Title: Optimizing batched winograd convolution on GPUs

JX-Ma

2024/11/10

ABSTRACT

This article introduces two new fast Fourier transform convolution implementations, one based on NVIDIA's cuFFT library and the other based on Facebook's FFT implementation fbfft. Then we discussed different performance modes of convolution and compared how simple time-domain convolution outperforms Fourier frequency-domain convolution in certain fields.

INTRODUCE

This section introduces the content of each subsequent section and provides the location of the source code for this article. http://github.com/facebook.

CONVOLUTION

This section mainly introduces the forward propagation and directional propagation of convolution

- One convolutional optimization method is to expand the data until the computational form becomes large matrix multiplication.
- Another approach is to increase the step size and sacrifice accuracy to reduce computational costs
- Convolution based on FFT transform is also known as frequency domain convolution, corresponding to time-domain convolution implemented through direct computation

FFT CONVOLUTION DETAILS

This section mainly introduces the implementation of cuFFT convolution

- The complex part after FFT conversion has Hermitian symmetry, so the space for storing complex numbers is only half of the space for storing real

numbers, and the remaining half can be obtained using complex conjugation

- Before performing FFT conversion, 0 padding was also used because it is necessary for handling boundary conditions and requires all operands to be interpolated on the same Fourier basis. Padding also affects the actual FFT algorithm used and the floating-point operation count of non FFT operations.
- CuFFT implements the widely used Cooley Tukey algorithm, which utilizes recursive decomposition of trigonometric equations and reuses computations.
- Zero padding is applied to the input tensor and filter tensor to perform FFT on any larger size, which may be more efficient.
- The cuBLAS library provides different implementations for batch processing and single operation modes
  - For larger batches of small matrices, use the`cublasCgemmBatched` library to call;
  - For smaller batches of large matrices, call multiple `cublasCgemm`from the host
  - For medium batches and matrix sizes, devices with computing power of 3.5 or above support dynamic parallelism, allowing the CUDA kernel to boot other kernels. This is beneficial for initiating multiple small matrices.

CUFFT CONVOLUTION PERFORMANCE

Firstly, this paper compared the cuFFT convolution results with NVIDIA's cuDNN 1.0 library. Different batches of input tensors were set, with batch sizes ranging from 1-64 and convolution kernel widths and heights of 3x3, 5x5, 7x7, 9x9, 11x11, and 13x13. Through experimental results, it was found that FFT convolutions make large kernel sizes inexpensive. Subsequently, testing was conducted on the entire neural network and it was found that larger width and height of the input tensor or smaller convolution kernel would lead to a decrease in FFT performance. However, significant acceleration can be achieved when the convolution kernel width and height are 3x3.

Next is the implementation of fbfft, which heavily relies on the shuffle instruction.

When implementing frequency domain convolution, temporary memory overhead is a very important issue because frequency domain convolution requires the use of complex numbers, which requires a large amount of space to store the imaginary parts of the complex data.

CONCLUSION

In this article, two open-source implementation methods based on FFT convolution are presented. FFT convolution is fundamentally different from the convolution we usually implement. FFT convolution is a convolution performed in the frequency domain, while the convolution we usually directly calculate is a convolution performed in the time domain. A major drawback of FFT convolution is that it requires a large amount of temporary memory to store the imaginary part of complex numbers after FFT transformation. As mentioned in this article, due to the Hermitian symmetry of the matrix after FFT transformation, additional memory usage can be reduced based on these characteristics. After comparing different input tensors and filter tensors for cuFFT convolution, this article found that larger input tensor heights and widths, or smaller filter tensors, have a significant negative impact on FFT performance. However, when the height and width of the filter tensor are 3x3, it does exhibit good performance. Coincidentally, when we called the FFT algorithm in cuDNN, we found that its applicable condition is that the height and width of the filter tensor are 3x3.