Instructor: Vishnu Boddeti
CSE 891-001: Deep Learning
Homework 3
December, 03

Instructions:

- **Filename:** Submit solutions in PDF format titled `written-assignment-3-msunetid.pdf`. Submissions in other formats or with other filenames will not be graded. You can produce your file however you like, LaTeX, Word or scan. Handwritten scans that are not legible will not be graded.
- **Submission:** Only homeworks uploaded to Google Classroom will be graded. Make sure to show all the steps of your derivations in order to receive full credit.
- **Integrity and Collaboration:** You are expected to work on the homeworks by yourself. You are not permitted to discuss them with anyone except the instructor. The homework that you hand in should be entirely your own work. You may be asked to demonstrate how you got any results that you report.
- **Clarifications:** If you have any question, please look at Google Classroom first. Other students may have encountered the same problem, and is solved already. If not, post your question there. We will respond as soon as possible.

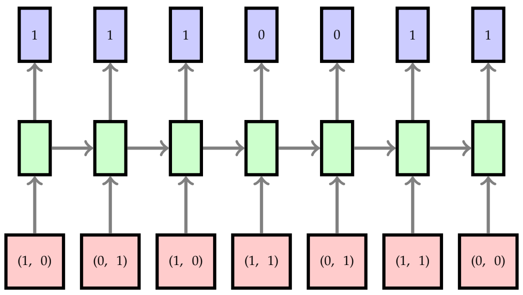
1 Binary Addition (2.5pts): In this problem, you will design a recurrent neural network to implement binary addition. The inputs are provided as binary sequences, starting with the least significant bit. The sequences are padded with at least one zero at the end. Here is an example:

$$0101101 + 0111010 = 1100111$$

where the inputs and outputs would be represented as:

- **Input 1:** 1, 0, 1, 1, 0, 1, 0
- **Input 2:** 0, 1, 0, 1, 1, 1, 0
- **Correct Output:** 1, 1, 1, 0, 0, 1, 1

At each time instance, the RNN would have two inputs and one output. Therefore, the pattern of inputs and outputs for the above example would be:



Design, by hand, the weights and biases for a RNN that can perform binary addition. The RNN should consist of two inputs, three hidden nodes and one output. You can assume that all the nodes use the hard threshold activation function. In particular, find the weights $\boldsymbol{W}_{xh}$, $\boldsymbol{W}_{hh}$ and $\boldsymbol{W}_{hy}$, the bias vector $\boldsymbol{b}_h$ and scalar b_y.

2 LSTM Gradient (3.5pts): In this problem you will derive the backpropagation-through-time equations for a univariate version of the Long-Term Short-Term Memory (LSTM) architecture we saw in class. Consider the case when the bias terms are assumed to be zero. So the LSTM computations are:

$$\begin{aligned}
i^{(t)} &= \sigma(w_{xi}x^{(t)} + w_{hi}h^{(t-1)}) \\
f^{(t)} &= \sigma(w_{xf}x^{(t)} + w_{hf}h^{(t-1)}) \\
o^{(t)} &= \sigma(w_{xo}x^{(t)} + w_{ho}h^{(t-1)}) \\
g^{(t)} &= tanh(w_{xg}x^{(t)} + w_{hg}h^{(t-1)}) \\
c^{(t)} &= f^{(t)}c^{(t-1)} + i^{(t)}g^{(t)} \\
h^{(t)} &= o^{(t)}tanh(c^{(t)})
\end{aligned}$$

1. (**3pts**) Derive the backpropagation-through-time equations for the activations and the gates:

$$\begin{aligned}
\overline{h^{(t)}} &= \ldots \\
\overline{c^{(t)}} &= \ldots \\
\overline{g^{(t)}} &= \ldots \\
\overline{o^{(t)}} &= \ldots \\
\overline{f^{(t)}} &= \ldots \\
\overline{i^{(t)}} &= \ldots
\end{aligned}$$

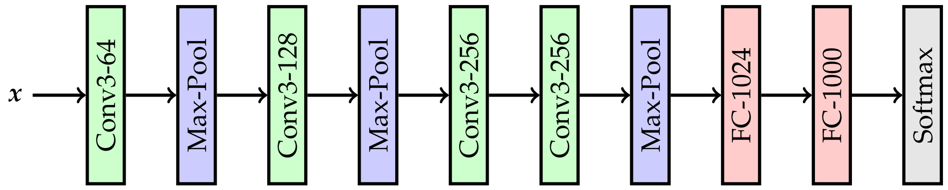
2. (**0.5pt**) Derive the update equation for the weight w_{xi}:

$$\overline{w_{xi}} = \ldots$$

3. (**Extra Credit: 0.5pt**) Based on your answers above, explain why the gradient does not explode if the values of the forget gates are very close to 1 and the values of the input and output gates are very close to 0. **Hint:** Your answer should involve both $\overline{h^{(t)}}$ and $\overline{c^{(t)}}$.

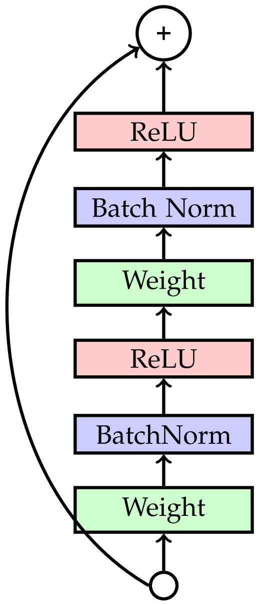
3 Convolutional Neural Networks (2pts):

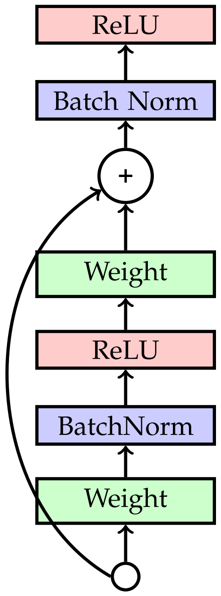
1. (**1pt**) Consider a CNN with 4 conv layers like in the diagram below. All 4 conv layers have kernel size of 3×3. The number after the hyphen specifies the number of output channels or units of a layer (e.g. Conv3-64 layer has 64 output channels and FC-1024 has 1024 output units). All the Max Pool in the diagram has size of 2×2. Assume zero padding of 1 for conv layers and stride 2 for Max Pool.



Size of the input image is 224×224 with 3 channels. Calculate the total number of parameters in the network including the bias units.

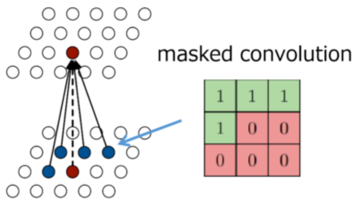
2. (**1pt**) Consider the following two ResNet architectures for the placement of the batch norm and the residual connection. "Weight" layer can be either a matrix multiplication or convolution. Which architecture is easier to learn in terms of exploding / vanishing gradient? Provide a brief justification for your answer.





4 Autoregressive Generative Models (2pt): In this question we will consider autoregressive models that model the distribution of images in an autoregressive manner. For simplicity, assume that each conditional distribution is a Gaussian with a fixed variance, so the model predicts the mean of the next pixel given all the previous pixels.

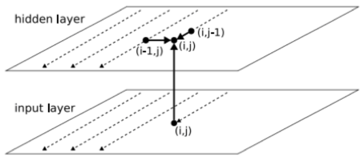
1. (**1pt**) Here we will consider PixelCNN, which models the distribution of images using a convolutional architecture. PixelCNN masks each convolutional filter to only see the pixels that appear before the current pixel in a raster scan order. See figure below for a visualization. Several of such layers are stacked sequentially.



   (a) Consider a d layer PixelCNN model, what is the total number of connections? Give your answers in terms of d, k, H, W. You only need to give $\mathcal{O}(\cdot)$, not an exact count.

   (b) Suppose that in each step we compute as many matrix-vector multiplications in parallel as we can, what is the minimum number of the sequential operations to compute the output of a Pixel CNN in terms of d, k, H, W. You only need to give $\mathcal{O}(\cdot)$, not an exact count.

2. (**1pt**) Here we will consider Multidimensional RNN (MDRNN). This is like the RNNs we discussed in the lecture, except that instead of a 1-D sequence, we have a 2-D grid structure. Analogous to how ordinary RNNs have an input vector and a hidden vector for every time step, MDRNNs have an input vector and hidden vector for every grid square. Each hidden unit receives bottom-up connections from the corresponding input square, as well as recurrent connections from its north and west neighbors as shown in the figure below.



   The activations are computed as:

$$\boldsymbol{h}^{(i,j)} = \boldsymbol{\phi}(\boldsymbol{W}_{in}^{T}\boldsymbol{x}^{(i,j)} + \boldsymbol{W}_{W}^{T}\boldsymbol{x}^{(i-1,j)} + \boldsymbol{W}_{N}^{T}\boldsymbol{x}^{(i,j-1)}) \tag{1}$$

   Denote the number of input channels and the number of recurrent neurons at each layer to be k. The input image size is $H \times W$. For simplicity, we assume there are no bias parameters.

   (a) Consider a d layer MDRNN model, what is the total number of connections? Give your answers in terms of d, k, H, W. You only need to give $\mathcal{O}(\cdot)$, not an exact count.

   (b) Suppose that in each step we compute as many matrix-vector multiplications in parallel as we can, what is the minimum number of the sequential operations to compute the output of a Pixel CNN in terms of d, k, H, W. You only need to give $\mathcal{O}(\cdot)$, not an exact count.

3. (**Extra Credit: 1pt**) What is the benefit of using PixelCNN over MDRNN. Discuss the pros and cons of the two models in terms of their computational, memory complexity, parallelization potential and the size of their context windows.