COEN 140 Machine Learning and Data Mining

Homework 5

Complete the following problems and submit your homework as a pdf file to Camino lecture section. Append/print the code for Problem 3 at the end of the homework. **Calculation questions need detailed derivations. We cannot accept only a final result.**

Problem 1: Consider a convolutional neural network that has an input of size $75 \times 75 \times 3$ ($height \times width \times depth$). For the following 6 layers in the network, calculate (1) the layer output dimension, and (2) the number of multiplications required to generate that layer.

Assume the weights do not have bias terms. For the number of channels (depth) of each filter, please figure it out by yourself.

Layer 1: convolutional layer
10 filters of size: 3×3 (filter height $\times$ filter width)
Stride $S = [1,1]$ (in the direction of [height, width])
Zero-padding P=[0,0] (in the direction of [height, width])

Layer 2: convolutional layer
20 filters of size: 5×5
Stride $S = [2,2]$
Zero-padding P=[0,0]

Layer 3: max pooling layer
filter size: 2×2
Stride $S = [2,2]$
Zero-padding P=[0,0]

Layer 4: convolutional layer
40 filters of size: 5×5
Stride $S = [2,2]$
Zero-padding P=[0,0]

Layer 5: flattening Layer

Layer 6: A fully connected layer, with a single output node.

Problem 2

Consider a CNN. Its first 3 layers are all convolutional layers, each with a 3×3 kernel, a stride of 2, and no zero padding. The 1st layer outputs 100 feature maps, the 2nd one outputs 200, and the third one outputs 400. The input images are RGB (3 channels: red, green, blue) images of 200×300 pixels. What is the total number of parameters in the first three layers of this CNN? Assume that the weights do not have bias terms.

Problem 3

You are provided with lena256.jpeg, a 256×256 gray-scale image. Load it in python, and implement a 2D convolution from scratch with the following filter. Let the stride S=[1,1] ([height, width]), and there's no zero-padding.

$$Filter = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Store the convolution output in a matrix "output", then take the absolute values of "output", and normalize the values to be in the range [0,255]. This can be done with the following code:

```
output = numpy.abs(output)
o_max = max(output.flatten())
o_min = min(output.flatten())

output_normalized = (output-o_min)/o_max*255
```

Then display output_normalized as a gray-scale image. What do you observe from the result? What do you think the filter do?