Homework 4 Neural Networks

Instructions

Answer the questions and upload your answers to courseville. Answers can be in Thai or English. Answers can be either typed or handwritten and scanned. the assignment is divided into several small tasks. Each task is weighted equally (marked with **T**). For this assignment, each task is awarded 0.4 points. There are also optional tasks (marked with **OT**) counts for 0.3 points each.

The Basics

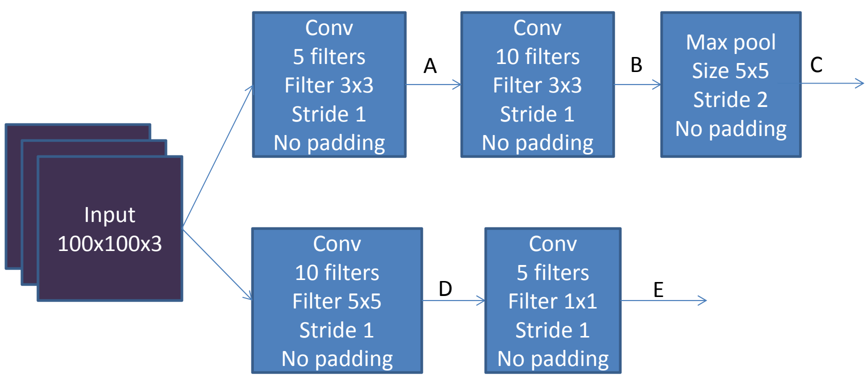
In this section, we will review some of the basic materials taught in class. These are simple tasks and integral to the understanding of deep neural networks, but many students seem to misunderstand.

T1. Compute the forward and backward pass of the following computation. Note that this is a simplified residual connection.

$$
\begin{aligned}
x_1 &= ReLU(x_0 * w_0 + b_0) \\
y_1 &= x_1 * w_1 + b_1 \\
z &= ReLU(y_1 + x_1)
\end{aligned}
$$

Let $x_0 = 1.0$, $w_0 = 0.5$, $w_1 = -0.2$, $b_0 = 0.1$, $b_1 = 0.15$. Find the gradient of z with respect to w_0, w_1, b_0, and b_1.

T2. Given the following CNN architecture specifications, determine the size of all the intermediate feature maps.



OT1. Note how at point B and D we have feature maps of the same size. If you think carefully, having 3x3 followed by 3x3 will cover a region of 5x5 in the input. This is why in Inception v2, the 5x5 convolution is replaced by two 3x3 convolutions. Compute the amount of multiplies to compute A and B. Compare it to the amount of multiplies to compute D. Compute the amount of parameters in the path to A and B. Compare it to the amount of parameters in path to D.

Deep Learning from (almost) scratch

In this section we will code simple a neural network model from scratch (numpy). However, before we go into coding let's start with some loose ends, namely the gradient of the softmax layer.

Recall in class we define the softmax layer as:

$$P(y = j) = \frac{exp(h_j)}{\Sigma_k exp(h_k)} \tag{1}$$

where h_j is the output of the previous layer for class index j

The cross entropy loss is defined as:

$$L = -\Sigma_j y_j log P(y = j) \tag{2}$$

where y_j is 1 if y is class j, and 0 otherwise.

T3. Prove that the derivative of the loss with respect to h_i is $P(y = i) - y_i$. In other words, find $\frac{\partial L}{\partial h_i}$ for $i \in \{0, ..., N-1\}$ where N is the number of classes. Hint: first find $\frac{\partial P(y=j)}{\partial h_i}$ for the case where $j = i$, and the case where $j \neq i$. Then, use the results with chain rule to find the derivative of the loss.

Next, we will code a simple neural network using numpy. Use the starter code `hw4_prob_part1.zip` on github.

Hints: In order to do this part of the assignment, you will need to find gradients of vectors over matrices. We have done gradients of scalars (Traces) over matrices before, which is a matrix (two-dimensional). However, gradients of vectors over matrices will be a tensor (three-dimensional), and the properties we learned will not work. I highly recommend you find the gradients in parts. In other words, compute the gradient for each element in the the matrix/vector separately. Then, combine the result back into matrices. For more information, you can read this simple guide `http://cs231n.stanford.edu/vecDerivs.pdf`

End of part 1