In [1]:

```
from keras.models import load_model

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

```
WARNING:tensorflow:From C:WProgramDataWAnaconda3WlibWsite-packagesWtensorflowWpyth
onWframeworkWop_def_library.py:263: colocate_with (from tensorflow.python.framewor
k.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:WProgramDataWAnaconda3WlibWsite-packagesWkerasWbackendWt
ensorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) wit
h keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_p
rob`.
WARNING:tensorflow:From C:WProgramDataWAnaconda3WlibWsite-packagesWtensorflowWpyth
onWopsWmath_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is depreca
ted and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (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
_________________________________________________________________
```

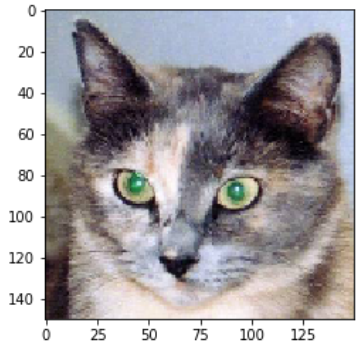
In [2]:

```
from keras.preprocessing import image
import numpy as np

img_path= './datasets/cats_and_dogs_small/test/cats/cat.1700.jpg'
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.

print(img_tensor.shape)
```

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

In [4]:

```
import matplotlib.pyplot as plt

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



In [7]:

```
from keras import models

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

In [8]:

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

In [9]:

```
first_layer_activation = activations[0]
first_layer_activation.shape
```

Out[9]:

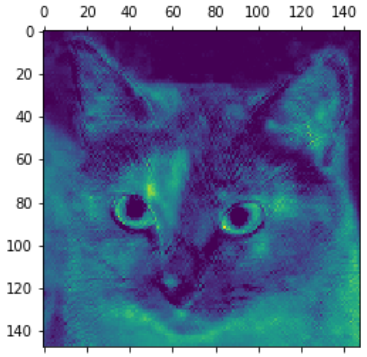
```
(1, 148, 148, 32)
```

In [10]:

```
plt.matshow(first_layer_activation[0,:,:,19], cmap='viridis')
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x26e7bca9f98>
```

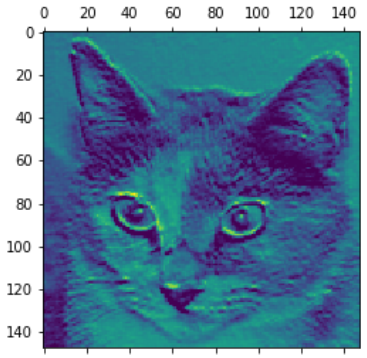


In [11]:

```
plt.matshow(first_layer_activation[0,:,:,15], cmap='viridis')
```

Out[11]:

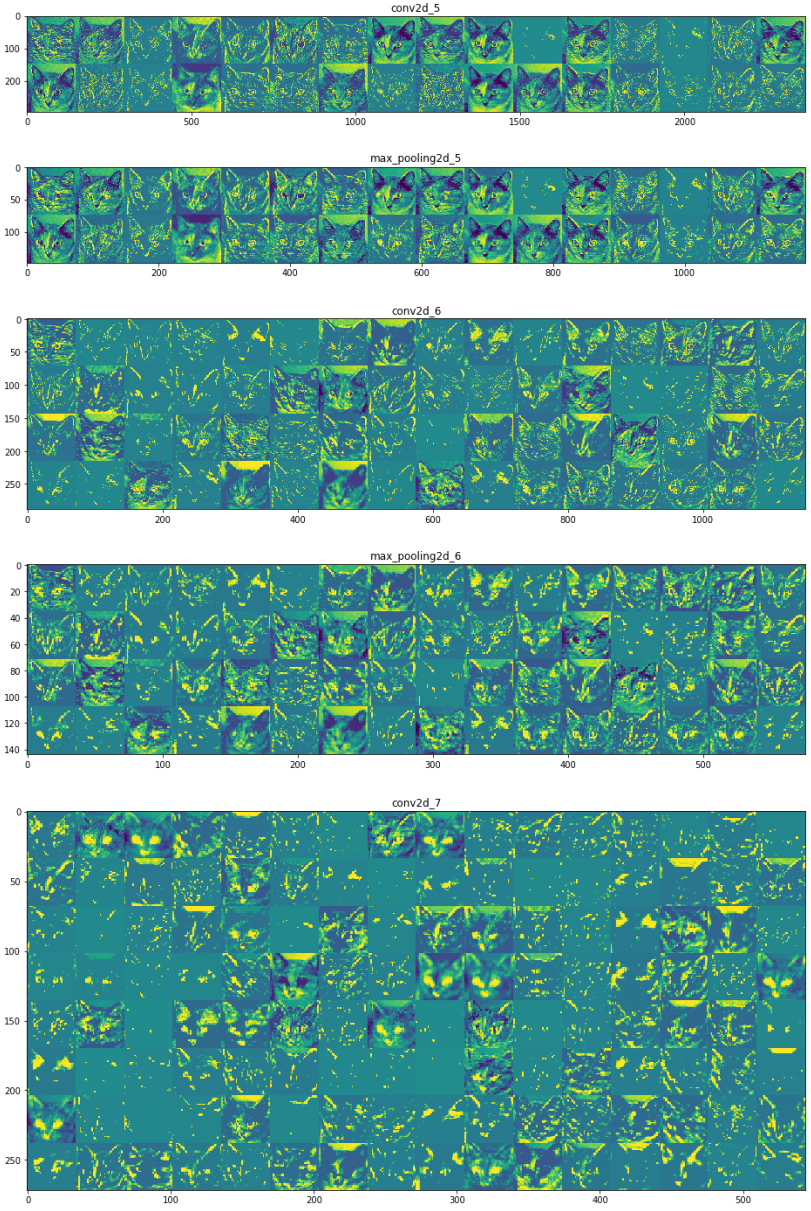
```
<matplotlib.image.AxesImage at 0x26e7b7ada20>
```

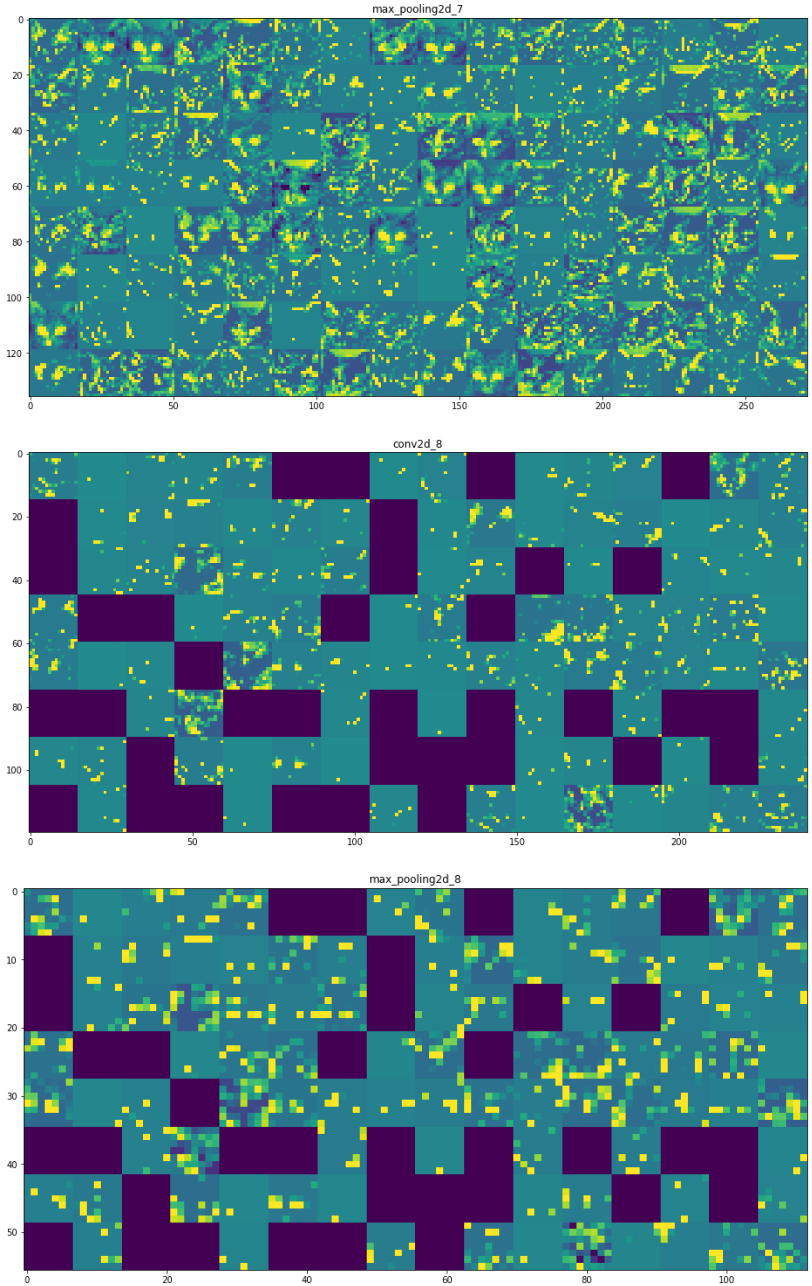


In [12]:

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

image_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 // image_per_row
    display_grid = np.zeros((size * n_cols, image_per_row * size))

    for col in range(n_cols):
        for row in range(image_per_row):
            channel_image = layer_activation[0,:,:,col * image_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()
```

C:₩ProgramData₩Anaconda3₩lib₩site-packages₩ipykernel_launcher.py:19: RuntimeWarning: invalid value encountered in true_divide





In [13]:

```
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])
```

그래디언트 정규화하기

In [14]:

```
grads = K.gradients(loss, model.input)[0]
grads /=  (K.sqrt(K.mean(K.square(grads))) + 1e-5)
```

입력 값에 대한 넘파이 값 추출하기

In [15]:

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

loss_value, grads_value = iterate([np.zeros((1,150,150,3))])
```

확률적 경사 상승법을 사용한 손실 최대화하기

In [16]:

```
input_img_data = np.random.random((1,150,150,3)) * 20 + 128

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

텐서를 이미지 형태로 변환하기 위한 유틸리티 함수

In [17]:

```
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
```

필터 시각화 이미지를 만드는 함수

In [20]:

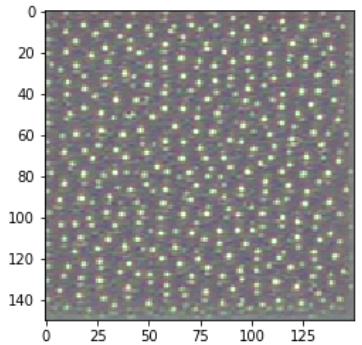
```
def generate_pattern(layer_name, filter_index, size=150):
    layer_output = model.get_layer(layer_name).output
    loss = K.mean(layer_output[:,:,:,filter_index])

    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)
```

In [21]:

```
plt.imshow(generate_pattern('block3_conv1', 0))
'''
block3_conv1 층 0번째 채널 최대 반응 패턴
'''
```

Out[21]:

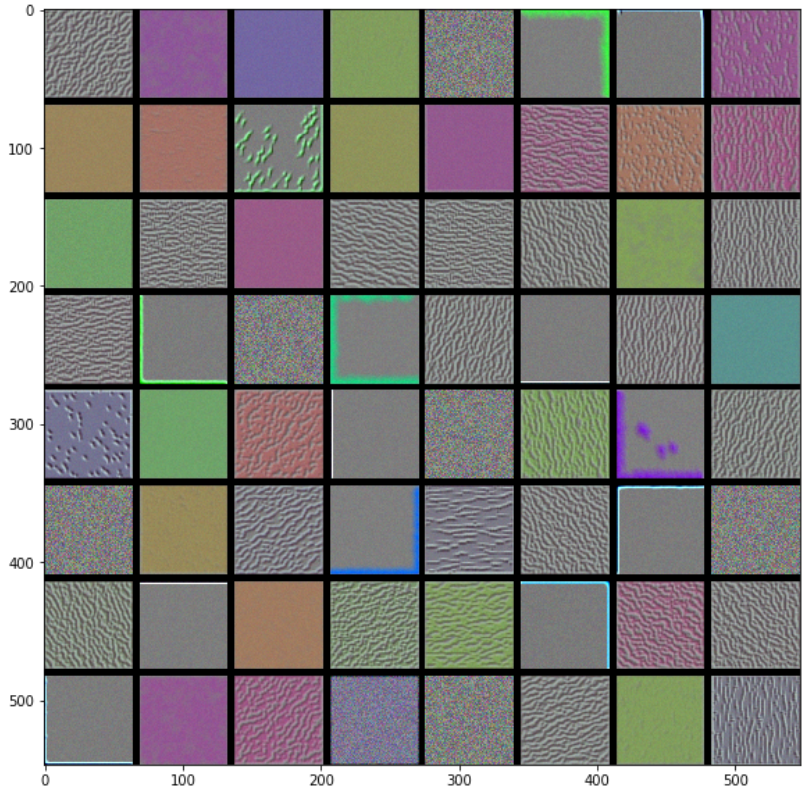
<matplotlib.image.AxesImage at 0x26c322c09b0>

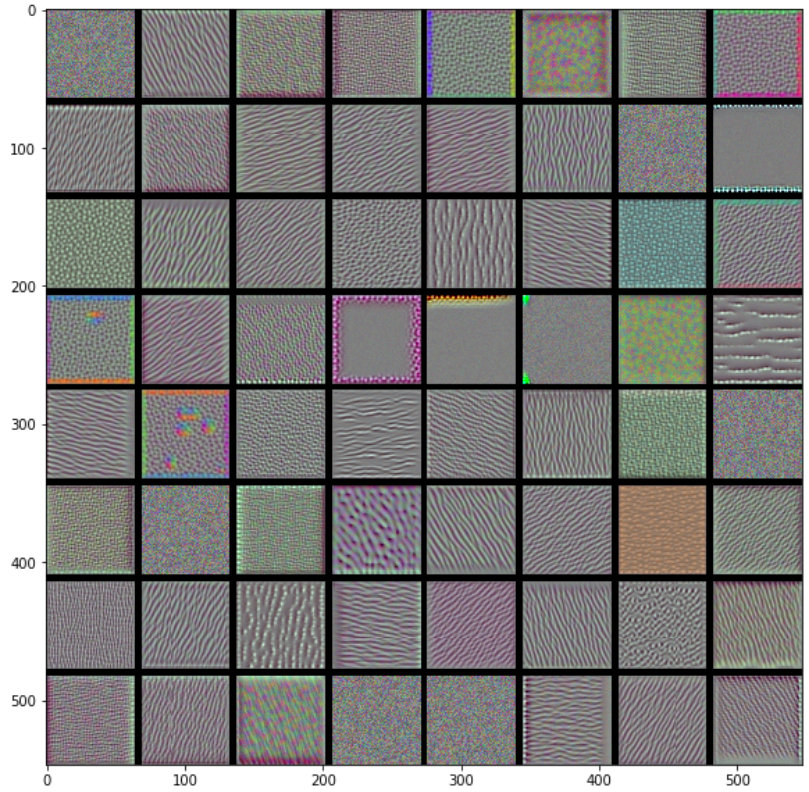


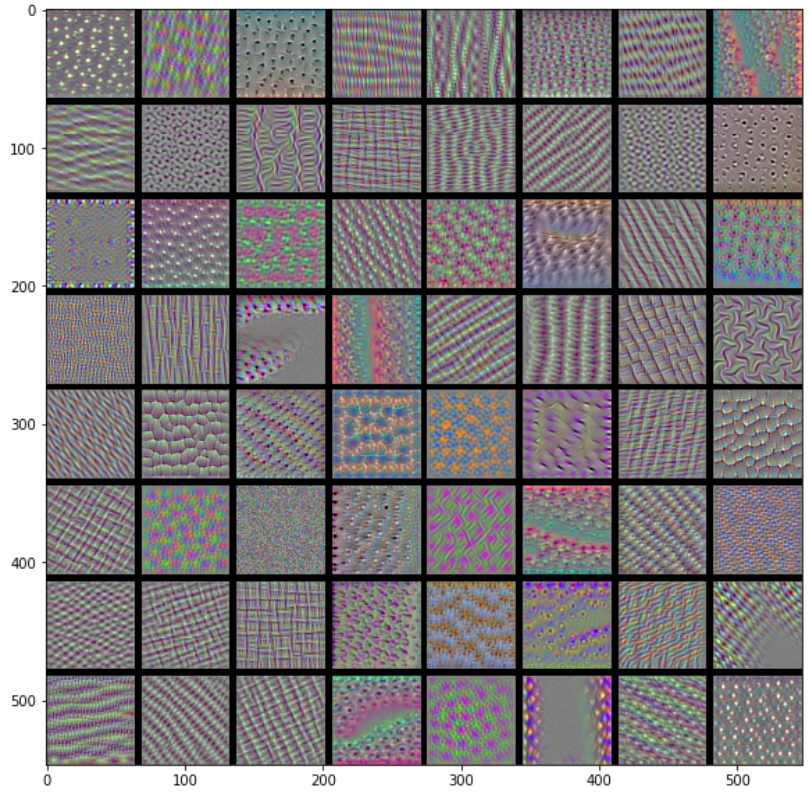
층에 있는 각 필터에 반응하는 패턴 생성하기

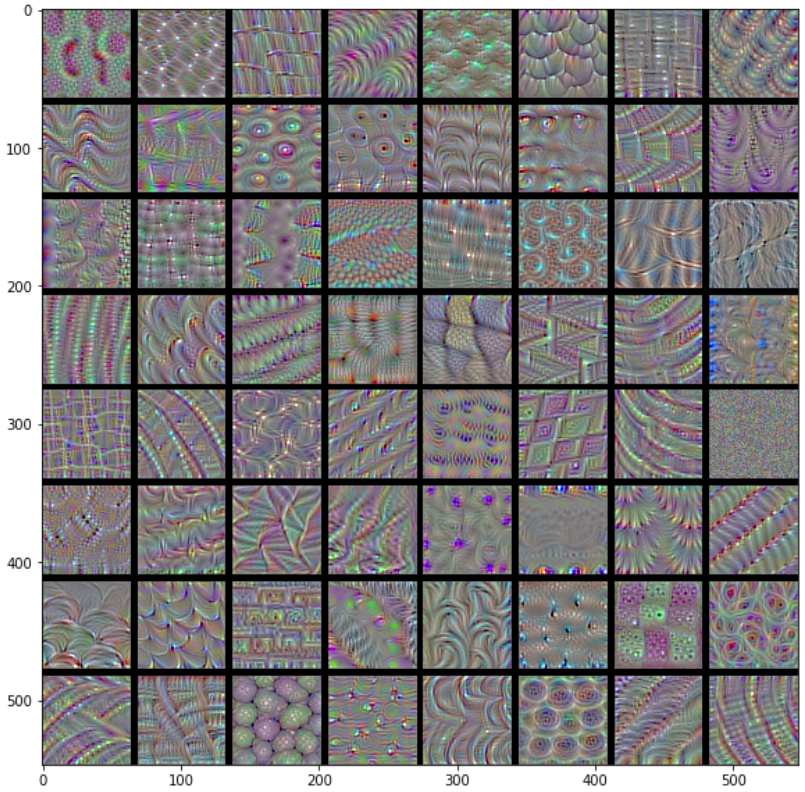
In [24]:

```
for layer_name in ['block1_conv1', 'block2_conv1', 'block3_conv1', 'block4_conv1']:
    size = 64
    margin = 5

    results = np.zeros((8 * size + 7 * margin, 8 * size + 7 * margin, 3), dtype='uint8')

    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=(10,10))
    plt.imshow(results)
    plt.show()
```




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


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



In [25]:

```
from keras.applications.vgg16 import VGG16, preprocess_input, decode_predictions
from keras.preprocessing import image

model = VGG16(weights='imagenet')

img_path = './creative_commons_elephant.jpg'
img = image.load_img(img_path, target_size=(224,224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
```

```
Downloading data from https://github.com/fchollet/deep-learning-models/releases/do
wnload/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467904/553467096 [==============================] - 69s 0us/step
```

In [26]:

```
preds = model.predict(x)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/
imagenet_class_index.json
40960/35363 [================================] - 0s 1us/step
Predicted: [('n02504458', 'African_elephant', 0.90942144), ('n01871265', 'tusker',
0.08618243), ('n02504013', 'Indian_elephant', 0.0043545845)]
```

In [27]:

```
np.argmax(preds[0])
```

Out[27]:

```
386
```

In [28]:

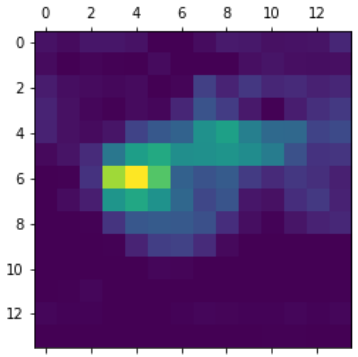
```
african_elephant_output = model.output[:, 386]

last_conv_layer = model.get_layer('block5_conv3')
grads = K.gradients(african_elephant_output, last_conv_layer.output)[0]
pooled_grads = K.mean(grads, axis=(0,1,2))
iterate = K.function([model.input],[pooled_grads, last_conv_layer.output[0]])
pooled_grads_value, conv_layer_output_value = iterate([x])
for i in range(512):
    conv_layer_output_value[:,:,i] *= pooled_grads_value[i]

heatmap = np.mean(conv_layer_output_value, axis=-1)
```

In [29]:

```
heatmap = np.maximum(heatmap, 0)
heatmap /= np.max(heatmap)
plt.matshow(heatmap)
```

Out[29]:

```
<matplotlib.image.AxesImage at 0x26ecdb08e48>
```



In [36]:

```
import cv2

img = cv2.imread(img_path)

heatmap = cv2.resize(heatmap, (img.shape[1], img.shape[0]))
heatmap = np.uint8(255 * heatmap)
heatmap = cv2.applyColorMap(heatmap, cv2.COLORMAP_JET)
superimposed_img = heatmap * 0.4 + img
cv2.imwrite('./elephant_cam.jpg', superimposed_img)
```

Out[36]:

```
True
```