CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 Problem 1

Read: A guide to convolution arithmetic for deep learning.

Solution: Complete ■

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)

Homework 2 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 multplication 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 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:

$$\texttt{Compute} = (1 * 28 * 28)^2 = 6.15\,\mathrm{E}^6 \tag{10}$$
$$\texttt{Spatial} = (1 * 28 * 28)^2 + 2(1 * 28 * 28) = 6.16\,\mathrm{E}\,6 \tag{11}$$

■

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

Solution:

$$\texttt{Compute} = (3 * 32 * 32)^2 = 9.44\,\mathrm{E}\,6 \tag{12}$$
$$\texttt{Spatial} = (3 * 32 * 32)^2 + 2(3 * 32 * 32) = 9.44\,\mathrm{E}\,6 \tag{13}$$

■

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

Solution:

$$\texttt{Compute} = (3 * 224 * 224)^2 = 2.27\,\mathrm{E}\,10 \tag{14}$$
$$\texttt{Spatial} = (3 * 224 * 224)^2 + 2(3 * 224 * 224) = 2.27\,\mathrm{E}\,10 \tag{15}$$

■

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

Solution:

$$\texttt{Compute} = (3 * 512 * 1024)^2 = 2.47\,\mathrm{E}\,12 \tag{16}$$
$$\texttt{Spatial} = (3 * 512 * 1024)^2 + 2(3 * 512 * 1024) = 2.47\,\mathrm{E}\,12 \tag{17}$$

■

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

Solution:

$$\texttt{Compute} = (3 * 1024 * 2048)^2 = 3.96\,\mathrm{E}\,13 \tag{18}$$
$$\texttt{Spatial} = (3 * 1024 * 2048)^2 + 2(3 * 1024 * 2048) = 3.96\,\mathrm{E}\,13 \tag{19}$$

■

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 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. 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. ∎

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)
Homework 2 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 accomodate 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 Problem 6

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

Proof: Let this $\square$

CS 6301.503 Spring 2019 — Scott C. Waggener (scw180000)

Homework 2 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: From the result of the previous problem we know that the CNN style 2D convolution layer can be lowered to the sum of $F_r * F_c$ matrix matrix between matrices of the following dimensions in BLAS notation.

$$M = N_o \qquad K = N_i \qquad N = M_r \times M_c \tag{25}$$

This gives the total number of MACs for all output points as

$$\begin{aligned} \texttt{Compute} &= (F_r * F_c) * M * N * K && (26)\\ &= F_r * F_c * N_o * N_i * M_r * M_c && (27)\\ & && (28) \end{aligned}$$

For a single output point we discard factors N_o, M_r, M_c since we are considering a single output point in one output feature map. This is similar to conducting $F_r * F_c$ vector vector multiplications where the vectors are of length N_i. In either case the number of MACs for a single output point is given by

$$\texttt{Compute}_1 = F_r * F_c * N_i \tag{29}$$

This result makes intuitive sense. To compute a single output point we multiply each of $F_r * F_c$ filter weights by the the input values across N_i feature maps and acculmulate to a single output point of a single output feature map.

■

CS 6301.503 Spring 2019	Scott C. Waggener (scw180000)
Homework 2 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:

$$\texttt{Compute} \rightarrow \texttt{Linearly} \tag{30}$$
$$\texttt{Memory} \rightarrow \texttt{Linearly} \tag{31}$$
$$\tag{32}$$

■

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

Solution:

$$\texttt{Compute} \rightarrow \texttt{Linearly} \tag{33}$$
$$\texttt{Memory} \rightarrow \texttt{Linearly} \tag{34}$$
$$\tag{35}$$

■

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

Solution:

$$\texttt{Compute} \rightarrow \texttt{Linearly} \tag{36}$$
$$\texttt{Memory} \rightarrow \texttt{Linearly} \tag{37}$$
$$\tag{38}$$

■

We have already established the number of MACs in a CNN style 2D convolution layer which clearly scales linearly for all the given alterations. Memory movement for the same layer is given by

$$N_i L_r L_c + N_o M_r M_c + N_i N_o F_r F_c \tag{39}$$

Which again scales linearly for all of the given inputs

CS 6301.503 Spring 2019 **Homework 2 Problem 9**	Scott C. Waggener (scw180000)

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

$$P_l + P_r = F_c - 1 \qquad P_t + P_b = F_r - 1 \tag{40}$$

■

(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:

$$P_l + P_r = 3 - 1 \qquad P_t + P_b = 3 - 1 \tag{41}$$

$$P_l = P_r = 1 \qquad P_t = P_b = 1 \tag{42}$$

(43)

■

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

Solution:

$$P_l + P_r = 5 - 1 \qquad P_t + P_b = 5 - 1 \tag{44}$$

$$P_l = P_r = 2 \qquad P_t = P_b = 2 \tag{45}$$

(46)

■

CS 6301.503 Spring 2019 — Scott C. Waggener (scw180000)

Homework 2 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: $M_r \times M_c = (L_r - F_r + 1) \times (L_c - F_c + 1)$ ■

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

Solution: $\frac{M_r}{2} \times \frac{M_c}{2} = (L_r - F_r + 1) \times (L_c - F_c + 1)$ ■

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

Solution: In the equivalent lowered matrix matrix operation, striding alters the matrix of shape $N_i \times (M_r * M_c)$, dividing M_r, M_c by S_r, S_c respectively. ■

CS 6301.503 Spring 2019 Scott C. Waggener (scw180000)
Homework 2 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. ■

(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 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 its value in the previous state.

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. 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 $y_i \in \boldsymbol{y}_{t-1}$. ■

CS 6301.503 Spring 2019	Scott C. Waggener (scw180000)
Homework 2 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 reduce 1024 feature maps of size 16×32 to an average of those 16×32 features for each feature map. ■

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

Solution:

$$\begin{aligned}
\texttt{Compute} &= N_i * L_r * L_c && (47)\\
&= 1024 * 16 * 32 && (48)\\
&= 524288 && (49)\\
& && (50)
\end{aligned}$$

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

$$M = N_i \qquad K = L_r * L_c \qquad N = 1 \tag{51}$$

We can then compute the familiar MAC count for matrix matrix (really matrix vector in this case) multiplication. ■