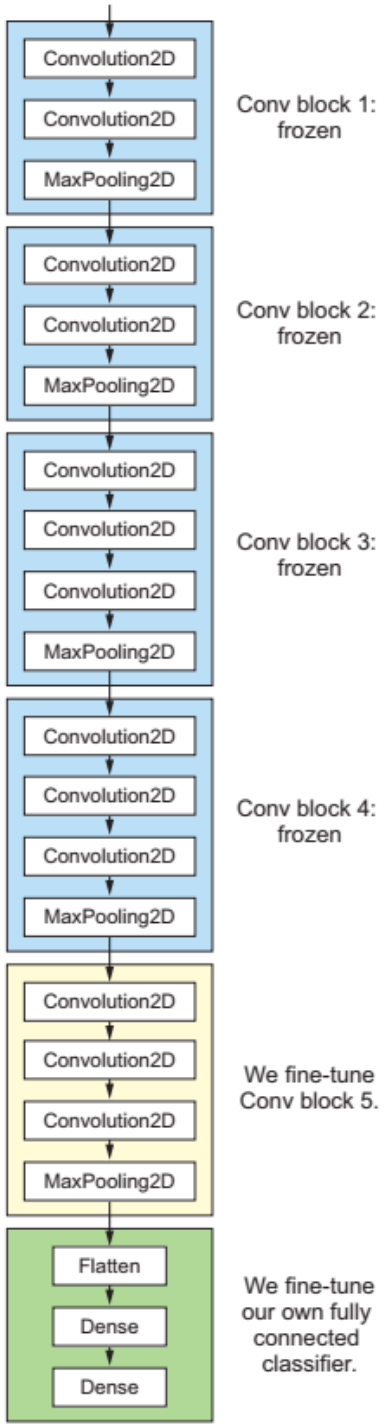

بسم الله الرحمن الرحيم

شبکه‌های عصبی کانولوشنی

Convolutional Neural Networks

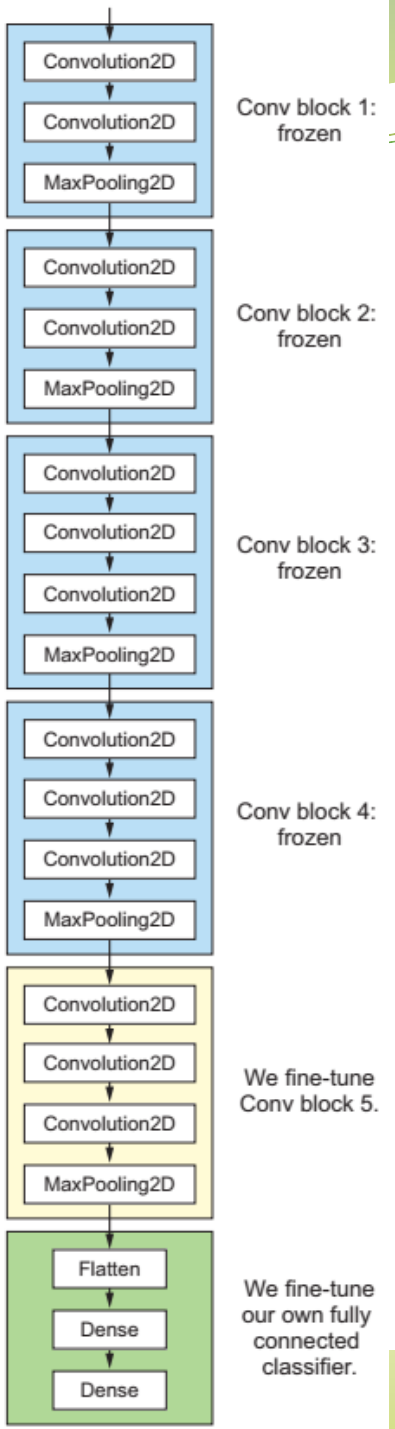
شبکه‌های پیش‌آموخته (Pretrained)

- دو روش رایج برای استفاده از یک شبکه پیش‌آموخته وجود دارد:
  - استخراج ویژگی (feature extraction)
  - تنظیم دقیق (fine tuning)



تنظیم دقیق

- یکی دیگر از شیوه‌های پرکاربرد برای استفاده مجدد از مدل، تنظیم دقیق است
- تعدادی از لایه‌های انتهایی شبکه را از حالت منجمد (freeze) خارج می‌کنیم و آنها را در کنار لایه‌های جدید با نرخ آموزش کمتر آموزش می‌دهیم
- پیشنهاد می‌شود که ابتدا لایه‌های جدید به تنهایی آموزش ببینند و سپس همراه با تعدادی از لایه‌های انتهایی شبکه تنظیم دقیق شوند
- اگر لایه‌های جدید دسته‌بند آموزش داده نشده باشند، سیگنال خطایی که در طول آموزش از طریق شبکه منتشر می‌شود بسیار بزرگ خواهد بود و بازنمایی‌هایی که قبلاً آموزش دیده بودند تخریب می‌شوند



تنظیم دقیق

- بنابراین مراحل تنظیم دقیق شبکه به شرح زیر است:
  - لایه‌های جدید را در انتهای یک شبکه پیش‌آموخته اضافه کنید
  - شبکه پایه را منجمد کنید
  - لایه‌هایی که اضافه کردید را آموزش دهید
  - برخی از لایه‌های انتهایی را از حالت منجمد خارج کنید
  - این لایه‌ها و لایه‌های اضافه شده را به طور مشترک آموزش دهید

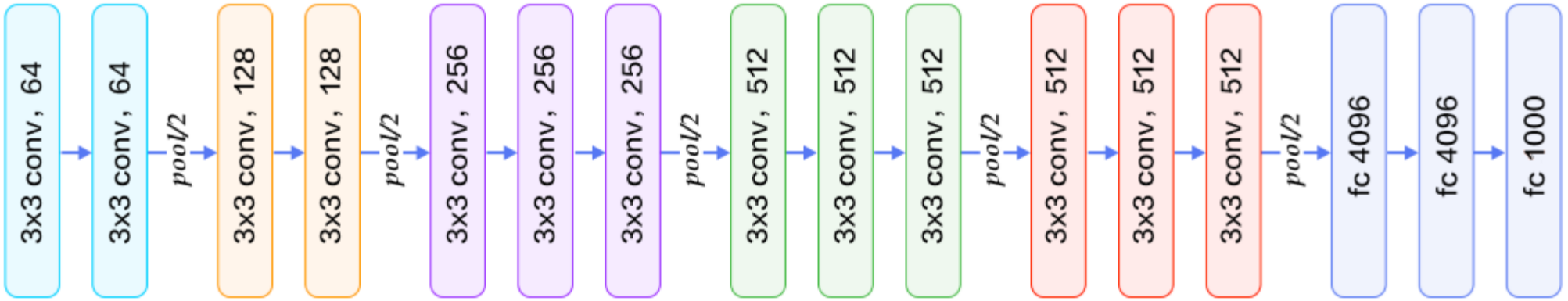
https://colab.research.google.com/drive/1t6DF8L1cZfhtgE6qBHHgELmNf5TjbZPI?usp=sharing

مصوّرسازی

Visualization

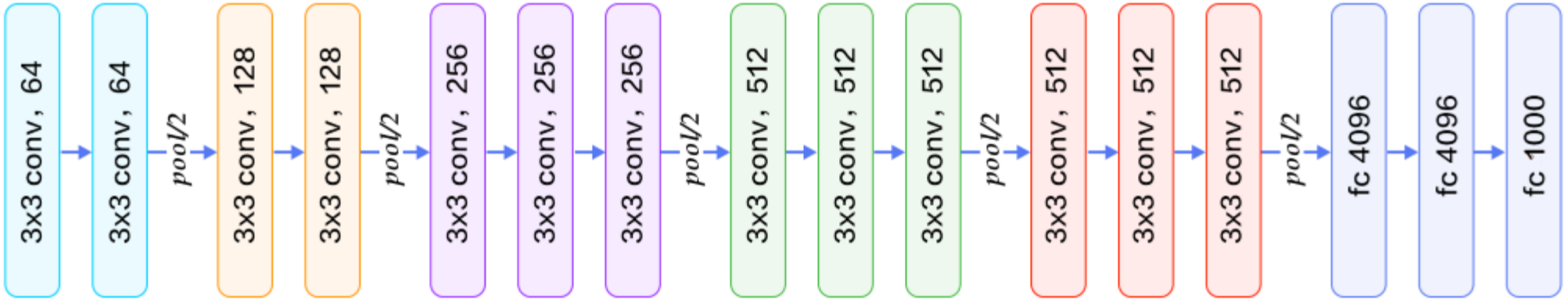
شبکه چه آموخته است؟

- اغلب گفته می‌شود که مدل‌های یادگیری عمیق «جعبه سیاه» هستند
  - قطعا برای شبکه‌های کانولوشنی درست نیست
- بازنمایی‌هایی که توسط شبکه‌های کانولوشنی آموخته می‌شوند قابلیت مصوّرسازی بالایی دارند
  - تا حد زیادی به این دلیل که بازنمایی مفاهیم بصری هستند



شبکه چه آموخته است؟

- روش‌های مختلفی برای مصورسازی یا تفسیر بازنمایی‌های آموخته شده توسط شبکه وجود دارد که سه روش رایج عبارتند از:
  - نمایش خروجی‌های لایه‌های میانی (فعال شدن لایه‌های میانی)
    - برای درک اینکه لایه‌های متوالی ورودی خود را چگونه تبدیل می‌کنند
  - نمایش فیلترهای آموخته شده
    - برای درک دقیق الگوی بصری یا مفهومی که هر فیلتر آموخته است

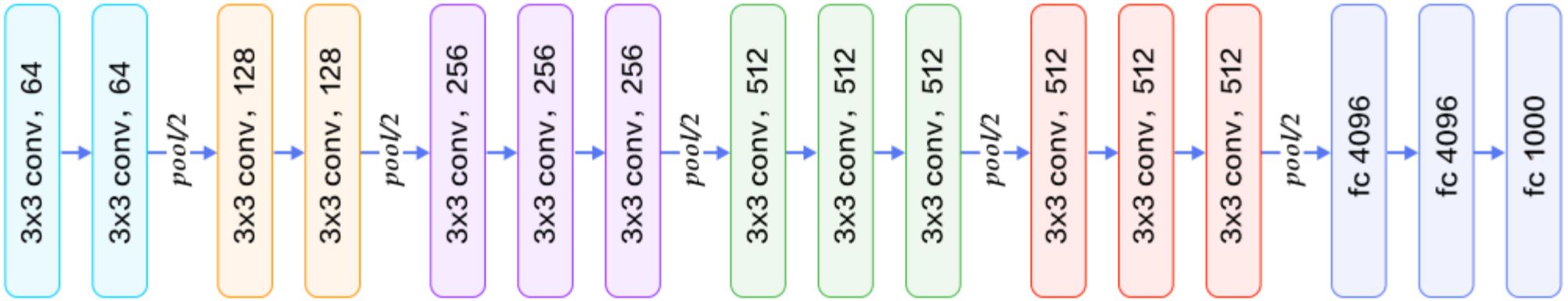


شبکه چه آموخته است؟

- روش‌های مختلفی برای مصورسازی یا تفسیر بازنمایی‌های آموخته شده توسط شبکه وجود دارد که سه روش رایج عبارتند از:
  - نمایش خروجی‌های لایه‌های میانی (فعال شدن لایه‌های میانی)
    - برای درک اینکه لایه‌های متوالی ورودی خود را چگونه تبدیل می‌کنند
  - نمایش فیلترهای آموخته شده
    - برای درک دقیق الگوی بصری یا مفهومی که هر فیلتر آموخته است
  - نمایش نقشه‌های حرارتی (heatmaps) مربوط به فعالیت هر کلاس در یک تصویر
    - برای درک اینکه کدام بخش از یک تصویر باعث شده است شبکه کلاس مربوطه را تشخیص دهد
    - این امکان را می‌دهد که مکان اشیاء در تصاویر را تخمین زد

خروجی‌های لایه‌های میانی

- نمایش نقشه‌های ویژگی که توسط لایه‌های کانولوشنی و تجمیع مختلف در یک شبکه با توجه به ورودی مشخصی محاسبه می‌شوند
- نقشه‌های ویژگی دارای سه بعد عرض، ارتفاع و عمق (کانال) هستند



```
>>> from keras.models import load_model
>>> model = load_model('cats_and_dogs_small_2.h5')
>>> model.summary()
```

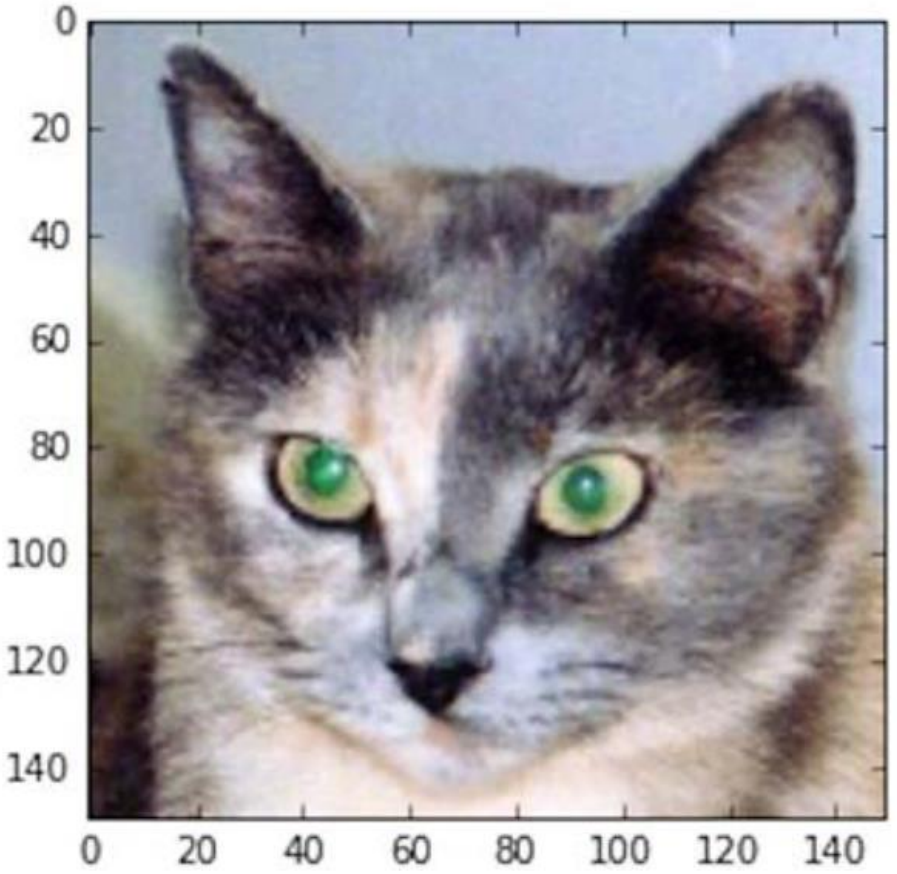
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
maxpooling2d_5 (MaxPooling2D)(None, 74, 74, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
maxpooling2d_6 (MaxPooling2D)(None, 36, 36, 64)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
maxpooling2d_7 (MaxPooling2D)(None, 17, 17, 128)       0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
maxpooling2d_8 (MaxPooling2D)(None, 7, 7, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 6272)              0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               3211776
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
```

Listing 5.25 Preprocessing a single image

```
img_path = '/Users/fchollet/Downloads/cats_and_dogs_small/test/cats/cat.1700.jpg'

from keras.preprocessing import image
import numpy as np

img = image.load_img(img_path, target_size=(150, 150))
img_tensor = image.img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis=0)
img_tensor /= 255.

<1> Its shape is (1, 150, 150, 3)
print(img_tensor.shape)


import matplotlib.pyplot as plt

plt.imshow(img_tensor[0])
plt.show()
```

Preprocesses the image into a 4D tensor

Remember that the model was trained on inputs that were preprocessed this way.



Listing 5.27 Instantiating a model from an input tensor and a list of output tensors

```
from keras import models

layer_outputs = [layer.output for layer in model.layers[:8]]
activation_model = models.Model(inputs=model.input, outputs=layer_outputs)
```

Extracts the outputs of the top eight layers

Creates a model that will return these outputs, given the model input

Listing 5.28 Running the model in predict mode

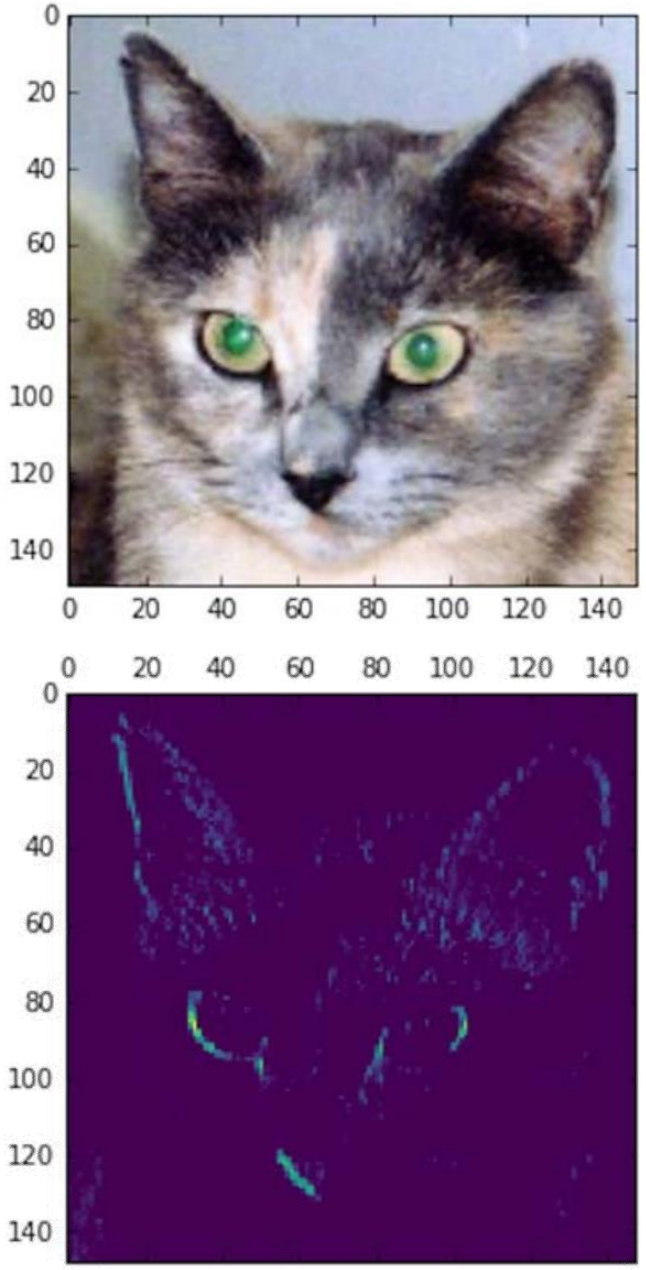
```
activations = activation_model.predict(img_tensor)
```

Reeight a list of five eight Numpy arrays: one array per layer activation

```
>>> first_layer_activation = activations[0]
>>> print(first_layer_activation.shape)
(1, 148, 148, 32)
```

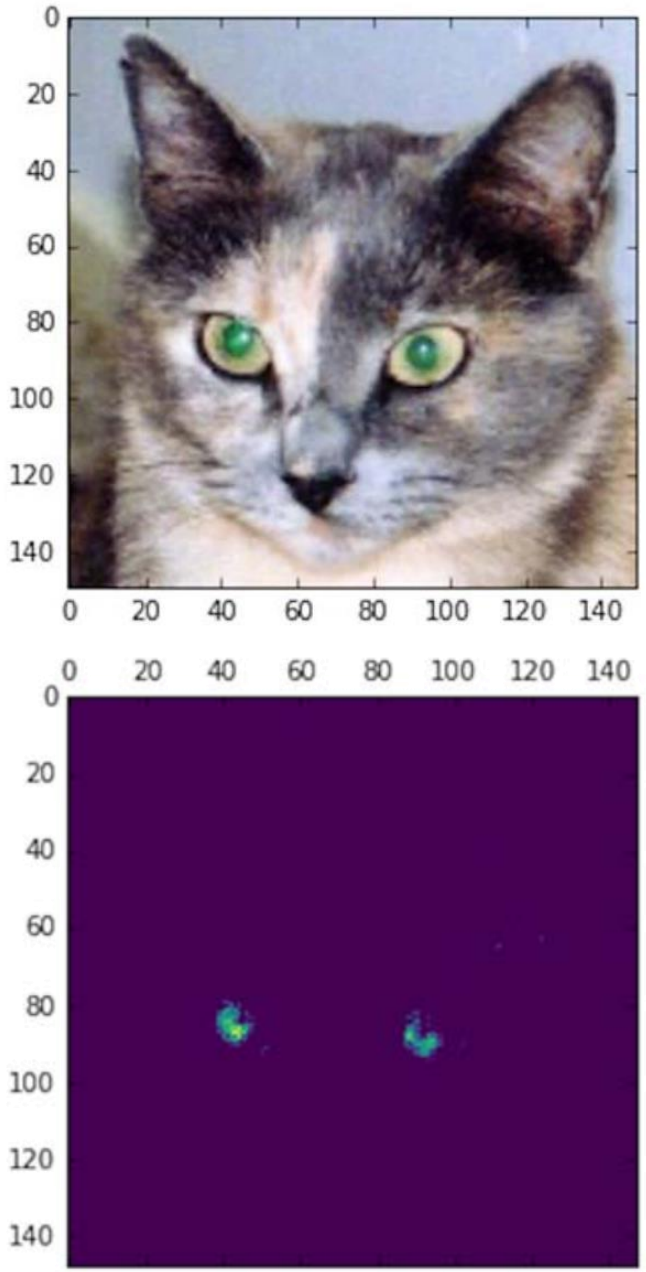
Listing 5.29 Visualizing the fourth channel

```
import matplotlib.pyplot as plt

plt.matshow(first_layer_activation[0, :, :, 4], cmap='viridis')
```



Listing 5.30 Visualizing the seventh channel

```
plt.matshow(first_layer_activation[0, :, :, 7], cmap='viridis')
```



Listing 5.31 Visualizing every channel in every intermediate activation

```
layer_names = []
for layer in model.layers[:8]:
    layer_names.append(layer.name)

images_per_row = 16

for layer_name, layer_activation in zip(layer_names, activations):
    n_features = layer_activation.shape[-1]

    size = layer_activation.shape[1]

    n_cols = n_features // images_per_row
    display_grid = np.zeros((size * n_cols, images_per_row * size))

    for col in range(n_cols):
        for row in range(images_per_row):
            channel_image = layer_activation[0,
                                             :, :,
                                             col * images_per_row + row]

            channel_image -= channel_image.mean()
            channel_image /= channel_image.std()
            channel_image *= 64
            channel_image += 128
            channel_image = np.clip(channel_image, 0, 255).astype('uint8')
            display_grid[col * size : (col + 1) * size,
                         row * size : (row + 1) * size] = channel_image

    scale = 1. / size
    plt.figure(figsize=(scale * display_grid.shape[1],
                        scale * display_grid.shape[0]))
    plt.title(layer_name)
    plt.grid(False)
    plt.imshow(display_grid, aspect='auto', cmap='viridis')
```

Names of the layers, so you can have them as part of your plot

Displays the feature maps

Number of features in the feature map

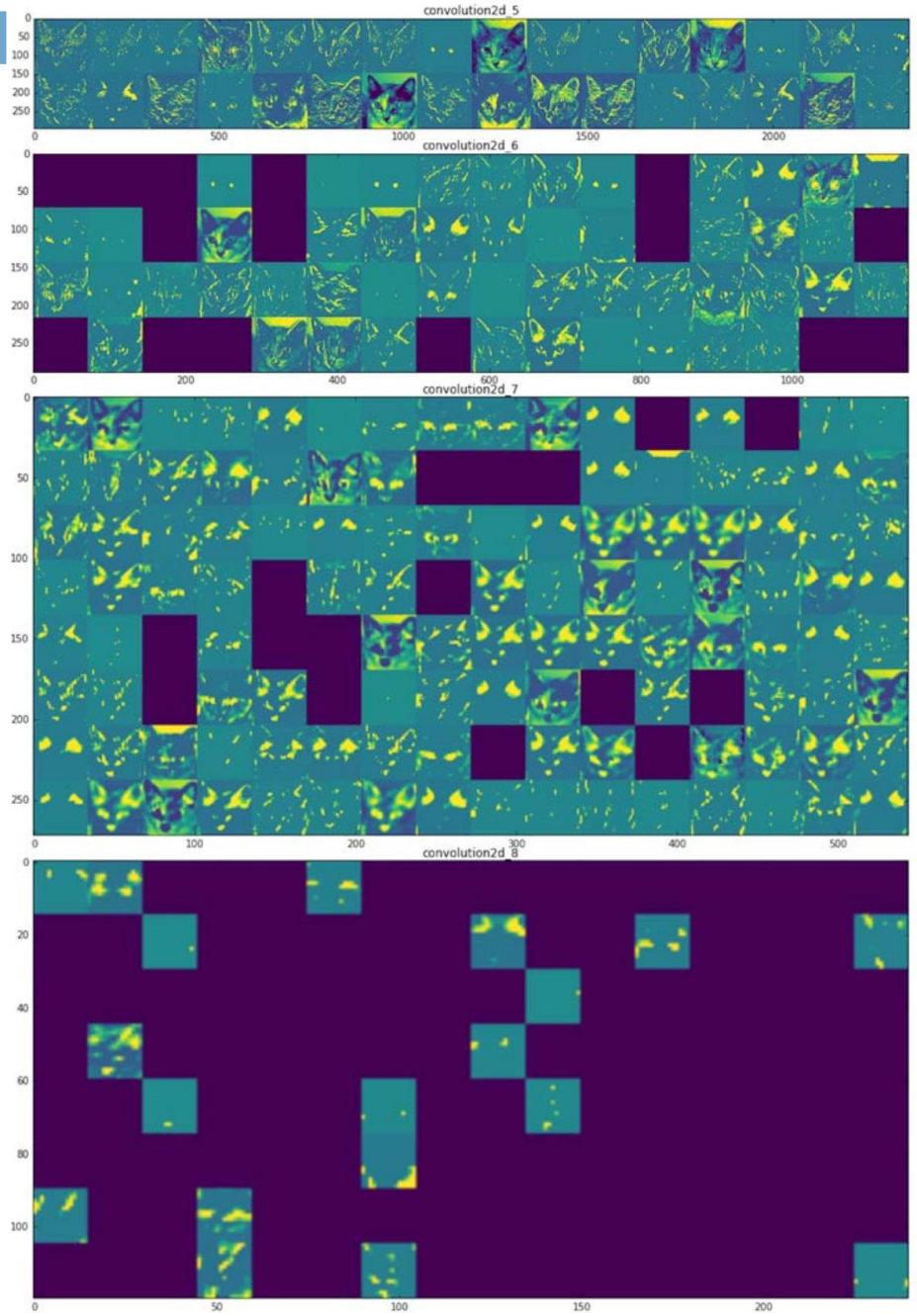
The feature map has shape (1, size, size, n_features).

Tiles the activation channels in this matrix

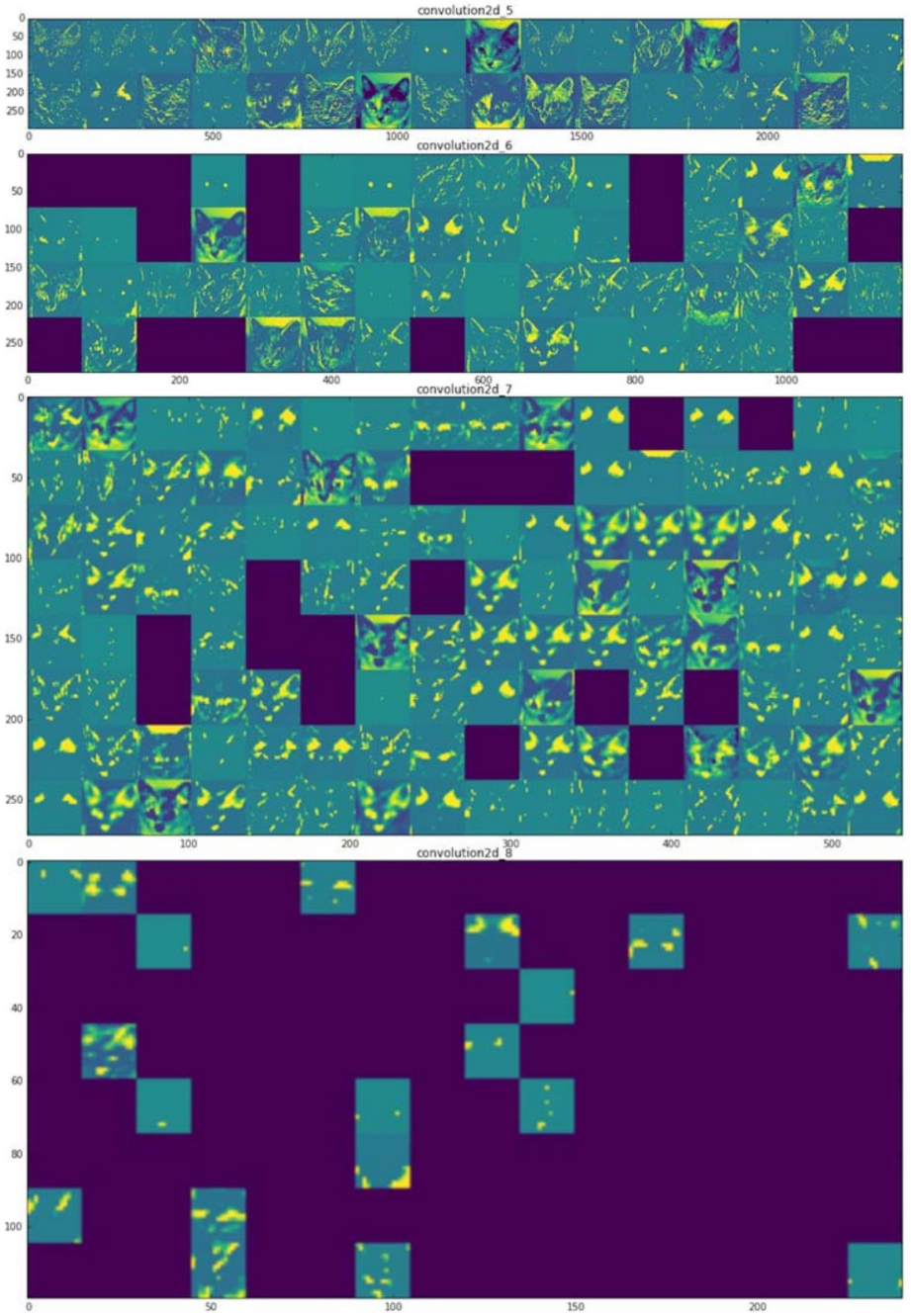
Tiles each filter into a big horizontal grid

Post-processes the feature to make it visually palatable

Displays the grid



- لایه اول به عنوان مجموعه‌ای از آشکارسازهای مختلف لبه و رنگ عمل می‌کند
  - در این مرحله، فعال‌سازی‌ها تقریباً تمام اطلاعات موجود در تصویر اولیه را حفظ می‌کنند
- هرچه جلوتر برویم، فعال‌سازی‌ها به طور فزاینده‌ای انتزاعی‌تر و کمتر قابل تفسیر بصری می‌شوند
  - شروع به تشخیص مفاهیم سطح بالاتری مانند "گوش سگ" و "چشم گربه" می‌کنند
  - بازنمایی‌های بالاتر اطلاعات کمتری را در مورد محتوای بصری تصویر و اطلاعات بیشتر مربوط به کلاس تصویر را در خود دارند
- تنک (sparse) شدن فعال‌سازی‌ها با عمق لایه افزایش می‌یابد
  - این بدان معناست که الگوی کدگذاری شده توسط فیلتر مورد نظر در تصویر ورودی یافت نمی‌شود



بازنمایی‌ها

- ویژگی‌های استخراج شده توسط یک لایه با افزایش عمق به طور فزاینده‌ای انتزاعی می‌شوند
  - خروجی لایه‌های بالاتر اطلاعات کمتری در مورد ورودی خاص مشاهده شده دارد و اطلاعات بیشتری در مورد هدف دارد
  - داده‌های خام وارد شبکه می‌شوند (در این مورد، تصاویر RGB) و به طور مکرر تبدیل می‌شوند به طوری که اطلاعات نامربوط فیلتر می‌شوند (به عنوان مثال، ظاهر بصری خاص تصویر)، و اطلاعات مفید بزرگنمایی و پالایش می‌شوند (به عنوان مثال، دسته تصویر)