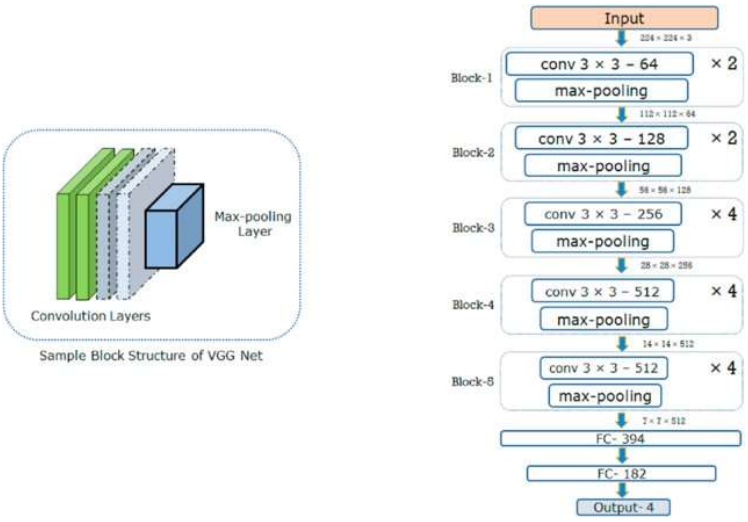
تحلیل بصری خروجی هر لایه در مدلهای یادگیری عمیق

شبکه های عصبی عموما مدلهای غیرشفاف هستند به این معنا که توضیح ساده و قابل درکی برای چگونگی و چرایی خروجی تولید شده توسط آنها وجود ندارد. شبکه های عصبی کانولوشن (CNN) نوع خاصی از شبکه عصبی با چندین لایه هستند که داده‌هایی را که نمایش ماتریسی دارند، پردازش کرده و ویژگی‌های مهم آن‌ها را استخراج می‌کنند. درواقع این شبکه‌ها با یادگیری فیلترهای مختلف، انواع ویژگی‌های تصاویر ورودی را استخراج می‌کنند. تحلیل بصری فیلترهای آموخته‌شده در هر لایه و خروجی هر لایه (نتیجه اعمال فیلتر روی ورودی) می‌تواند تا حد زیادی به درک نحوه عملکرد این شبکه‌ها کمک کند.

در این پروژه از شبکه VGG19 که یک شبکه مبتنی بر کانولوشن است و ساختار آن در شکل 1 نشان داده شده‌است، برای تحلیل بصری استفاده می‌شود. مراحل انجام کار و نتایج در ادامه شرح داده شده است.



شکل 1 – معماری شبکه VGG19

همانطور که در شکل 1 مشاهده می‌شود، شبکه VGG19 از پنج بلاک اصلی تشکیل شده است. این شبکه در ورودی تصاویر رنگی در اندازه 224*224 را دریافت می‌کند. این تصویر از لایه‌های مختلف عبور می‌کند و در هر لایه ویژگی‌های خاصی استخراج می‌شود.

1- ایجاد مدل و درک ساختار آن

ابتدا مدل VGG19 را بارگذاری میکنیم و با استفاده از متد summary ساختار آن را نمایش می دهیم:

```
model = VGG19()
model.summary()
```

```
=================================================================
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
=================================================================
```

همانطور که مشاهده میشود این شبکه 5 بلاک اصلی دارد که به ترتیب شامل 2، 2، 4، 4 و 4 لایه کانولوشن هستند. لایه‌های مختلف به شکل #block#_conv نامگذاری شده‌اند که # شماره بلاک و شماره لایه کانولوشن در هر بلاک می‌باشد.

به طور کلی شبکه VGG19 از 26 لایه تشکیل شده که 16 لایه آن لایه‌های Conv هستند. کد زیر تعداد دقیق همه لایه‌های این شبکه را نشان می‌دهد:

```
n_layers = len(model.layers)
print('Number of Layers= %d' % n_layers)
```

```
Number of Layers= 26
```

کد زیر تعداد لایه‌های کانولوشن و تعداد و اندازه فیلترهای هر لایه را نشان می‌دهد. لایه‌های کانولوشن در خروجی این کد از شماره 1 تا 16 شماره‌گذاری می‌شوند که شماره واقعی متناظر با هر لایه در لیست کل لایه‌ها در قالب یک دیکشنری با عنوان conv_layers_index ذخیره می‌شود. (این دیکشنری در مراحل بعدی کد، برای دسترسی صحیح به لایه‌های کانولوشن مورد استفاده قرار خواهد گرفت.)

```
n_conv_layers = 0
index = 0
conv_layers_index = {}

for layer in model.layers:
  if 'conv' in layer.name:
```

```
    n_conv_layers +=1
    conv_layers_index[n_conv_layers] = index

    filters, biases = layer.get_weights()
    print(n_conv_layers , layer.name, filters.shape)

  index += 1

print('\nNumber of Conv Layers= %d' % n_conv_layers)
print('\nConv Layers Index:\n' , conv_layers_index)
```

```
1 block1_conv1 (3, 3, 3, 64)
2 block1_conv2 (3, 3, 64, 64)
3 block2_conv1 (3, 3, 64, 128)
4 block2_conv2 (3, 3, 128, 128)
5 block3_conv1 (3, 3, 128, 256)
6 block3_conv2 (3, 3, 256, 256)
7 block3_conv3 (3, 3, 256, 256)
8 block3_conv4 (3, 3, 256, 256)
9 block4_conv1 (3, 3, 256, 512)
10 block4_conv2 (3, 3, 512, 512)
11 block4_conv3 (3, 3, 512, 512)
12 block4_conv4 (3, 3, 512, 512)
13 block5_conv1 (3, 3, 512, 512)
14 block5_conv2 (3, 3, 512, 512)
15 block5_conv3 (3, 3, 512, 512)
16 block5_conv4 (3, 3, 512, 512)

Number of Conv Layers= 16

Conv Layers Index:
 {1: 1, 2: 2, 3: 4, 4: 5, 5: 7, 6: 8, 7: 9, 8: 10, 9: 12, 10: 13, 11: 14, 12: 15, 13: 17, 14:
18, 15: 19, 16: 20}
```

همانطور که در خروجی این کد مشاهده می‌شود، ساختار هر لایه conv با 4 عدد نشان داده شده است. دو عدد اول اندازه فیلترها، عدد سوم تعداد کانالها، و عدد چهارم تعداد فیلترها را نشان می‌دهد. به عنوان مثال اولین لایه شامل 64 فیلتر با اندازه 3*3 که دارای سه کانال است (چون تصویر ورودی یک تصویر رنگی است که با سه کانال RGB مشخص می‌شود) می‌باشد.

2- بصری سازی فیلترها

در این شبکه در همه لایه‌ها از فیلترهایی با اندازه 3*3 استفاده شده است. بنابراین می‌توان هر فیلتر را مانند یک تصویر 3*3 در نظر گرفت و آن را نمایش داد.

ابتدا شماره لایه Conv مورد نظر (عددی بین 1 تا 16) که مایل به نمایش فیلترهای آن هستیم را مشخص می‌کنیم:

```
conv_layer_index = int(input('Which Conv Layer do you want to visualize (1,%d)?' %n_conv_layers))

if conv_layer_index not in range(1,n_conv_layers+1):
  print('Please Enter a value between 1 and %d' %n_conv_layers)
```

```
Which Conv Layer do you want to visualize (1,16)?1
```

در گام بعدی نام لایه منتخب و تعداد فیلترها و تعداد کانالهای آن نمایش داده می‌شود و تعداد فیلتر و کانال مورد نظر برای بصری سازی تعیین می‌شود.

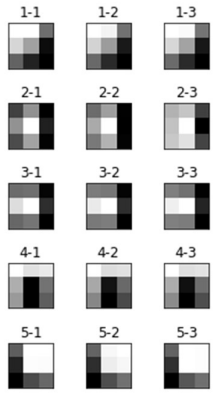
```
layer_index = conv_layers_index[conv_layer_index]
print('Selected Layer: %s'  %model.layers[layer_index].name)

filters, biases = model.layers[layer_index].get_weights()

number_of_filters = filters.shape[3]
print('Number of Filters = %d' %number_of_filters)

number_of_channels = filters.shape[2]
print('Number of Channels = %d' %number_of_channels)

number_of_filters_to_visualize = int(input('How many filters do you want to visualize (1,%d)?
' %number_of_filters))

if number_of_filters_to_visualize not in range(1,number_of_filters+1):
  print('Please Enter a value between 1 and %d' %number_of_filters)


number_of_channels_to_visualize = int(input('How many channels do you want to visualize (1,%d
)?' %number_of_channels))

if number_of_channels_to_visualize not in range(1,number_of_channels+1):
  print('Please Enver a value between 1 and %d' %number_of_channels)
```

```
Selected Layer: block1_conv1
Number of Filters = 64
Number of Channels = 3
How many filters do you want to visualize (1,64)?5
How many channels do you want to visualize (1,3)?3
```

برای نمایش فیلترها ابتدا مقادیر آنها در بازه 0 تا 1 نرمالسازی می‌شود. سپس تصویری با اندازه تعداد کانالها*تعداد فیلترها ایجاد میشود و در هر سطر کانالهای مربوط به هر فیلتر نمایش داده می شود.

```
#-- Plot Filters ----------------------------------------------------------------

#-- Normalize Values to the Range 0-1 --
f_min, f_max = filters.min(), filters.max()
filters = (filters - f_min) / (f_max - f_min)

n_filters, ix = number_of_filters_to_visualize, 1
n_channels = number_of_channels_to_visualize

#-- Set Figure Size --
fig = plt.figure(figsize=(n_channels,n_filters ))

for i in range(n_filters):
  #-- Get the Filter --
  f = filters[:, :, :, i]

  #-- Plot Each Channel Separately --
```

```
  for j in range(n_channels):

    #-- Specify Subplot and Turn of Axis --
    ax = plt.subplot(n_filters, n_channels, ix)
    ax.set_xticks([])
    ax.set_yticks([])

    #-- Set Title: Channle#-Filter# --
    ax.set_title(str(i+1) + '-' + str(j+1))

    #-- Plot Filter Channel in Grayscale --
    plt.imshow(f[:, :, j], cmap='gray')
    ix += 1


plt.tight_layout()
plt.show()
```



به عنوان مثال این تصویر هر سه کانال از 5 فیلتر اول از لایه کانولوشن اول را نشان میدهد. عنوان هر تصویر به ترتیب شماره فیلتر و شماره کانال مربوط به آن فیلتر را نشان میدهد.

3- بصری سازی خروجی هر لایه

به طور کلی در یک شبکه Conv در هر لایه یک تصویر به عنوان ورودی دریافت می‌شود. فیلترهای یادگرفته شده در این لایه، روی این ورودی اعمال می‌شود و تصویر جدیدی در خروجی تولید می‌شود. با بصری سازی تصویر خروجی هر لایه، می‌توانیم ویژگی‌های مختلفی که در هر لایه یادگرفته می‌شود را نمایش دهیم.

در این بخش برای بصری سازی خروجی هر لایه از تصویر زیر استفاده می‌کنیم:

شکل 2: تصویر مورد استفاده برای بصری سازی خروجی لایه‌های Conv

(لینک دانلود تصویر: https://machinelearningmastery.com/wp-content/uploads/2019/02/bird.jpg)

ابتدا شماره هر لایه Conv (1 تا 16) و ابعاد خروجی آن نمایش داده می‌شود:

```
index = 1
for layer in model.layers:
  if 'conv' not in layer.name:
    continue

  print(index , layer.name, layer.output.shape)

  index +=1
```

```
1 block1_conv1 (None, 224, 224, 64)
2 block1_conv2 (None, 224, 224, 64)
3 block2_conv1 (None, 112, 112, 128)
4 block2_conv2 (None, 112, 112, 128)
5 block3_conv1 (None, 56, 56, 256)
6 block3_conv2 (None, 56, 56, 256)
7 block3_conv3 (None, 56, 56, 256)
8 block3_conv4 (None, 56, 56, 256)
9 block4_conv1 (None, 28, 28, 512)
10 block4_conv2 (None, 28, 28, 512)
11 block4_conv3 (None, 28, 28, 512)
12 block4_conv4 (None, 28, 28, 512)
13 block5_conv1 (None, 14, 14, 512)
14 block5_conv2 (None, 14, 14, 512)
15 block5_conv3 (None, 14, 14, 512)
16 block5_conv4 (None, 14, 14, 512)
```

سپس لایه مورد نظر برای نمایش خروجی انتخاب می‌شود:

```
conv_layer_index = int(input('Which Layer do you want to visualize Feature Maps (1,%d)?' %n_c
onv_layers))

if conv_layer_index not in range(1,n_conv_layers+1):
  print('Please Enter a value between 1 and %d' %n_conv_layers)
```

```
Which Layer do you want to visualize Feature Maps (1,16)?1
```

در گام بعدی بخشی از مدل VGG19 انتخاب می‌شود که شامل لایه ورودی تا لایه انتخاب شده می‌باشد. بنابراین شبکه‌ای خواهیم داشت که تصویر را در ورودی دریافت می‌کند و تا لایه منتخب پیش می‌رود.

```
layer_index = conv_layers_index[conv_layer_index]
```

```
print(model.layers[layer_index].name)

#-- Redefine Model to Output Right After the Selected Layer
refined_model = Model(inputs=model.inputs, outputs=model.layers[layer_index].output)

refined_model.summary()
```

```
block1_conv1
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

=================================================================
Total params: 1,792
Trainable params: 1,792
Non-trainable params: 0
_________________________________________________________________
```

به عنوان مثال در اینجا اولین لایه کانولوشن انتخاب شده است. این شبکه جدید، تصویر ورودی را دریافت می‌کند و آن را از لایه کانولوشن عبور میدهد و در خروجی نتیجه پردازش توسط این لایه، تولید می‌شود. اکنون می‌توانیم یک تصویر را به عنوان ورودی به این شبکه ارسال کنیم نتیجه پردازش را مشاهده کنیم.

ابتدا تصویر مورد نظر بارگذاری می‌شود. با توجه به اینکه شبکه VGG19 تصویری در ابعاد 224*224 دریافت می‌کند لازم است ابعاد تصویر ورودی به این مقادیر تبدیل شود.

```
input_size = refined_model.layers[0].input.shape
img_size = (input_size[1] , input_size[2])

#-- Load the Image with the Required Shape --
img = load_img('bird.jpg', target_size=img_size)
```

برای ارسال این تصویر به شبکه لازم است این تصویر به یک ارایه Numpy از مقادیر پیکسلها تبدیل شود و در قالب یک آرایه 4 بعدی به صورت [samples, rows, cols, channels] تبدیل بشود (که در اینجا samples=1). و در نهایت لازم است مقادیر پیکسلها به مقیاس مناسب برای شبکه VGG19 تبدیل شوند.

```
#-- Convert the Image to an Array --
img = img_to_array(img)

#-- Expand Dimensions so that it Represents a Single 'sample' --
img = expand_dims(img, axis=0)

#-- Prepare the Image (e.g. scale pixel values for the vgg) --
img = preprocess_input(img)
```

برای تولید Feature Map های لایه منتخب، کافی است تصویر مورد نظر به شبکه ایجاد شده از روی مدل اصلی، ارسال شود و از متد Predict برای تولید خروجی استفاده شود. شبکه تصویر را دریافت می‌کند. فیلترهایی که در فرایند آموزش یادگرفته شده را روی آن اعمال می‌کند و خروجی نهایی تولید می‌شود.

```
#-- Get Feature Map for Selected Layer --
feature_maps = refined_model.predict(img)

#-- Show output Size --
print(feature_maps.shape)
```

```
1/1 [==============================] - 1s 508ms/step
(1, 224, 224, 64)
```

به عنوان مثال، اولین لایه کانولوشن انتخاب شده است. همانطور که در بخشهای فوق اشاره شد، این لایه از 64 فیلتر 3*3 استفاده میکند. بنابراین در خروجی 64 تصویر خواهیم داشت. به ازای هر فیلتر اعمال شده، یک تصویر در خروجی تولید می‌شود.

در گام بعدی تعداد Feature Map های حاصل در خروجی نمایش داده می شود و تعداد مورد نظر برای نمایش، تعیین می‌شود.

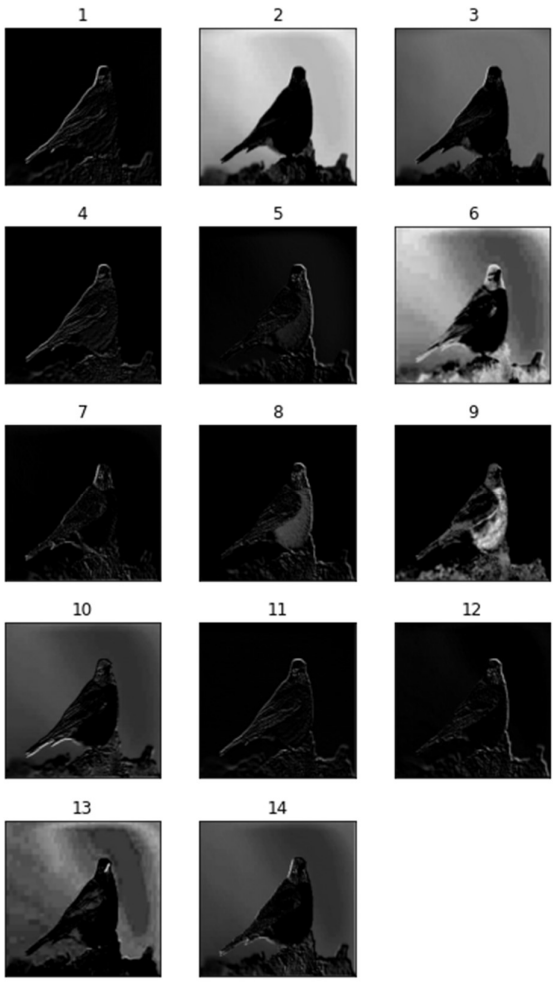
```
number_of_feature_maps = feature_maps.shape[3]
print('Number of Feature Maps = %d' %number_of_feature_maps)

number_of_feature_maps_to_visualize = int(input('How many feature maps do you want to visuali
ze (1,%d)?' %number_of_feature_maps))

if number_of_feature_maps_to_visualize not in range(1,number_of_feature_maps+1):
  print('Please Enter a value between 1 and %d' %number_of_feature_maps)
```

```
Number of Feature Maps = 64
How many feature maps do you want to visualize (1,64)?14
```

در نهایت feature map های تولید شده (به تعداد انتخاب شده) در خروجی نمایش داده می‌شوند.

```
#-- Set size of Figure --
h = int(math.sqrt(number_of_feature_maps_to_visualize))
w = math.ceil(number_of_feature_maps_to_visualize/h)
fig = plt.figure(figsize=(h*2,w*2 ))

ix = 1
for _ in range(w):
  for _ in range(h):

    #-- Specify Subplot And Turn of Axis --
    ax = plt.subplot(w, h, ix)
    ax.set_xticks([])
    ax.set_yticks([])

    #-- Set Title: feature_map# --
    ax.set_title(ix)

    #-- Plot Feature Maps --
    plt.imshow(feature_maps[0, :, :, ix-1], cmap='gray')

    ix += 1
    if ix>number_of_feature_maps_to_visualize:
      break
```

```
plt.tight_layout()
plt.show()
```



تصویر فوق 14 خروجی اول تولید شده در اولین لایه کانولوشن را نشان می‌دهد. همانطور که مشاهده می‌شود هر فیلتر ویژگی‌های متفاوتی را یاد می‌گیرد.

4- بصری سازی فیلترها و خروجی هر بلاک

همانطور که در شکل 1 نشان داده شده است، شبکه VGG19 شامل 5 بلاک است. در این بخش، فیلترها و Feature Map های خروجی در هر بلاک از شبکه VGG19 نمایش داده می‌شود.

کد زیر شماره اندیس لایه‌های Conv در هر بلاک را نمایش میدهد (مقادیر هایلایت شده، اندیس اخرین لایه Conv در هرلایه را نشان می‌دهد):

```
n_conv_layers = 0
index = 0

#-- key: conv_number : value:index --
conv_layers_index = {}

for layer in model.layers:
```

```
  if 'conv' in layer.name:
    n_conv_layers +=1
    filters, biases = layer.get_weights()
    print(index , n_conv_layers , layer.name, filters.shape)

  index += 1
```

```
1 1 block1_conv1 (3, 3, 3, 64)
2 2 block1_conv2 (3, 3, 64, 64)
4 3 block2_conv1 (3, 3, 64, 128)
5 4 block2_conv2 (3, 3, 128, 128)
7 5 block3_conv1 (3, 3, 128, 256)
8 6 block3_conv2 (3, 3, 256, 256)
9 7 block3_conv3 (3, 3, 256, 256)
10 8 block3_conv4 (3, 3, 256, 256)
12 9 block4_conv1 (3, 3, 256, 512)
13 10 block4_conv2 (3, 3, 512, 512)
14 11 block4_conv3 (3, 3, 512, 512)
15 12 block4_conv4 (3, 3, 512, 512)
17 13 block5_conv1 (3, 3, 512, 512)
18 14 block5_conv2 (3, 3, 512, 512)
19 15 block5_conv3 (3, 3, 512, 512)
20 16 block5_conv4 (3, 3, 512, 512)
```

همانطور که در خروجی کد فوق مشخص شده، اخرین لایه‌های Conv در هر بلاک به ترتیب لایه‌ها 2، 5، 10، 15 و 20 هستند. در ادامه از روی مدل VGG19 مدلی می‌سازیم که خروجی همه این لایه‌ها را در خروجی تولید می‌کند.

```
ixs = [2, 5, 10, 15, 20]
outputs = [model.layers[i].output for i in ixs]
refined_model = Model(inputs=model.inputs, outputs=outputs)

refined_model.summary()
```

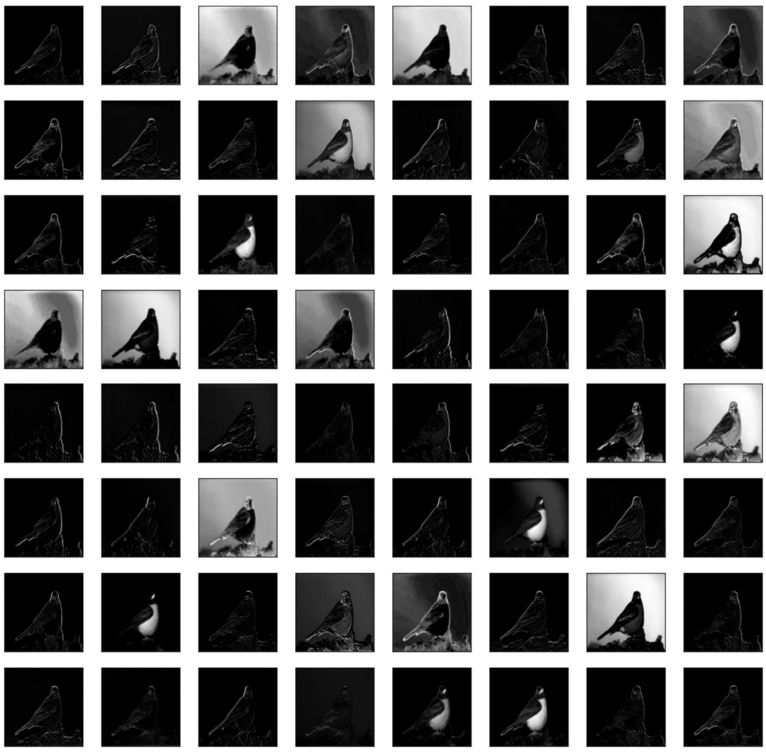
در ادامه یک تصویر به عنوان ورودی به این مدل ارسال می‌شود و Feature Map های تولید شده در هر بلاک در خروجی تولید می‌شود.

```
#-- Load the Image --
img = load_img('bird.jpg', target_size=(224, 224))

#-- Convert the Image to an Array --
img = img_to_array(img)

#-- Expand Dimensions --
img = expand_dims(img, axis=0)

#-- Prepare the Image --
img = preprocess_input(img)

#-- Get Feature Maps --
feature_maps = refined_model.predict(img)
```

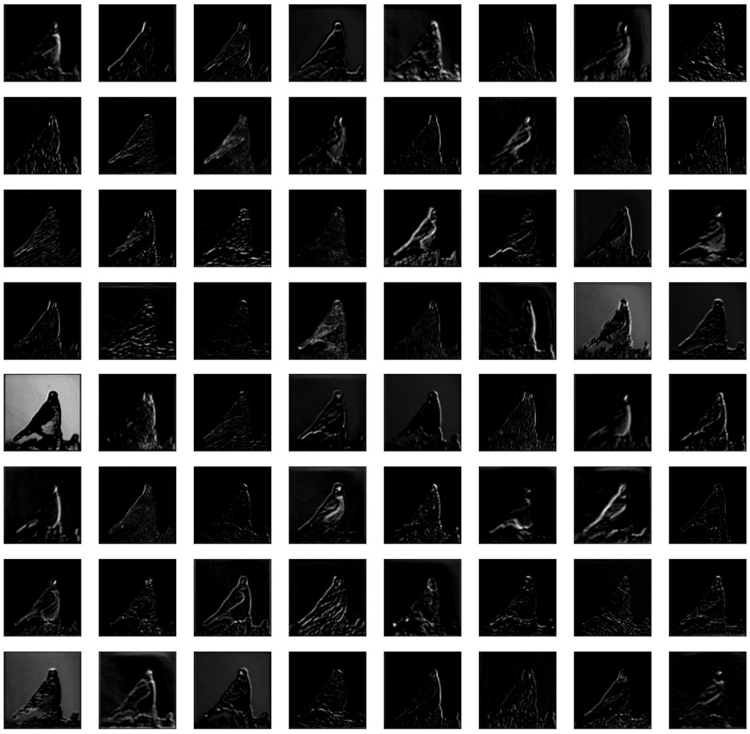
و در نهایت Feature Map های تولید شده نمایش داده می‌شوند. با توجه به اینکه تعداد کانالها و فیلترها در لایه‌های مختلف متفاوت است، Feature Map های تولید شده در انتهای هر بلاک متفاوت است. در هر لایه حداقل 64 خروجی تولید می‌شوند. در این بخش از هر لایه، 64 خروجی اول در قالب یک تصویر 8*8 نشان داده می‌شوند:

```
square = 8

for fmap in feature_maps:
  print("=============================================================================")
  fig = plt.figure(figsize=(square*2 , square*2 ))
  #-- Plot 64 Maps in an 8x8 Squares
  ix = 1
  for _ in range(square):
    for _ in range(square):
      #-- Specify Subplot and Turn of Axis
      ax = plt.subplot(square, square, ix)
      ax.set_xticks([])
      ax.set_yticks([])
      #-- Plot Filter Channel in Grayscale
      plt.imshow(fmap[0, :, :, ix-1], cmap='gray')
      ix += 1
  # show the figure
  plt.show()
  print("=============================================================================")
```

Feature Map های خروجی در بلاک 1

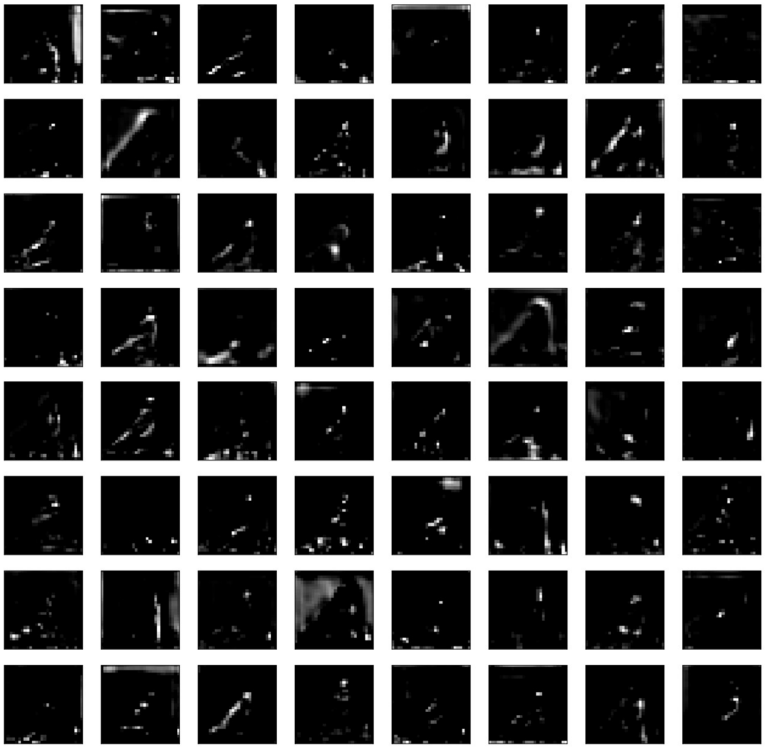


Feature Map های خروجی در بلاک 2

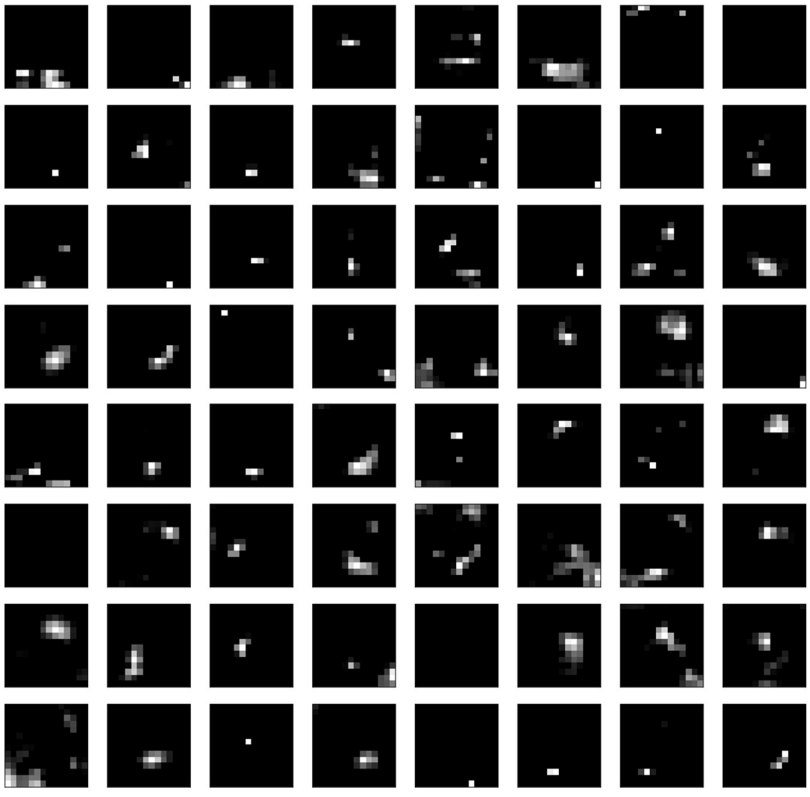


Feature Map های خروجی در بلاک 3

Feature Map های خروجی در بلاک 4



Feature Map های خروجی در بلاک 5



5- نتیجه‌گیری

به طور کلی، در شبکه‌های کانولوشن لایه‌های ابتدایی ویژگی‌های عمومی (مانند لبه) در تصاویر را استخراج می‌کنند و لایه‌های پایانی ویژگی‌های انتزاعی که در دسته‌بندی صحیح تصاویر موثر هستند (مانند نوک پرنده، گوش گربه و ...) را استخراج می‌کنند.

نتایج نمایش‌داده شده در بخش قبلی، این مساله را تایید می‌کند. همانطور که در تصاویر مشخص است خروجی بلاک 1، تصویر پرنده قابل مشاهده و تشخیص است و فیلترهای عمومی مانند لبه بالا، لبه پایین و ... استخراج شده‌اند. هر چه به سمت بلاک 5 پیش می‌رویم، ویژگی‌ها پیچیده‌تر می‌شوند به عنوان مثال در تصاویر حاصل از بلاک 4 و 5، پرنده در تصویر خیلی قابل تشخیص نیست درواقع خروجی خیلی مشابه تصویر اصلی نیست و نوعی نمایش انتزاعی از تصویر اولیه است. به طور کلی فیلترهای پایانی برای انسان خیلی قابل درک و تفسیر نیستند اما در عین حال حاوی اطلاعات مهمی در تشخیص کلاس تصاویر می‌باشند.