01_filter_example

September 29, 2021

1 Visualize Image Filter

1.1 Imports

[1]:
```
%matplotlib inline
import cv2
import numpy as np
import scipy.misc
import matplotlib.pyplot as plt
import matplotlib.cm as cm

from keras.models import Sequential
from keras.layers.convolutional import Convolution2D
```

```
Using TensorFlow backend.
```

1.2 Import the Image

[2]:
```
# Feel free to try out your own images
img_path = 'images/building.png'
```

[3]:
```
# load color image
bgr_img = cv2.imread(img_path)
```

[4]:
```
### convert to grayscale
gray_img = cv2.cvtColor(bgr_img, cv2.COLOR_BGR2GRAY)
```

[5]:
```
# resize to smaller
small_img = scipy.misc.imresize(gray_img, 0.3)
```

```
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.6/site-
packages/ipykernel_launcher.py:2: DeprecationWarning: `imresize` is deprecated!
`imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.3.0.
Use Pillow instead: ``numpy.array(Image.fromarray(arr).resize())``.
```

[6]:
```
# rescale entries to lie in [0,1]
small_img = small_img.astype("float32")/255
```

[7]:
```
# plot image
plt.imshow(small_img, cmap='gray')
plt.axis('off');
```

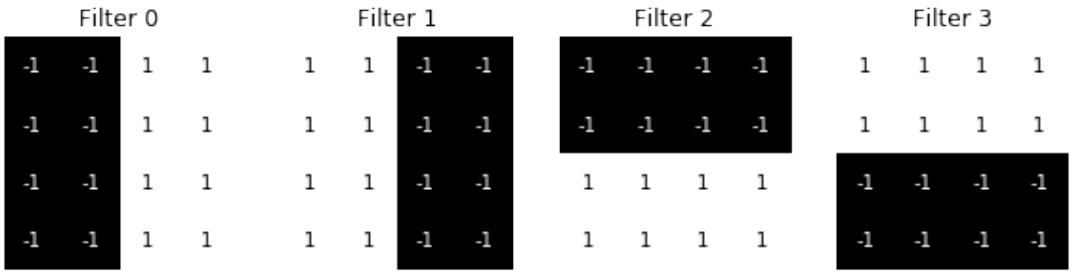
1.3 Specify the Filters

[8]:
```
# Feel free to modify the numbers here, to try out another filter!
filter_vals = np.array([[-1, -1, 1, 1],
                        [-1, -1, 1, 1],
                        [-1, -1, 1, 1],
                        [-1, -1, 1, 1]])
```

[9]:
```
# define four filters
filter_1 = filter_vals
filter_2 = -filter_1
filter_3 = filter_1.T
filter_4 = -filter_3
filters = [filter_1, filter_2, filter_3, filter_4]
```

[10]:
```
# visualize all filters
fig, axes = plt.subplots(ncols=4, figsize=(10, 5))
for i, ax in enumerate(axes):
    ax.imshow(filters[i], cmap='gray')
    ax.axis('off')
    ax.set_title(f'Filter {i}')
```

```
    width, height = filters[i].shape
    for x in range(width):
        for y in range(height):
            ax.annotate(str(filters[i][x][y]), xy=(y,x),
                        horizontalalignment='center',
                        verticalalignment='center',
                        color='white' if filters[i][x][y]<0 else 'black')
```



1.4 Visualize the Activation Maps for Each Filter

[11]:
```
plt.imshow(small_img, cmap='gray')
plt.axis('off')
```

[11]: (-0.5, 1531.5, 1020.5, -0.5)

1.4.1 Define single-layer CNN

[12]:
```
cnn = Sequential([
    Convolution2D(1, (4, 4),
                  activation='relu',
                  input_shape=(small_img.shape[0], small_img.shape[1], 1))
])
```

```
WARNING:tensorflow:From
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.6/site-
packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Colocations handled automatically by placer.
```

[13]:
```
cnn.summary()
```

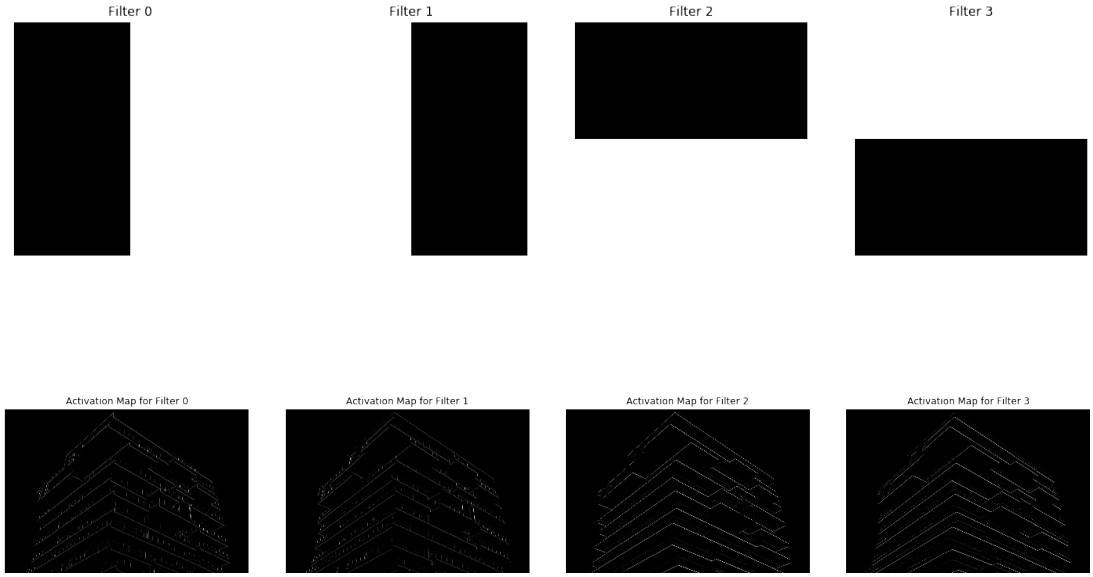
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 1018, 1529, 1)     17
=================================================================
Total params: 17
Trainable params: 17
Non-trainable params: 0
_________________________________________________________________
```

1.4.2 Apply convolutional filter and return output

[14]:
```
def apply_filter(img, index, filter_list, ax):
    # set the weights of the filter in the convolutional layer to filter_list[i]
    cnn.layers[0].set_weights([np.reshape(filter_list[i], (4,4,1,1)), np.
 ↪array([0])])
    # plot the corresponding activation map
    ax.imshow(np.squeeze(cnn.predict(np.reshape(img, (1, img.shape[0], img.
 ↪shape[1], 1)))), cmap='gray')
```

[15]:
```
# visualize all filters
fig, axes = plt.subplots(ncols=4, figsize=(15, 5))
for i, ax in enumerate(axes):
    ax.imshow(filters[i], cmap='gray')
    ax.axis('off')
    ax.set_title(f'Filter {i}')
fig.tight_layout()
fig.savefig('images/filters', dpi=300)
```

```
# visualize all activation maps
fig, axes = plt.subplots(ncols=4, figsize=(20, 20))
for i, ax in enumerate(axes):
    apply_filter(small_img, i, filters, ax)
    ax.axis('off')
    ax.set_title(f'Activation Map for Filter {i}')
fig.tight_layout()
```



[ ]: