ECE/CS 6524 HW2 Math Problem

For the math problems, you can submit your scanned hand-written answers (please make sure they are clear to read), or electronic version. Please submit a **PDF file** of math problems independent from the coding part. At the end, you will submit:

(i) **Your solutions to the math problems** (PDF file, name format: Assiment2_Math_[YOUR VT PID])

(ii) **Your code** (zip file, including .ipynb and all other codes you used for the coding part. No data. Name format: Assignment_2_Code_[YOUR PID NUMBER].zip)

(iii) **Your notebook** (PDF file, name format: Assignment2_NB_[YOUR PID NUMBER].pdf)

1 Understanding Convolutional Neural Network Basics [10 pts]

This portion [1] of the assignment is to build your intuition and understanding the basics of convolutional neural networks - namely how convolutional neural layers learn feature maps and how max pooling works - by manually simulating these. This part of the assignment has to be submitted independently.
For questions 1-3, consider the $(5 \times 5 \times 2)$ input with values in {-1,0,1} and the corresponding $(3 \times 3 \times 1)$ filter shown in Figure 1.
(As in PyTorch, we refer to this single filter as being $(3 \times 3 \times 1)$ despite having two sub-filters, because it would produce an output with 1 channel. In a case where the input had 4 channels, the filter would have 4 sub-filters but still be $(3 \times 3 \times 1)$.)
1. **[3 pts]** In the provided area in Figure 2, fill in the values resulting from applying a convolutional layer to the input with 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 $\times$ in them. You can assume a bias of 0. The Output Feature Map will be a combination of the Filtered inputs, as in the Stanford example.
2. **[2 pts]** Think about what ideal 3×3 patch (values in range [-1, 1]) from each of the input channels would maximally activate (i.e. maximizing the output) the corresponding 3×3 filter. Fill in these maximally activating patches in Figure 3. If there are any cells for which the input values don't matter, place an $\times$ in them.
3. **[1 pt] Spatial pooling**: Using your **output feature map** in question 1, apply max-pooling using a $[2 \times 2]$ kernel with a stride of 1. Fill in your result in Figure 4.
4. **[4 pts] Number of learnable parameters**. Suppose we had the following architecture:

$$inputs \rightarrow conv1 \rightarrow conv2 \rightarrow conv3 \rightarrow maxpool \rightarrow fc1 \rightarrow fc2,$$

where all convs have a stride of 1 and no zero-padding. 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 ReLU is the activation function between each layer and the inputs are $[512 \times 512]$ greyscale images, what are:

[1] This problem is inspired by Garrison W. Cottrell 's CSE 253 at UCSD

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

Figure 1: Input and Filter

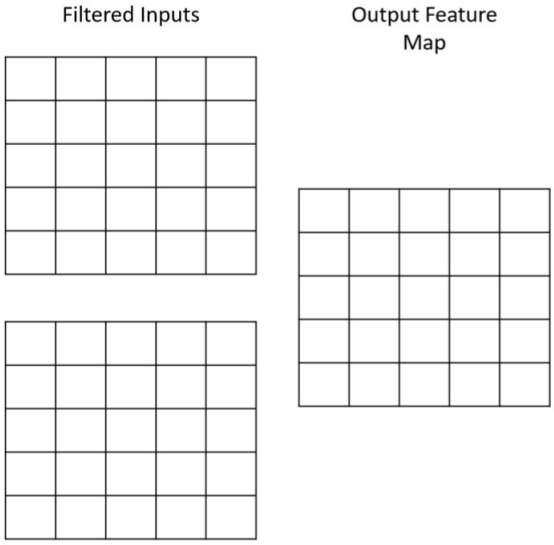

Figure 2: Question1

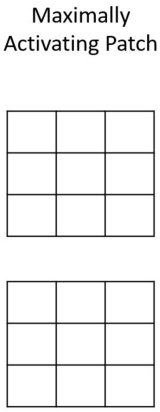

Figure 3: Max Activation

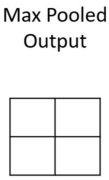

Figure 4: Max Pooling

(i) The number of input channels to **conv1**, **conv2** and **conv3**, respectively? **[2 pts]**

(ii) In order to add a fully-connected layer following the maxpool (stride = 1) layer, we need to reshape the convolutional outputs to feed them into this hidden layer. Based on this architecture, what will the incoming dimensions to **fc1** be? Show your work. **[2 pts]**

Logistic Regression: (10 Marks)

2. Logistic regression for two-class classification:
   - N pairs of data points: ($\mathbf{x}_i$, C_i), i = 1, …, N, where $\mathbf{x}_i$ is the feature vector, and C_i is the binary class label (i.e. C_i is either 0 or 1).
   - Logistic regression is a (special) linear classifier: $y_i = \boldsymbol{\beta}_1^T \mathbf{x}_i + \beta_0$, but y_i is interpreted as log-odds, i.e., $y_i = \log(p / (1 - p))$, where the probability of class 1 ($C_i = 1$) is p, and the probability of class 0 ($C_i = 0$) is 1 - p. The prediction is based on sigmoid(y_i) = 1/[1+exp(-y_i)]; if sigmoid(y_i) >= 0.5, $\mathbf{x}_i$ is predicted as a data sample from class 1; otherwise, class 0.
   - The likelihood function is: $\pi_{i=1,\ldots,N} (p(\mathbf{x}_i))^{C_i} (1- p(\mathbf{x}_i))^{(1 - C_i)}$, where $p(\mathbf{x}_i)$ is the probability of $\mathbf{x}_i$ from class 1.

   Prove that:
   1. The **log-likelihood** function can be rewritten/formulated as:
      $\sum_{i=1,\ldots,N} \{C_i (\boldsymbol{\beta}_1^T \mathbf{x}_i + \beta_0) – \log[1 + \exp(\boldsymbol{\beta}_1^T \mathbf{x}_i + \beta_0)]\}$
   2. **Compute the gradient** of the log-likelihood function.