Part I

Individual Assignment: Understanding Convolutional Network Basics

This portion of the assignment is to build your intuition and understanding the basics of convolutional networks - namely how convolutional layers learn feature maps and how max pooling works - by manually simulating these. We highly recommend reading the Stanford page on convolutional networks. Especially scroll down and take a look at the animated **Convolution Demo**.

For questions 1-3, consider the $(5 \times 5 \times 2)$ input with values drawn from $\{-1, 0, 1\}$ and the corresponding $(3 \times 3 \times 2)$ filter shown below.

Input Features

-1	0	0	-1	1
0	1	1	-1	0
1	1	1	1	1
-1	0	-1	1	0
0	0	1	1	1

-1	-1	-1	0	-1
-1	0	-1	-1	0
0	-1	0	0	1
0	1	1	1	1
1	0	1	1	1

Filter 0

-1	-1	-1
0	0	0
1	1	1

1	0	-1
1	0	0
1	1	1

1. In the provided area below, fill in the values resulting from applying a convolutional layer to the input with no zero-padding and a stride of 1. Calculate these numbers before any activation function is applied. If there are any unused cells after completing the process in the provided tables, place an "X" in them.{5pts}

X	X	X	X	X
X	2	2	3	X
X	-2	2	2	X
X	0	1	3	X
X	X	X	X	X

2. Think about what ideal 3×3 patch from each of the input channels would maximally activate the corresponding 3×3 filter. Fill in these maximally activating patches in the area below. Note that the numbers should come from the same set as before: $\{-1, 0, 1\}$, and use 0 where the element of the patch doesn't matter. (Hint: You will use *all* of these cells).

Maximally
Activating Patch

-1	-1	-1
0	0	0
1	1	1

1	0	-1
1	0	0
1	1	1

3. **Spatial pooling:** Using your **output feature map** in question 1, first apply ReLU, and then apply *max-pooling* using a $[2\times2]$ kernel with a stride of 1. Recall from lecture that spatial pooling gives a CNN invariance to small transformations in the input, and max-pooling is more widely used over sum or average pooling due to empirically better performance. Again, if there are any unused cells after completing the process in the provided tables, place an "X" in them.

2	3	X
2	3	X
X	X	X

4. **Number of learnable parameters**
Suppose we had the following architecture:

```
inputs -> conv1 -> conv2 -> conv3 -> maxpool -> fc1 -> fc2 (outputs)
```

where all convs have a stride of 1 and no zero-padding, and the inputs are a greyscale image (single channel). Conv1 has a kernel size of $[8 \times 8]$ with 12 output channels, conv2 has a $[8 \times 8]$ kernel with 10 output channels, conv3 has a $[6 \times 6]$ kernel with 8 output channels, and maxpool has a $[3 \times 3]$ kernel.
If the inputs are $[512 \times 512]$ greyscale images, what are (Hint: Don't forget the bias!):

(i) The number of parameters required for **conv1**: 8 x 8 x 12 + 12 =780

(ii) The number of parameters required for **conv2**: 8 x 8 x12 x 10 + 10 = 7690

(iii) The number of parameters required for **conv3**: 6 x 6 x 10 x 8 + 8 = 2888

(iv) Size of the effective receptive field of a neuron in **conv1**: 8 x 8

(v) Size of the effective receptive field of a neuron in **conv2**: 15 x 15

(vi) Size of the effective receptive field of a neuron in **conv3**: 20x20

(vii) What are the number of inputs to **fc1**? Show your work.

conv1: (512 - 8) /1 + 1 = 505. Shape: 505 x 505 x 12

conv2: (505 - 8) /1 + 1 = 498. Shape: 498 x 498 x 10

conv3: (498 - 6) /1 + 1 = 493. Shape: 493 x 493 x 8

maxpool: (493-3)/1 +1 = 491. Shape: 491 x 491 x 8

Input FC1: 491 x 491 x 8 = 1928648

(viii) If fc1 is 100-dimensional and fc2 is 100 dimensional, how many parameters are there between the two layers (Again, include the bias!)? How does this compare to the total number of parameters in the convolutional layers?

100 x 100 +100 = 10100 (8 x 8 x 12 + 12) + (8 x 8 x12 x 10 + 10) + (6 x 6 x 10 x 8 + 8) = 11358

(ix) Dropout is a very effective regularization technique in deep neural networks. It's only (meta-)parameter is a regularization term called the drop_probability which is the probability that a neuron will be dropped from the network. **Write 2-3 lines about what you understand about dropout and why you think dropout works. Feel free to read about it from it's original paper[2] or on the web, but put this in your own words!**

Deep neural nets always contains a lot of parameters which easily leads to overfitted.Dropout is a tool for tackle this issue. The main operation is to randomly shut down some units with their connections from the neural network during training. The reason why dropout can bring advantage in my opinion is
(1) dropout acts what similar to regularization, it enables some of the units to become 0 namely makes the units to be sparse
(2) drop outs randomly select units as input, which can reduce the relationship between the neighbouring units
(3) dropout can break up the co-adaptations by making the presence of any particular hidden unit unreliable