
بسم الله الرحمن الرحيم

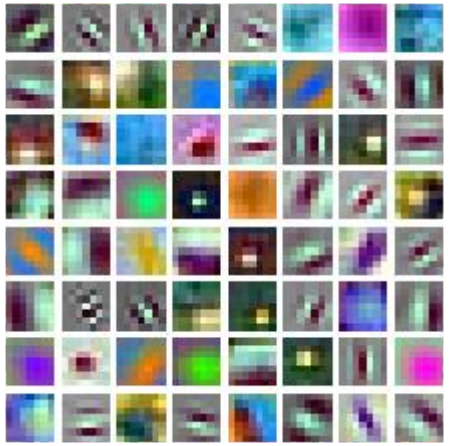
مصوّرسازی

Visualization

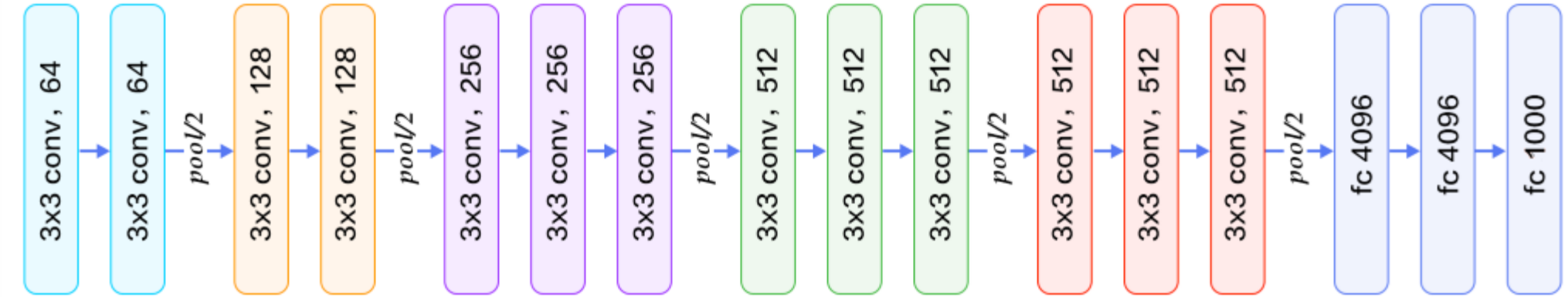
نمایش فیلترها

VGG16

ResNet101



- فیلترهای لایه اول به راحتی قابل نمایش و تفسیر هستند
- فیلترهای لایه‌های بعد به سادگی قابل تفسیر نیستند
  - ابعاد فیلترها خیلی بالا است
  - روی خروجی لایه قبل اعمال می‌شود



نمایش فیلترها

- یکی راه ساده دیگر برای بررسی فیلترهای آموخته شده توسط شبکه‌های کانولوشنی، نمایش الگوی بصری است که هر فیلتر قرار است به آن پاسخ دهد
- این کار را می توان با گرادیان افزایشی در فضای ورودی انجام داد
  - بهینه‌سازی تصویر ورودی به منظور به حداکثر رساندن پاسخ یک فیلتر خاص
- تصویر ورودی به دست آمده تصویری خواهد بود که فیلتر انتخاب شده حداکثر پاسخ را به آن می‌دهد
- برای این منظور، یک تابع ضرر تعریف می‌کنیم که مقدار خروجی یک فیلتر معین در یک لایه کانولوشنی مشخص را محاسبه کند

Listing 5.32 Defining the loss tensor for filter visualization

```
from keras.applications import VGG16
from keras import backend as K

model = VGG16(weights='imagenet',
              include_top=False)

layer_name = 'block3_conv1'
filter_index = 0

layer_output = model.get_layer(layer_name).output
loss = K.mean(layer_output[:, :, :, filter_index])
```

Listing 5.38 Function to generate filter visualizations

Builds a loss function that maximizes the activation of the nth filter of the layer under consideration

Computes the gradient of the input picture with regard to this loss

Normalization trick: normalizes the gradient

Returns the loss and grads given the input picture

Starts from a gray image with some noise

Runs gradient ascent for 40 steps

```
def generate_pattern(layer_name, filter_index, size=150):
    layer_output = model.get_layer(layer_name).output
    loss = K.mean(layer_output[:, :, :, filter_index])

    grads = K.gradients(loss, model.input)[0]

    grads /= (K.sqrt(K.mean(K.square(grads))) + 1e-5)

    iterate = K.function([model.input], [loss, grads])

    input_img_data = np.random.random((1, size, size, 3)) * 20 + 128.

    step = 1.
    for i in range(40):
        loss_value, grads_value = iterate([input_img_data])
        input_img_data += grads_value * step

    img = input_img_data[0]
    return deprocess_image(img)
```

Listing 5.37 Utility function to convert a tensor into a valid image

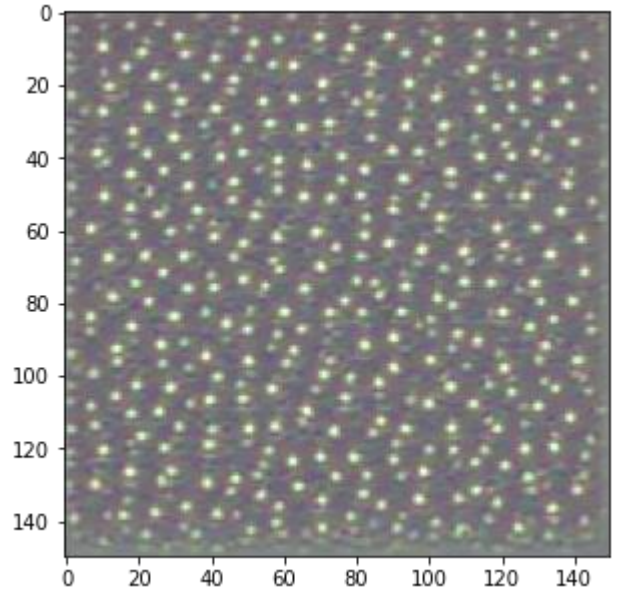
```
def deprocess_image(x):
    x -= x.mean()
    x /= (x.std() + 1e-5)
    x *= 0.1

    x += 0.5
    x = np.clip(x, 0, 1)

    x *= 255
    x = np.clip(x, 0, 255).astype('uint8')
    return x
```

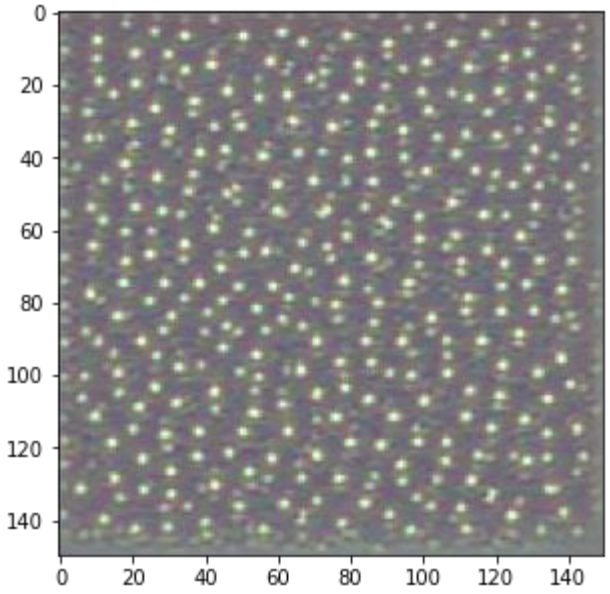
Normalizes the tensor: centers on 0, ensures that std is 0.1

Clips to [0, 1]

Converts to an RGB array

```
>>> plt.imshow(generate_pattern('block3_conv1', 0))
```





Listing 5.39 Generating a grid of all filter response patterns in a layer

```
layer_name = 'block1_conv1'
size = 64
margin = 5

results = np.zeros((8 * size + 7 * margin, 8 * size + 7 * margin, 3))

for i in range(8):
    for j in range(8):
        filter_img = generate_pattern(layer_name, i + (j * 8), size=size)

        horizontal_start = i * size + i * margin
        horizontal_end = horizontal_start + size
        vertical_start = j * size + j * margin
        vertical_end = vertical_start + size
        results[horizontal_start: horizontal_end,
                vertical_start: vertical_end, :] = filter_img

plt.figure(figsize=(20, 20))
plt.imshow(results)
```

Empty (black) image to store results

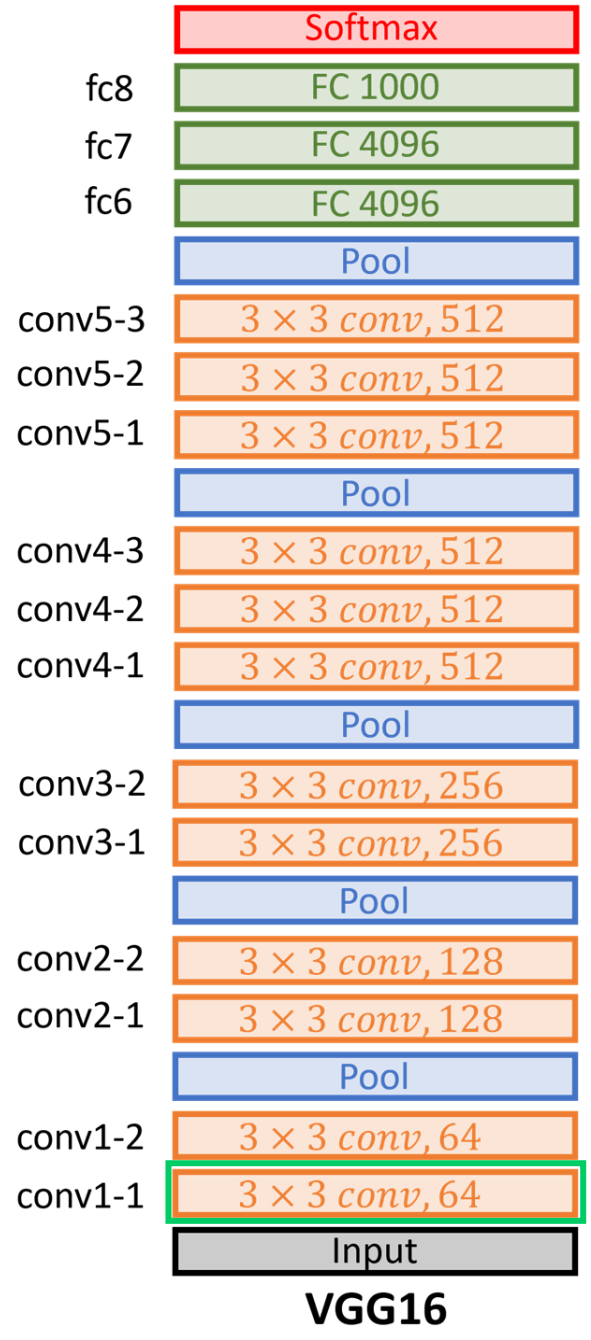
Iterates over the rows of the results grid

Iterates over the columns of the results grid

Generates the pattern for filter i + (j * 8) in layer_name

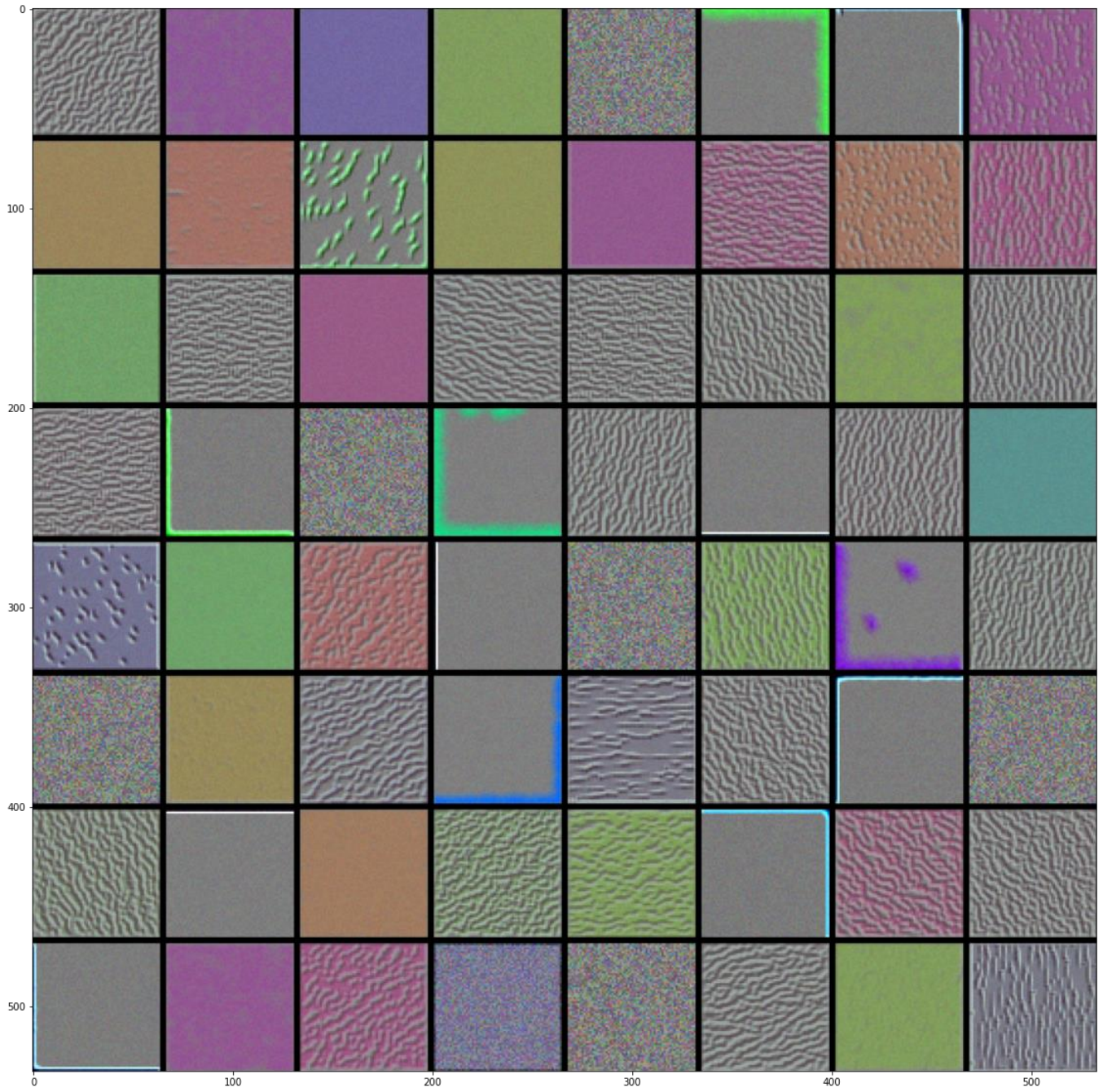
Puts the result in the square (i, j) of the results grid

Displays the results grid



VGG16

64, 3×3×3 filters



Softmax

fc8 FC 1000

fc7 FC 4096

fc6 FC 4096

Pool

conv5-3 $3 \times 3\ conv, 512$

conv5-2 $3 \times 3\ conv, 512$

conv5-1 $3 \times 3\ conv, 512$

Pool

conv4-3 $3 \times 3\ conv, 512$

conv4-2 $3 \times 3\ conv, 512$

conv4-1 $3 \times 3\ conv, 512$

Pool

conv3-2 $3 \times 3\ conv, 256$

conv3-1 $3 \times 3\ conv, 256$

Pool

conv2-2 $3 \times 3\ conv, 128$

conv2-1 $3 \times 3\ conv, 128$

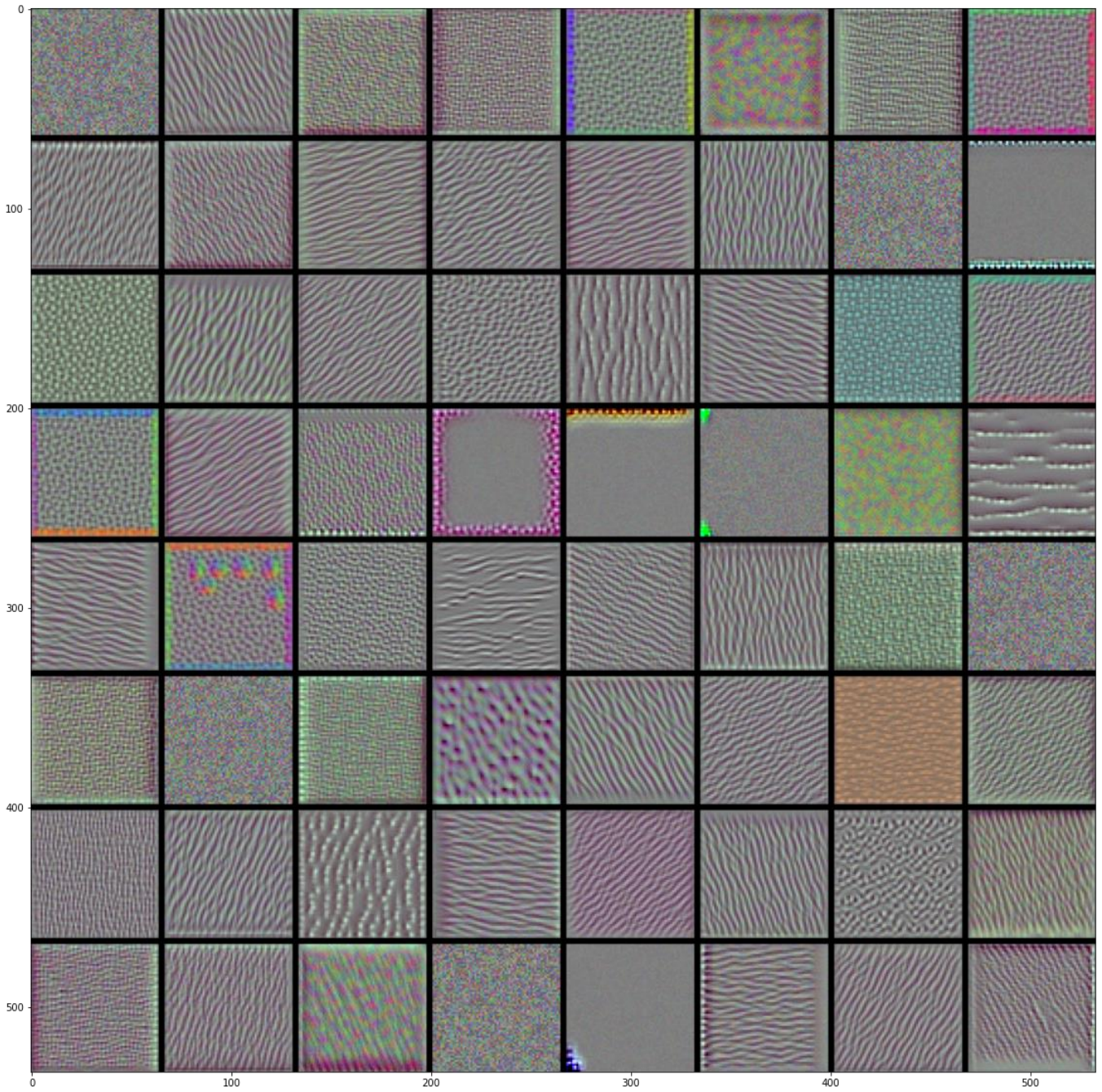
Pool

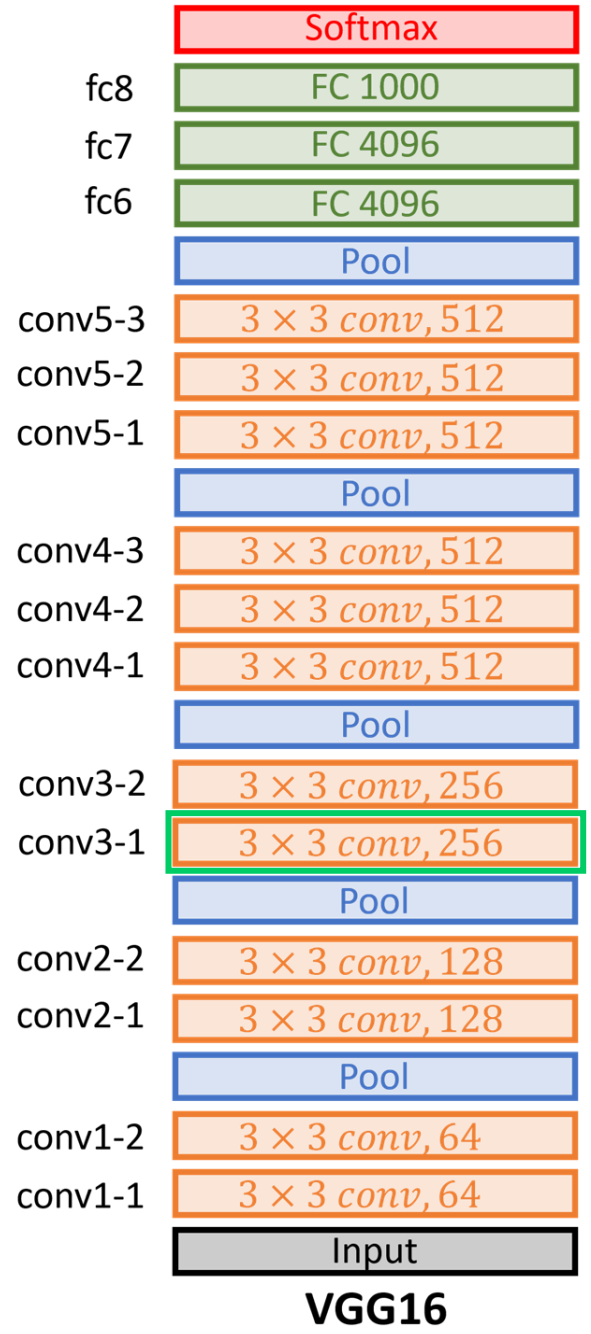
conv1-2 $3 \times 3\ conv, 64$

conv1-1 $3 \times 3\ conv, 64$

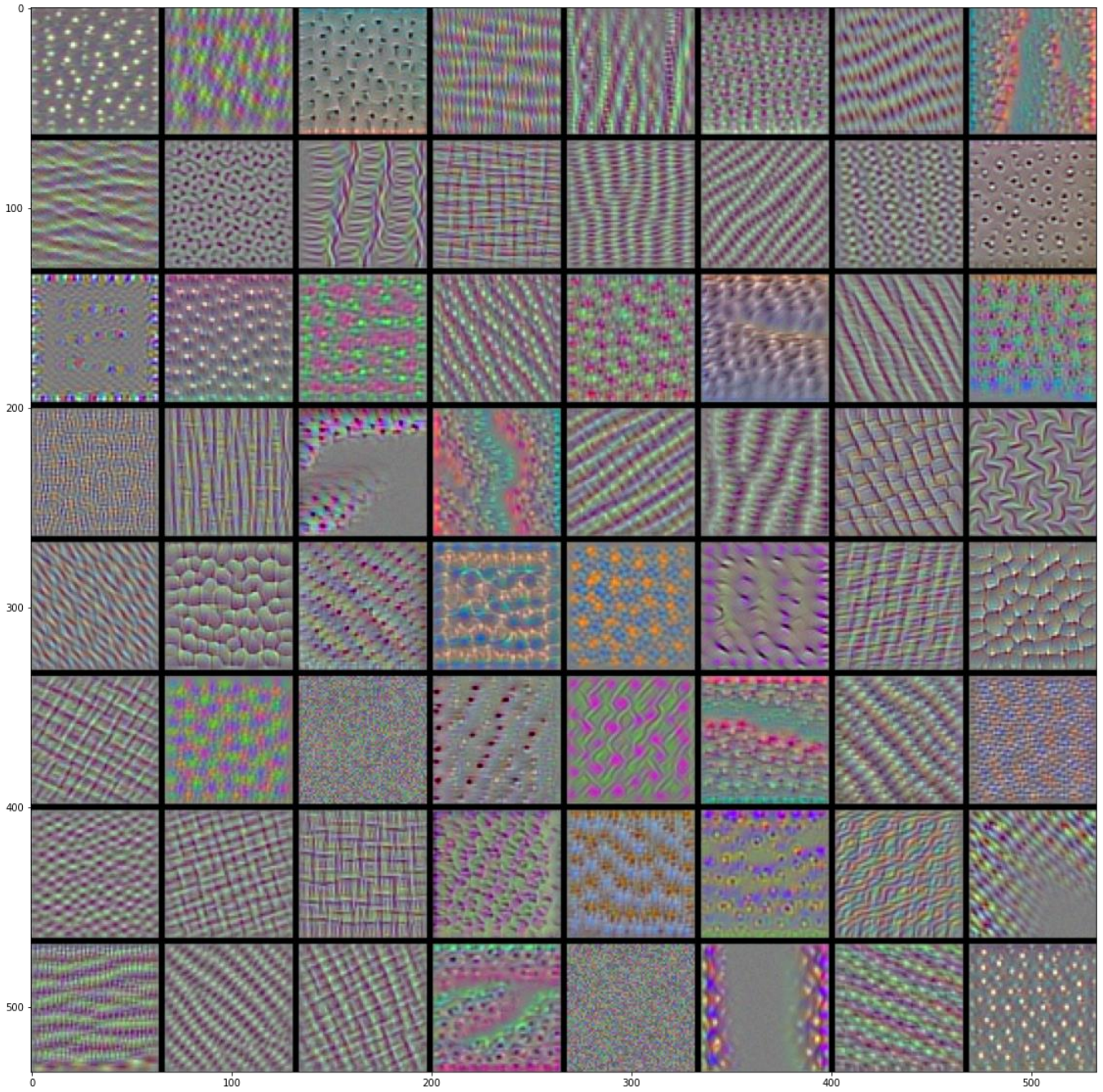
Input

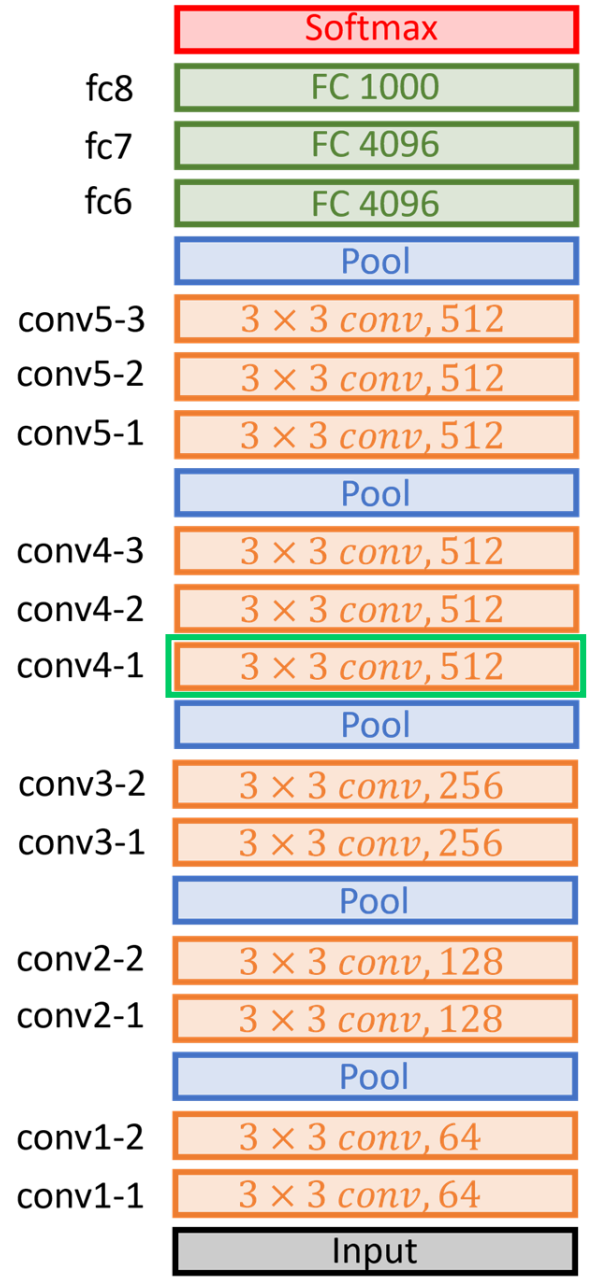
VGG16



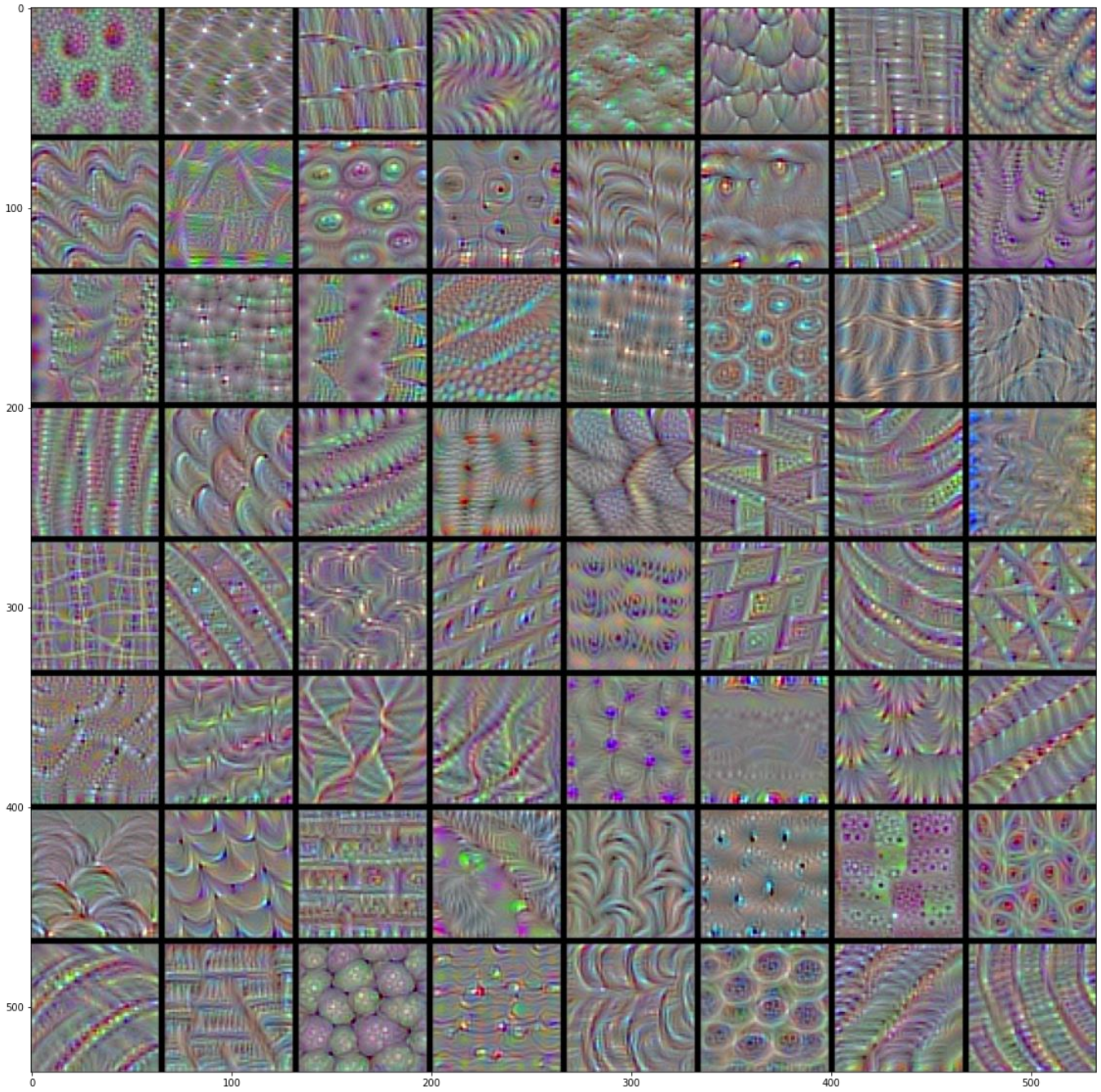

Softmax
fc8
FC 1000
fc7
FC 4096
fc6
FC 4096
Pool
conv5-3
3 × 3 conv, 512
conv5-2
3 × 3 conv, 512
conv5-1
3 × 3 conv, 512
Pool
conv4-3
3 × 3 conv, 512
conv4-2
3 × 3 conv, 512
conv4-1
3 × 3 conv, 512
Pool
conv3-2
3 × 3 conv, 256
conv3-1
3 × 3 conv, 256
Pool
conv2-2
3 × 3 conv, 128
conv2-1
3 × 3 conv, 128
Pool
conv1-2
3 × 3 conv, 64
conv1-1
3 × 3 conv, 64
Input

VGG16

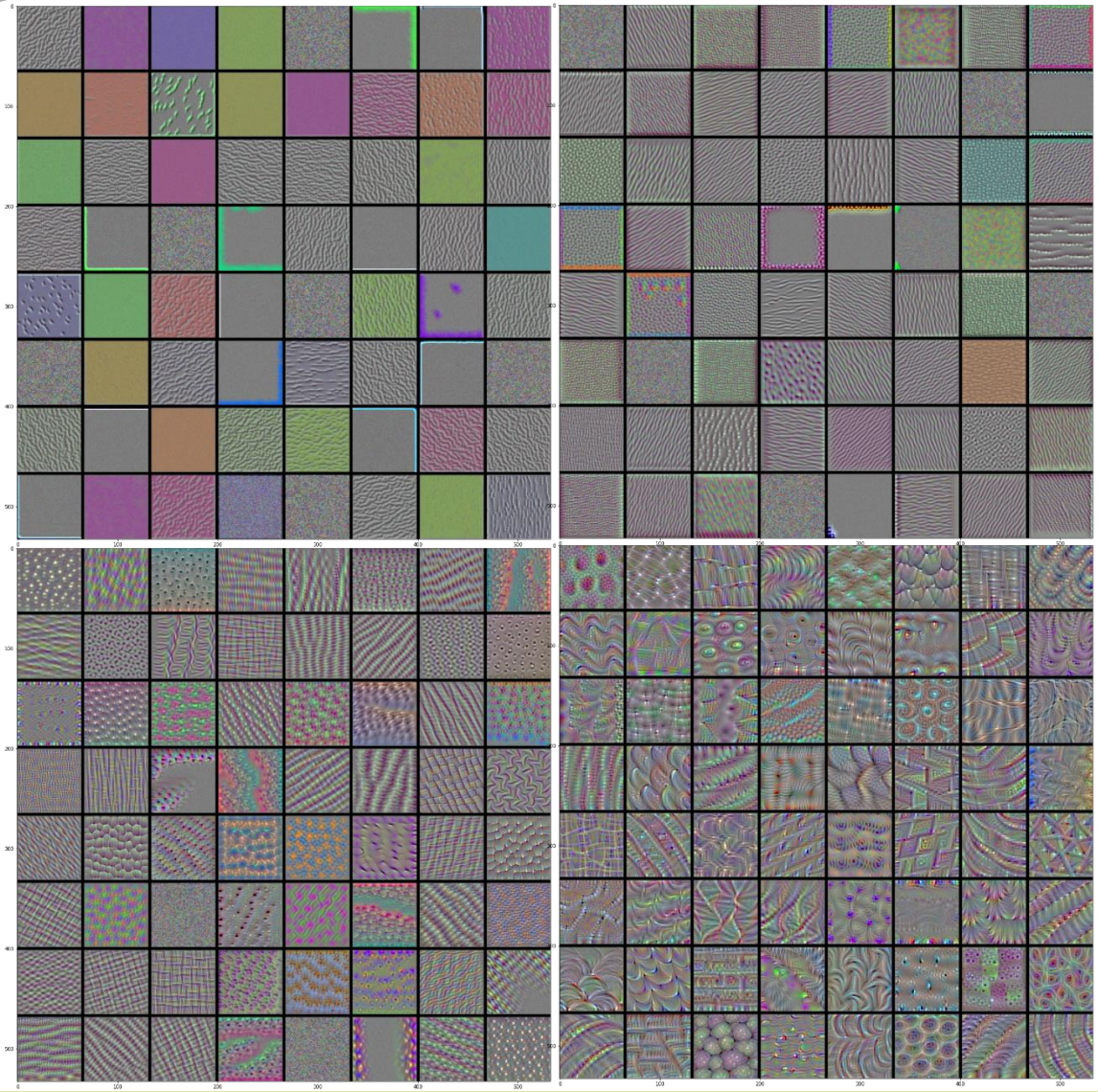

0
100
200
300
400
500
0
100
200
300
400
500


Softmax
fc8
FC 1000
fc7
FC 4096
fc6
FC 4096
Pool
conv5-3
3 × 3 conv, 512
conv5-2
3 × 3 conv, 512
conv5-1
3 × 3 conv, 512
Pool
conv4-3
3 × 3 conv, 512
conv4-2
3 × 3 conv, 512
conv4-1
3 × 3 conv, 512
Pool
conv3-2
3 × 3 conv, 256
conv3-1
3 × 3 conv, 256
Pool
conv2-2
3 × 3 conv, 128
conv2-1
3 × 3 conv, 128
Pool
conv1-2
3 × 3 conv, 64
conv1-1
3 × 3 conv, 64
Input

VGG16


0
100
200
300
400
500
0
100
200
300
400
500

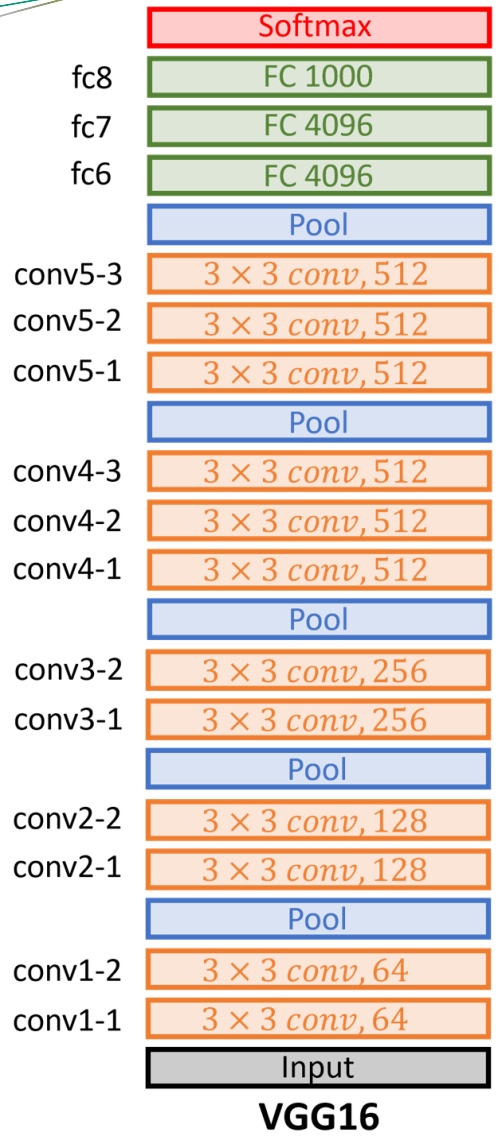
نمایش فیلترها



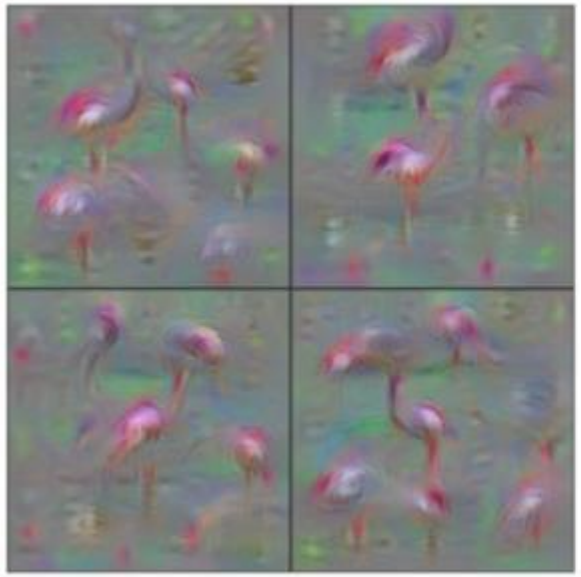
- هرچه در مدل جلو برویم، فیلترها پیچیده‌تر می‌شوند
  - فیلترهای لایه اول مدل لبه‌ها و رنگ‌های جهت‌دار ساده (یا لبه‌های رنگی، در برخی موارد) را آشکار می‌کنند
  - فیلترهای block2_conv1 بافت های ساده ساخته شده از ترکیب لبه‌ها و رنگ‌ها را آشکار می‌کنند
  - فیلترها در لایه های بالاتر شبیه بافت‌های موجود در تصویر طبیعی می شوند، مانند پر، چشم، برگ و غیره.

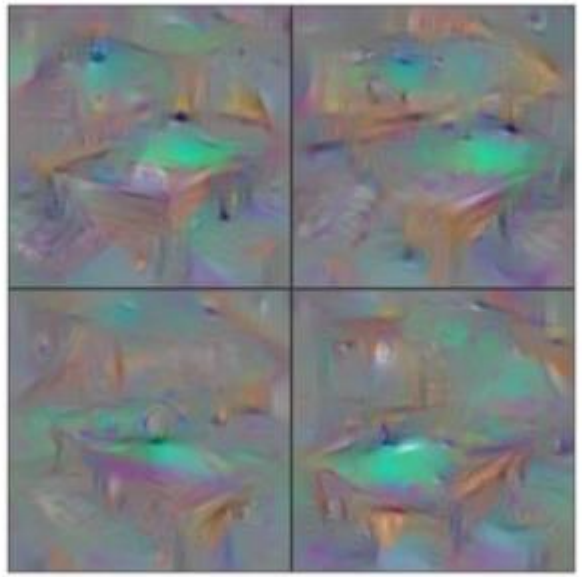
DeepVis

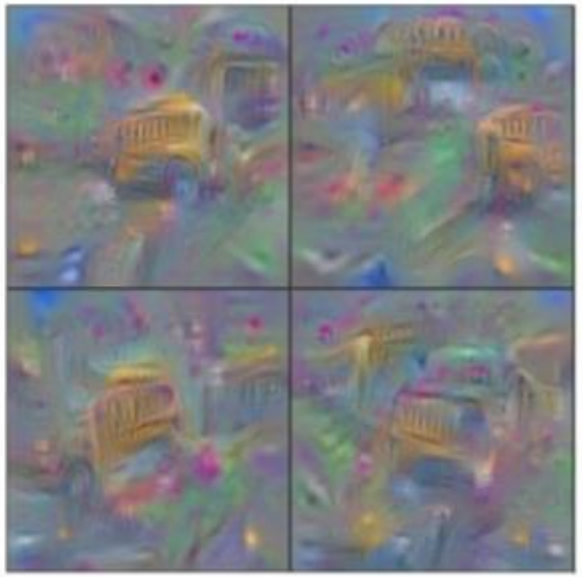
- مصورسازی نورون‌های خروجی
  - تصویر ورودی مصنوعی که به بهترین نحو آن نورون را فعال می‌کند



VGG16


Flamingo

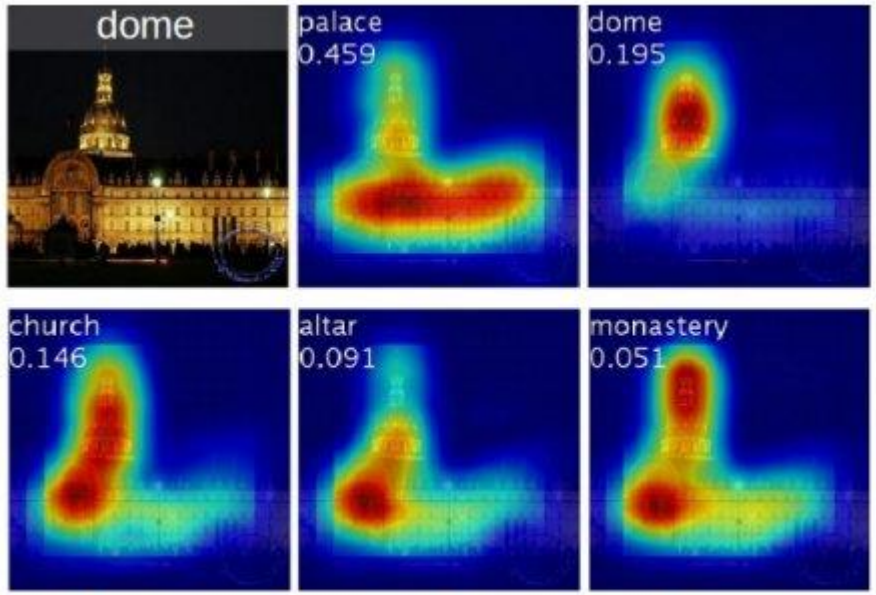

Billiard Table


School Bus

Yosinski, Jason, et al. "Understanding neural networks through deep visualization." *arXiv preprint arXiv:1506.06579* (2015).

نمایش نقشه‌های حرارتی (heatmaps)

- نقشه فعالیت کلاس (Class Activation Map)
- نشان می‌دهد که هر مکان با توجه به کلاس مورد بررسی چقدر اهمیت دارد
- برای درک اینکه یک شبکه بر اساس کدام بخش از یک تصویر به تصمیم نهایی رسیده است



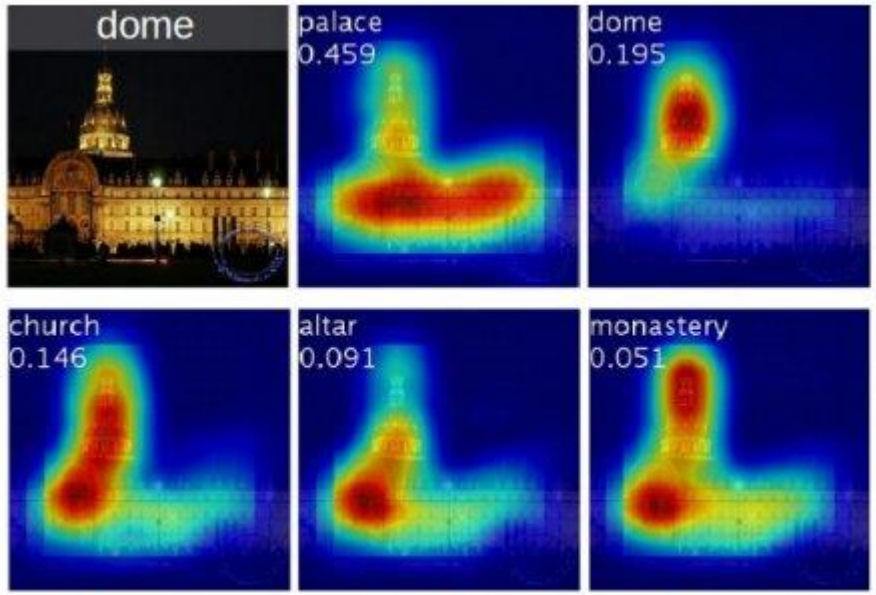
Class activation maps of top 5 predictions

Class activation maps for one object class

https://github.com/zhoubolei/CAM

نمایش نقشه‌های حرارتی (heatmaps)

- نقشه فعالیت کلاس (Class Activation Map)
- نشان می‌دهد که هر مکان با توجه به کلاس مورد بررسی چقدر اهمیت دارد
- برای درک اینکه یک شبکه بر اساس کدام بخش از یک تصویر به تصمیم نهایی رسیده است مفید است
- برای اشکال‌زدایی فرآیند تصمیم‌گیری یک شبکه، به ویژه در مواردی که اشتباه کرده است کمک‌کننده است
- همچنین امکان می‌دهد مکان اشیاء مورد نظر را در یک تصویر تخمین بزنیم



Class activation maps of top 5 predictions

https://github.com/zhoubolei/CAM