

## TTIC 31230 Fundamentals of Deep Learning, winter 2019

### CNN Problems

In these problems, as in the lecture notes, capital letter indices are used to indicate subtensors (slices) so that, for example,  $M[I, J]$  denotes a matrix while  $M[i, j]$  denotes one element of the matrix,  $M[i, J]$  denotes the  $i$ th row, and  $M[I, j]$  denotes the  $j$ th column.

Throughout these problems we assume a word embedding matrix  $e[W, I]$  where  $e[w, I]$  is the word vector for word  $w$ . We then have that  $e[w, I]^\top h[t, I]$  is the inner product of the word vector  $w[w, I]$  and the hidden state vector  $h[t, I]$ .

We will adopt the convention, similar to true Einstein notation, that repeated capital indices in a product of tensors are implicitly summed. We can then write the inner product  $e[w, I]^\top h[t, I]$  simply as  $e[w, I]h[t, I]$  without the need for the (meaningless) transpose operation.

**Problem 1.** Consider convolving a filter  $W[\Delta x, \Delta y, i, j]$  with thresholds  $B[j]$  on a “data box”  $L[b, x, y, i]$  where  $B, X, Y, I, J, \Delta X, \Delta Y$  are the number of possible values for  $b, x, y, i, j, \Delta x$  and  $\Delta y$  respectively. How many floating point multiplies are required in computing the convolution on the batch (without any activation function)?

**Solution:**

$$BXY \Delta X \Delta Y IJ$$

**Problem 2:** Suppose that we want a video CNN producing layers of the form  $L[b, x, y, t, i]$  which are the same as the layers of an image CNN but with an additional time index. Write the equation for computing  $L_{\ell+1}[b, x, y, t, j]$  from the tensor  $L_\ell[B, X, Y, T, I]$ . Your filter should include an index  $\Delta t$  and handle a stride  $s$  applied to both space and time.

**Problem 3.** Consider a bottleneck multi-layer perceptron (MLP) with residual connections defined as follows where  $h[0, J]$  is an input vector and where  $I$  is small compared to  $J$ .

$$\begin{aligned} b[\ell, I] &= \text{ReLU}(W^b[I, J]h[\ell, J]) \\ h[\ell + 1, J] &= h[\ell, J] + \text{ReLU}(W^h[\ell, J, I]b[\ell, I]) \end{aligned}$$

(a) What is the number of multiplications done by this network as a function of the hidden layer dimension  $J$ , the bottleneck vector dimension  $I$  and the number of layers  $L$ ? Under what conditions does this give fewer multiplications than the standard MLP with one matrix between layers? (For the tensor  $h[L, I]$  we have that  $\ell$  ranges from 0 to  $L - 1$ . We use this as a standard convention for tensor indices. The input layer has  $\ell = 0$  and the final layer has  $\ell = L - 1$ .)

**Solution:** The number of multiplications is  $2JI(L-1)$ . For a standard MLP (with no bottleneck) the number of multiplications is  $J^2(L-1)$ . The bottleneck layer has fewer multiplications for  $I < J/2$ .

(b) We now consider introducing a multiplicative constant  $\gamma$  into the residual connection.

$$\begin{aligned} b[\ell, I] &= \text{ReLU}(W^b[I, J]h[\ell, J]) \\ h[\ell + 1, J] &= \gamma(h[\ell, J] + \text{ReLU}(W^h[\ell, J, I]b[\ell, I])) \end{aligned}$$

If the network is initialized such that each of  $h[\ell, j]$  and  $\text{ReLU}(W^h[\ell, j, I]b[\ell, I])$  are zero mean and unit variance, and are assumed to be independent, what value of  $\gamma$  gives that  $h[\ell + 1, j]$  has zero mean and unit variance.

**Solution:**  $1/\sqrt{2}$

(c) The main advantage of a stack of residual connections is that there is direct additive path from the loss to each layer of the stack, including the input layer. Give a reason why the introduction of the constant  $\gamma < 1$  as in part (b) might be damaging to the optimization of the lower layers of the residual stack.

**Solution:** When we introduce  $\gamma < 1$  as in (b) the gradient update on the bottom layer is reduced by  $\gamma^{L-2}$ . This could harm the learning along the direct connection between the loss and the first layer of the network.

**Problem 4:** Images have translation invariance — a person detector must look for people at various places in the image. Translation invariance is the motivation for convolution — all places in the image are treated the same.

Images also have some degree of scale invariance — a person detector must look for people of different sizes (near the camera or far from the camera). We would like to design a deep architecture that treats all scales (sizes) the same just as CNNs treat all places the same.

Consider a batch of images  $I[b, x, y, c]$  where  $c$  ranges over the three color values red, green, blue. We start by constructing an “image pyramid”  $I_s[x, y, c]$ . We assume that the original image  $I[b, x, y, c]$  has spatial dimensions  $2^k$  and construct images  $I_s[b, x, y, c]$  with spatial dimensions  $2^{k-s}$  for  $0 \leq s \leq k$ . The image pyramid  $I_s[b, x, y, c]$  for  $0 \leq s \leq k$  is defined by the following equations.

$$\begin{aligned} I_0[b, x, y, c] &= I[b, x, y, c] \\ I_{s+1}[b, x, y, c] &= \frac{1}{4} \left( I_s[b, 2x, 2y, c] + I_s[b, 2x+1, 2y, c] \right. \\ &\quad \left. + I_s[b, 2x, 2y+1, c] + I_s[b, 2x+1, 2y+1, c] \right) \end{aligned}$$

We want to compute a set of layers  $L_{\ell, s}[b, x, y, c]$  where  $s$  is the scale and  $\ell$  is the level of processing with  $\ell + s \leq k$  and where  $L_{\ell, s}[b, x, y, c]$  has spatial dimensions

$2^{k-\ell-s}$  (increasing either the processing level or the scale reduces the spatial dimensions by a factor of 2). First we set

$$L_{0,s}[b, x, y, c] = I_s[b, x, y, c].$$

Give an equation for a linear threshold unit to compute  $L_{\ell+1,s}[b, x, y, j]$  from  $L_{\ell,s}[b, x, y, j]$  **and**  $L_{\ell,s+1}[b, x, y, j]$ . Use parameters  $W_{\ell+1,\leftarrow}[\Delta x, \Delta y, i, j]$  for the dependence of  $L_{\ell+1,s}$  on  $L_{\ell,s+1}$  and parameters  $W_{\ell+1,\uparrow}[\Delta x, \Delta y, i, j]$  for the dependence of  $L_{\ell+1,s}$  on  $L_{\ell,s}$ . Use  $B_{\ell+1}[j]$  for the threshold. Note that these parameters do not depend on  $s$  — they are scale invariant.

**Solution:**

$$L_{\ell+1,s}[b, x, y, j] = \sigma \left( \begin{array}{l} \sum_{\Delta x, \Delta y, i} W_{\ell+1,\leftarrow}[\Delta x, \Delta y, i, j] L_{\ell,s+1}[b, x + \Delta x, y + \Delta y, i, j] \\ + \sum_{\Delta x, \Delta y, i} W_{\ell+1,\uparrow}[\Delta x, \Delta y, i, j] L_{\ell,s}[b, 2x + \Delta x, 2y + \Delta y, i, j] \\ + B_{\ell+1}[j] \end{array} \right)$$

Note: I am not aware of this architecture in the literature. A somewhat related architecture is “Feature Pyramid Networks” arxiv 1612.03144.