CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 (Linear Algebra) Problem 1

Read: A guide to convolution arithmetic for deep learning.

Solution: Complete ■

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 (Linear Algebra) Problem 2

(a) What is the arithmetic intensity for matrix matrix multiplication with sizes M, N, K.

Solution:

$$\texttt{Compute} = MNK \tag{1}$$

$$\texttt{Spatial} = KN + MK + MN \tag{2}$$

When multiplying MxK and KxN matrices we will produce a MxN output matrix where each element was the result of a linear combination of K elements. As such, we must compute M*N sums of K elements. Similarly, for memory operations we must bring in the original matrices and output the resulting matrix, leading to a sum over the dimensions of these matrices. ■

(b) Prove that arithmetic intensity for matrix matrix multiplication is maximized when $M = N = K$.

Proof: Choose N such that it is the largest matrix dimension. Let $I(M, N, K)$ designate the total arithmetic intensity. We can express M, K in terms of a factor of N.

$$M = C_m N \qquad K = C_k N \qquad C_m, C_n \in \mathbb{R}^+ \tag{3}$$

Using the result of part a we can express the total arithmetic intensity as a sum of the compute and memory components.

$$I(M, N, K) = MNK + (KN + MK + MN) \tag{4}$$

$$= C_m C_k N^3 + C_m C_k N^2 + C_m N^2 + C_k N^2 \tag{5}$$

Since N was chosen to be the largest of the terms, the coefficients C_m, C_k must be on the interval

$$0 < C_m, C_k \leq 1 \tag{6}$$

From the form of $I(M, N, K)$ given above and the interval on which C_m, C_k fall, it is clear that arithmetic intensity will be maximized for

$$C_m = 1 \qquad C_k = 1 \tag{7}$$

So

$$M = C_m N = N \qquad K = C_k N = N \tag{8}$$

$$\therefore M = N = K \tag{9}$$

□

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 (Linear Algebra) Problem 3

What is the complexity (MACs and memory) of a $N_o = N_i$ dense layer applied to vectorized versions of the following inputs:

(a) MNIST $1 \times 28 \times 28$

Solution:

$$\texttt{Compute} = (1 * 28 * 28)^2 = 6.15\,\text{E}^6 \tag{10}$$

$$\texttt{Spatial} = (1 * 28 * 28)^2 + 2(1 * 28 * 28) = 6.16\,\text{E}\,6 \tag{11}$$

■

(b) CIFAR $3 \times 32 \times 32$

Solution:

$$\texttt{Compute} = (3 * 32 * 32)^2 = 9.44\,\text{E}\,6 \tag{12}$$

$$\texttt{Spatial} = (3 * 32 * 32)^2 + 2(3 * 32 * 32) = 9.44\,\text{E}\,6 \tag{13}$$

■

(c) ImageNet $3 \times 224 \times 224$

Solution:

$$\texttt{Compute} = (3 * 224 * 224)^2 = 2.27\,\text{E}\,10 \tag{14}$$

$$\texttt{Spatial} = (3 * 224 * 224)^2 + 2(3 * 224 * 224) = 2.27\,\text{E}\,10 \tag{15}$$

■

(d) Quasi 1/4 HD $3 \times 512 \times 1024$

Solution:

$$\texttt{Compute} = (3 * 512 * 1024)^2 = 2.47\,\text{E}\,12 \tag{16}$$

$$\texttt{Spatial} = (3 * 512 * 1024)^2 + 2(3 * 512 * 1024) = 2.47\,\text{E}\,12 \tag{17}$$

■

(e) Quasi HD $3 \times 1024 \times 2048$

Solution:

$$\texttt{Compute} = (3 * 1024 * 2048)^2 = 3.96\,\text{E}\,13 \tag{18}$$

$$\texttt{Spatial} = (3 * 1024 * 2048)^2 + 2(3 * 1024 * 2048) = 3.96\,\text{E}\,13 \tag{19}$$

■

Vectorizing our input (using L_d to denote feature map depth) gives us

$$N_i = L_d \times L_r \times L_c \tag{20}$$

The dense layer transforms an input to an output vector of the same length. As such, our transformation matrix will be N_i^2. We are then left with matrix vector multiplication with the following dimensions under BLAS notation.

$$M = N_i \qquad K = N_i \qquad N = 1 \tag{21}$$

$$M = L_d \times L_r \times L_c \qquad K = L_d \times L_r \times L_c \qquad N = 1 \tag{22}$$

From the result of Problem 2 we have

$$\texttt{Compute} = MNK \qquad \texttt{Spatial} = KN + MK + MN \tag{23}$$

$$= \left(L_d \times L_r \times L_c\right)^2 \qquad = \left(L_d \times L_r \times L_c\right)^2 + 2 * \left(L_d \times L_r \times L_c\right) \tag{24}$$

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 (Linear Algebra) Problem 4

In practice, why can't you flatten a quasi HD input image to a vector for input to a dense layer

Solution: The results of Problem 3 illustrate how arithmetic intensity for vectorized image inputs to a dense layer grows rapidly, reaching tens of trillions of operations. This level of arithmetic intensity for large inputs is impractical on today's hardware. Furthermore, the use of matrix vector multiplication (as opposed to matrix matrix for CNN style 2D convolution layers) creates a constant factor ratio of compute to memory intensity. This leads to a memory wall, as memory movement operations are much slower than compute operations. It is also worth noting that dense layers acting on a vectorized 3D input will have no information about the topology of the true input, and will be forced to look for relationships between features that are separated by a great distance in the original input. ■

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 (Linear Algebra) Problem 5

Can a dense layer trained on an input of 1024×1 be applied to an input of size 2048×1 or 512×1

Solution: No to both. Dense layers learn weights such that a component in the output vector has a variable dependency on each of items in the input vector. The alterations needed to accommodate inputs of varying size would likely reduce the validity of the layer's output. ■

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 (Linear Algebra) Problem 6

Prove that CNN style 2D convolution with $N_o \times N_i \times F_r \times F_c$ filter, $N_i \times L_r \times L_c$ input and $N_o \times M_r \times M_c$ output can be lowered to the sum of $(F_r * F_c)$ matrix matrix multiplications.

Proof: Let the CNN style 2D convolution layer be of the usual form

$$\boldsymbol{Y}^{3D} = \boldsymbol{H}^{4D}\boldsymbol{X}^{3D} + \boldsymbol{V}^{3D} \tag{25}$$

If we consider a single output point $m_r \in M_r, m_c \in M_c$ of a single output feature map $n_o \in N_o$ we can express this as

$$Y(n_o, m_r, m_c) = \sum_{n_i}^{N_i} \sum_{f_r}^{F_r} \sum_{f_c}^{F_c} H(n_o, n_i, f_r, f_c)\, X(n_i, m_r + f_r, m_c + f_c) \tag{26}$$

If we expand the summation over n_i only we get a vector vector multiplication as follows

$$Y(n_o, m_r, m_c) = \sum_{f_r}^{F_r} \sum_{f_c}^{F_c} \begin{bmatrix} H(n_o, 0, f_r, f_c) \\ H(n_o, 1, f_r, f_c) \\ \cdots \\ H(n_o, N_i, f_r, f_c) \end{bmatrix}^T \begin{bmatrix} X(0, f_r, f_c) \\ X(1, f_r, f_c) \\ \cdots \\ X(N_i, f_r, f_c) \end{bmatrix} \tag{27}$$

We must next generalize this to all output points of all feature maps. If we extend row vector H into a matrix by adding N_o rows for $n_o \in N_o$, and if we extend column vector X into a matrix by adding $M_r \times M_c$ columns for the Cartesian product combinations of m_r, m_c, we get the following

$$Y = \sum_{f_r}^{F_r} \sum_{f_c}^{F_c} \underbrace{\begin{bmatrix} H(0,0,f_r,f_c) & \cdots & H(0,N_i,f_r,f_c) \\ \vdots & \ddots & \vdots \\ H(N_o,0,f_r,f_c) & \cdots & H(N_o,N_i,f_r,f_c) \end{bmatrix}}_{N_o \times N_i} \underbrace{\begin{bmatrix} X(0,f_r,f_c) & \cdots & X(0,f_r+M_r,f_c+M_c) \\ \vdots & \ddots & \vdots \\ X(N_i,f_r,f_c) & \cdots & X(N_i,f_r+M_r,f_c+M_c) \end{bmatrix}}_{N_i \times M_r * M_C} \tag{28}$$

From here we need only trivially reshape Y from a 2D matrix into a $3D$ tensor. The expression above indicates that CNN style 2D convolution can be lowered to a sum of $F_r * F_c$ matrix matrix multiplications. Furthermore, this result suggests that there are many alternative lowerings possible where each is determined by the which summations are expanded.

□

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)
Homework 2 (Linear Algebra) Problem 7

How many MACs are required to compute each output point in a CNN style 2D convolution layer $N_o \times N_i \times F_r \times F_c$

Solution: In an intermediate step of Problem 6 we expressed a single output point as a summation as follows

$$Y(n_o, m_r, m_c) = \sum_{n_i}^{N_i} \sum_{f_r}^{F_r} \sum_{f_c}^{F_c} H(n_o, n_i, f_r, f_c)\, X(n_i, m_r + f_r, m_c + f_c) \tag{29}$$

Recall that this summation can be expanded to an inner product with vectors of length $N_i * F_r * F_c$. We know trivially that the number of MACs for this operation will be given by

$$\texttt{Compute}_1 = F_r * F_c * N_i \tag{30}$$

■

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)
Homework 2 (Linear Algebra) Problem 8

How does CNN style 2D convolution complexity (MACs and memory) scale as a function of

(a) Product of image rows and columns ($L_r * L_c$)

Solution:

$$\texttt{Compute} \rightarrow \texttt{Linearly} \tag{31}$$
$$\texttt{Memory} \rightarrow \texttt{Linearly} \tag{32}$$
$$\tag{33}$$

■

(b) Product of filter rows and columns ($F_r * F_c$)

Solution:

$$\texttt{Compute} \rightarrow \texttt{Linearly} \tag{34}$$
$$\texttt{Memory} \rightarrow \texttt{Linearly} \tag{35}$$
$$\tag{36}$$

■

(c) Product of input and output feature maps ($N_o * N_i$)

Solution:

$$\texttt{Compute} \rightarrow \texttt{Linearly} \tag{37}$$
$$\texttt{Memory} \rightarrow \texttt{Linearly} \tag{38}$$
$$\tag{39}$$

■

From previous problems we know that MACs are determined by

$$\texttt{Compute} = N_i * F_r * F_c * N_o * M_r * M_c \tag{40}$$

and memory is determined by

$$\texttt{Memory} = N_i L_r L_c + N_o M_r M_c + N_i N_o F_r F_c \tag{41}$$

Both of these

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 (Linear Algebra) Problem 9

Consider a CNN style 2D convolution layer with filter size $N_o \times N_i \times F_r \times F_c$.

(a) How many padding 0s such that output feature map is the same size as input?

Solution: Padding to maintain size across the CNN style 2D convolution layer is given by

$$P_l + P_r = F_c - 1 \qquad P_t + P_b = F_r - 1 \tag{42}$$

This can easily be confirmed by realizing that

$$M_r = (L_r + P_r) - F_r + 1 \tag{43}$$
$$\tag{44}$$

So total padding along rows is given by

$$M_r = L_r \implies L_r = L_r + P_r - F_r + 1 \tag{45}$$
$$\implies P_r = F_r - 1 \tag{46}$$

■

(b) What is the size of the border of 0s for $F_r = F_c = 1$?

Solution: 0

There is no need for padding since a 1×1 filter will map each point in the input feature maps to a point in an output feature map. ■

(c) What is the size of the border of 0s for $F_r = F_c = 3$?

Solution:

$$P_l + P_r = 3 - 1 \qquad P_t + P_b = 3 - 1 \tag{47}$$
$$P_l = P_r = 1 \qquad P_t = P_b = 1 \tag{48}$$
$$\tag{49}$$

■

(d) What is the size of the border of 0s for $F_r = F_c = 5$?

Solution:

$$P_l + P_r = 5 - 1 \qquad P_t + P_b = 5 - 1 \tag{50}$$
$$P_l = P_r = 2 \qquad P_t = P_b = 2 \tag{51}$$
$$\tag{52}$$

■

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 (Linear Algebra) Problem 10

Consider a CNN style 2D convolution layer with filter size $N_o \times N_i \times F_r \times F_c$, input $N_i \times L_r \times L_c$, $P_r = F_r - 1$ and $P_c = F_c - 1$.

(a) What is the size of the output feature map with striding $S_r = S_c = 1$

Solution:

$$M_r = L_r \qquad\qquad M_c = L_c \tag{53}$$

Using a modified form of Relationship 6 in the reading we can write

$$M_r = \left\lfloor \frac{L_r + P_r - F_r}{S_r} \right\rfloor + 1 \tag{54}$$

$$= \left\lfloor \frac{L_r + (F_r - 1) - F_r}{S_r} \right\rfloor + 1 \tag{55}$$

$$= \left\lfloor \frac{L_r - 1}{S_r} \right\rfloor + 1 \tag{56}$$

Symmetry in the given input values allows this result to be extended to M_c. For a stride of one and padding given by filter dimension minus one, we get the expected result of $M_r = L_r$

■

(b) What is the size of the output feature map with striding $S_r = S_c = 2$

Solution:

$$M_r = \left\lfloor \frac{L_r - 1}{2} \right\rfloor + 1 \qquad\qquad M_c = \left\lfloor \frac{L_c - 1}{2} \right\rfloor + 1 \tag{57}$$

We use the result of the previous part with $S_r = S_c = 2$

■

(c) How does this change the shape of the equivalent lowered matrix equation

Solution: In the equivalent lowered matrix matrix operation, striding alters the input matrix of shape $N_i \times (M_r * M_c)$. Values M_r, M_c are divided by S_r, S_c respectively. This in turn alters the shape of the output matrix in a similar way. Intuitively, striding is a downsampling operation where the input feature map is roughly downsampled by a factor of S_r, S_c along L_r, L_c respectively. ■

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 (Linear Algebra) Problem 11

Can a CNN style 2D convolution layer trained on an input of size $3 \times 1024 \times 2048$ be applied to an input of size:

(a) $3 \times 512 \times 1024$

Solution: Yes, there are upsampling techniques like deconvolution or interpolation that could be used.

This is the strength of CNN style 2D convolution layers. Since each output point is determined by input points exposed to the filter kernel at a given position, there will be no dependencies between distant points in the input feature map. Upsampling and downsampling can be applied to an input without compromising the validity of outputs from the CNN layer. ■

(b) $3 \times 512 \times 512$

Solution: Again, upsampling techniques could be used, however care must be taken since $U_r \neq U_c$. ■

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 (Linear Algebra) Problem 12

In a standard RNN, if the state update matrix is constrained to a diagonal, what does this do for the mixing of the previous state with new inputs.

Solution: It forces an output element y_i to depend only on the value of y_i at time $t-1$, rather than all outputs at time $t-1$.

An RNN layer is given by

$$\boldsymbol{y}_t = \boldsymbol{HX} + \boldsymbol{G}\boldsymbol{y}_{t-1} + \boldsymbol{V} \tag{58}$$

The state update matrix $\boldsymbol{G}$ acts on vector $\boldsymbol{y}_{t-1}$ which was produced by the previous RNN state. For non-diagonal $\boldsymbol{G}$ this will produce a vector of length N where each element is a linear combination of all outputs from the previous state, as follows

$$\boldsymbol{G}\boldsymbol{y}_{t-1} = \begin{pmatrix} G_{00}y_0^{t-1} + G_{01}y_1^{t-1} + \cdots + G_{0N_o}y_{N_o}^{t-1} \\ G_{10}y_0^{t-1} + G_{11}y_1^{t-1} + \cdots + G_{0N_o}y_{N_o}^{t-1} \\ \vdots \\ G_{N_o0}y_0^{t-1} + G_{N_o1}y_1^{t-1} + \cdots + G_{N_oN_o}y_{N_o}^{t-1} \end{pmatrix} \tag{59}$$

Constraining $\boldsymbol{G}$ to the diagonal means that a given output $y_i \in \boldsymbol{y}$ will be determined only be a scaled factor of $y_i \in \boldsymbol{y}_{t-1}$ as follows

$$\boldsymbol{G}\boldsymbol{y}_{t-1} = \begin{pmatrix} G_{00}y_0^{t-1} \\ G_{11}y_1^{t-1} \\ \vdots \\ G_{N_oN_o}y_{N_o}^{t-1} \end{pmatrix} \tag{60}$$

It is clear that in the addition of column vectors there will be no interplay between different output features. ■

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 (Linear Algebra) Problem 13

The size of the input to the global average pooling layer is $1024 \times 16 \times 32$.

(a) What is the size of the output

Solution: 1024×1

The global average pooling layer computes a global average for each of the input feature maps. As such, we will average feature maps of size 16×32 to a single value, repeated for each of the 1024 given feature maps. ■

(b) What is the complexity (MACs) of the layer

Solution:

$$\begin{aligned} \texttt{Compute} &= N_i * L_r * L_c && (61)\\ &= 1024 * 16 * 32 && (62)\\ &= 524288 && (63)\\ & && (64) \end{aligned}$$

Global average pooling can be thought of as a matrix vector multiplication with the following BLAS dimensions

$$M = N_i \qquad K = L_r * L_c \qquad N = 1 \tag{65}$$

where the $K \times N$ matrix has each value set to $1/(L_r * L_c)$ for averaging. Based on this interpretation we can determine the complexity of the layer.

■