Visualizing_Activations

October 21, 2018

1 Visualizing Activations

In [23]:
```
# First lets import our cats and dogs model
from keras.models import load_model

import warnings
warnings.filterwarnings('ignore')

model = load_model('cats_and_dogs_small_2.h5')
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 6272)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6272)              0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               3211776
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 513
```

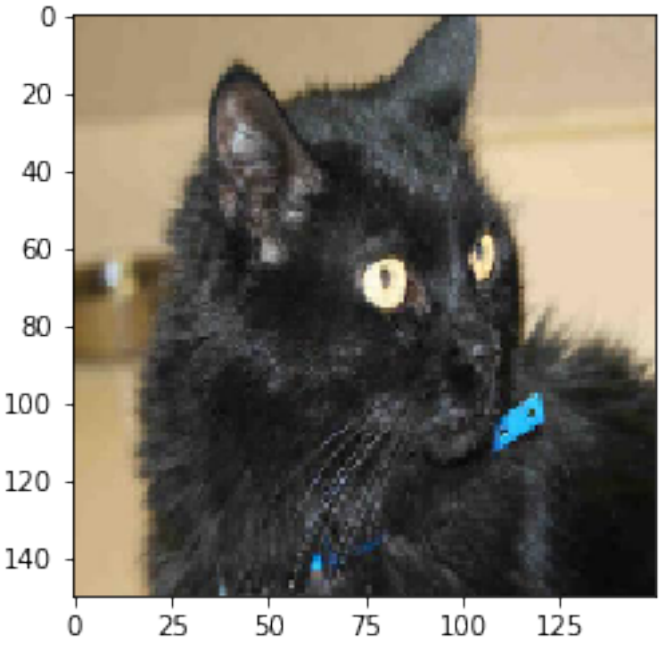
```
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

In [2]:
```
# now we need an image not used to train the network

img_path = 'PetImages/Cat/10095.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
```

In [3]:
```
print(img_tensor.shape)
```

```
(1, 150, 150, 3)
```

In [5]:
```
# lets see our image
# side note I had to run this block twice, idk why
import matplotlib.pyplot as plt

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



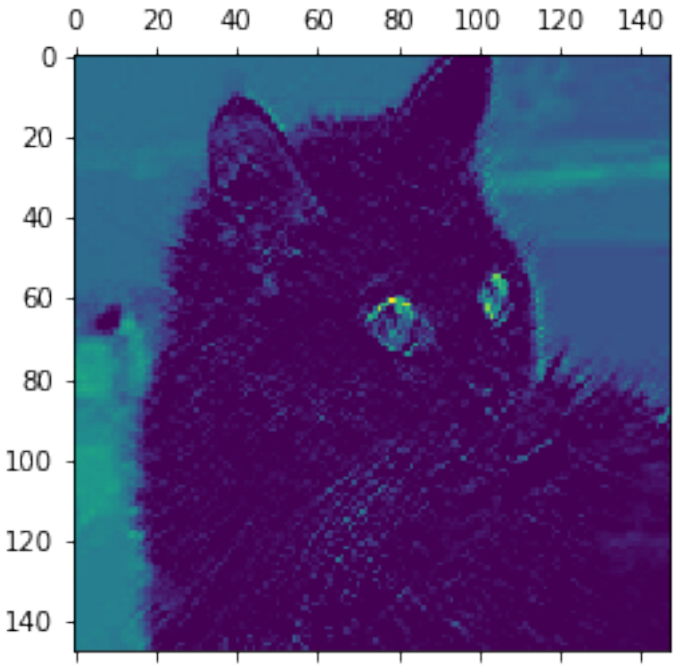
In [6]:
```
from keras import models

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

In [7]:
```
activations = activation_model.predict(img_tensor)
```

In [8]:
```
first_layer_activation = activations[0]
print(first_layer_activation.shape)
```

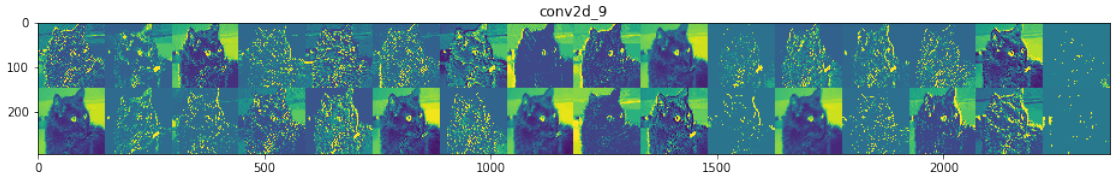
```
(1, 148, 148, 32)
```

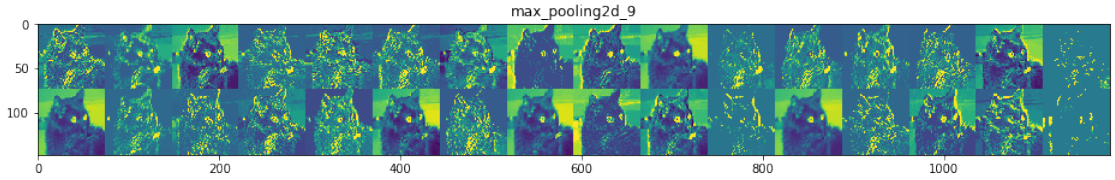
In [9]:
```
plt.matshow(first_layer_activation[0, :, :, 24], cmap='viridis')
plt.show()
```

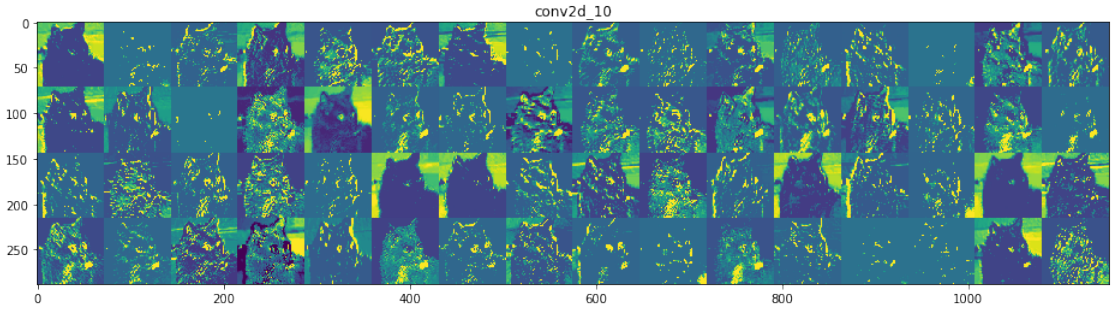


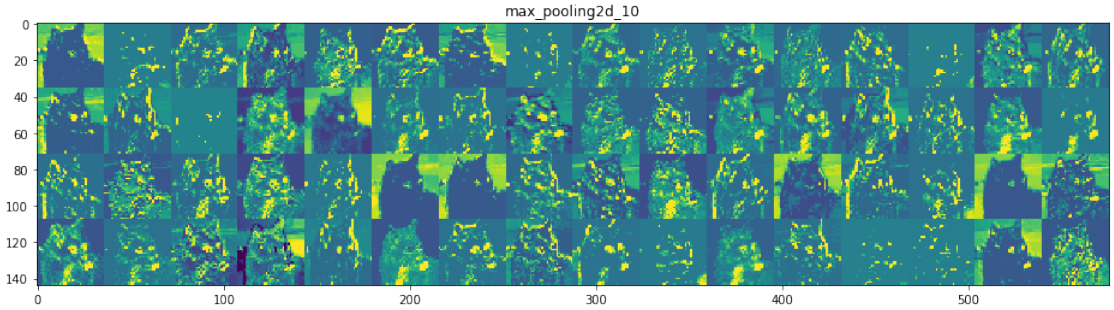
In [22]:
```
layer_names = []
for layer in model.layers[:8]:
    layer_names.append(layer.name)

images_per_row = 16

for layer_name, layer_activation in zip(layer_names, activations):
```

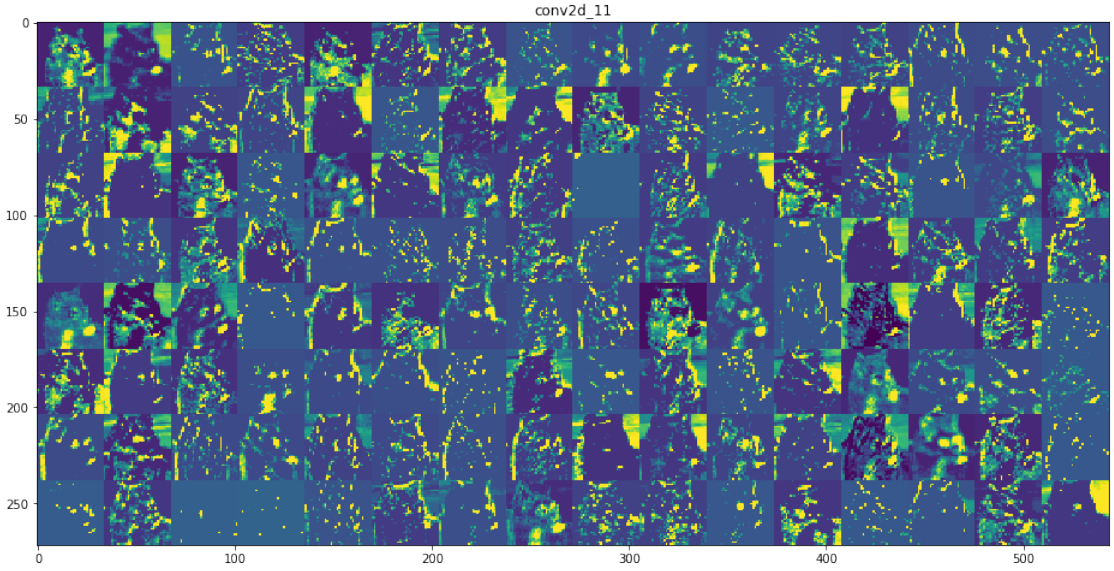
```
n_features = layer_activation.shape[-1]

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')
plt.show()
```

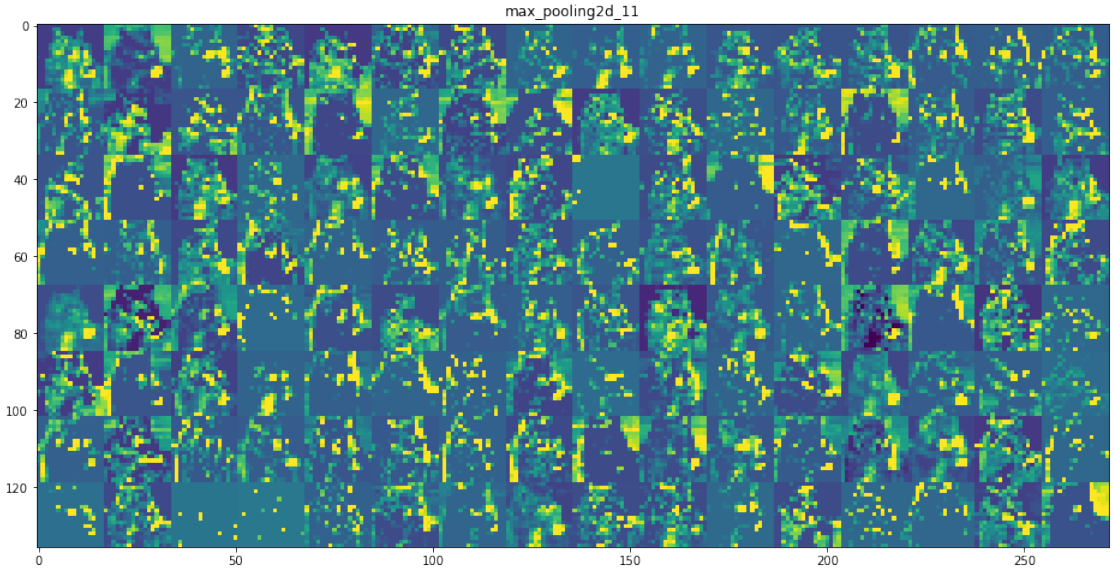


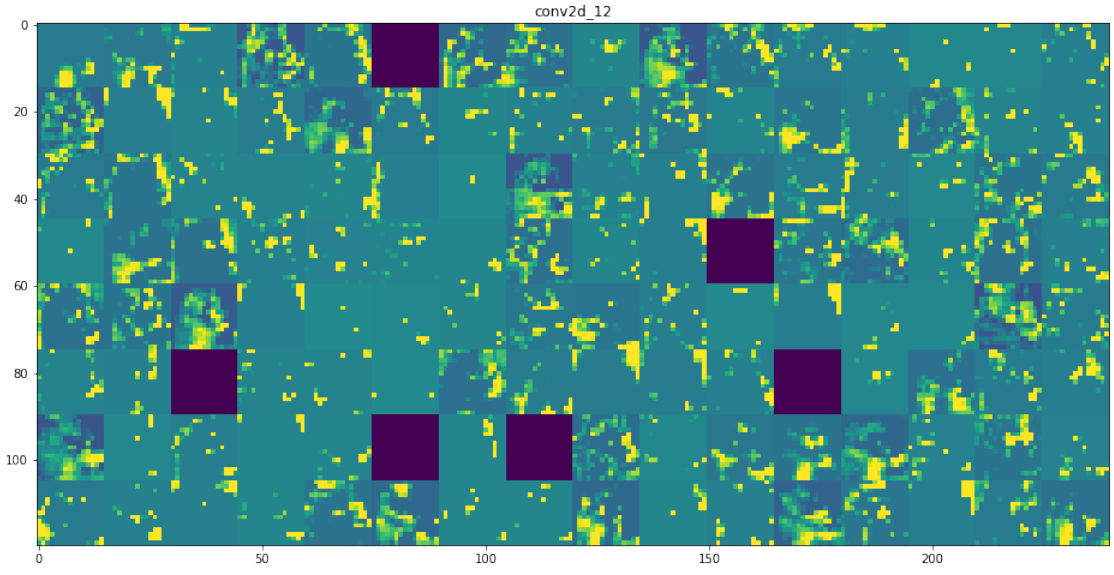


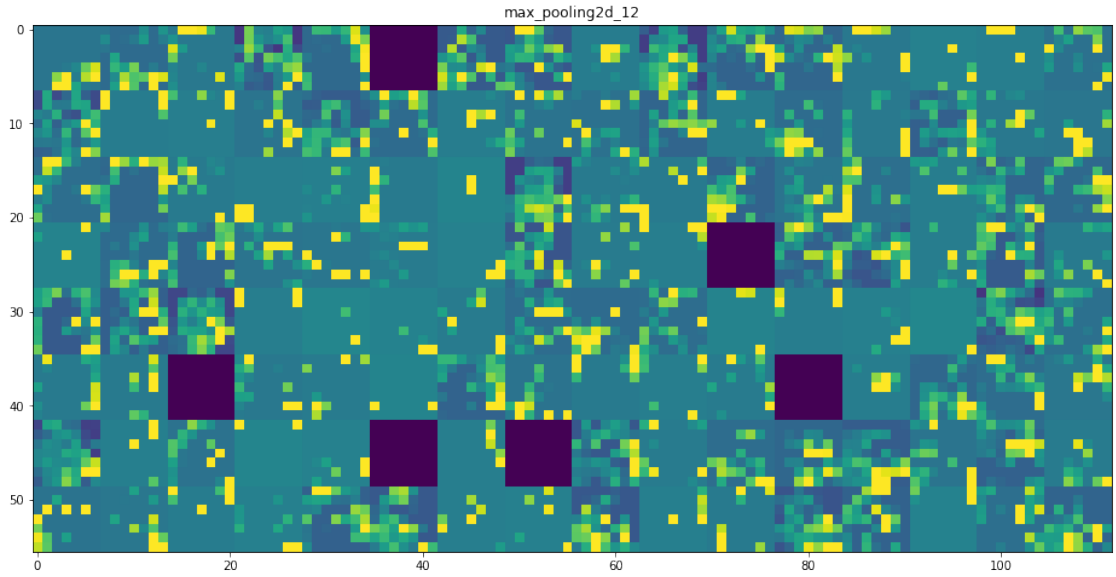

conv2d_10
0
50
100
150
200
250
0
200
400
600
800
1000


max_pooling2d_10
0
20
40
60
80
100
120
140
0
100
200
300
400
500


conv2d_11
0
50
100
150
200
250
0
100
200
300
400
500


max_pooling2d_11
0
20
40
60
80
100
120
0
50
100
150
200
250


conv2d_12
0
20
40
60
80
100
0
50
100
150
200



1.1 Some things to note

- This first layer is a collection of various edge detectors. At this stage the activations retain almost all of the information present in the picture.
- Going higher the activations become increasingly abstract. They are encoding things like 'cat ear' or 'cat eye'. Higher representation contain less information about the visual contents of the image, and increasingly more information about the class of the image
- The sparcity of the activations inreases with the depth of the layer. In the first image all of the filters are activated by the input image. In later layers more and more filters are blank. This shows that the pattern encoded by the filter isn't found in the input image.