Convolutional Neural Networks

In this notebook, we visualize four activation maps in a CNN layer.

1. Import the Image

In [1]:

```
import cv2
import scipy.misc
import matplotlib.pyplot as plt
%matplotlib inline

# TODO: Feel free to try out your own images here by changing img_path
# to a file path to another image on your computer!
img_path = 'images/WIN_20170808_20_28_09_Pro.jpg'

# load color image
bgr_img = cv2.imread(img_path)
# convert to grayscale
gray_img = cv2.cvtColor(bgr_img, cv2.COLOR_BGR2GRAY)
# resize to smaller
small_img = scipy.misc.imresize(gray_img, 0.3)

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

# plot image
plt.imshow(small_img, cmap='gray')
plt.show()
```

2. Specify the Filters

In [2]:

```
import numpy as np
```

In [130]:

```
# TODO: Feel free to modify the numbers here, to try out another filter!
# Please don't change the size of the array ~ :D
filter_vals = np.array([[-1, -1, 1, 1], [-1, -1, 1, 1], [-1, -1, 1, 1], [-1, -1, 1, 1]])
```

In [131]:

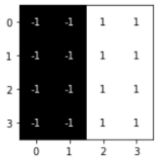
```
### do not modify the code below this line ###

# 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]
```

In [132]:

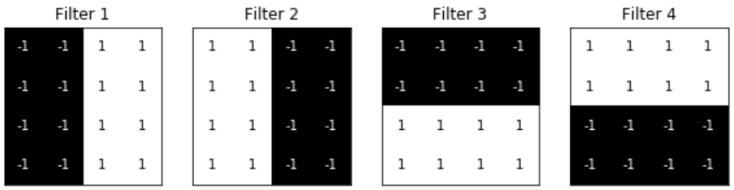
```
fig1 = plt.figure(figsize=(5,2.5))
#ax = fig.add_subplot()
plt.imshow(filters[0],cmap='gray')
```

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



In [133]:

```
# visualize all filters
fig = plt.figure(figsize=(10, 5))
for i in range(4):
    ax = fig.add_subplot(1, 4, i+1, xticks=[], yticks=[])
    ax.imshow(filters[i], cmap='gray')
    ax.set_title('Filter %s' % str(i+1))
    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')
```



3. Visualize the Activation Maps for Each Filter

In [134]:

```
from keras.models import Sequential
from keras.layers.convolutional import Convolution2D
import matplotlib.cm as cm
```

In [200]:

```
# define a neural network with a single convolutional layer with one filter
model = Sequential()
model.add(Convolution2D(1, (4, 4),
                        activation='relu', #padding='Same', strides=1,
                        input_shape=(small_img.shape[0],small_img.shape[1],1)))
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_30 (Conv2D)           (None, 141, 189, 1)       17
=================================================================
Total params: 17.0
Trainable params: 17
Non-trainable params: 0.0
```

In [201]:

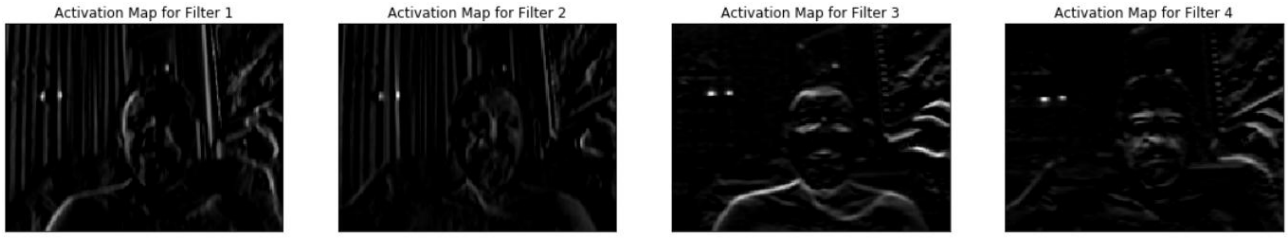
```
# apply convolutional filter and return output
def apply_filter(img, index, filter_list, ax):
    # set the weights of the filter in the convolutional layer to filter_list[i]
    model.layers[0].set_weights([np.reshape(filter_list[i], (4,4,1,1)), np.array([0])])
    # plot the corresponding activation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

In [202]:

```
# visualize all filters
fig = plt.figure(figsize=(12, 6))
fig.subplots_adjust(left=0,right=1.5,bottom=0.8,top=1, hspace=0.05, wspace=0.05)
for i in range(4):
    ax = fig.add_subplot(1, 4, i+1, xticks=[], yticks=[])
    ax.imshow(filters[i], cmap='gray')
    ax.set_title('Filter %s' % str(i+1))
```

In [203]:

```
# visualize all activation maps
fig = plt.figure(figsize=(20, 20))
for i in range(4):
    ax = fig.add_subplot(1, 4, i+1, xticks=[], yticks=[])
    apply_filter(small_img, i, filters, ax)
    ax.set_title('Activation Map for Filter %s' % str(i+1))
```



In [187]:

```
def visualize_input(img, ax):
    ax.imshow(img, cmap='gray')
    width, height = img.shape
    thresh = img.max()/2.5
    print(img.max(),thresh)
#     for x in range(width):
#         for y in range(height):
#             ax.annotate(str(round(img[x][y],2)), xy=(y,x),
#                         horizontalalignment='center',
#                         verticalalignment='center',
#                         color='white' if img[x][y]<thresh else 'black')

fig = plt.figure(figsize = (14,19))
ax = fig.add_subplot(111)
visualize_input(small_img, ax)
```

```
0.972549 0.389019608498
```