



Denglin HammingTM V2

dIBLAS Introduction

DL-DG/SW-041A-06

2025-02-25

Copyright©苏州登临科技有限公司，2019 - 2025，版权所有。

未经苏州登临科技有限公司事先书面同意，不得以任何形式或方式复制或传播本文件的任何部分。

商标和许可



和其它苏州登临科技有限公司的其它登临科技的图标为苏州登临科技有限公司的商标。本手册中提及的所有其他商标均为其各自所有者的财产。

通知

所购买的产品、服务和特性由苏州登临科技有限公司与客户签订的合同规定。本文件中描述的所有或部分产品、服务和特性可能不在采购范围或使用范围内。除非合同中另有规定，本文件中的所有声明、信息和建议均按“原样”提供，无任何明示或暗示的保证或陈述。

本手册中的信息如有更改，恕不另行通知。本文件在编制过程中已尽一切努力确保内容的准确性，本文件中的所有声明、信息和建议不构成任何明示或暗示的保证。

苏州登临科技有限公司

苏州工业园区扬富路11号南岸新地一期商务楼5号1101室，江苏，中国

<http://www.denglin.ai>

Email: support@denglin.ai

更新历史

| 版本 | 更新描述 |
|----|--|
| 06 | 新增 dBLAS ext API dlblasMatrixElementwise、dlblasMatrixTransform 和 dlblasGemmExV2。 |
| 05 | 新增 dBLAS ext API dlblasGemmMarlinEx。 |
| 04 | 修订 dBLAS ext API dlblasGemmSparseEx，新增 version update instruction module。 |
| 03 | 修订 dBLAS ext API 相关内容。 |
| 02 | 适配 CUDA 11。 |
| 01 | 第一次发布。 |

章节目录

- 1 dBLAS 简介
- 2 dBLAS 版本
- 3 dBLAS 已支持API
 - 3.1 Helper Function
 - 3.2 Level-1 Function
 - 3.3 Level-2 Function
 - 3.4 Level-3 Function
 - 3.5 BLAS-like Extension
- 4 dBLAS 扩展API
 - 4.1 当前版本
 - 4.1.1 当前版本已支持的API
 - 4.2 dBLAS 版本更新说明
 - 4.2.1 版本1.0.0
 - 4.2.2 版本1.1.0
 - 4.2.3 版本1.2.0
 - 4.2.4 版本1.3.0
 - 4.2.5 版本1.4.0
 - 4.2.6 版本1.5.0
 - 4.2.7 版本1.6.0
 - 4.3 dblasGemmEx
 - 4.3.1 量化类型与CUDA数据类型对应关系
 - 4.3.2 支持的类型组合
 - 4.3.3 int4 或者 uint4 量化时数据内存布局
 - 4.3.4 dblasGemmEx 计算代码示例
 - 4.4 dblasGemmStridedBatchedEx
 - 4.5 dblasGemmSparseEx
 - 4.6 dblasGemmMarlinEx
 - 4.7 dblasMatrixElementwise
 - 4.8 dblasMatrixTransform
 - 4.9 dblasGemmExV2

1 dIBLAS 简介

dIBLAS库是在CUDA运行时之上的BLAS（基本线性代数子程序）的实现。

2 dIBLAS 版本

dIBLAS 当前版本为1.0.0。

3 dIBLAS 已支持API

dIBLAS 支持的算子列表分为5个部分，分别为Helper Function， Level-1 Function， Level-2 Function， Level-3 Function， BLAS-like Extension。

每个部分API简要介绍如下：

Helper Function：一些辅助计算的API。

Level-1 Function：向量之间运算相关API，如向量求和，向量点乘等。

Level-2 Function：矩阵与向量之间运算相关API，如矩阵乘向量。

Level-3 Function：矩阵与矩阵之间运算相关API，如矩阵乘。

BLAS-like Extension：前面已有部分API的扩展，具有更高灵活性，可以通过cudaDataType，控制输入和输出类型。

3.1 Helper Function

| Helper Function |
|----------------------|
| cublasCreate |
| cublasDestroy |
| cublasGetVersion |
| cublasGetProperty |
| cublasSetStream |
| cublasSetWorkspace |
| cublasGetStream |
| cublasGetPointerMode |
| cublasSetPointerMode |
| cublasSetVector |

| Helper Function |
|----------------------|
| cublasGetVector |
| cublasSetMatrix |
| cublasGetMatrix |
| cublasSetVectorAsync |
| cublasGetVectorAsync |
| cublasSetMatrixAsync |
| cublasGetMatrixAsync |
| cublasSetAtomicsMode |
| cublasGetAtomicsMode |
| cublasSetMathMode |
| cublasGetMathMode |

3.2 Level-1 Function

| Level-1 Function |
|------------------|
| cublasIsamax |
| cublasIcamax |
| cublasIsamin |
| cublasIcamin |
| cublasSasum |
| cublasScasum |
| cublasSaxpy |
| cublasCaxpy |
| cublasScopy |
| cublasCcopy |
| cublasSdot |
| cublasCdotu |
| cublasCdotc |

| Level-1 Function |
|------------------|
| cublasSnrm2 |
| cublasScnrm2 |
| cublasSrot |
| cublasCrot |
| cublasCsrot |
| cublasSrotg |
| cublasCrotg |
| cublasSrotm |
| cublasSrotmg |
| cublasSscal |
| cublasCscal |
| cublasCsscal |
| cublasSswap |
| cublasCswap |

3.3 Level-2 Function

| Level-2 Function |
|------------------|
| cublasSgbmv |
| cublasCgbmv |
| cublasSgemv |
| cublasCgemv |
| cublasSger |
| cublasCgeru |
| cublasCgerc |
| cublasSsbmv |
| cublasSspmv |
| cublasSspr |

| Level-2 Function |
|--------------------|
| cublasSspr2 |
| cublasSsymv |
| cublasCsymv |
| cublasSsyr |
| cublasCsyr |
| cublasSsyr2 |
| cublasCsyr2 |
| cublasStbmv |
| cublasCtbmv |
| cublasStbsv |
| cublasCtbsv |
| cublasStpmv |
| cublasCtpmv |
| cublasStpsv |
| cublasCtpsv |
| cublasStrmv |
| cublasCtrmv |
| cublasStrsv |
| cublasCtrsv |
| cublasChemv |
| cublasChbmv |
| cublasChpmv |
| cublasCher |
| cublasCher2 |
| cublasChpr |
| cublasChpr2 |
| cublasSgemvBatched |
| cublasCgemvBatched |

| Level-2 Function |
|---------------------------|
| cublasSgemvStridedBatched |
| cublasCgemvStridedBatched |

3.4 Level-3 Function

| Level-3 Function |
|-----------------------------|
| cublasSgemm |
| cublasCgemm |
| cublasHgemm |
| cublasCgemm3m |
| cublasHgemmBatched |
| cublasSgemmBatched |
| cublasCgemmBatched |
| cublasHgemmStridedBatched |
| cublasSgemmStridedBatched |
| cublasCgemm3mStridedBatched |
| cublasSsymm |
| cublasCsymm |
| cublasSsyrk |
| cublasCsyrk |
| cublasSsyr2k |
| cublasCsyr2k |
| cublasSsyrkx |
| cublasCsyrkx |
| cublasStrmm |
| cublasCtrmm |
| cublasStrsm |
| cublasStrsmBatched |

| Level-3 Function |
|------------------|
| cublasChemmm |
| cublasCherk |
| cublasCher2k |
| cublasCherkx |

3.5 BLAS-like Extension

| BLAS-like Extension |
|----------------------|
| cublasSgeam |
| cublasCgeam |
| cublasSdgmm |
| cublasCdggmm |
| cublasSgetrfBatched |
| cublasCgetrfBatched |
| cublasSgetrsBatched |
| cublasCgetrsBatched |
| cublasSgetriBatched |
| cublasCgetriBatched |
| cublasSmatinvBatched |
| cublasSgeqrfBatched |
| cublasCgeqrfBatched |
| cublasSgelsBatched |
| cublasCgelsBatched |
| cublasStpttr |
| cublasCtpttr |
| cublasStrttp |
| cublasCtrttp |
| cublasSgemmEx |

| BLAS-like Extension |
|----------------------------|
| cublasCgemmEx |
| cublasGemmEx |
| cublasGemmBatchedEx |
| cublasGemmStridedBatchedEx |
| cublasCsyrkEx |
| cublasCsyrk3mEx |
| cublasCherkEx |
| cublasCherk3mEx |
| cublasNrm2Ex |
| cublasAxpypEx |
| cublasDotEx |
| cublasDotcEx |
| cublasRotEx |
| cublasScalEx |

4 dBLAS 扩展API

4.1 当前版本

dBLAS 扩展API当前版本为1.6.0。

4.1.1 当前版本已支持的API

| dlblas_ext API | 备注 |
|----------------------------|----------------------------|
| dlblasGemmEx | 量化矩阵乘法 |
| dlblasGemmStridedBatchedEx | 多batch量化矩阵乘法 |
| dlblasGemmSparseEx | 稀疏矩阵乘法 |
| dlblasGemmMarlinEx | Marlin量化矩阵乘法 |
| dlblasMatrixElementise | 三维矩阵对应元素乘，加操作，支持在batch维度广播 |
| dlblasMatrixTransform | 三维矩阵转置，可转置batch维度到最低维 |

| dlblas_ext API | 备注 |
|----------------|-----------------------------------|
| dlblasGemmExV2 | dlblasGemmEx的v2版本，支持block-wise的量化 |

4.2 dBLAS 版本更新说明

4.2.1 版本1.0.0

扩展了量化矩阵乘法API。

dlblasGemmEx：量化矩阵乘法。

dlblasGemmStridedBatchedEx：多batch量化矩阵乘法。

4.2.2 版本1.1.0

新增稀疏矩阵乘法API。

dlblasGemmSparseEx

4.2.3 版本1.2.0

dlblasGemmSparseEx新增quantParameters参数。

4.2.4 版本1.3.0

拓展了Marlin矩阵乘法API。

dlblasGemmMarlinEx：fp16xint4量化矩阵乘法。

4.2.5 版本1.4.0

支持了三维矩阵在batch维度做广播的矩阵点乘。

dlblasMatrixElementise。

4.2.6 版本1.5.0

支持了三维矩阵的转置，当前仅支持将最高维的batch转到最低维上。

dlblasMatrixTransform。

4.2.7 版本1.6.0

对dblasGemmEx做了改进，增加如下功能：

- 1.支持block-wise的量化方式，当前仅支持量化windows为正方形，即在行和列上量化的group size大小必须相等。
- 2.zeropoints 和 scales量化系数不再强制为half类型，可通过cuda type指定。

dblasGemmExV2。

4.3 dblasGemmEx

```
cublasStatus_t dblasGemmEx(cublasHandle_t handle,
                           cublasOperation_t transa,
                           cublasOperation_t transb,
                           int m,
                           int n,
                           int k,
                           const void *alpha,
                           const void *A,
                           cudaDataType_t Atype,
                           int lda,
                           const void *B,
                           cudaDataType_t Btype,
                           int ldb,
                           const void *beta,
                           void *C,
                           cudaDataType_t Ctype,
                           int ldc,
                           cudaDataType_t computeType,
                           cublasGemmAlgo_t algo,
                           dblasExtQuantParameters_t* quantParameters)
```

dblasGemmEx计算量化矩阵乘法，支持int4，uint4，int8，uint8，fp8(e5m2)，fp8(e4m3)量化类型，需include "dblas_ext.h"。

dblasGemmEx支持perTensor，perChannel，perGroup 3 种量化方式，除最后一个参数quantParameters外，其余参数均与cublasGemmEx保持一致，可参考cublas官方文档，了解其他参数含义。本文档仅重点说明不同的量化方式下quantParameters的设置。

perTensor量化：被量化矩阵所有元素共用一个zeropoints，scales，且zeropoints，scales必须在host上。

```
__half zeropoints = xxx;
__half scales = xxx;
```

A是量化矩阵时，quantParameters参数如下：

```
quantParameters.a_axis = kNone;
quantParameters.a_group_size = 0;
quantParameters.a_zeropoints = &zeropoints;
quantParameters.a_scales = &scales
```

B是量化矩阵时，quantParameters参数如下：

```
quantParameters.b_axis = kNone;
quantParameters.b_group_size = 0;
quantParameters.b_zeropoints = &zeropoints;
quantParameters.b_scales = &scales
```

perChannel量化：zeropoints, scales 在device上，被量化矩阵的某一维度所有元素共用一个量化参数。

A是量化矩阵时，若a_axis = kM，表示是在m维度上量化，m维上所有k个元素共用一个量化参数，a_zeropoints和a_scales维度为[m]。若a_axis = kK，表示是在k维度上量化，k维上所有m个元素共用一个量化参数，a_zeropoints和a_scales维度为[k]，quantParameters参数如下：

```
quantParameters.a_axis = kM or kK;
quantParameters.a_group_size = 0;
quantParameters.a_zeropoints = (device_pointer);
quantParameters.a_scales = (device_pointer);
```

B是量化矩阵时，若b_axis = kK，表示是在k维度上量化，k维上所有n个元素共用一个量化参数，b_zeropoints和b_scales维度为[k]。若b_axis = kN，表示是在n维度上量化，n维上所有k个元素共用一个量化参数，b_zeropoints和b_scales维度为[n]，quantParameters参数如下：

```
quantParameters.b_axis = kK or kN;
quantParameters.b_group_size = 0;
quantParameters.b_zeropoints = (device_pointer);
quantParameters.b_scales = (device_pointer);
```

perGroup量化：zeropoints, scales 在device上，被量化矩阵的某一维度每group_size个元素共用一个量化参数。

A是量化矩阵时，若a_axis = kM，表示perChannel在m维度上，且在k维度上每a_group_size个元素共用一个量化参数，如果transa = CUBLAS_OP_N，a_zeropoints和a_scales维度为[k/a_group_size, m]，否则维度为[m, k/a_group_size]。a_axis = kK表示perChannel在k维度上，且在m维度上每a_group_size个元素共用一个量化参数，如果transa = CUBLAS_OP_N，a_zeropoints和a_scales维度为[k, m/a_group_size]，否则维度为[m/a_group_size, k]。

quantParameters参数如下：

```
quantParameters.a_axis = kM or kK;  
quantParameters.a_group_size = xxx;  
quantParameters.a_zeropoints = (device_pointer);  
quantParameters.a_scales = (device_pointer);
```

B是量化矩阵时，若b_axis = kK，表示perChannel在k维度上，且在n维度上每b_group_size个元素共用一个量化参数，如果transb = CUBLAS_OP_N，b_zeropoints和b_scales维度为[n/b_group_size, k]，否则维度为[k, n/b_group_size]。b_axis = kN表示perChannel在n维度上，且在k维度上每b_group_size个元素共用一个量化参数，如果transb = CUBLAS_OP_N，b_zeropoints和b_scales维度为[n, k/b_group_size]，否则维度为[k/b_group_size, n]。

quantParameters参数如下：

```
quantParameters.b_axis = kK or kN;  
quantParameters.b_group_size = xxx;  
quantParameters.b_zeropoints = (device_pointer);  
quantParameters.b_scales = (device_pointer)
```

说明：
如果zeropoints为0或者没有该参数则zeropoints可传nullptr。

4.3.1 量化类型与CUDA数据类型对应关系

量化类型与CUDA数据类型如下表所示：

| 量化类型 | CUDA 数据类型 |
|-----------|--------------------------------|
| int4 | CUDA_R_4I |
| uint4 | CUDA_R_4U |
| int8 | CUDA_R_8I |
| uint8 | CUDA_R_8U |
| fp8(e5m2) | (cudaDataType_t)CUDA_R_8F_E5M2 |
| fp8(e4m3) | (cudaDataType_t)CUDA_R_8F_E4M3 |

4.3.2 支持的类型组合

dlblasGemmEx supports the following Scale Type(alpha and beta), Atype, Btype, Ctype:

| Scale Type(alpha and beta) | Atype | Btype | Ctype |
|----------------------------|----------------|----------------|------------|
| CUDA_R_16F/CUDA_R_32F | CUDA_R_8F_E5M2 | CUDA_R_8F_E5M2 | CUDA_R_16F |

| Scale Type(alpha and beta) | Atype | Btype | Ctype |
|----------------------------|----------------|-----------------------|------------|
| CUDA_R_16F/CUDA_R_32F | CUDA_R_8F_E5M2 | CUDA_R_16F | CUDA_R_16F |
| CUDA_R_16F/CUDA_R_32F | CUDA_R_8F_E4M3 | CUDA_R_8F_E4M3 | CUDA_R_16F |
| CUDA_R_16F/CUDA_R_32F | CUDA_R_8F_E4M3 | CUDA_R_16F | CUDA_R_16F |
| CUDA_R_16F/CUDA_R_32F | CUDA_R_8I | CUDA_R_8I | CUDA_R_16F |
| CUDA_R_16F/CUDA_R_32F | CUDA_R_8I | CUDA_R_16F | CUDA_R_16F |
| CUDA_R_16F/CUDA_R_32F | CUDA_R_8U | CUDA_R_8U | CUDA_R_16F |
| CUDA_R_16F/CUDA_R_32F | CUDA_R_8U | CUDA_R_16F | CUDA_R_16F |
| CUDA_R_16F/CUDA_R_32F | CUDA_R_4I | CUDA_R_4I | CUDA_R_16F |
| CUDA_R_16F/CUDA_R_32F | CUDA_R_4I | CUDA_R_16F | CUDA_R_16F |
| CUDA_R_16F/CUDA_R_32F | CUDA_R_4U | CUDA_R_4U | CUDA_R_16F |
| CUDA_R_16F/CUDA_R_32F | CUDA_R_4U | CUDA_R_16F | CUDA_R_16F |
| | | | |
| CUDA_R_32F | CUDA_R_8F_E5M2 | CUDA_R_8F_E5M2 | CUDA_R_32F |
| CUDA_R_32F | CUDA_R_8F_E5M2 | CUDA_R_16F/CUDA_R_32F | CUDA_R_32F |
| CUDA_R_32F | CUDA_R_8F_E4M3 | CUDA_R_8F_E4M3 | CUDA_R_32F |
| CUDA_R_32F | CUDA_R_8F_E4M3 | CUDA_R_16F/CUDA_R_32F | CUDA_R_32F |
| CUDA_R_32F | CUDA_R_8I | CUDA_R_8I | CUDA_R_32F |
| CUDA_R_32F | CUDA_R_8I | CUDA_R_16F/CUDA_R_32F | CUDA_R_32F |
| CUDA_R_32F | CUDA_R_8U | CUDA_R_8U | CUDA_R_32F |
| CUDA_R_32F | CUDA_R_8U | CUDA_R_16F/CUDA_R_32F | CUDA_R_32F |
| CUDA_R_32F | CUDA_R_4I | CUDA_R_4I | CUDA_R_32F |
| CUDA_R_32F | CUDA_R_4I | CUDA_R_16F/CUDA_R_32F | CUDA_R_32F |
| CUDA_R_32F | CUDA_R_4U | CUDA_R_4U | CUDA_R_32F |
| CUDA_R_32F | CUDA_R_4U | CUDA_R_16F/CUDA_R_32F | CUDA_R_32F |

4.3.3 int4 或者 uint4 量化时数据内存布局

若Atype = CUDA_R_4U, transa = CUBLAS_OP_N, m = 2, k = 3, A 中的数据如下:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

则在列主序的情况下A中数据在内存上为: A={1,4,2,5,3,6}

对应的内存上的bit数据为: |1100|0011|0101|0010|0100|0001|

也就是A中的数据在内存上是按照从低位到高位每4bit连续摆放的。

4.3.4 dlblasGemmEx 计算代码示例

若仅perChannel量化A, 且Atype = CUDA_R_8I, Btype = CUDA_R_16F, Ctype = CUDA_R_16F, transa = CUBLAS_OP_N, transb = CUBLAS_OP_N, a_axis=kM, alpha=1, beta=0, 则dlblasGemmEx的计算过程可由如下代码表示:

```
void dlblasGemmEx(int8_t* A, __half* B, __half* C, __half* zeropoints, __half* scales
                  int m, int n, int k){
    for (int i = 0; i < n; i++){
        for (int j = 0; j < m; j++){
            float temp = 0.0f;
            for (int q = 0; q < k; q++){
                __half dequantA = 0;
                dequantA = (__half(A[q * m + j]) - zeropoints[j])*scales[j];
                temp += __half2float(dequantA * B[i * k +q]);
            }
            c[i * m + j] = __float2half(temp);
        }
    }
}
```

若仅perGroup量化A, 且Atype = CUDA_R_8I, Btype = CUDA_R_16F, Ctype = CUDA_R_16F, transa = CUBLAS_OP_N, transb = CUBLAS_OP_N, a_axis=kM, alpha=1, beta=0, 则dlblasGemmEx的计算过程可由如下代码表示:

```
void dlblasGemmEx(int8_t* A, __half* B, __half* C, __half* zeropoints, __half* scales
                  int m, int n, int k, int group_size){
    for (int i = 0; i < n; i++){
        for (int j = 0; j < m; j++){
            float temp = 0.0f;
            for (int q = 0; q < k; q++){
                __half dequantA = 0;
                int scale_idx = q / a_group_size * m + j;
                dequantA = (__half(A[q * m + j]) -
zeropoints[scale_idx])*scales[scale_idx];
                temp += __half2float(dequantA * B[i * k +q]);
            }
            c[i * m + j] = __float2half(temp);
        }
    }
}
```

4.4 dblasGemmStridedBatchedEx

dblasGemmStridedBatchedEx是dblasGemmEx的多batch版本，除最后一个参数quantParameters外，其余参数均与cublasGemmStridedBatchedEx保持一致，可参考cublas官方文档，对于quantParameters的设置，可参考dblasGemmEx中的说明。

```
cublasStatus_t dblasGemmStridedBatchedEx(cublasHandle_t handle,
                                           cublasOperation_t transa,
                                           cublasOperation_t transb,
                                           int m,
                                           int n,
                                           int k,
                                           const void *alpha,
                                           const void *A,
                                           cudaDataType_t Atype,
                                           int lda,
                                           long long int strideA,
                                           const void *B,
                                           cudaDataType_t Btype,
                                           int ldb,
                                           long long int strideB,
                                           const void *beta,
                                           void *C,
                                           cudaDataType_t Ctype,
                                           int ldc,
                                           long long int strideC,
                                           int batchSize,
                                           cudaDataType_t computeType,
                                           cublasGemmAlgo_t algo,
                                           dblasExtQuantParameters_t* quantParameters)
```

4.5 dblasGemmSparseEx

dblasGemmSparseEx是稀疏矩阵乘法计算。其中Aval、Aidx分别指向稀疏矩阵经过压缩以后所对应的数据和位置索引。其余参数均与cublasGemmEx保持一致，可参考cublas官方文档。默认alpha=1, beta=0, 其它值暂时不支持。如果transa=CUBLAS_OP_T且Atype=CUDA_R_16F或者Atype=CUDA_R_32F, 则lda和lda_i需要满足32Byte对齐的要求。

```
cublasStatus_t dblasGemmSparseEx(cublasHandle_t handle,
                                  cublasOperation_t transa,
                                  cublasOperation_t transb,
                                  int m,
                                  int n,
                                  int k,
                                  const void *alpha,
                                  const void *Aval,
                                  cudaDataType_t Atype,
                                  int64_t lda,
                                  const void *Aidx,
                                  int64_t lda_i,
```

```

const void *B,
cudaDataType_t Btype,
int64_t ldb,
const void *beta,
void *C,
cudaDataType_t Ctype,
int64_t ldc,
cudaDataType_t computeType,
cublasGemmAlgo_t algo,
dlblasExtQuantParameters_t* quantParameters)

```

4.6 dlblasGemmMarlinEx

dlblasGemmMarlinEx是内部拓展fp16xint4量化矩阵乘法的API, 其中A(fp16), B(int4), C(fp16) 分别为输入输出矩阵,s(fp16)为量化参数scale, prob_m, prob_n, prob_k代表矩阵形状。

```

cublasStatus_t DLBLASEXTWINAPI dlblasGemmMarlinEx(const void* A,
                                                    const void* B,
                                                    void* C,
                                                    void* s,
                                                    int prob_m,
                                                    int prob_n,
                                                    int prob_k,
                                                    cublasHandle_t handle,
                                                    void* workspace = 0,
                                                    int groupsize = -1,
                                                    int dev = 0,
                                                    int thread_k = -1,
                                                    int thread_n = -1,
                                                    int sms = -1,
                                                    int max_par = 16);

```

4.7 dlblasMatrixElementwise

三维矩阵对应元素相加或相乘。目前仅支持矩阵的点乘, 并可在batch维度上做broadcasted。

```

cublasStatus_t DLBLASEXTWINAPI dlblasMatrixElementwise(cublasHandle_t handle,
                                                         cublasOperation_t transa,
                                                         cublasOperation_t transb,
                                                         int mA,
                                                         int nA,
                                                         const void *A,
                                                         cudaDataType_t Atype,
                                                         int batchA,
                                                         int lda,
                                                         long long int strideA,
                                                         int mB,
                                                         int nB,

```

```

const void *B,
cudaDataType_t Btype,
int batchB,
int ldb,
long long int strideB,
void *C,
cudaDataType_t Ctype,
int ldc,
long long int strideC,
dblasMatrixElementwiseOp_t* op)

```

4.8 dblasMatrixTransform

三维矩阵的转置。当前仅支持将最高维的batch转置到最低维度。

```

cublasStatus_t DLBLASEXTWINAPI dblasMatrixTransform(cublasHandle_t handle,
                                                    cublasOperation_t transa,
                                                    int m,
                                                    int n,
                                                    const void *A,
                                                    cudaDataType_t Atype,
                                                    int lda,
                                                    long long int strideA,
                                                    void *C,
                                                    cudaDataType_t Ctype,
                                                    int ldc,
                                                    long long int strideC,
                                                    int batchSize,
                                                    dblasMatrixTransformDesc_t
transformDesc)

```

4.9 dblasGemmExV2

dblasGemmEx的v2版本，支持指定zeropoints和scales的数据类型，支持block-wise量化。当前仅支持block-wise量化时两个group_size相等的情形。

```

cublasStatus_t DLBLASEXTWINAPI dblasGemmExV2(cublasHandle_t handle,
                                                cublasOperation_t transa,
                                                cublasOperation_t transb,
                                                int m,
                                                int n,
                                                int k,
                                                const void *alpha,
                                                const void *A,
                                                cudaDataType_t Atype,
                                                int lda,
                                                const void *B,
                                                cudaDataType_t Btype,

```

```
int ldb,  
const void *beta,  
void *C,  
cudaDataType_t Ctype,  
int ldc,  
cudaDataType_t computeType,  
cublasGemmAlgo_t algo,  
dblasExtQuantParametersV2_t*  
quantParameters)
```