BST 261: Data Science II
Lecture 7
Convolutional Neural Networks (CNNs): Visualizing what CNNs learn

Heather Mattie
Harvard T.H. Chan School of Public Health
Spring 2 2019

Visualizing What CNNs Learn

- It is possible to visualize and interpret the learned representations of your CNN

- 3 of the most useful visualizations are
  - **Visualizing intermediate activations** (intermediate activations)
    - Useful for understanding how successive layers transform their input and getting an idea of the meaning of individual filters
  - **Visualizing filters**
    - Useful for understanding what visual pattern or concept each filter in a CNN is receptive to
  - **Visualizing heatmaps** of class activations in an image
    - Useful for understanding which parts of an image were identified as belonging to a given class

Visualizing Intermediate Outputs

Visualizing Intermediate Outputs

- Display the feature maps that are output by various convolution and pooling layers
- You should look at each channel separately

```
from keras.models import load_model

model = load_model('cats_and_dogs_small_2.h5')
model.summary()  # As a reminder.
```

```
img_path = '/Users/heathermattie/Dropbox/Teaching/2019/BST261/cats_dogs_small/test/cats/cat.1700.jpg'

# We preprocess the image into a 4D tensor
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)
# Remember that the model was trained on inputs
# that were preprocessed in the following way:
img_tensor /= 255.
```

Let's look at this particular image as an example

Visualizing Intermediate Outputs

```
from keras import models

# Extracts the outputs of the top 8 layers:
layer_outputs = [layer.output for layer in model.layers[:8]]
# Creates a model that will return these outputs, given the model input:
activation_model = models.Model(inputs=model.input, outputs=layer_outputs)

# This will return a list of 5 Numpy arrays:
# one array per layer activation
activations = activation_model.predict(img_tensor)

first_layer_activation = activations[0]
print(first_layer_activation.shape)

import matplotlib.pyplot as plt

plt.matshow(first_layer_activation[0, :, :, 11], cmap = 'viridis')
plt.show()
```

This will save the outputs or "activations" for each filter in every layer

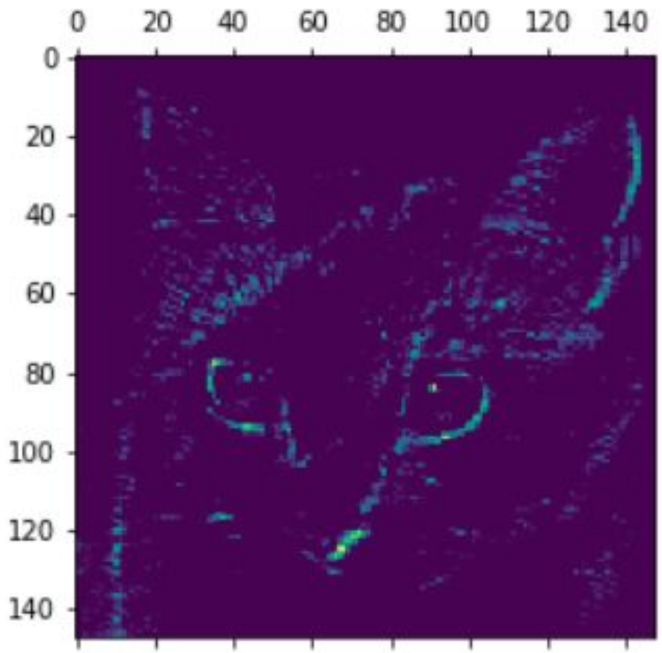
Let's look at the filters in the first layer

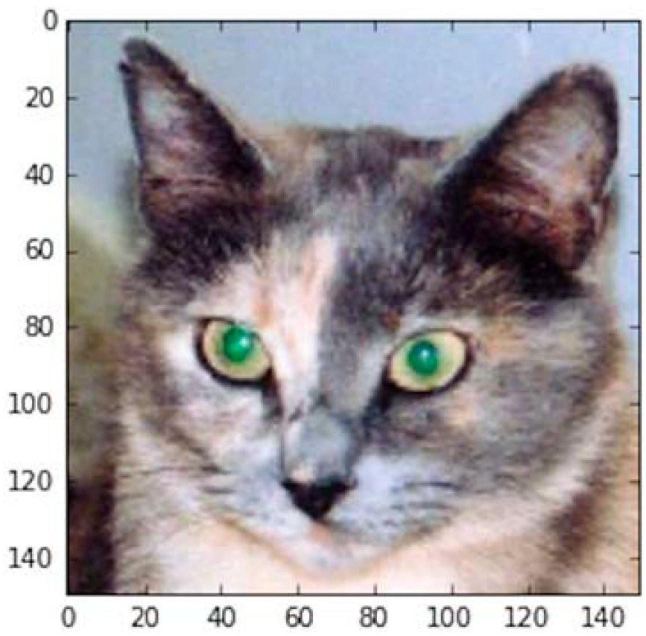
We'll visualize what patterns this filter is picking up

Visualizing Intermediate Outputs

```
import matplotlib.pyplot as plt

plt.matshow(first_layer_activation[0, :, :, 11], cmap = 'viridis')
plt.show()
```



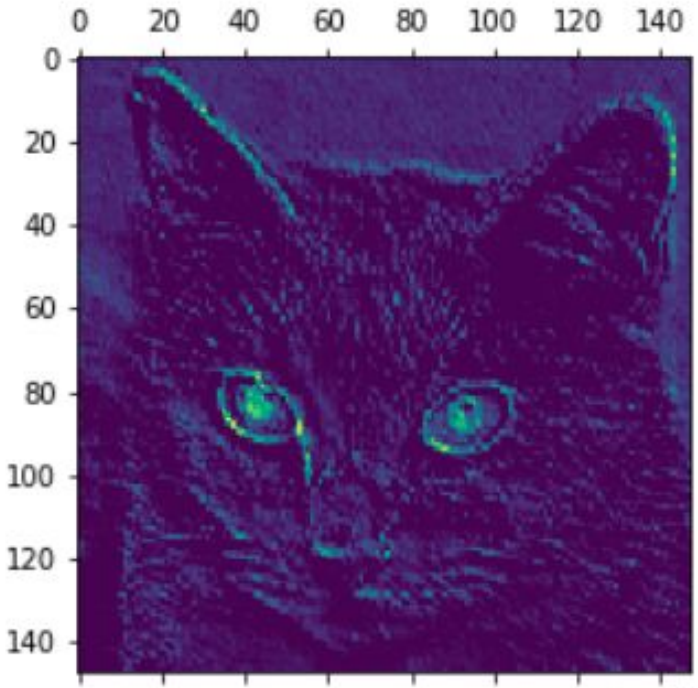


Diagonal/rounded edges filter?

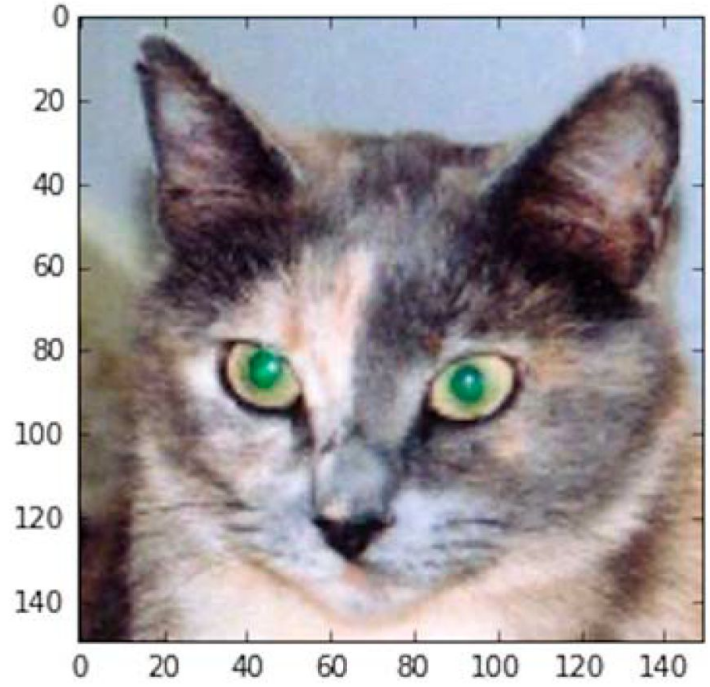
Visualizing Intermediate Outputs

```
plt.matshow(first_layer_activation[0, :, :, 26], cmap = 'viridis')
plt.show()
```

Now let's look at what pattern this filter picks up

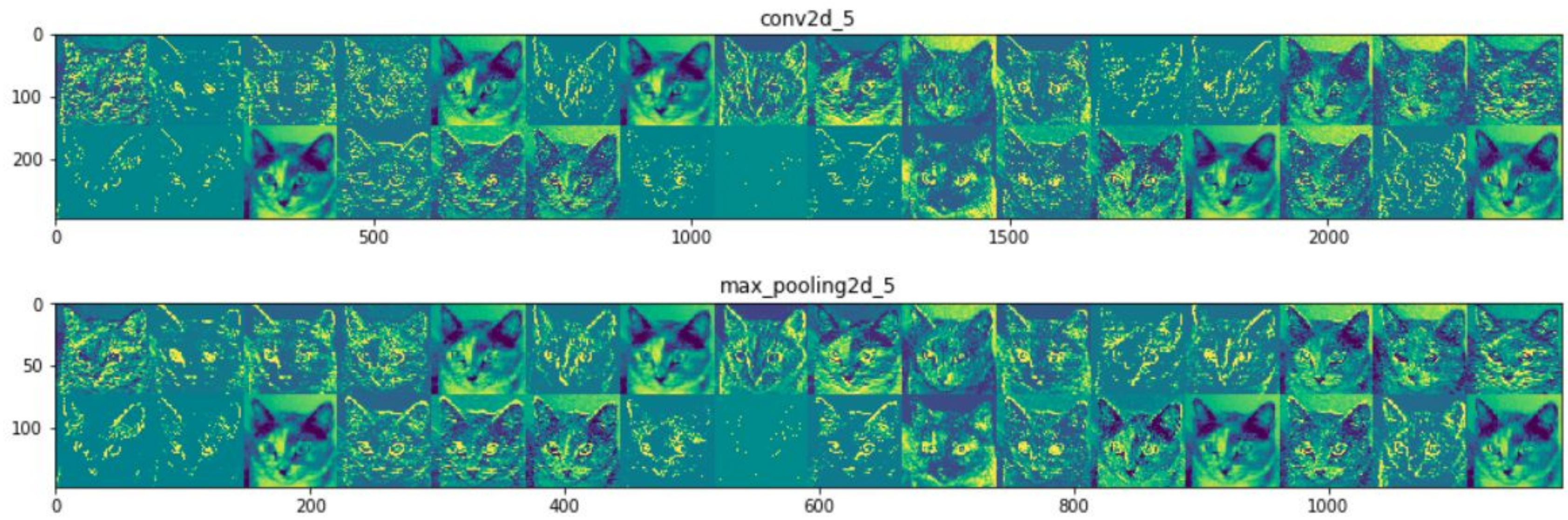


"Green dots" filter?

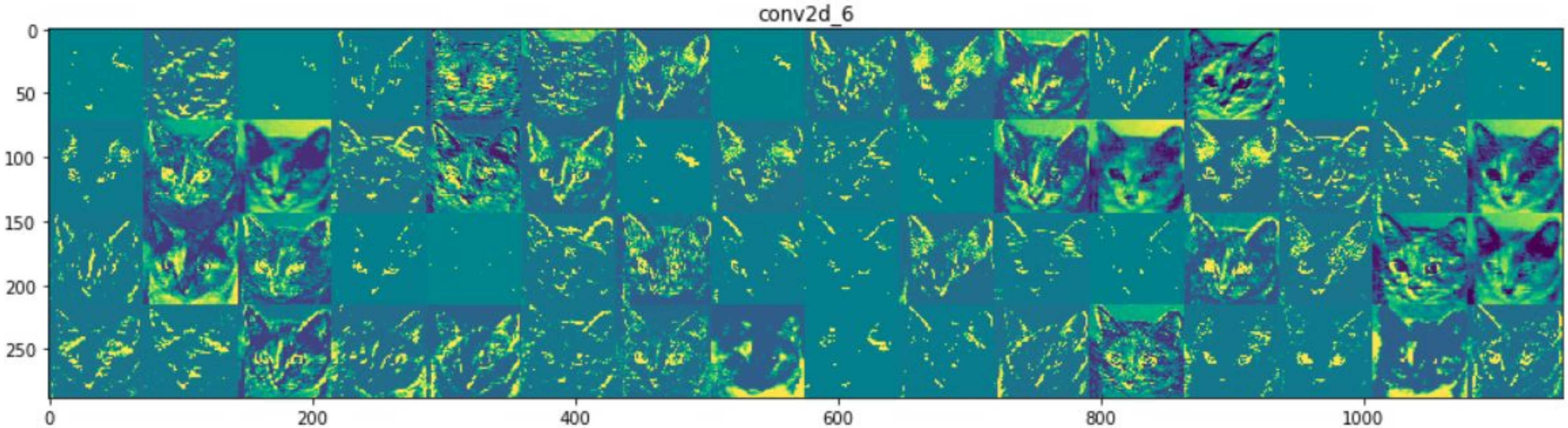


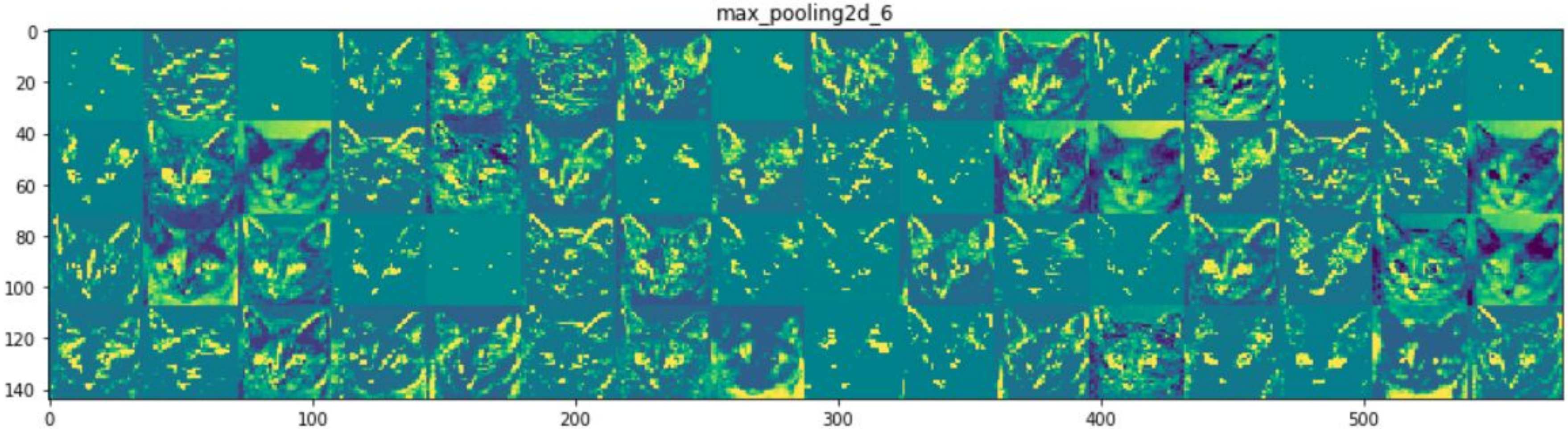
Visualizing Intermediate Outputs

- We can also look at what pattern each filter in every layer is picking up on

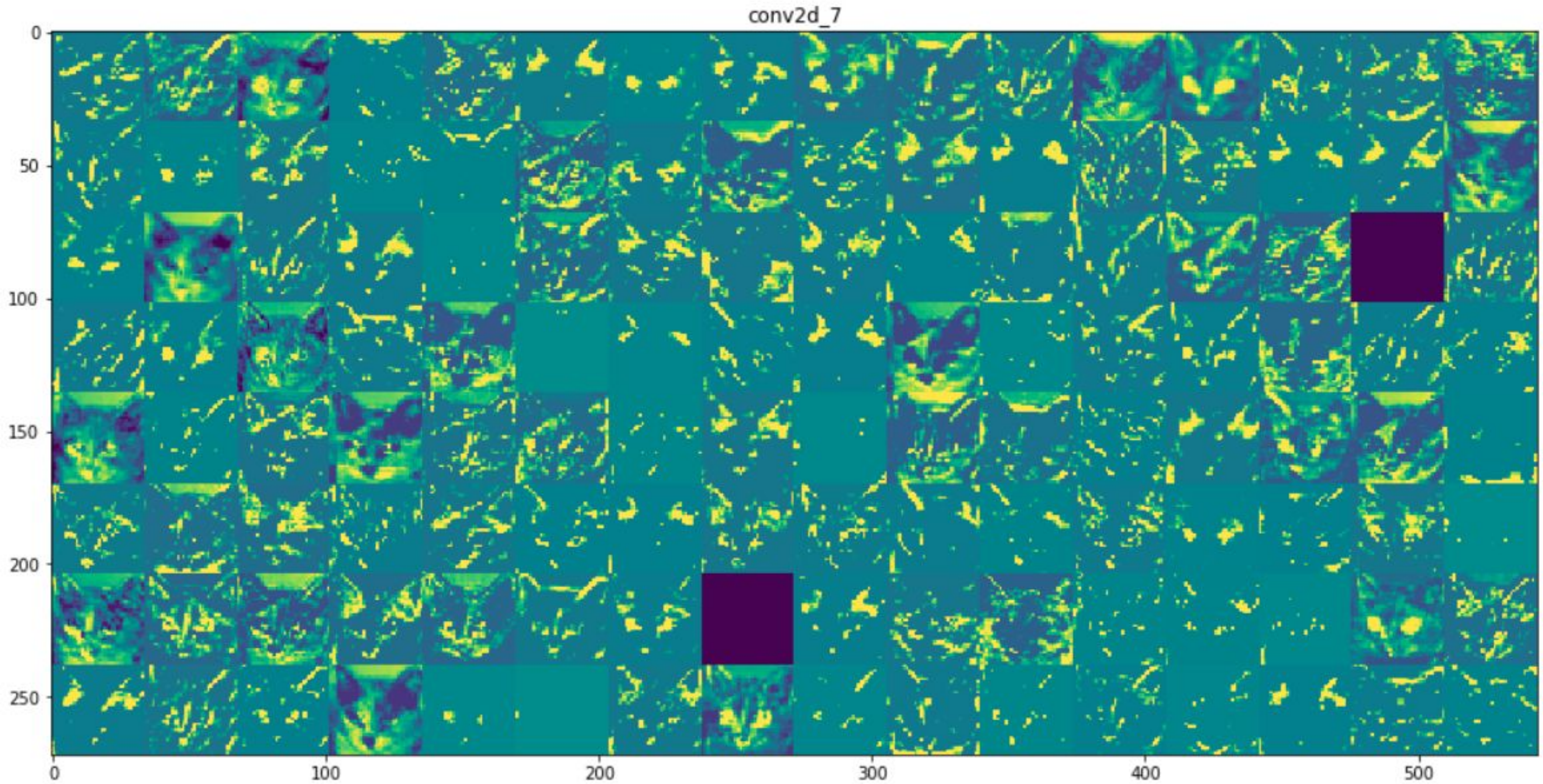


Visualizing Intermediate Outputs

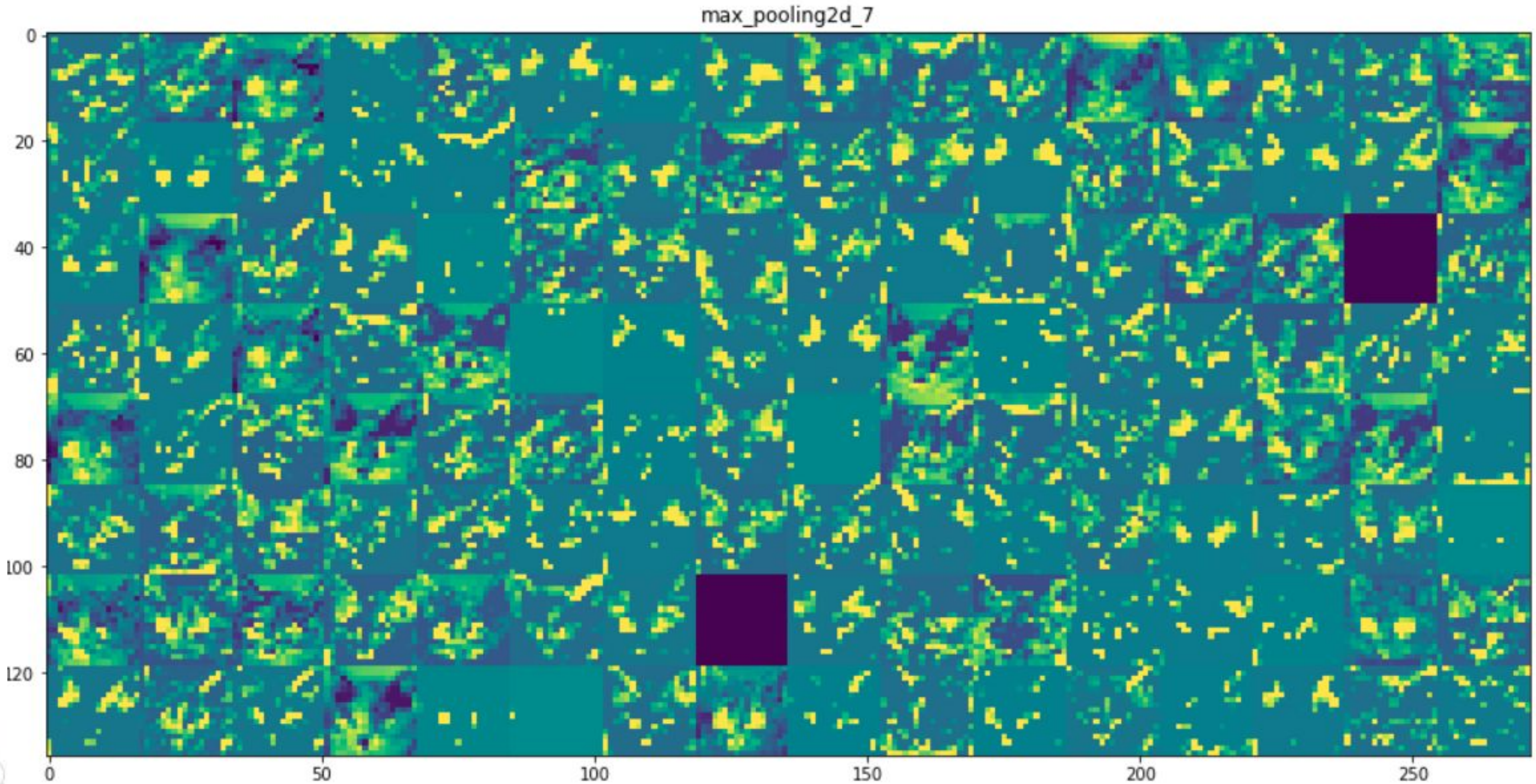




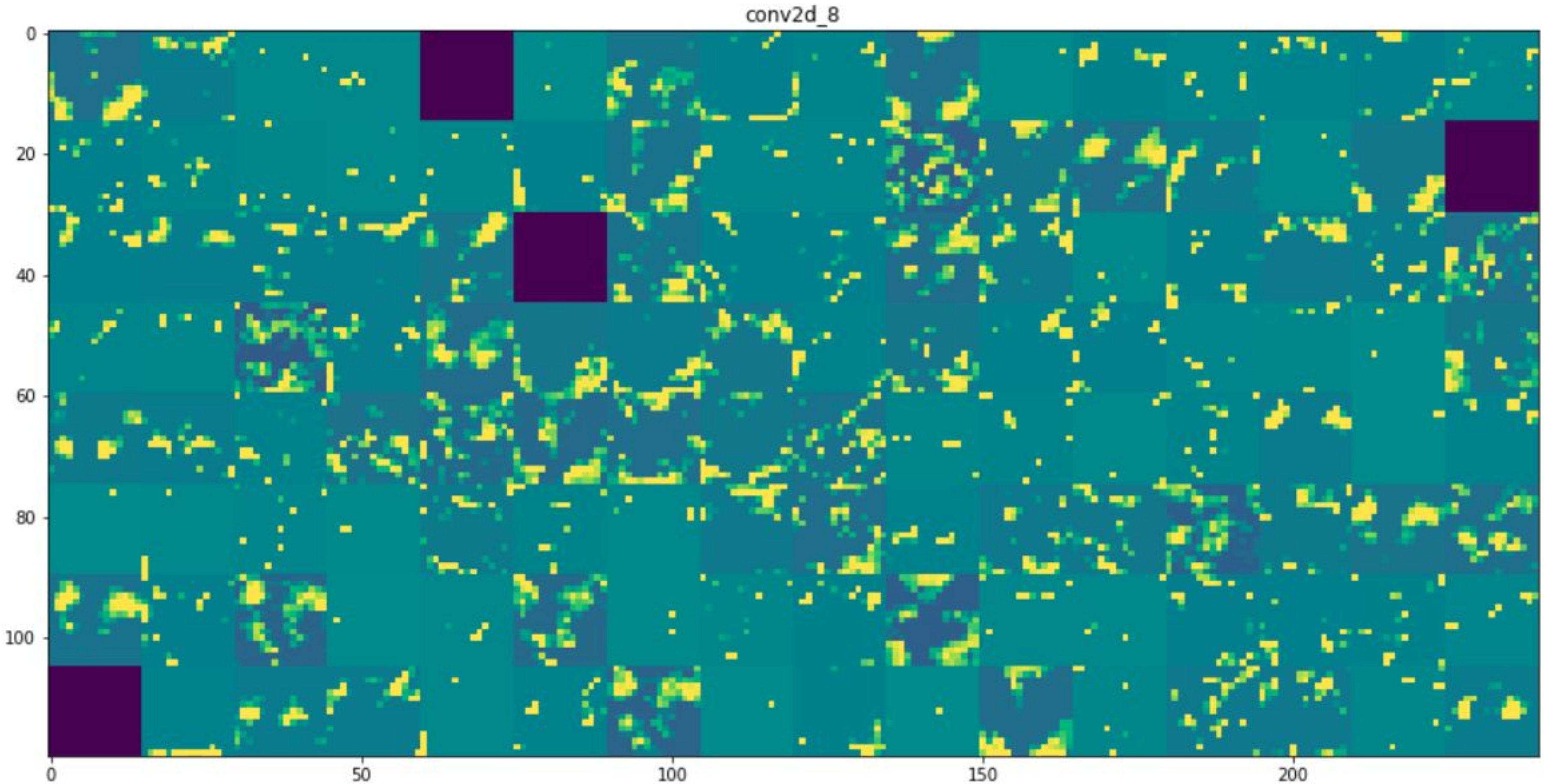
Visualizing Intermediate Outputs



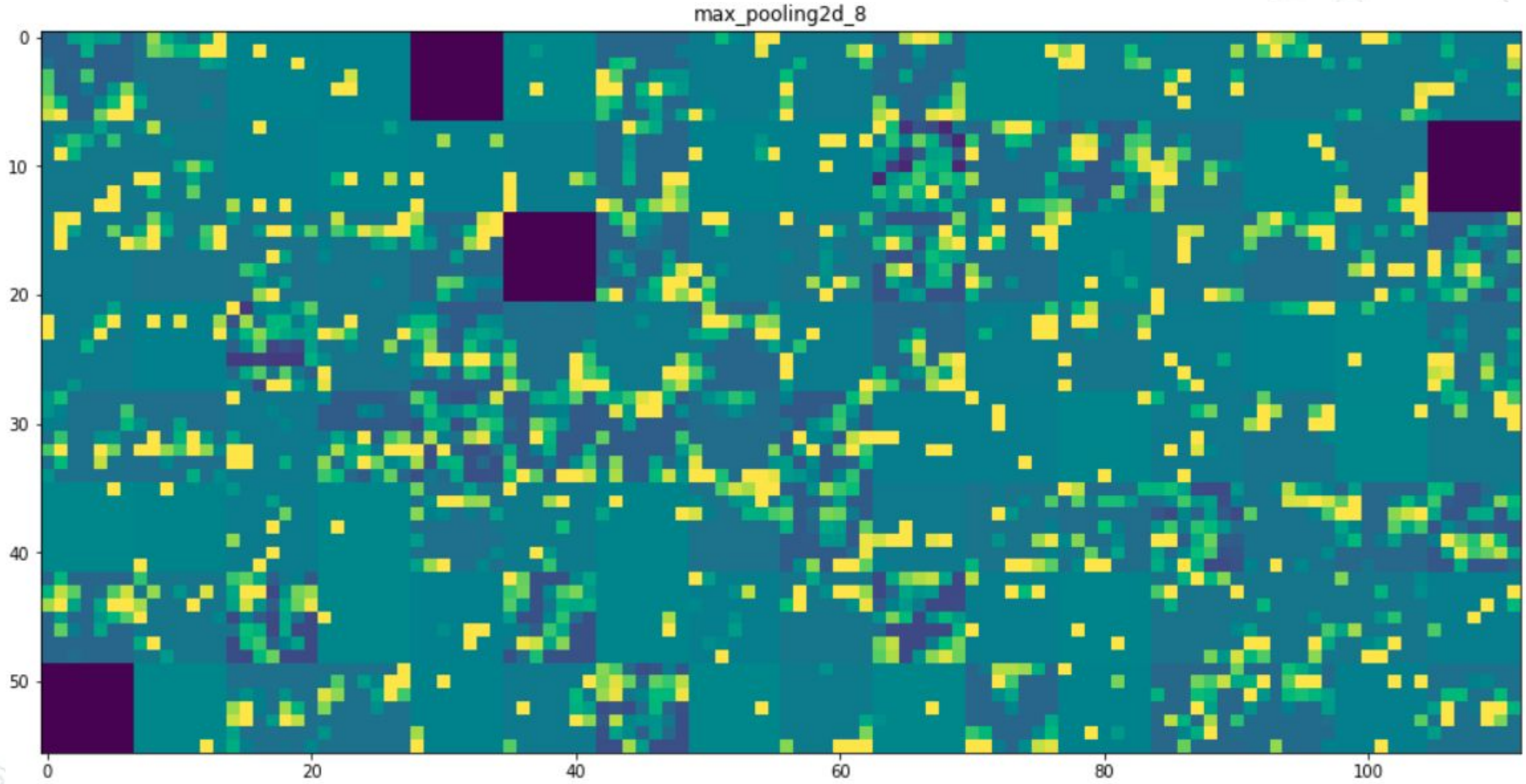
Visualizing Intermediate Outputs



Visualizing Intermediate Outputs



Visualizing Intermediate Outputs



Visualizing Intermediate Outputs

- The first layer acts as a collection of edge detectors
- The later layers contain more abstract activations that are less visually interpretable
- Deeper layers carry less information about visual contents of the image, and more information related to the class of the image
- The sparsity of the activations increases with the depth of the layer
- Blank activations mean the pattern encoded by that filter isn’t found in the input image

Visualizing Filters

Visualizing Filters

- Shows the visual pattern that each filter is meant to respond to

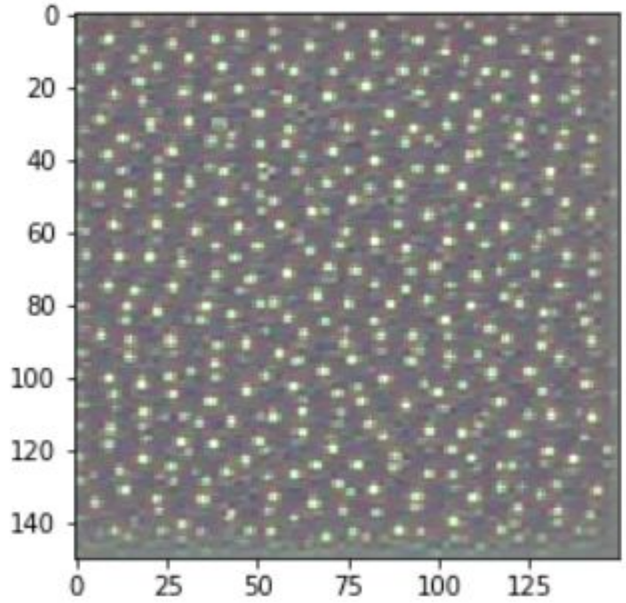
- This is done with gradient ascent in input space: applying gradient descent to the value of the input image to maximize the response of a specific filter, starting with a blank input image

- The resulting image will be one that the chosen filter is maximally responsive to

- Steps:
  - Build a loss function that maximizes the value of a given filter in a given convolution layer
  - Use stochastic gradient descent to adjust the values of the input image in order to maximize the activation value

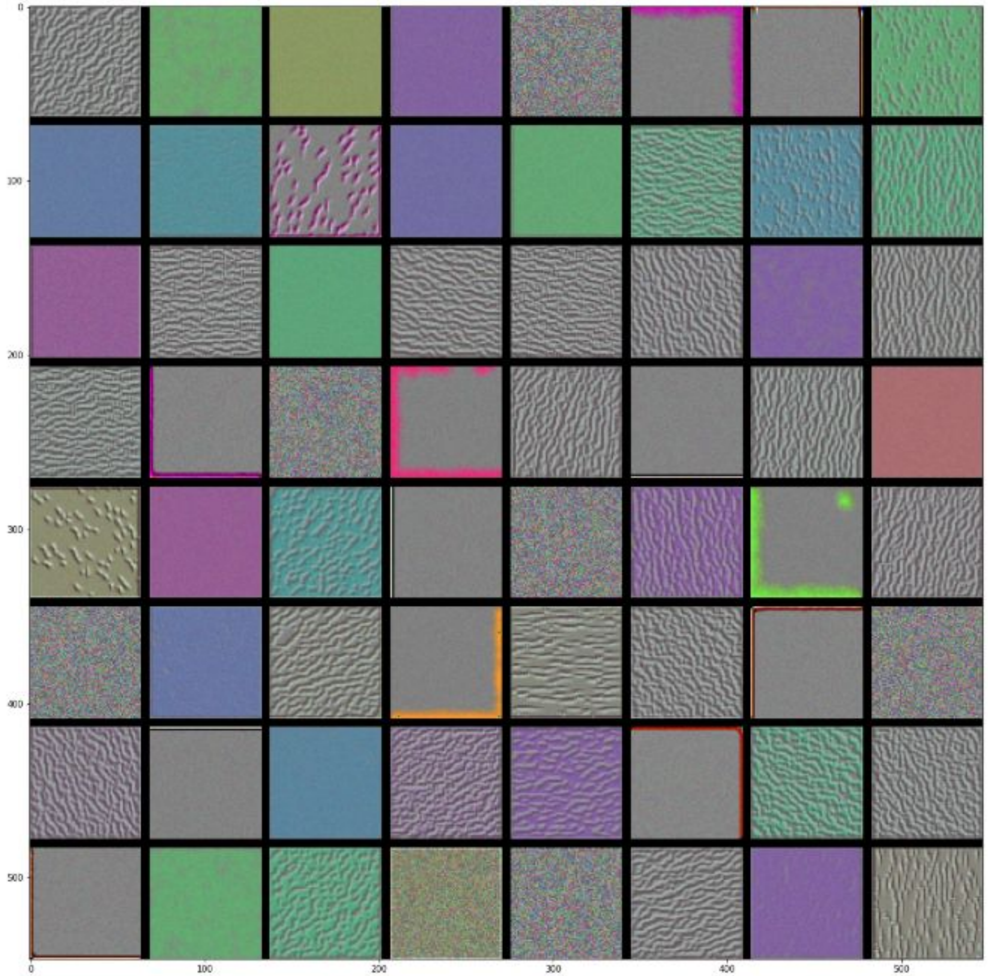
Visualizing Filters

The "polka dots" filter



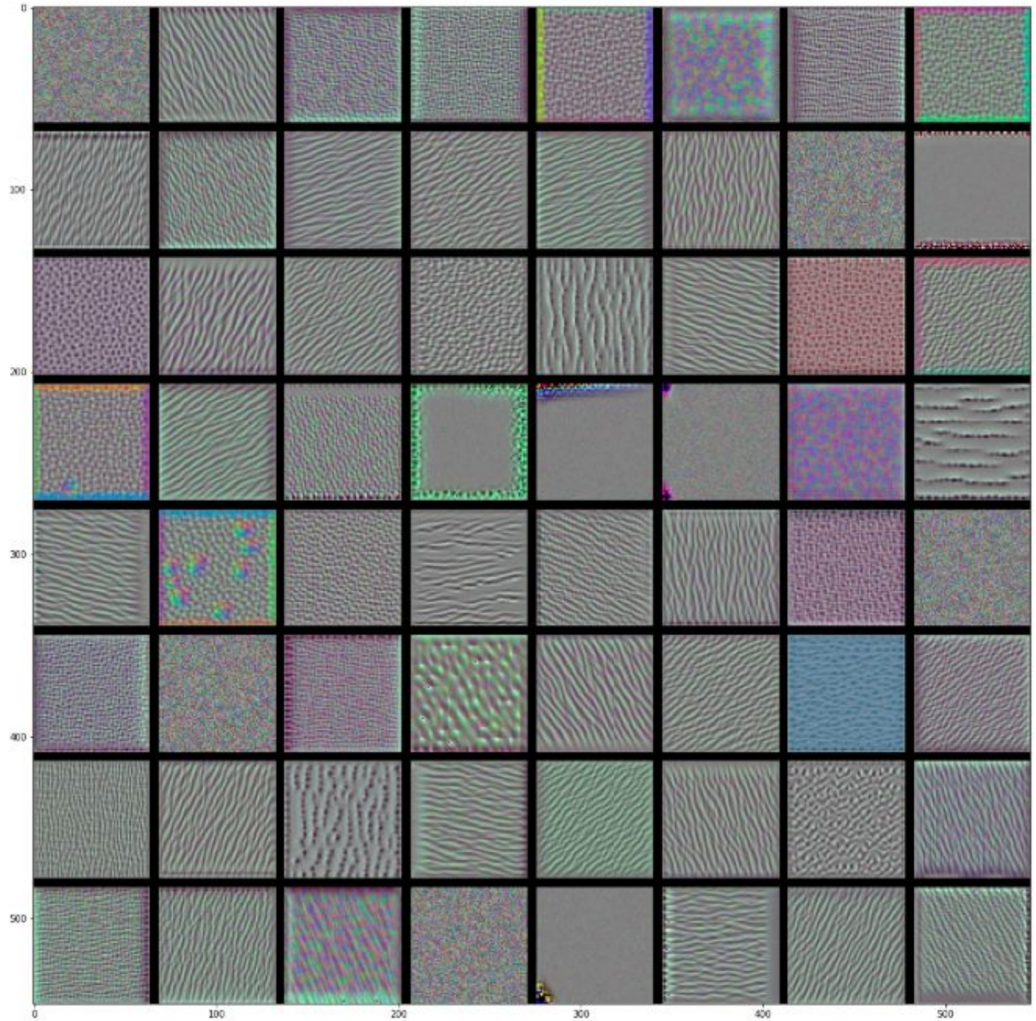
Visualizing Filters

- Filters from the 1st convolution block



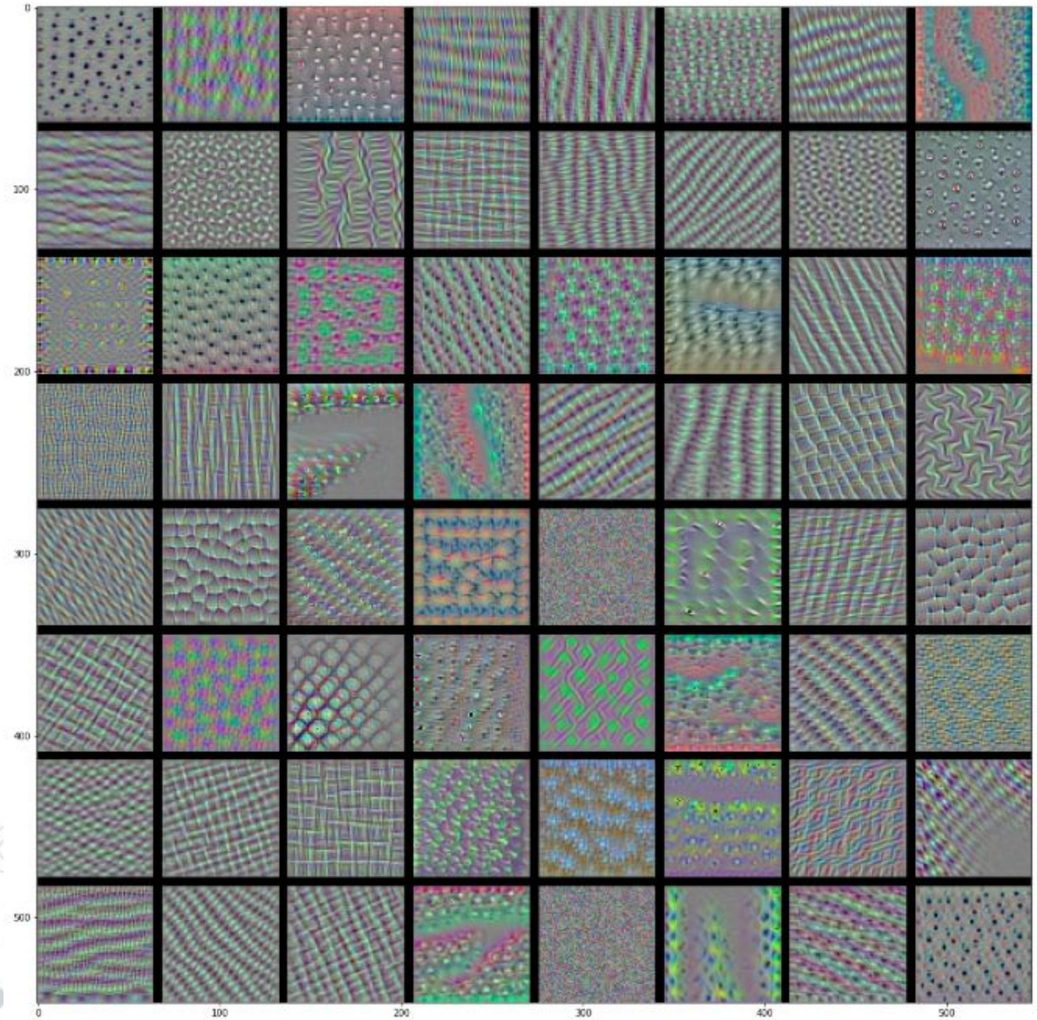
Visualizing Filters

- Filters from the second convolution block



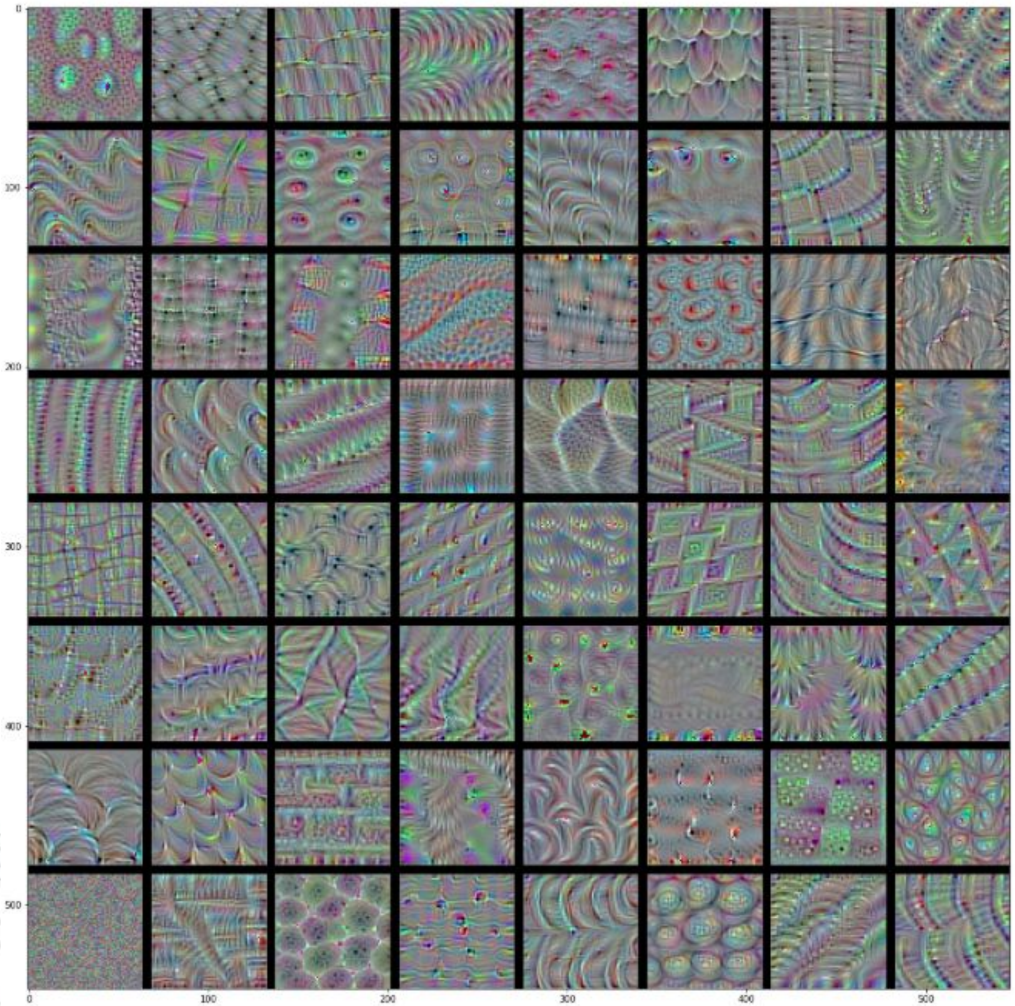
Visualizing Filters

- Filters from the third convolution block



Visualizing Filters

- Filters from the fourth convolution block



Visualizing Filters

- The filters get increasingly complex and refined as you go deeper in the model

- The filters from the first layer encode single directional edges and colors

- The next set of filters encode simple textures made from combinations of edges and colors

- The filters in later layers resemble textures found in natural images - eyes, feathers, leaves, etc.

Visualizing Heatmaps of Class Activation

Visualizing Heatmaps of Class Activation

- Great for understanding which parts of an image led the network to its final classification

- Helpful for debugging the decision process

- This also allows you to locate specific objects in an image

- Called class activation map (CAM) visualization

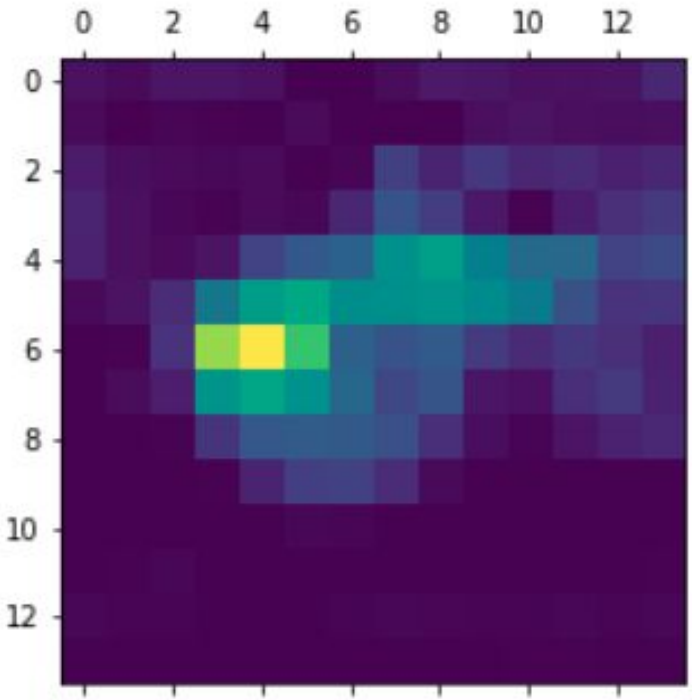
- A class activation heatmap is a 2D grid of scores associated with a specific output class, computed for every location in an input image, indicating how important each location is with respect to the class under consideration

Visualizing Heatmaps of Class Activation

- When we run this image of African elephants through the VGG16 network, the following are the top 3 predictions:
  - African elephant (with 92.5% probability)
  - Tusker (with 7% probability)
  - Indian elephant (with 0.4% probability)

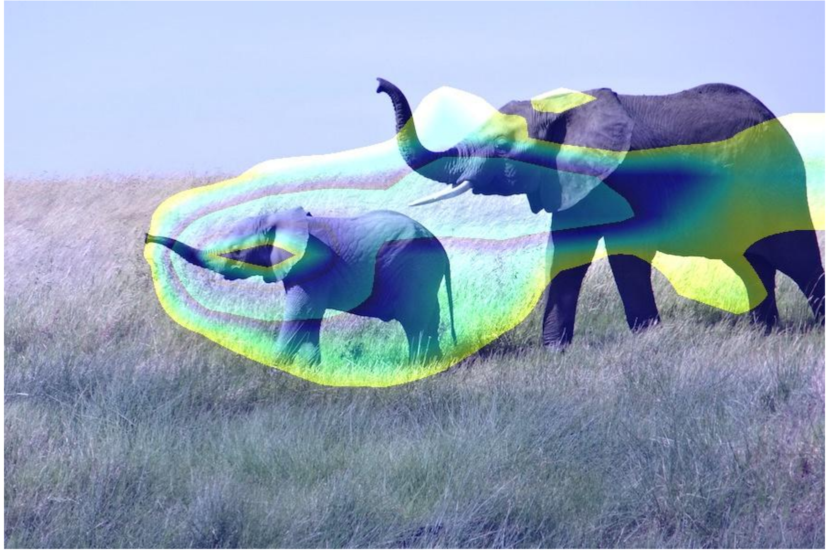
Visualizing Heatmaps of Class Activation

- Lighter colors (yellow, green) correspond to greater activation and darker colors (blue, purple) to less or no activation, allowing us to see which parts of the image were used for the classification



Visualizing Heatmaps of Class Activation

- We can then overlap these activations with the original image to see exactly what and where in the image was used in classification



Visualizing Heatmaps of Class Activation

- When we run this image of a Turkish Shepherd through the VGG16 network, the following are the top 3 predictions:
    - Saluki (with 65.9% probability)
    - Whippet (with 6.3% probability)
    - Labrador retriever (with 3.9% probability)

Visualizing Heatmaps of Class Activation

- When we run this image of a Turkish Shepherd through the VGG16 network, the following are the top 3 predictions:
  - Saluki (with 65.9% probability)
  - Whippet (with 6.3% probability)
  - Labrador retriever (with 3.9% probability)

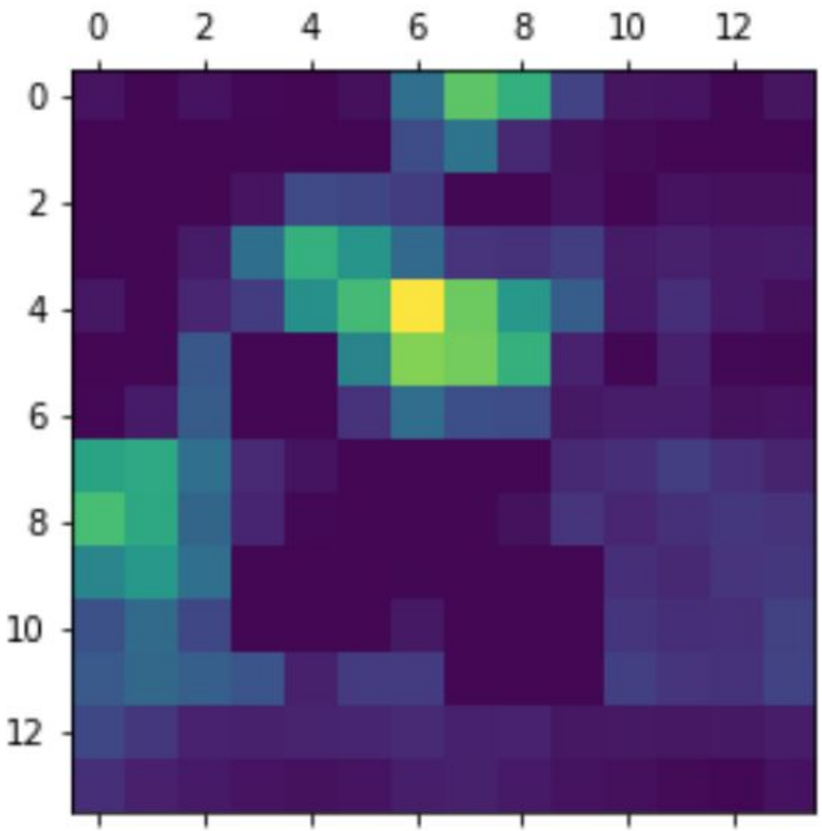
Visualizing Heatmaps of Class Activation

- When we run this image of a Turkish Shepherd through the VGG16 network, the following are the top 3 predictions:
    - Saluki (with 65.9% probability)
    - $\boxed{\text{Whippet (with 6.3\% probability)}}$
    - Labrador retriever (with 3.9% probability)

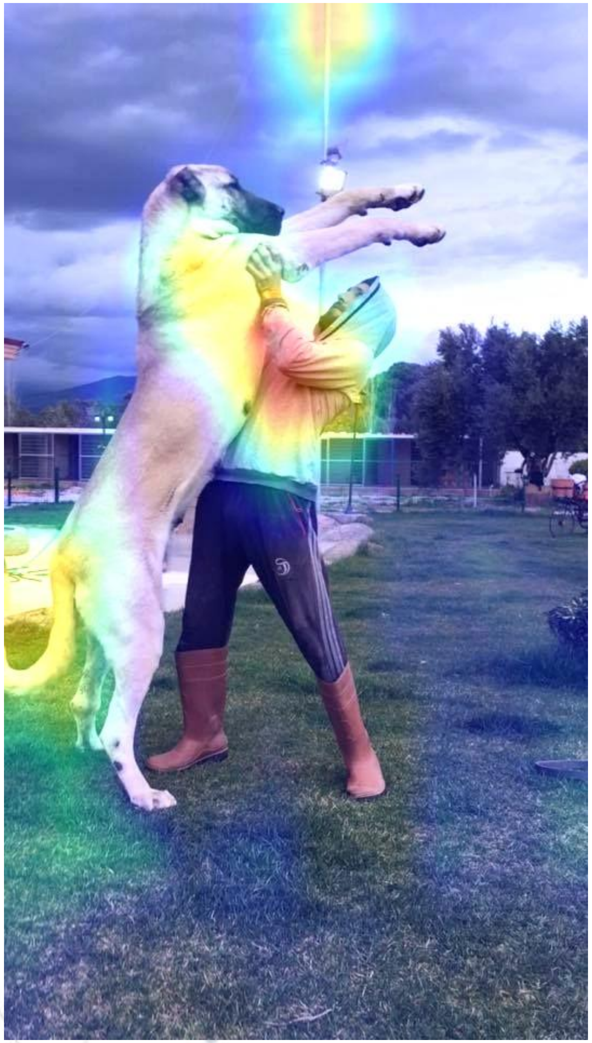
Visualizing Heatmaps of Class Activation

- When we run this image of a Turkish Shepherd through the VGG16 network, the following are the top 3 predictions:
  - Saluki (with 65.9% probability)
  - Whippet (with 6.3% probability)
  - Labrador retriever (with 3.9% probability)

Visualizing Heatmaps of Class Activation



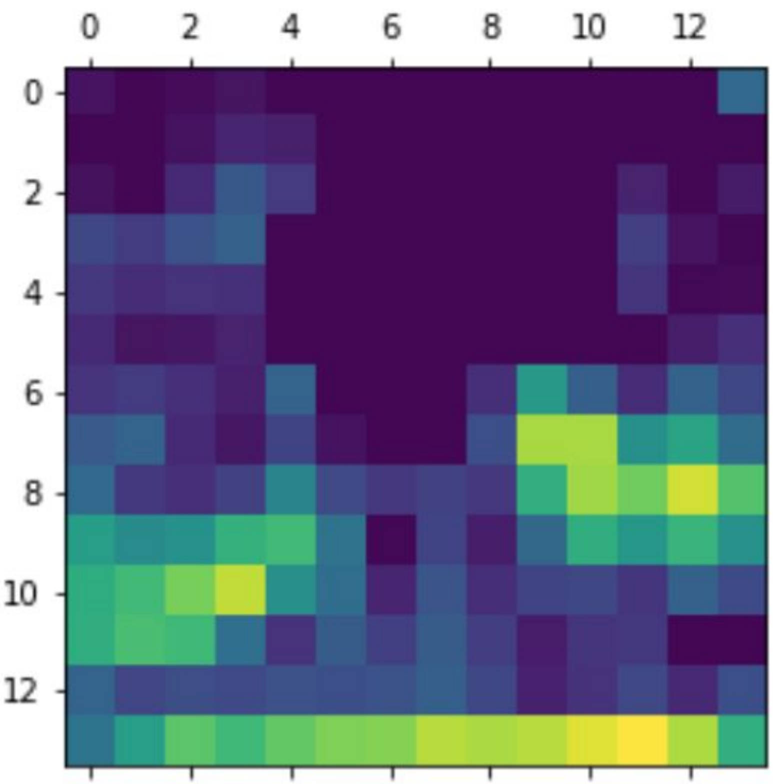
Visualizing Heatmaps of Class Activation



Visualizing Heatmaps of Class Activation

- When we run this image of a Harvard gate through the VGG16 network, the following are the top 3 predictions:
  - Prison (with 41.3% probability)
  - Fire screen (with 10.6% probability)
  - Monastery (with 7.7% probability)

Visualizing Heatmaps of Class Activation



Visualizing Heatmaps of Class Activation