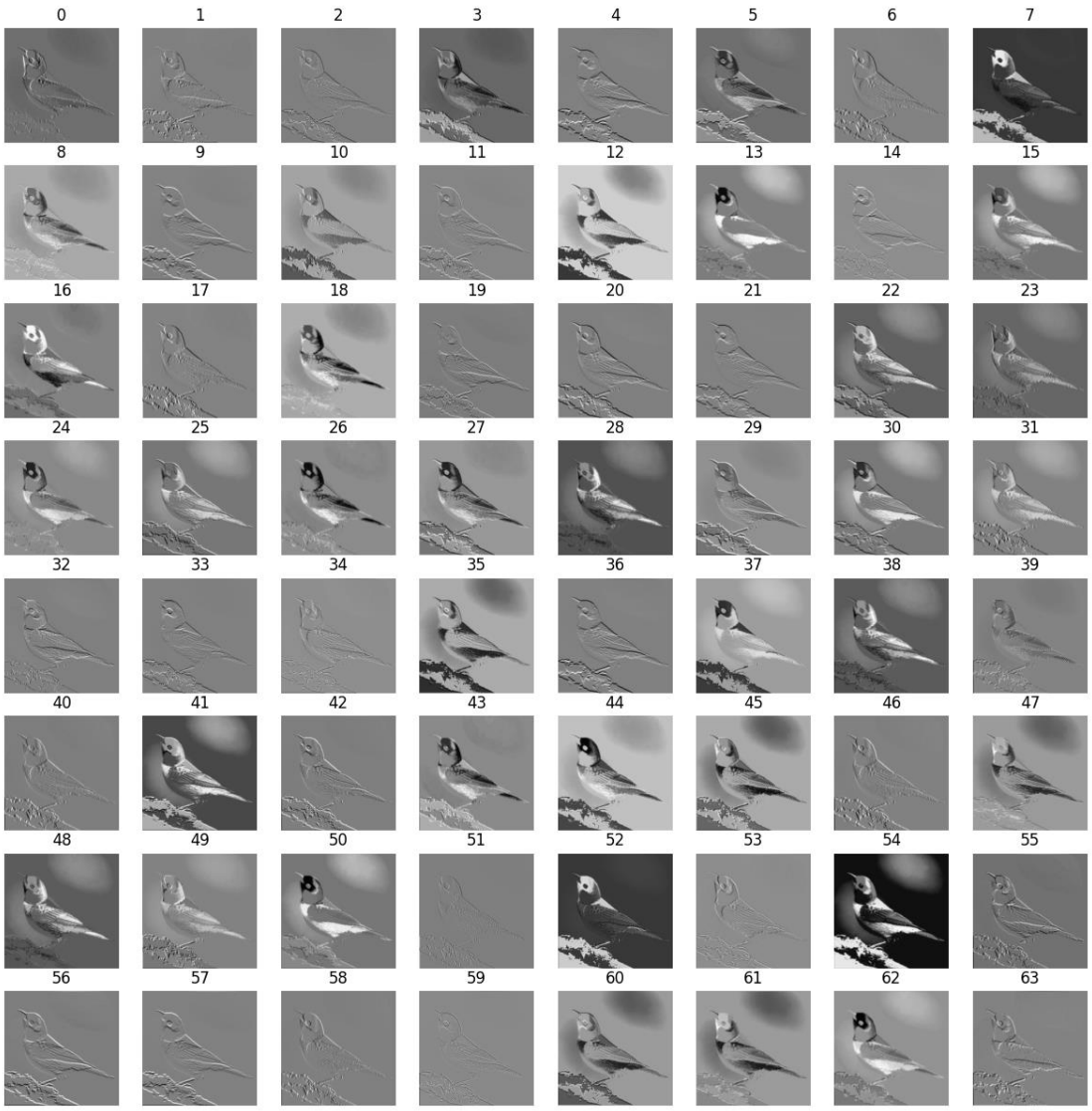
ทำการเปลี่ยน normalized_image จาก numpy ให้เป็น PyTorch tensor แล้วทำการ permute(2, 0, 1) ซึ่งจะเป็นการเรียงมิติของ tensor ใหม่จาก [224, 224, 3] เป็น [3, 224, 224] แล้วทำการ unsqueeze(0) ซึ่งจะเป็นการเพิ่มมิติ1 เข้าไปที่ตำแหน่งที่ 0 เมื่อแสดงผล torch_image.shape จะได้เป็น torch.Size([1, 3, 224, 224])

```
### START CODE HERE ###

def plot_featuremap(img, title):
    num_feature_maps = img.shape[1]
    grid_size = int(np.ceil(num_feature_maps**0.5))
    fig, axes = plt.subplots(grid_size, grid_size, figsize=(15, 15))
    fig.suptitle(title)
    k = 0
    for i in range(grid_size):
        for j in range(grid_size):
            ax = axes[i, j]
            index = i * grid_size + j
            if index < num_feature_maps:
                feature_map = img[0, index].detach().numpy()
                ax.imshow(feature_map, cmap='gray')
                ax.set_title(f'{k}')
                k += 1
            ax.axis('off')

    plt.show()

### END CODE HERE ###
```

สร้าง function plot_featuremap() ซึ่งจะนำผลจากการ convolution หรือ ReLU ที่เรียกว่า feature map มาแสดงผลโดยใช้ grid_size เป็น มิติของ num_feature_maps ยกกำลัง ½ แล้วปัดขึ้น เพื่อให้แสดงผลได้ครบทุกภาพ จากนั้นก็ทำการ loop เพื่อ plot feature map ด้วยการนำ img[0, index].detach().numpy() ซึ่งจะเป็นการแปลง tensor ให้เป็น numpy array และลบมิติของ batch ออก

```
### START CODE HERE ###

features_map = first_conv(torch_image)
plot_featuremap(features_map, 'conv')

### END CODE HERE ###
```

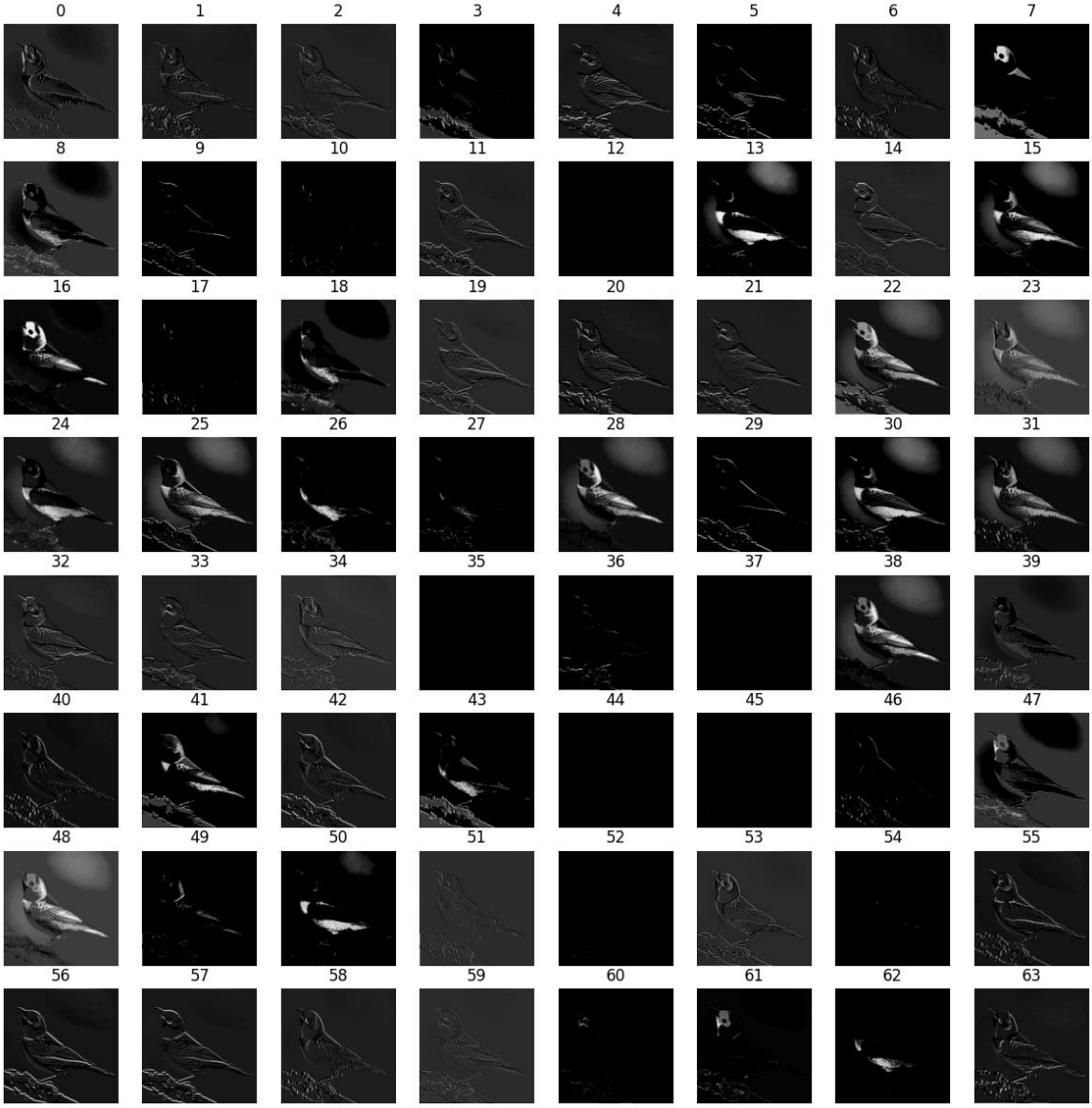
ทำการแสดงผลของการ convolution ครั้งแรกจะได้ผลลัพท์ดังรูปด้านล่าง




```
features_map = first_relu(features_map)
plot_featuremap(features_map, 'ReLU')
```

ทำการแสดงผลของการใช้ activation function ReLU ครั้งแรกจะได้ผลลัพท์ดังรูปด้านล่าง

ReLU

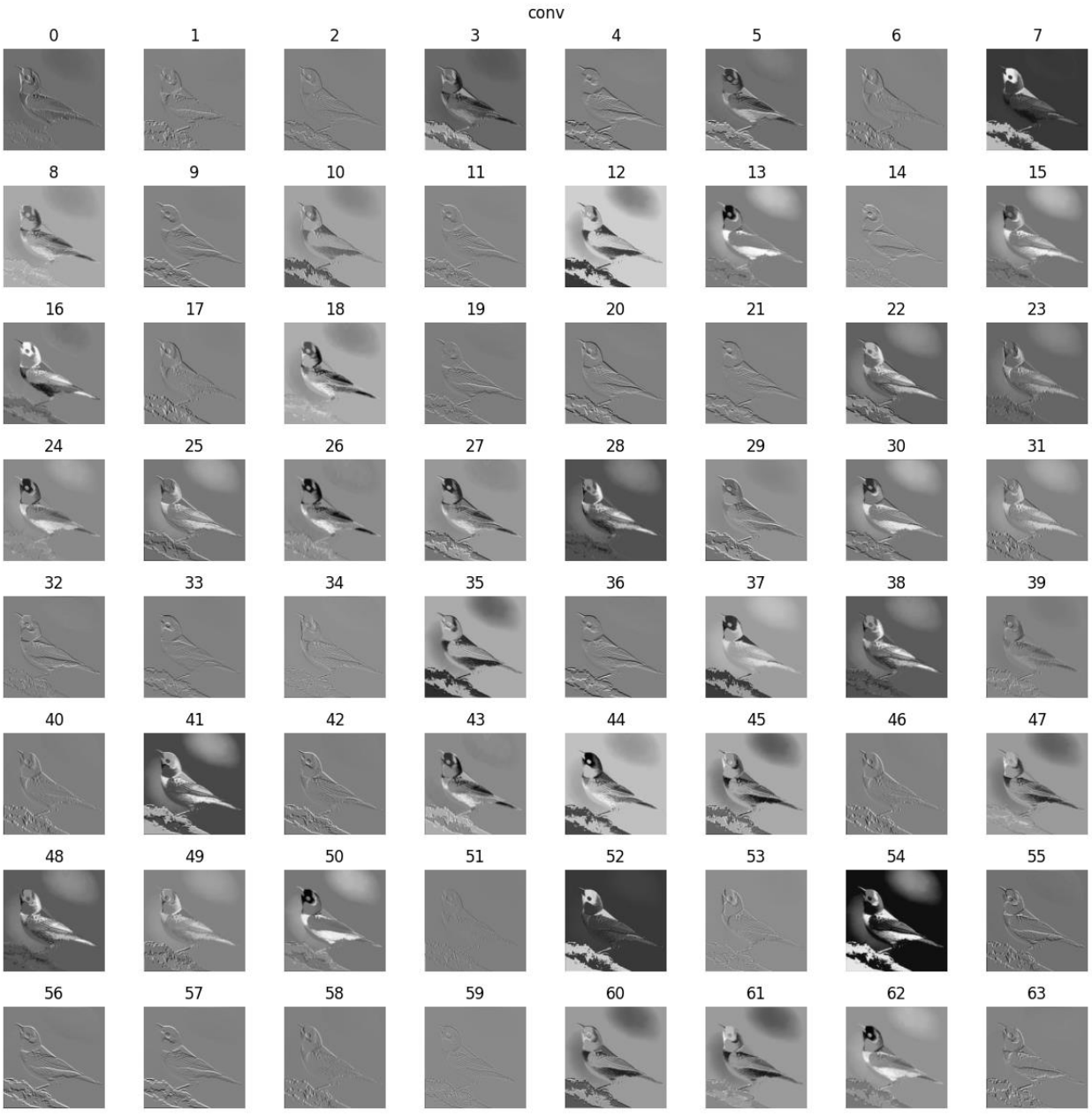

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

```python
### START CODE HERE ###
def convolution2d(img,kernel,padding=0, stride=1):
    img_h, img_w = img.shape
    kernel_h, kernel_w = kernel.shape

    img_padded = np.pad(img, [(padding, padding), (padding, padding)], mode='constant', constant_values=0)

    out_h = (img_h + 2 * padding - kernel_h) // stride + 1
    out_w = (img_w + 2 * padding - kernel_w) // stride + 1

    output = np.zeros((out_h, out_w))

    for y in range(out_h):
        for x in range(out_w):
            region = img_padded[y * stride:y * stride + kernel_h, x * stride:x * stride + kernel_w]
            output[y, x] = np.sum(region * kernel)

    return output
### END CODE HERE ###
```

สร้าง function convolution2d() โดยจะทำการดึงขนาดของ img และ kernel แล้วทำการ zero padding ในกรณีที่มี Input ใน parameter padding โดยจะตั้ง default เป็น 0 ต่อจากนั้นคำนวณความสูงและความกว้างของ output ตามขนาด img, kernel, padding, และ stride เพื่อนำมาสร้าง output array ที่เป็น array 0 ขนาดตามที่คำนวณไว้ ก่อนจะ loop ไล่ดึง region มาแล้ว plot ลง output array

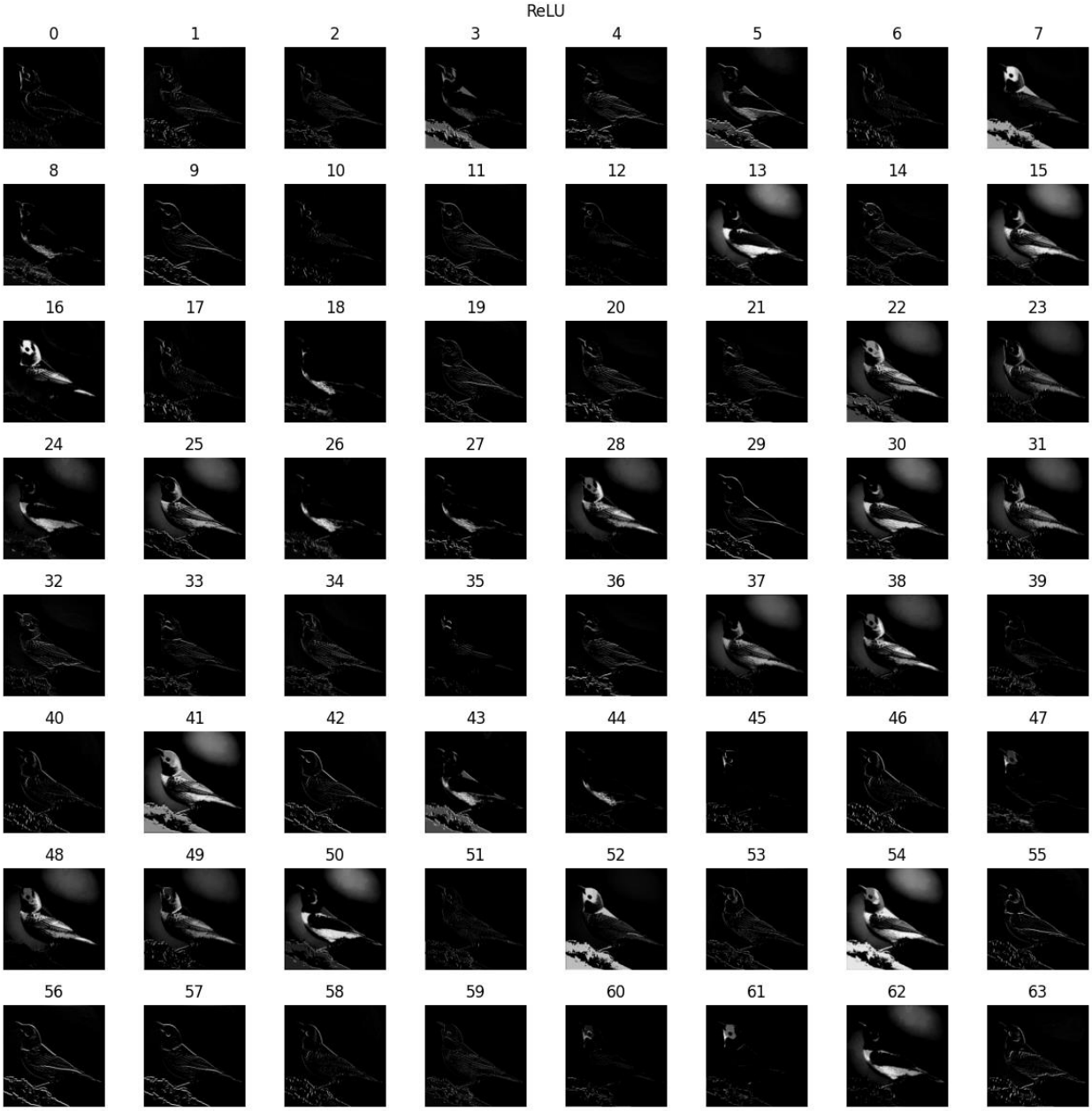
```python
### START CODE HERE ###

weights = first_conv.weight.detach().numpy()
biases = first_conv.bias.detach().numpy()
img_tensor = torch.from_numpy(normalized_image).permute(2, 0, 1).float()

fig, axes = plt.subplots(8, 8, figsize=(12, 12))
fig.suptitle('conv')

for i in range(weights.shape[0]):
    feature_map = np.zeros((224, 224))

    for j in range(3):
        kernel = weights[i, j]
        feature_map += convolution2d(img_tensor[j], kernel, padding=1, stride=1)

    ax = axes[i // 8, i % 8]
    ax.imshow(feature_map, cmap='gray')
    ax.axis('off')
    ax.set_title(f'{i}')

plt.tight_layout()
plt.show()

### END CODE HERE ###
```

ทำการดึงค่า weights และ biases ของ convolution layer แรกแล้วแปลงภาพจาก numpy array เป็น pytorch tensor (Channel First) เพื่อให้ convolution ได้ ก่อนจะ loop ตามจำนวน output channels ใน convolution layer โดยการเรียก weights.shape[0] ในกรณีนี้เป็น 64 โดยแต่ละ output channel ก็จะต้อง loop อีก 3 ครั้งเนื่องจากมี 3 kernels สำหรับ RGB โดยจะนำ kernel ที่ได้มาใส่ใน function convolution2d() แล้วนำผลลัพท์มาเก็บใน feature map แล้วแสดงผลเมื่อ loop เสร็จได้ผลลัพท์ดังรูปด้านล่าง



```
### START CODE HERE ###

fig, axes = plt.subplots(8, 8, figsize=(12, 12))
fig.suptitle('ReLU')

for i in range(weights.shape[0]):
    feature_map = np.zeros((224, 224))

    for j in range(3):
        kernel = weights[i, j]
        feature_map += convolution2d(img_tensor[j], kernel, padding=1, stride=1)

    feature_map = np.maximum(feature_map, 0)

    ax = axes[i // 8, i % 8]
    ax.imshow(feature_map, cmap='gray')
    ax.axis('off')
    ax.set_title(f'{i}')

plt.tight_layout()
plt.show()

### END CODE HERE ###
```

ในการทำ ReLU activation ก็จะทำเหมือนกับการ convolution ด้านบนโดยการใช้ loop และ function convolution2d() เหมือนเดิมเพียงแต่จะต้องเพิ่ม feature_map = np.maximum(feature_map, 0) เพื่อให้ค่าใน feature_map ที่ติดลบกลายเป็น 0 จะได้ผลลัพท์ดังนี้


ReLU
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63