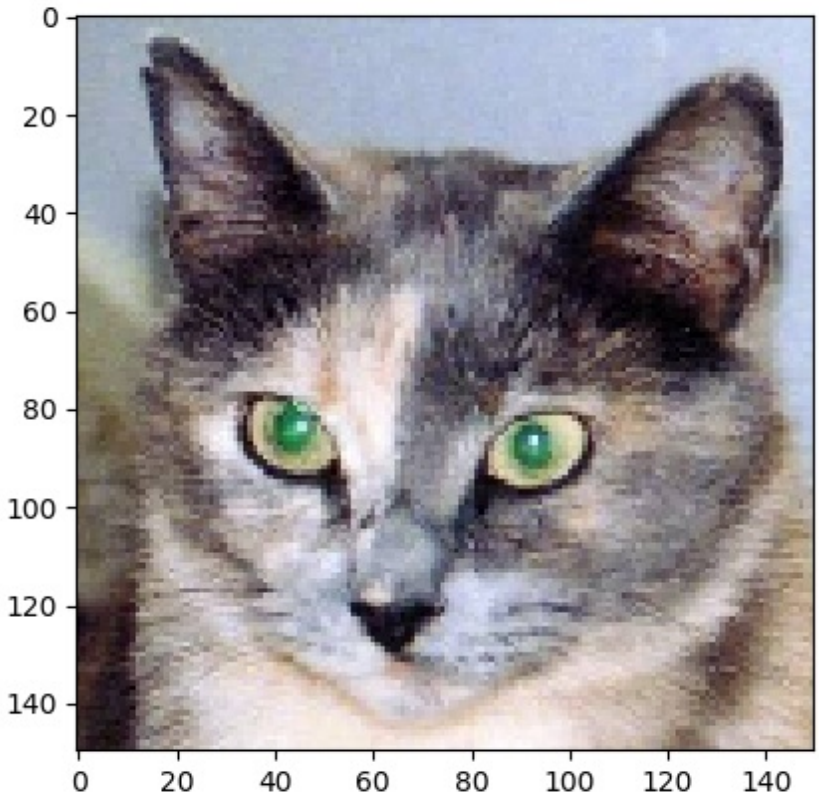
卷积神经网络的可视化 - 简书

- 可视化卷积神经网络的中间输出（中间激活）：有助于理解卷积神经网络连续的层如何对输入进行变换，也有助于初步了解卷积神经网络每个过滤器的含义。
- 可视化卷积神经网络的过滤器：有助于精确理解卷积神经网络中每个过滤器容易接受的视觉模式或视觉概念。
- 可视化图像中类激活的热力图：有助于理解图像的哪个部分被识别为属于某个类别，从而可以定位图像中的物体。

可视化中间激活

```
from keras.models import load_model
model = load_model('cats_and_dogs_small_2.h5')
print(model.summary())  # 作为提醒

# 预处理单张图像
img_path = 'D:/DeepLearning/kaggle/cats_and_dogs_small/test/cats/cat.1700.jpg'

from keras.preprocessing import image  # 将图像处理为一个4D张量
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.  # 训练模型的输入数据都用这种方法预处理
print(img_tensor.shape)  # 形状为（1，150,150,3）

# 显示测试图像
import matplotlib.pyplot as plt

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



测试的猫图像.png

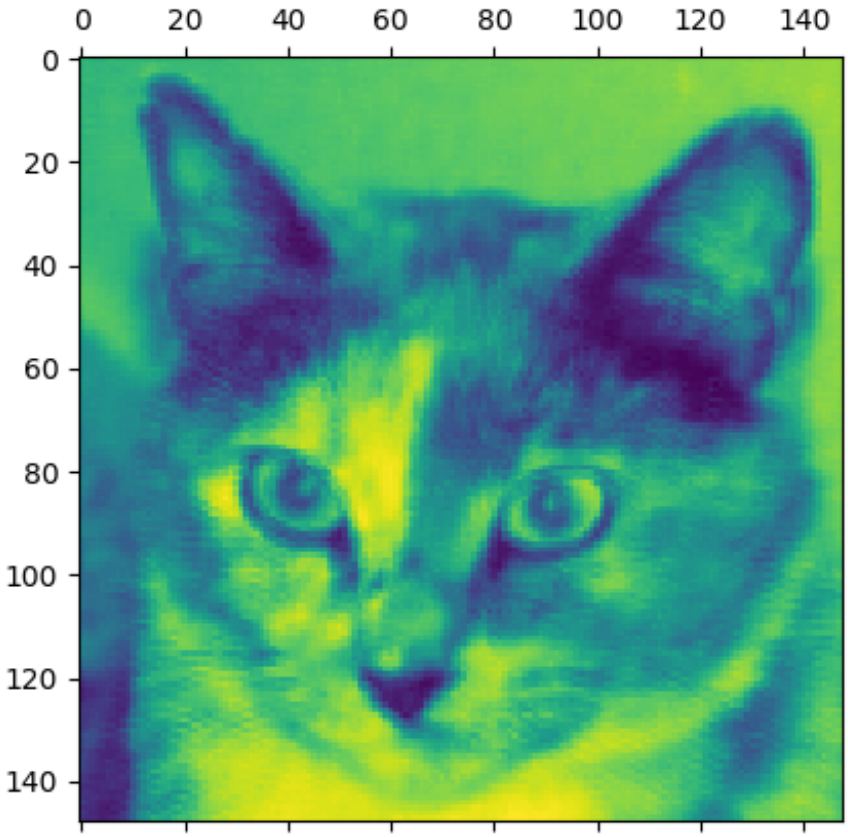
```
# 用一个输入张量和一个输出张量将模型实例化
from keras import models

layer_outputs = [layer.output for layer in model.layers[:8]] # 提取前3层的输出
# 创建一个模型，给定模型输入，可以返回这些输出
activation_model = models.Model(inputs=model.input, outputs=layer_outputs)
```

使用 keras 的 Model 类，得到的类是一个 Keras 模型，如同 Sequential 模型一样，将特定输入映射为特定输出。Model 类允许有多个输出，Sequential 模型允许一个输入和一个输出。
上述代码，输入一张图像将返回原始模型前 8 层的激活值。

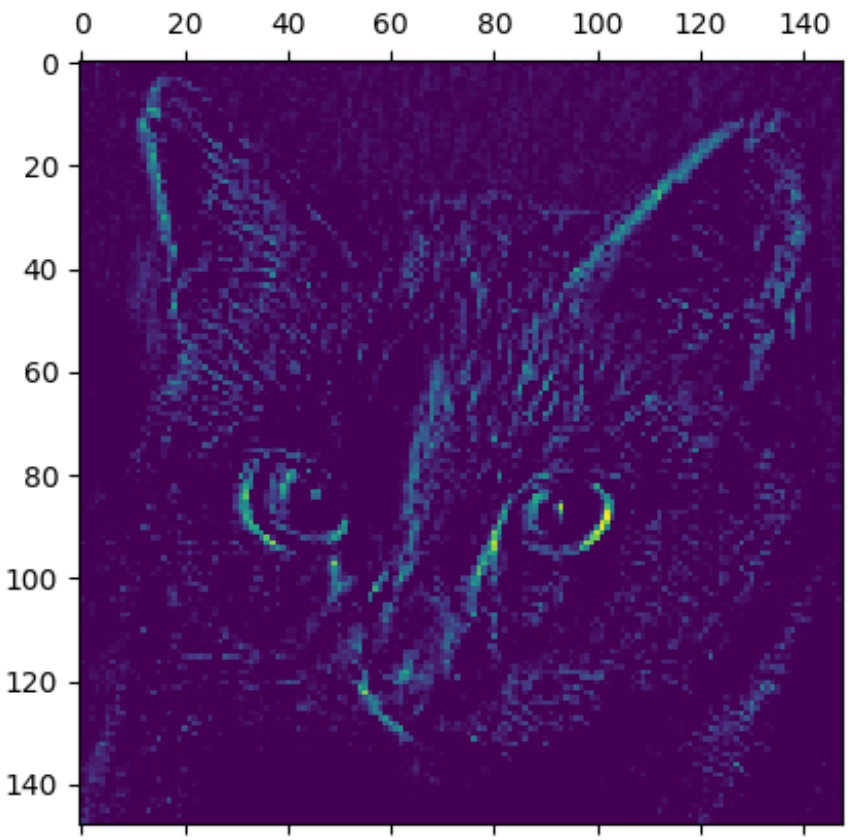
```
# 以预测模式运行模型
activations = activation_model.predict(img_tensor) # 返回8个Numpy数组组成的列表，每个层激活对应一个Numpy数组
first_layer_activation = activations[0]
print(first_layer_activation.shape)
```

```
(1, 148, 148, 32)
```

```
# 将第4个、第7通道可视化
import matplotlib.pyplot as plt

plt.matshow(first_layer_activation[0, :, :, 4], cmap='viridis')
plt.matshow(first_layer_activation[0, :, :, 7], cmap='viridis')
plt.show()
```



对于测试的猫图像，第一层激活的第4个通道.png



对于测试的猫图像，第一层激活的第7个通道.png

`matplotlib.pyplot.matshow` 矩阵可视化
这是一个绘制矩阵的函数: `matplotlib.pyplot.matshow(A, fignum=None, **kwargs)`
A是绘制的矩阵，一个矩阵元素对应一个图像像素。
`plt.matshow(Mat, cmap=plt.cm.gray)` ，cmap代表一种颜色映射方式。

```
# 将每个中间激活的所有通道可视化
layer_names = []
for layer in model.layers[:8]:
    layer_names.append(layer.name)  # 层的名称，这样你可以将这些名称画到图中

images_per_row = 16

for layer_name, layer_activation in zip(layer_names, activations):  # 显示特征图
    n_features = layer_activation.shape[-1]  # 特征图中的特征个数

    size = layer_activation.shape[1]  # 特征图的形状为(1,size,size,n_features)

    n_cols = n_features // images_per_row  # 在这个矩阵中将激活通道平铺，向下取整
    display_grid = np.zeros((size * n_cols, images_per_row * size))

    for col in range(n_cols):  # 将每个过滤器平铺到一个大的水平网络中
        for row in range(images_per_row):
            channel_image = layer_activation[0, :, :, col * images_per_row + row]
            channel_image -= channel_image.mean()    # 对特征值进行后续处理，使其看起来更美观,对所有元素求均值
            channel_image /= channel_image.std()  # 计算全局标准差
            channel_image *= 64
            channel_image += 128
            channel_image = np.clip(channel_image, 0, 255).astype('uint8')  # 数据裁剪到0到255内，8位图像节省内存空间
            display_grid[col * size : (col + 1) * size,
                         row * size : (row + 1) * size] = channel_image  # 显示网格
            scale = 1. / size
            plt.figure(figsize=(scale * display_grid.shape[1],
                                scale * display_grid.shape[0]))
            plt.title(layer_name)
            plt.grid(False)  # 不显示网格线
            plt.imshow(display_grid, aspect='auto', cmap='viridis')
```

`np.clip()` 的用法

`numpy.clip(a, a_min, a_max, out=None)`

参数说明

- a : 输入的数组
- a_min: 限定的最小值 也可以是数组 如果为数组时 shape必须和a一样
- a_max:限定的最大值 也可以是数组 shape和a一样
- out：剪裁后的数组存入的数组

`plt.imshow()`

`imshow(X, cmap=None, norm=None, aspect=None, interpolation=None, alpha=None, vmin=None, vmax=None, origin=None, extent=None, shape=None, filternorm=1, filterrad=4.0, imlim=None, resample=None, url=None, hold=None, data=None, **kwargs)`

其中，X变量存储图像，可以是浮点型数组、unit8数组以及PIL图像，如果其为数组，则需满足一下形状：

(1) M X N 此时数组必须为浮点型，其中值为该坐标的灰度；

(2) M X N X 3 RGB（浮点型或者unit8类型）

(3) M X N X 4 RGBA（浮点型或者unit8类型）
参数 `aspect`
`aspect：{'equal'，'auto'}` 控制轴的纵横比。该方面与图像特别相关，因为它可能使图像失真，即像素不是方形的。

- `equal` ：确保宽高比为1.像素将为正方形（除非像素大小在使用* extent *的数据坐标中明确地为非正方形）。
- `auto` ：轴保持固定，方向调整，使数据适合轴。通常，这将导致非方形像素。

深度神经网络学到表示的一个重要普遍特征：随着层数的加深，层所提取的特征越来越抽象。更高层激活包含关于特定输入的信息越来越少，而关于目标的信息越来越多（猫或狗）。深度神经网络可以有效地作为信息蒸馏管道，输入原始数据（RGB图像），反复对其进行变换，将无关信息过滤掉（图像的具体外观），并放大和细化有用信息（图像的类别）。