Title: Study of cuBLAS document

JX-Ma

2024/11/16

INTRODUCE

The main reading material for this week is the Cublas document, and the cuBLAS library is available at NVIDIA CUDA Implementation of BLAS (Basic Linear Algebra Subroutine) above runtime. It allows users to access the computing resources of NVIDIA graphics processing units (GPUs).

This week, I mainly learned about the convolution algorithm in cuBLAS. The convolution algorithm in cuBLAS is mainly divided into vector X vector in Level 1, matrix X vector in Level 2, and matrix X matrix in Level 3. I focused on learning the matrix X matrix in Level 3.

cuBLAS Level-3 Function Reference

The matrix multiplication function in Level 3 is as follows:

cublasgemm

where T represents data type

The formula for matrix multiplication is as follows:

$$C = \alpha op(A) op(B) + \beta C$$

- A B is the input matrix C is the output matrix
- Op (A) represents the operation performed on matrix A
  - $op(A) = A$Do nothing transa==CUBLAS-OPNN
  - $op(A) = A^T$Transposition transa==CUBLAS-OP_T
  - $op(A) = \ hatA$complex conjugate transa==CUBLAS-OP_C
- $\alpha\beta$is a scalar that is the scaling factor of the input matrix and output matrix

Cublasgemm3m()

this function uses Gaussian complexity reduction algorithm to perform complex matrix multiplication. This can improve performance by 25%

cublasgemmBatched()

The formula for matrix multiplication is as follows:

$$C[i] = \alpha op(A[i])op(B[i]) + \beta(C[i])i = [0, batchCount - 1]$$

This function performs matrix multiplication on a batch of matrices. This batch is considered "uniform", meaning that all instances of matrices A, B, and C have the same dimensions (m, n, k), leading dimensions (lda, ldb, ldc), and transposes (transa, transb). Read the addresses of the input matrix and output matrix for each instance of batch processing from the pointer array passed to the function by the caller.

cublasgemmStridedBatched()

This function performs matrix multiplication on a batch of matrices. This batch is considered "uniform", meaning that all instances of matrices A, B, and C have the same dimensions (m, n, k), leading dimensions (lda, ldb, ldc), and transposes (transa, transb). The input matrices A, B, and output matrix C of each instance in batch processing have a fixed offset in the number of elements compared to their positions in the previous instance. The user passes pointers to the A, B, and C matrices of the first instance, as well as the offset of the number of elements, to the function, which determines the position of the input and output matrices in future instances.

$$C+i*strideC = \alpha op(A+i*strideA)op(B+i*strideB)+\beta(C+i*strideC)i = [batchCount-1]$$

cuBlas of function in pytorch

In PyTorch, there are three functions for matrix multiplication, namely cublasSgemm, cublasSgemmStrodedBatched, and cublasSgemmEx.

cublasSgemm

- function: Basic matrix multiplication for calculating single precision floating-point numbers
- operation: Perform a matrix multiplication operation (C= A B+ C).

- input data: Single batch matrix multiplication operation. The input matrices A, B, and C must be specified separately, and this function only processes matrix multiplication once.
- application: Suitable for single matrix multiplication calculations, without involving batch processing.

cublasSgemmStridedBatched

- function: Used for handling batch single precision floating-point matrix multiplication
- operation: Perform multiple matrix multiplications and calculate multiple independent matrix multiplication operations (Ci= Ai Bi+ Ci), where (i) and (i) represent the index of the batch.
- input data: Use block memory layout to represent the matrix of batches, allowing for setting stride to indicate memory offset between batches. Even if the input matrix is discontinuous, different batches of matrices can still be accessed using the `strip` parameter.
- application: Suitable for computing multiple matrix multiplications simultaneously, avoiding the overhead of video memory copying and function calling in batch processing scenarios, such as matrix operations for processing multiple samples in deep learning

cublasSgemmEx

- function: This is an extended version of `cublasSgemm`, providing more data type support and calculation accuracy options.
- operation: The operation of computing (C= A B+ C) (C= A B+ C), but provides support for different data types, such as `half` (half precision floating-point), `float`, and `int8`.
- input data: Allow `A`, `B`, and `C` to use different data types, for example, the input can be half precision `__half`, but the result is saved in single precision `float`. This feature is particularly important in scenarios where mixed precision computing and storage requirements are saved.
- application: Suitable for scenarios with special requirements for computational accuracy and storage, such as using mixed precision computing in deep learning to improve efficiency while ensuring result accuracy.