week-14

JX-Ma

2024/11/2

INTORDUCE

This week, I mainly completed the description of FFT and the flops calculation part of Winograd.

Winograd

The Winograd algorithm was first mentioned in the article 'Arithmetic Complexity of Computations' in 1980,
Later, at the CVPR2016 conference, Andraw Lavin et al. proposed using the Winograd algorithm to accelerate convolution and published a paper called Fast Algorithms for Convolutional Neural NetWorks, which became popular. Winograd mainly achieves the goal of accelerating convolution by reducing multiplication floating-point calculations.

In 1D convolution, we assume that the input tensor contains 4 elements $d = (d_0, d_1, d_2, d_3)$ and the filter tensor contains 3 elements $g = (g_0, g_1, g_2)$.

$$d \cdot g = \begin{bmatrix} m_1 + m_2 + m_3 \\ m_2 - m_3 - m_4 \end{bmatrix}$$

The following are the values of each m, These have been demonstrated in the paper on arithmetic complexity of computations.

$$m_1 = (d_0 - d_2)g_0 \qquad m_2 = (d_1 + d_2)\frac{g_0 + g_1 + g_2}{2} \tag{1}$$

$$m_4 = (d_1 - d_3)g_2 \qquad m_3 = (d_2 - d_1)\frac{g_0 - g_1 + g_2}{2} \tag{2}$$

We can use the following algorithm to describe the calculation of Winograd:

$$Y = A^T[(Gg) \odot (B^T d)]$$

The G is filter tensor transformation matrix, The B^T is input tensor transformation matrix. A^T is output transformation matrix.

flops of winograd We can use $\mu(F(m, r)) = m + r - 1$ to calculate the number of multiplications using Winograd

m is size of out tensor and r is size of filter tensor.(Winograd)

For 2D matrix multiplication, assuming the output tensor size is m x n and the convolution size is r x s, we can obtain the multiplication

$\mu(F(m \times n, r \times s)) = \mu(F(m, r))\mu(F(n, s)) = (m + r - 1)(n + s - 1)$

For 4D, assuming the output tensor size is N_O, C_O, H_O, W_O and the filter tensor is N_F, C_F, H_F, W_F,

The required number of multiplications to obtain $H_O \times W_O$ output tensor elements is $F(H_O \times W_O, H_F \times W_F) \times C_F$, so, The total number of multiplications is:

$$flops_{mul} = N_O \times C_O \times C_F \times (H_O + H_F - 1) \times (W_O + W_F - 1)$$

The number of additions is the same as the addition required for ordinary matrix multiplication convolution

$$flops_{add} = N_O \times C_O \times H_O \times W_O \times C_F \times H_F \times W_F$$

(cudnn Winograd winograd NONFUSED,)

FFT

Fast Fourier Transform (FFT) is an efficient algorithm for computing Discrete Fourier Transform (DFT) and its reverse, The Fourier transform transforms a time domain signal into a frequency domain signal, allowing us to analyse the frequency components of the signal.

DFT is defined by the following equation

$$X(k) = \sum_{n=0}^{N-1} x(n) \times W_N^{nk} = \sum_{n=0}^{N-1} x(n) \cdot e^{-\frac{2\pi i}{N} nk}$$

where x is the input sequence of n complex numbers, X is the corresponding output sequence, N is the length of the data, the $W_N^{nk} = e^{-\frac{2\pi i}{N} nk}$ is called the rotation factor. The FFT is the use of the properties of symmetry, periodicity, decomposition and recursive computation of W_N^{nk}, which significantly reduces the memory and computational complexity required for the Fourier transform.

Expanding the above DFT transform in detail as shown below:

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{bmatrix} = \cdot \begin{bmatrix} W_N^0 & W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & W_N^2 & \cdots & W_N^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W_N^0 & W_N^{n-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}$$

where X represents the signal in the frequency domain and x is the data in the time domain. If we need to transform the signal from the frequency domain to the time domain, we need the data in the frequency domain to be left into a rotation factor inverse matrix. So for FFT transform and inverse FFT transform, they have equal number of floating point calculations.

(FFT)

In FFT-based convolutional computation, we first need to transform the input tensor and the filter tensor from the time domain to the frequency domain through the FFT, then multiply the result of the conversion to the frequency domain element by element to obtain the value of the output tensor in the frequency domain, and finally convert the result of the output tensor in the frequency domain to the result in the time domain to obtain the final value through the FFT inverse transformation.

For a 1D FFT of length N and execution time t seconds, we can compute Gflops using the formula proposed by the well-known benchmark (benchFFT)[Frigo and SG Johnson. 2019. benchFFT. http://www.fftw.org/benchfft.]

$$Gflops = \frac{5N \cdot log_2 N \cdot 10^{-9}}{t}$$

flops of FFT For the FFT1D convolution operation, we assume that the number of elements of the input tensor is m, the number of elements of the filter tensor is n. To ensure that every element of the input tensor is used, we first need to populate the input tensor and the filter tensor, The number of input tensor and filter tensor elements after padding are both (m + n - 1). The size of the output tensor is m+n-1 So for the number of flops calculations required for FFT transform and inverse FFT transform can be obtained using the above equation.

$$flops_{fft} = flops_I + flops_F + flops_O = 3 * 5(m + n - 1) * log_2(m + n - 1)$$

The number of multiplications required to multiply the input tensor and the frequency domain is (m + n -1), Note that the number of multiplications referred to at this point is complex multiplication,When the calculator stores complex numbers, the real and imaginary parts are stored separately, so a complex multiplication $(a+bi)(c+di)$requires four floating-point multiplications and two floating-point additions.

Therefore, the number of flops calculations required for frequency domain multiplication is $6(m+n-1)$

For the FFT2D convolution operation, Assuming that the size of the input tensor is NxN and the size of the filter tensor is SXS, we first need to fill the size of the input tensor and the filter tensor to (N+S-1)x (N+S-1), and the number of floating-point computations required for the FFT transform is $3*5(N+S-1)^2*log_2(N+S-1)^2$. the number of flops calculations required for frequency domain multiplication is $6(m+n-1)^2$

For the FFT4D convolution operation, assuming the input tensor size is N_I, C_I, H_I, W_I and the filter tensor is N_F, C_F, H_F, W_F, We first need to fill the width and height of the input tensor and the filter tensor by filling them as $(H_I+H_F-1)\times(W_I+W_F-1)$, At this point the number of floating points required to perform the FFT transform of the input tensor is :

$5((H_I+H_F-1)\times(W_I+W_F-1)\times C_I\times N_I)\times log_2((H_I+H_F-1)\times(W_I+W_F-1)\times C_I\times N_I)$

The number of flops calculations required to perform the FFT of the filter tensor is:

$5((H_I+H_F-1)\times(W_I+W_F-1)\times C_F\times N_F)\times log_2((H_I+H_F-1)\times(W_I+W_F-1)\times C_F\times N_F)$

The number of flops computations required to perform the FFT inverse transformation of the output tensor is:

$5((H_I+H_F-1)\times(W_I+W_F-1)\times C_O\times N_O)\times log_2((H_I+H_F-1)\times(W_I+W_F-1)\times C_O\times N_O)$

Next consider the number of flops computations required to compute the $W_O\times H_O$ elements of the output tensor,Firstly, we need to multiply the elements of the different channels of the input tensor and the filter tensor after the FFT transform element by element, and finally add the corresponding elements of each channel to get the result of the output tensor in the frequency domain.

The number of complex multiplications we need at this point is $(H_I+H_F-1)\times(W_I+W_F-1)\times C_F$

The number of complex additions is : $(H_I+H_F-1)\times(W_I+W_F-1)\times(C_F-1)$

The total number of flops is $8\times(H_I+H_F-1)\times(W_I+W_F-1)\times C_F-2\times(H_I+H_F-1)\times(W_I+W_F-1)$.

Calculate flops for all output tensors is $(8\times(H_I+H_F-1)\times(W_I+W_F-1)\times C_F-2\times(H_I+H_F-1)\times(W_I+W_F-1))\times C_O\times N_O$