CSC321H5 Homework 4.

Deadline: Thursday, March. 12, by 9pm

Submission: You must submit your solutions as a PDF file through MarkUs. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.

Late Submission: Please see the syllabus for the late submission criteria.

Question 1. – 12 pts

For this homework, you will first read the following paper:

> A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (NIPS), 2012.
> http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

This is a highly influential paper (nearly 10,000 citations on Google Scholar!) because it was one of the first papers to demonstrate impressive performance for a neural network on a modern computer vision benchmark. It generated lots of excitement both in academia and in the tech industry. The architecture presented in this paper widely used today, and is known as "AlexNet", after the first author. Reading this paper will also help you review a lot of the important concepts from this class.

The authors use a conv net architecture which has five convolution layers and three fully connected layers (one of which is the output layer). Your job is to count the number of units, the number of parameters, and the number of connections in each layer. I.e., you should complete the following table:

	# Input Units	# Output Units	# Parameters	# Connections
Convolution Layer 1	$3 \times 224 \times 224$			
Convolution Layer 2				
Convolution Layer 3				
Convolution Layer 4				
Convolution Layer 5				
Fully Connected Layer 1				
Fully Connected Layer 2				
Output Layer		1000		

You can ignore the pooling layers when doing these calculations, i.e., you don't need to consider the units in the pooling layers or the connections between convolution and pooling layers.

You can use Figure 2 of the paper to help fill out the table. If you prefer to look at an implementation of AlexNet in PyTorch, you can check output [26] of the lecture 6 demo here:
https://www.cs.toronto.edu/~lczhang/321/lec/conv_notebook.html

Question 2. – 6 pts

Part (a) – 4 points

Suppose we have a 4×4 input that looks like this:

1	2	1	-1
0	0	1	-1
1	0	1	-1
1	1	0	0

Apply the following 3×3 convolutional kernel to the input. Use a zero-padding of 1, and the default stride of 1. Assume that our bias is -1. Do not apply the ReLU activation or any pooling. Show your work computation for one of the output units (e.g. the multiplications and additions).

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

Part (b) 2 points

Consider the same input as above. Apply max pooling to the input (*not* your result from part a), with a kernel size of 2 and a stride of 2. Show your computation for one of the output units.

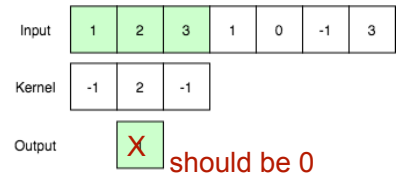
Question 3 – 12 points

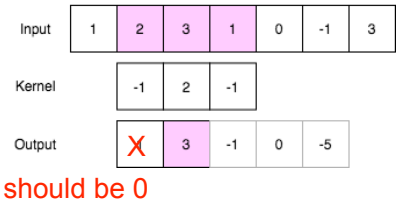
For this problem, we will compute the backward pass for a 1D convolution. In our treatment of convolutional neural networks, we focused on CNNs for images, which has a 2D geometry. Other types of data like audio, text, and time series data have a 1D geometry, and we can use convolutions for those data sets as well. For audio data, you can think of convolutions as detecting features in a small region in time.

Suppose we have audio or time series data that looks like this, where the target t is binary:

x	t
[1, 2, 3, 1, 0, -1, 3]	1
[0, 1, 4, 3, 1, 1, 1]	1
[1, 3, 2, 1, 1, 0, 1]	0

A 1D convolution can be applied by taking a 1D kernel (say [-1, 2, -1]), and sliding it across the sequence. At each position, we compute the dot product between the input with the kernel, like this:





should be 0

Part (a) – 3 pts

Suppose we are building a neural network to classify time series data like the one above. The architecture we choose will look like this:

- In the first layer, we'll apply a 1D convolution with kernel size 3. The number of input channels is 1, and the number of output channels is also 1. We won't use any zero padding. We'll initialize the weights of this 1D convolution to $\mathbf{w}^{(1)} = [-1, 2, -1]$ like in the figure, and initialize the bias to $\mathbf{b}^{(1)} = 0$. We will apply the ReLU activation after this layer.
- In the second layer, we'll have a fully-connected layer. There are 5 input units, and a single output unit. We'll initialize all the weights and biase to 1 so $w_k^{(2)} = 1$ and $b^{(2)} = 1$.
- We'll use the cross-entropy loss.

Compute the forward pass for the data point $x = [1, 3, 2, 1, 1, 0, 1]$, $t = 0$

Part (b) – 4 pts

Use backpropagation to compute the error signal $\overline{\mathbf{w}^{(1)}}$ of the 1D convolution kernel for the same data point $x = [1, 3, 2, 1, 1, 0, 1]$, $t = 0$. Show your steps. You do not need to compute the error signal for the other parameters.

Part (c) – 2 pts

1D convolutions can be and have been used for language modeling, where the sequence $\mathbf{x}$ represents a sequence of *words* in a sentence. Our target can represent (for example) whether a sentence conveys positive or negative sentiment.

The way we typically represent each *word* is either using a one-hot embedding (like in Project 2) or using distributed representations (like the one we learned in Project 2). In either case, each word will be represented using a vector. Here is a representation of a sentence $\mathbf{x} = w_1, w_2, \cdots, w_T$ using one-hot vectors:

w_1	w_2	w_3	$\cdots$	w_T
0	0	0	$\cdots$	1
1	0	0	$\cdots$	0
0	0	0	$\cdots$	0
0	1	0	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	0	0	$\cdots$	0

In particular, we can represent $\mathbf{x}$ as a matrix of shape $V \times T$, where V is the size of the vocabulary and T is the length of the sentence. This representation of $\mathbf{x}$ looks like a 2D, greyscale image!

Explain why it is *not* actually reasonable to use a 2D convolutional kernel (e.g., a 3x3 kernel like question 2).

Part (d) – 3 pts

Continuing from Part (c), what kind of computation should we use instead so that a 1D convolution can take account of the fact that each word w_j is a V-dimensional vector? How many trainable parameters will be in this new 1D convolution?