



天数智芯
Iluvatar CoreX

天数智芯

软件栈 API 参考

版本：V4.0.0-MR

日期：2024.4.15

适用产品：智铠 50 | 智铠 100

1 声明

1.1 版权声明

版权所有。未经天数智芯书面许可，不得以任何形式或方式将本文档的任何部分复制，传播，转录或翻译成任何语言。

1.2 免责声明

天数智芯可以随时对本文档或本文档中描述的产品进行改进和/或更改。本文档包括与天数智芯产品有关的信息，作为说明典型应用的一种方式，因此，不一定提供足以进行生产设计的完整信息。对于本文档中内容的准确性或完整性，天数智芯不做任何陈述或保证。

1.3 联系方式

地址：上海闵行区陈行公路 2168 号 3 幢

电话：021-68886607

网址：www.iluvatar.com

Contents

1 声明	2
1.1 版权声明	2
1.2 免责声明	2
1.3 联系方式	2
2 概述	6
2.1 修订记录	6
2.2 提供的函数库	6
3 ixJPEG	8
3.1 ixJPEG 库介绍	8
3.1.1 支持的功能	8
3.1.2 支持的特性	9
3.1.3 已知限制	9
3.2 JPEG Common API	10
3.2.1 nvjpegCreate	10
3.2.2 nvjpegCreateSimple	10
3.2.3 nvjpegCreateEx	11
3.2.4 nvjpegDestroy	11
3.3 JPEG Decode API	12
3.3.1 nvjpegGetImageInfo	12
3.3.2 nvjpegDecode	12
3.3.3 nvjpegDecodeBatchedInitialize	13
3.3.4 nvjpegDecodeBatched	14
3.3.5 nvjpegJpegStateCreate	14
3.3.6 nvjpegJpegStateDestroy	15
3.4 JPEG Encode API	15
3.4.1 nvjpegEncoderStateCreate	16
3.4.2 nvjpegEncoderStateDestroy	16
3.4.3 nvjpegEncoderParamsCreate	17
3.4.4 nvjpegEncoderParamsDestroy	17
3.4.5 nvjpegEncoderParamsSetQuality	18
3.4.6 nvjpegEncoderParamsSetOptimizedHuffman	18
3.4.7 nvjpegEncoderParamsSetSamplingFactors	19
3.4.8 nvjpegEncodeYUV	19
3.4.9 nvjpegEncodeImage	20
3.4.10 nvjpegEncodeRetrieveBitstream	21
3.4.11 cujpegEncodeYUVBatched	21
3.4.12 cujpegEncodeRetrieveBitstreamDeviceBatched	22
3.4.13 cujpegEncodeRetrieveBitstreamBatched	22
3.4.14 cujpegEncoderParamsSetFrameFormat	23
3.5 编解码用例使用指南	24
3.5.1 解码用例使用指南	24
3.5.2 编码用例使用指南	24

3.5.3 编解码混合用例使用指南	25
4 ixCODEC	26
4.1 ixCODEC 库介绍	26
4.2 cuvidCreateDecoder	26
4.3 cuvidDestroyDecoder	27
4.4 cuvidDecodePicture	27
4.5 cuvidGetDecodeStatus	28
4.6 cuvidMapVideoFrame	28
4.7 cuvidUnmapVideoFrame	29
4.8 cuvidDecodeGetPictureInfo	29
4.9 视频解码参考用例使用指南	30
4.9.1 使用方法	30
5 ixBLAS	32
5.1 ixBLAS API	32
5.1.1 Helper function	32
5.1.2 Level-1 function	33
5.1.3 Level-2 function	33
5.1.4 Level-3 function	34
5.1.5 Extension function	35
5.2 ixBLASLt API	36
6 ixFFT	37
6.1 cufft API	37
6.2 cufftw API	38
7 ixDNN	39
7.1 flashAttn	42
7.1.1 相关 API	44
7.1.2 相关 API 适用范围	44
7.1.3 API 调用方式	44
7.1.4 结构体说明	44
7.1.4.1 cudnnFlashAttnLayout_t	44
7.1.4.2 cudnnFlashAttnConfigInfo	45
7.1.5 函数说明	46
7.1.5.1 cudnnGetFlashAttnBuffers()	46
7.1.5.2 cudnnFlashAttnForward()	46
7.1.5.3 cudnnFlashAttnBackward()	48
7.1.6 接口调用说明	49
7.1.6.1 不同 Layout 调用说明	49
7.1.6.2 softmax 多种 Mask 接口调用说明	50
7.1.7 Layout 说明	52
7.1.7.1 CUDNN_FATTN_BHSD_UNPACK	52
7.1.7.2 CUDNN_FATTN_BSHD_UNPACK	53
7.1.7.3 CUDNN_FATTN_BSHD_QKVPACK	55
7.1.7.4 CUDNN_FATTN_BSHD_KVPACK	56
7.1.8 group-query-attn 支持	56

7.1.9 新增 Alibi 支持	59
7.1.9.1 Alibi 计算流程说明	60
7.1.9.2 Alibi 偏置矩阵	60
7.1.10 flashAttn 接口调用示例	62
8 ixRAND	63
9 ixCCL	64
9.1 功能说明	64
9.2 支持的多机通信协议	64
9.2.1 方式一：使用以太网卡	64
9.2.2 方式二：使用 InfiniBand 网卡 RDMA 通信	64
9.3 PyTorch 下指定通信后端	65
9.3.1 通信后端使用方式	66
9.3.1.1 指定通信后端	66
9.3.1.2 多机多卡：配置网络接口	67
10 ixSPARSE	68
10.1 Management Functions	68
10.2 Helper Functions	68
10.3 L1 Functions	69
11 CUB	70
12 THRUST	71
13 Driver API	73
14 Runtime API	78
15 Math API	82
15.1 Integer Function	82
15.2 Integer Intrinsic	83
15.3 Float Function	83
15.4 Float Intrinsic	85
15.5 Type Cast	86
15.6 Half Function	87
15.7 Half Arithmetic	88
15.8 Half Comparison	88
15.9 Half Precision Conversion and Data Movement	88
15.10 Half2 Function	90
15.11 Half2 Arithmetic	91
15.12 Half2 Comparison	91
16 CV-CUDA	93
17 商标声明	95

2 概述

2.1 修订记录

- COREX01-MR400-RF02-01：2024/4/15
 - 更新 ixDNN API，新增支持 flashAttn 算法相关内容
- COREX01-MR400-RF02-00：2024/4/3

文档本次发布内容与 V3.2.1-MR 文档相比 (COREX01-MR321-RF02-00)，有以下更新：

- ixCODEC (原 ixDEC):
 - * 更新 ixDEC 库名称为 ixCODEC
 - * VPU 模块新增支持 VP9 格式
 - * 更新 cuvidDecodePicture 的传参
 - * 新增“视频解码参考用例使用指南”小节，提供视频解码参考用例供用户快速上手 ixCODEC
- 新增 ixCCL API
- 更新适配的 CV-CUDA 版本为 v0.4，更新适配的算子

2.2 提供的函数库

天数智算软件栈提供以下函数库。

函数库	默认安装路径 [1]
ixJPEG	/usr/local/corex/lib64/libcjpeg.so
ixDEC	/usr/local/corex/lib64/libnvcuvid.so.1
ixBLAS	/usr/local/corex/lib64/libcublas.so
ixFFT	/usr/local/corex/lib64/libcufft.so
ixDNN	/usr/local/corex/lib64/libcudnn.so
ixRAND	/usr/local/corex/lib64/libcurand.so
ixCCL	/usr/local/corex/lib64/libnccl.so
ixSPARSE	/usr/local/corex/lib64/libcusparse.so
CUB	/usr/local/corex/include
THRUST	/usr/local/corex/include
Driver API	/usr/local/corex/lib64/libcuda.so
Runtime API	/usr/local/corex/lib64/libcudart.so
Math API	编译器默认支持

函数库	默认安装路径 [1]
CV-CUDA	/usr/local/corex/lib64/libcvcuda.so

Tip

[1] 天数适配版深度学习框架以及天数智芯编译器默认指向了相关路径，您在代码中直接调用所需 API 即可。

3 ixJPEG

3.1 ixJPEG 库介绍

天数智芯 ixJPEG 库为深度学习和超大规模多媒体应用程序中常用的图像格式提供高性能、GPU 加速的 JPEG 解码功能。该库提供单一和批量 JPEG 解码功能，可有效利用可用的 GPU 资源以获得最佳性能，以及用户管理解码所需的内存分配的灵活性。

天数智算软件栈提供 ixJPEG Decode 解码库与 ixJPEG Encode 编码库供您使用。

3.1.1 支持的功能

ixJPEG 库支持使用 JPEG 图像数据流作为输入，从数据流中检索图像的宽度和高度，并使用这些检索到的信息来管理 GPU 内存分配和解码。ixJPEG 提供专用的 API 用于从原始 JPEG 图像数据流中检索图像信息。

Note

在本文档中，术语“CPU”和“主机”是 synonym；术语“GPU”和“设备”是 synonym。

ixJPEG 库支持 MJPEG 解码/编码，基线与扩展顺序 ISO/IEC 10918-1 JPEG 兼容。扩展顺序仅支持 Huffman 编码与 12 位采样精度。

Note

暂不支持扩展序列的算术编码与 8 位采样精度。

具体如下：

- 支持 1 或 3 个颜色分量
- 一次扫描 3 个组件通道（仅限交错）
- 每个组件的 8 位或 12 位采样
- 最小编码大小为 16x16 像素
- 支持 ROI（Region of Interest）- 仅解码器支持
- 422/444 打包模式
 - 所有颜色格式的 422 打包模式
 - 444 打包模式仅适用于 444 格式
- 支持以下 3 个颜色通道 Y、Cb、Cr（Y、U、V）的色度子采样：
 - 4:4:4
 - 4:2:2
 - 4:2:0
 - 4:4:0
 - 4:0:0
- 支持以下解码输出格式：
 - RGB
 - RGBI
 - BGR

- BGRI
- YUV
- Unchanged
- NV12
- NV21

3.1.2 支持的特性

ixJPEG 库具有以下特性：

- 使用 CPU（即主机）与 GPU（即设备）的混合解码
- 支持基于 JPEG 解码的 JPU 硬件加速
- 支持库的输入在主机内存中，输出在 GPU 内存中
- 支持单图像与批量图像解码
- 支持单阶段与多阶段解码
- 支持色彩空间转换
- 支持用户提供的设备内存管理器与固定的主机内存进行分配
- 支持基于切片的编码
- 支持多实例模式与实例切换
- 支持动态旋转与镜像
- 支持解码器中的动态下采样

3.1.3 已知限制

ixJPEG 库具有以下限制：

- 多实例支持
 - 基于帧的切换：无限实例
 - 基于切片的切换，最多 4 个
- PP（前/后处理器）
 - 不适用于格式转换器
 - 不适用于 Scaler
 - 不适用于 ROI
- ROI
 - 不适用于打包的 YUY
 - 不适用于缩放
 - 不适用于旋转与镜像
 - 不适用于格式转换
- 切片编码
 - 切片编码与 PP 不支持同时工作

3.2 JPEG Common API

天智算软件栈提供以下 JPEG 公共 API：

3.2.1 nvjpegCreate

功能概述

指定参数部分，创建 JPEG 库句柄。

函数原型

```
nvjpegStatus_t nvjpegCreate(nvjpegBackend_t backend, nvjpegDevAllocator_t* dev_allocator,  
↪ nvjpegHandle_t* handle);
```

参数说明

- nvjpegBackend_t backend：指定后端解码模式，参数未实际生效，仅支持硬件 JPU 解码
- nvjpegDevAllocator_t* dev_allocator：设备内存管理函数，支持用户特化，默认使用标准 cudaMalloc 和 cudaFree
- nvjpegHandle_t* handle：JPEG 库句柄

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS：成功
- OTHERS：失败

3.2.2 nvjpegCreateSimple

功能概述

全部使用默认参数创建 JPEG 库句柄。

函数原型

```
nvjpegStatus_t nvjpegCreateSimple(nvjpegHandle_t* handle);
```

参数说明

- nvjpegHandle_t* handle：JPEG 库句柄

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS：成功
- OTHERS：失败

3.2.3 nvjpegCreateEx

功能概述

使用详细参数创建 JPEG 句柄，应初始化 handle=nullptr，用于显式区分编解码；Backend 当前只有默认值 JPU 解码，dev_allocator 和 pinned_allocator 为用户特化的内存分配接口可用 nullptr 设为默认 CUDA 接口，flags 暂未使用。

函数原型

```
nvjpegStatus_t nvjpegCreateEx(nvjpegBackend_t backend,
    nvjpegDevAllocator_t* dev_allocator,
    nvjpegPinnedAllocator_t* pinned_allocator,
    unsigned int flags,
    nvjpegHandle_t* handle);
```

参数说明

- nvjpegBackend_t backend：指定后端解码模式，参数未实际生效，仅支持硬件 JPU 解码
- nvjpegDevAllocator_t* dev_allocator：设备内存管理函数，支持用户特化，默认使用标准 cudaMalloc 和 cudaFree
- nvjpegPinnedAllocator_t* pinned_allocator：锁页内存管理函数，支持用户特化，默认使用标准 cudaHostAlloc 和 cudaFreeHost
- nvjpegHandle_t* handle：JPEG 库句柄

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS：成功
- OTHERS：失败

3.2.4 nvjpegDestroy

功能概述

析构 JPEG 库句柄。

函数原型

```
nvjpegStatus_t nvjpegDestroy(nvjpegHandle_t handle);
```

参数说明

- nvjpegHandle_t handle：JPEG 库句柄

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS：成功
- OTHERS：失败

3.3 JPEG Decode API

Tip

ixJPEG 解码库提供一个可执行文件用例 ixjpegdec，您可以直接使用该用例对图片进行解码。可执行文件的使用说明请参考[解码用例使用指南](#)。

天数智算软件栈提供以下 JPEG 解码 API：

3.3.1 nvjpegGetImageInfo

功能概述

根据图片数据 data 和尺寸 length，使用库句柄获取图片基本信息，包括颜色分量数 nComponents、采样格式 subsampling、宽 widths、高 heights。

函数原型

```
nvjpegStatus_t nvjpegGetImageInfo(nvjpegHandle_t handle,
    const unsigned char* data,
    size_t length,
    int* nComponents,
    nvjpegChromaSubsampling_t* subsampling,
    int* widths,
    int* heights);
```

参数说明

- nvjpegHandle_t handle：JPEG 库句柄
- const unsigned char* data：数据指针
- size_t length：数据长度
- int* nComponents：颜色分量数
- nvjpegChromaSubsampling_t* subsampling：图片采样格式
- int* widths：图片原始宽度
- int* heights：图片原始高度

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS：成功
- OTHERS：失败

3.3.2 nvjpegDecode

功能概述

根据图片数据 data 和尺寸 length，按照指定输出格式 output_format 进行解码，解码结果存放在用户提前分配好内存的 destination 中，stream 可由用户创建，也可传入 nullptr 来使用默认 stream。

函数原型

```
nvjpegStatus_t nvjpegDecode(nvjpegHandle_t handle,
    nvjpegJpegState_t jpeg_handle,
    const unsigned char* data,
    size_t length,
    nvjpegOutputFormat_t output_format,
    nvjpegImage_t* destination,
    cudaStream_t stream);
```

参数说明

- nvjpegHandle_t handle: JPEG 库句柄
- nvjpegJpegState_t jpeg_handle: JPEG 解码状态句柄
- const unsigned char* data: 数据指针
- size_t length: 数据长度
- nvjpegOutputFormat_t output_format: 图片解码输出格式
- nvjpegImage_t* destination: 解码输出地址描述符, 所描述内存由用户管理负责分配回收
- cudaStream_t stream: 配置解码使用的 CUDA 流, 支持使用默认 CUDA 流

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.3.3 nvjpegDecodeBatchedInitialize

功能概述

batch 解码初始化, 指定 batch 大小、batch_size 和输出格式 output_format, max_cpu_threads 当前未使用。

函数原型

```
nvjpegStatus_t nvjpegDecodeBatchedInitialize(nvjpegHandle_t handle,
    nvjpegJpegState_t jpeg_handle,
    int batch_size,
    int max_cpu_threads,
    nvjpegOutputFormat_t output_format);
```

参数说明

- nvjpegHandle_t handle: JPEG 库句柄
- nvjpegJpegState_t jpeg_handle: JPEG 解码状态句柄
- int batch_size: 配置 batch 解码时的批量大小
- int max_cpu_threads: 设置内部 CPU 线程数, 当前不支持配置

- nvjpegOutputFormat_t output_format: 图片解码输出格式

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.3.4 nvjpegDecodeBatched

功能概述

batch 解码接口，用户需保证 data、lengths、destinations 和初始化的 batch 大小匹配。

函数原型

```
nvjpegStatus_t nvjpegDecodeBatched(nvjpegHandle_t handle,
    nvjpegJpegState_t jpeg_handle,
    const unsigned char* const* data,
    const size_t* lengths,
    nvjpegImage_t* destinations,
    cudaStream_t stream);
```

参数说明

- nvjpegHandle_t handle: JPEG 库句柄
- nvjpegJpegState_t jpeg_handle: JPEG 解码状态句柄
- const unsigned char* const* data: 数据指针数组
- const size_t* lengths: 数据长度数组
- nvjpegImage_t* destinations: 解码输出地址描述符数组，用于 batch 解码
- cudaStream_t stream: 配置解码使用的 CUDA 流，支持使用默认 CUDA 流

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.3.5 nvjpegJpegStateCreate

功能概述

创建 JPEG 解码状态句柄。

函数原型

```
nvjpegStatus_t nvjpegJpegStateCreate(nvjpegHandle_t handle, nvjpegJpegState_t* jpeg_handle);
```

参数说明

- nvjpegHandle_t handle: JPEG 库句柄
- nvjpegJpegState_t* jpeg_handle: JPEG 解码状态句柄

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.3.6 nvjpegJpegStateDestroy

功能概述

析构 JPEG 解码状态句柄。

函数原型

```
nvjpegStatus_t nvjpegJpegStateDestroy(nvjpegJpegState_t jpeg_handle);
```

参数说明

- nvjpegJpegState_t jpeg_handle: JPEG 解码状态句柄

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.4 JPEG Encode API

Tip

天数智芯 ixJPEG 编码库提供一个可执行文件用例 `ixjpegenc`，您可以直接使用该用例对图片进行编码。可执行文件的使用说明请参考[编码用例使用指南](#)。

天数智算软件栈提供以下 JPEG 编码 API:

3.4.1 nvjpegEncoderStateCreate

功能概述

创建一个 encodeStatue 对象，包含编码所使用的内存资源信息。

函数原型

```
nvjpegStatus_t nvjpegEncoderStateCreate(nvjpegHandle_t handle, nvjpegEncoderState_t*
↪ encoder_state, cudaStream_t stream);
```

参数说明

- nvjpegHandle_t handle: 使用 cujpegCreate 函数创建的 encoder handle 指针，不能为 NULL
- nvjpegEncoderState_t* encoder_state: 创建的 encoderStatue 对象指针，不能为 NULL。创建成功，该指针指向 nvjpegEncoderState_t 对象的地址
- cudaStream_t stream: stream 对象，可以为 NULL

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.4.2 nvjpegEncoderStateDestroy

功能概述

释放 encoderStatue 对象。

函数原型

```
nvjpegStatus_t nvjpegEncoderStateDestroy(nvjpegEncoderState_t encoder_state);
```

参数说明

- nvjpegEncoderState_t encoder_state: 使用 nvjpegEncoderStateCreate 函数创建的 encoderStatue 对象指针，不能为 NULL

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.4.3 nvjpegEncoderParamsCreate

功能概述

创建编码所使用的编码参数对象，包含 quality huffman table, SamplingFactors 等信息。

函数原型

```
nvjpegStatus_t nvjpegEncoderParamsCreate(nvjpegHandle_t handle, nvjpegEncoderParams_t*
↪ encoder_params, cudaStream_t stream);
```

参数说明

- nvjpegHandle_t handle: 使用 cujpegCreate 函数创建的 encoder handle 指针，不能为 NULL
- nvjpegEncoderParams_t* encoder_params: 创建的 encoder params 对象指针，不能为 NULL。创建成功，该指针指向 nvjpegEncoderParams_t 对象的地址
- cudaStream_t stream: stream 对象，可以为 NULL

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.4.4 nvjpegEncoderParamsDestroy

功能概述

释放 encoderParams 对象。

函数原型

```
nvjpegStatus_t nvjpegEncoderParamsDestroy(nvjpegEncoderParams_t encoder_params);
```

参数说明

- nvjpegEncoderParams_t encoder_params: 使用 nvjpegEncoderParamsCreate 函数创建的 nvjpegEncoderParams_t 对象指针，不能为 NULL

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.4.5 nvjpegEncoderParamsSetQuality

功能概述

设置编码参数的 quality 参数。

函数原型

```
nvjpegStatus_t nvjpegEncoderParamsSetQuality(nvjpegEncoderParams_t encoder_params, const int
↳ quality, cudaStream_t stream);
```

参数说明

- nvjpegEncoderParams_t encoder_params: 使用 nvjpegEncoderParamsCreate 函数创建的 nvjpegEncoderParams_t 对象指针, 不能为 NULL
- const int quality: 要设置编码的 quality 参数, 数值范围为 [0-100]
- cudaStream_t stream: stream 对象, 可以为 NULL

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.4.6 nvjpegEncoderParamsSetOptimizedHuffman

功能概述

设置编码参数是否采用 huffman 编码。

函数原型

```
nvjpegStatus_t nvjpegEncoderParamsSetOptimizedHuffman(nvjpegEncoderParams_t
↳ encoder_params, const int optimized, cudaStream_t stream);
```

参数说明

- nvjpegEncoderParams_t encoder_params: 使用 nvjpegEncoderParamsCreate 函数创建的 nvjpegEncoderParams_t 对象指针, 不能为 NULL
- const int optimized: 是否参数优化的 huffman 参数编码, 数值范围为 [0, 1], 0 表示否, 1 表示是
- cudaStream_t stream: stream 对象, 可以为 NULL

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.4.7 nvjpegEncoderParamsSetSamplingFactors

功能概述

设置编码参数的 SamplingFactors 参数。

函数原型

```
nvjpegStatus_t nvjpegEncoderParamsSetSamplingFactors(nvjpegEncoderParams_t encoder_params,
    const nvjpegChromaSubsampling_t chroma_subsampling,
    cudaStream_t stream);
```

参数说明

- nvjpegEncoderParams_t encoder_params: 使用 nvjpegEncoderParamsCreate 函数创建的 nvjpegEncoderParams_t 对象指针，不能为 NULL
- const nvjpegChromaSubsampling_t chroma_subsampling: 要设置的 SamplingFactors 参数，目前支持 [444, 422, 420, 440, 400]，默认为 420
- cudaStream_t stream: stream 对象，可以为 NULL

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.4.8 nvjpegEncodeYUV

功能概述

对一个 YUV 文件进行编码。

函数原型

```
nvjpegStatus_t nvjpegEncodeYUV(nvjpegHandle_t handle,
    nvjpegEncoderState_t encoder_state,
    const nvjpegEncoderParams_t encoder_params,
    const nvjpegImage_t* source,
    nvjpegChromaSubsampling_t chroma_subsampling,
    int image_width,
    int image_height,
    cudaStream_t stream);
```

参数说明

- nvjpegHandle_t handle: 使用 cujpegCreate 函数创建的 encoder handle 指针，不能为 NULL
- nvjpegEncoderState_t encoder_state: 使用 nvjpegEncoderStateCreate 函数创建的对象，不能为 NULL

- `const nvjpegEncoderParams_t encoder_params`: 使用 `nvjpegEncoderParamsCreate` 函数创建的 `nvjpegEncoderParams_t` 对象指针, 不能为 `NULL`
- `const nvjpegImage_t* source`: 存放 YUV 文件的 buff 地址, 不能为 `NULL`
- `nvjpegChromaSubsampling_t chroma_subsampling`: YUV 文件的格式
- `int image_width`: YUV 文件的宽度参数
- `int image_height`: YUV 文件的高度参数
- `cudaStream_t stream`: stream 对象, 可以为 `NULL`

返回值

`nvjpegStatus_t`:

- `NVJPEG_STATUS_SUCCESS`: 成功
- `OTHERS`: 失败

3.4.9 nvjpegEncodeImage

功能概述

对 RGB 等格式的文件进行编码。

函数原型

```
nvjpegStatus_t nvjpegEncodeImage(nvjpegHandle_t handle,
    nvjpegEncoderState_t encoder_state,
    const nvjpegEncoderParams_t encoder_params,
    const nvjpegImage_t* source,
    nvjpegInputFormat_t input_format,
    int image_width,
    int image_height,
    cudaStream_t stream);
```

参数说明

- `nvjpegHandle_t handle`: 使用 `cujpegCreate` 函数创建的 encoder handle 指针, 不能为 `NULL`
- `nvjpegEncoderState_t encoder_state`: 使用 `nvjpegEncoderStateCreate` 函数创建的对象, 不能为 `NULL`
- `const nvjpegEncoderParams_t encoder_params`: 使用 `nvjpegEncoderParamsCreate` 函数创建的 `nvjpegEncoderParams_t` 对象指针, 不能为 `NULL`
- `const nvjpegImage_t* source`: 存放 RGB 文件的 buff 地址, 不能为 `NULL`
- `nvjpegInputFormat_t input_format`: RGB 输入文件的格式
- `int image_width`: 文件的宽度参数
- `int image_height`: 文件的高度参数
- `cudaStream_t stream`: stream 对象, 可以为 `NULL`

返回值

`nvjpegStatus_t`:

- `NVJPEG_STATUS_SUCCESS`: 成功
- `OTHERS`: 失败

3.4.10 nvjpegEncodeRetrieveBitstream

功能概述

将编码后的数据拷贝到指定的 buff 中。

函数原型

```
nvjpegStatus_t nvjpegEncodeRetrieveBitstream(nvjpegHandle_t handle,
nvjpegEncoderState_t encoder_state,
unsigned char* data,
size_t* length,
cudaStream_t stream);
```

参数说明

- nvjpegHandle_t handle: 使用 cujpegCreate 函数创建的 encoder handle 指针, 不能为 NULL
- nvjpegEncoderState_t encoder_state: 使用 nvjpegEncoderStateCreate 函数创建的对象, 不能为 NULL
- unsigned char* data: 保存编码后数据的 host buff 地址
- size_t* length: 保存编码后输出数据的长度
- cudaStream_t stream: stream 对象, 可以为 NULL

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.4.11 cujpegEncodeYUVBatched

功能概述

对 YUV 文件进行批量编码。

函数原型

```
nvjpegStatus_t cujpegEncodeYUVBatched(nvjpegHandle_t handle,
nvjpegEncoderState_t encoder_state,
const unsigned char* SouDevBuffer,
const jpegEncParam* EncParams,
int BatchSize,
CUjpegstream stream);
```

参数说明

- nvjpegHandle_t handle: 使用 cujpegCreate 函数创建的 encoder handle 指针, 不能为 NULL
- nvjpegEncoderState_t encoder_state: 使用 nvjpegEncoderStateCreate 函数创建的对象, 不能为 NULL
- const unsigned char* SouDevBuffer: 存放批量 YUV 文件的 device buff 地址

- `const jpegEncParam* EncParams`: 包含批量编码参数的 buff 地址
- `int BatchSize`: 要进行批量编码的 YUV 文件个数
- `cudaStream_t stream`: stream 对象, 可以为 NULL

返回值

`nvjpegStatus_t`:

- `NVJPEG_STATUS_SUCCESS`: 成功
- `OTHERS`: 失败

3.4.12 `cujpegEncodeRetrieveBitstreamDeviceBatched`

功能概述

获取批量编码后的数据, 并保存到指定的 device buff 内。

函数原型

```
nvjpegStatus_t cujpegEncodeRetrieveBitstreamDeviceBatched(nvjpegHandle_t handle,
    nvjpegEncoderState_t encoder_state,
    unsigned char* data,
    size_t* length,
    int BatchSize,
    CUjpegstream stream);
```

参数说明

- `nvjpegHandle_t handle`: 使用 `cujpegCreate` 函数创建的 encoder handle 指针, 不能为 NULL
- `nvjpegEncoderState_t encoder_state`: 使用 `nvjpegEncoderStateCreate` 函数创建的对象, 不能为 NULL
- `unsigned char* data`: 保存编码后数据的 device buff 地址
- `size_t* length`: 保存编码后输出数据的长度
- `cudaStream_t stream`: stream 对象, 可以为 NULL

返回值

`nvjpegStatus_t`:

- `NVJPEG_STATUS_SUCCESS`: 成功
- `OTHERS`: 失败

3.4.13 `cujpegEncodeRetrieveBitstreamBatched`

功能概述

获取批量编码后的数据, 并保存到指定的 host buff 内。

函数原型

```
nvjpegStatus_t cujpegEncodeRetrieveBitstreamBatched(nvjpegHandle_t handle,
    nvjpegEncoderState_t encoder_state,
    unsigned char* data,
    size_t* length,
    int BatchSize,
    CUjpegstream stream);
```

参数说明

- nvjpegHandle_t handle: 使用 cujpegCreate 函数创建的 encoder handle 指针, 不能为 NULL
- nvjpegEncoderState_t encoder_state: 使用 nvjpegEncoderStateCreate 函数创建的对象, 不能为 NULL
- unsigned char* data: 保存编码后数据的 host buff 地址
- size_t* length: 保存编码后输出数据的长度
- cudaStream_t stream: stream 对象, 可以为 NULL

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.4.14 cujpegEncoderParamsSetFrameFormat

功能概述

设置编码 Param 中的 Format 信息。

函数原型

```
nvjpegStatus_t DRIVERAPI_EXPORT cujpegEncoderParamsSetFrameFormat(nvjpegEncoderParams_t
    encoder_params,
    const unsigned int format,
    cudaStream_t stream);
```

参数说明

- nvjpegEncoderParams_t encoder_params: 使用 nvjpegEncoderParamsCreate 函数创建的 nvjpegEncoderParams_t 对象指针, 不能为 NULL
- const unsigned int format: format 参数的值, 目前支持 [NONE: 0, NV12: 1, NV21: 2, YUYV: 3, UYVY: 4, VYU: 5, VYUY: 6], 不指定时该值将被默认设置为 0
- cudaStream_t stream: stream 对象, 可以为 NULL

返回值

nvjpegStatus_t:

- NVJPEG_STATUS_SUCCESS: 成功
- OTHERS: 失败

3.5 编解码用例使用指南

天数智算软件栈提供以下三个 iJPEG 的参考用例：

- 解码用例，ixjpegdec：您可以使用该用例对图片进行解码。
- 编码用例，ixjpegenc：您可以使用该用例对图片进行编码。
- 编解码混合用例，ixjpegdecenc：您可以使用该用例对图片先进行解码，再进行编码，最终输出处理过的图片。

以上三个参考用例均是基于简化编解码接口函数实现的，呈现形式为可执行文件，您可以在 `/root/corex-samples-{v.r.m}_{ARCH}/samples/cudasamples/samples/` 目录下找到。

3.5.1 解码用例使用指南

本小节内容将详尽说明 ixjpegdec 可执行文件的使用参数规则，并结合可执行文件源码，对函数接口调用进行说明。

用户使用说明

您可以使用 `./ixjpegdec -h` 打印提示信息用于查看 ixjpegdec 的使用说明。

ixjpegdec 的使用方法是：

```
$ ./ixjpegdec -i images_dir [-b batch_size] [-t total_images] [-devid device_id] [-w
warmup_iterations] [-o output_dir] [-batched] [-fmt output_format]
```

参数具体说明如下：

- **images_dir**：输入文件路径，可以为单个文件或文件夹。文件夹模式需显示指定，即 `-i ./jpegFile/`，请务必带上“/”
- **batch_size**：batch 接口中的批量数目
- **total_images**：解码文件数目，当 image_dir 中图片数目小于此参数，会 loop 循环图片直到此数目
- **device_id**：指定需要运行的设备 ID
- **warmup_iterations**：此数量解码，不进入性能测试范围
- **output_dir**：仅支持文件夹形式，具体文件名需和输入的文件名保持一致。例如，i.jpeg 会解码为 1.yuv。因此，待解码文件夹下不能有同名文件，否则将导致解码结果异常。此处与输入路径不同，请勿带有“/”
- **batched**：使用 batch 接口解码，尽管指定 batch_size，还是需要显示指定 batched
- **output_format**：指定图片输出格式，支持 [rgb, rgbi, bgr, bgri, yuv, unchanged, nv12, nv21]

Note

当指定 **batch_size** 时，若 **total_images** 参数大小不能整除 **batch_size**，则解码的文件数量为 $(total_images // batch_size) * batch_size$ ，超出一个 **batch_size** 后不够下一个 **batch_size** 数量的文件将被忽略。

3.5.2 编码用例使用指南

本小节内容将详尽说明 ixjpegenc 可执行文件的使用参数规则，并结合可执行文件源码，对函数接口调用进行说明。

用户使用说明

您可以使用 **./ixjpegenc -h** 打印提示信息用于查看 ixjpegenc 的使用说明。

ixjpegenc 的使用方法是：

```
$ ./ixjpegenc [-i images_dir] [-o output_dir] [-d device_id] [-q quality] [-s subsampling] [-b  
↪ batch size] [-w width] [-he height] [-f frameformat]
```

参数具体说明如下：

- **images_dir**: 单个 YUV 文件名称或多个 YUV 文件路径
- **output_dir**: 保存编码后 JPEG 文件的目录
- **device_id**: 指定需要运行的设备 ID
- **quality**: 设置编码质量参数，支持输入的数值范围是 [0-100]，不指定时该值将被默认设置为 70
- **subsampling**: YUV 文件的格式，目前支持 [444, 422, 420, 440, 400]，不指定时该值将被默认设置为 420
- **batch size**: 单次编码的 batch size，默认为 8，设置为 0 表示单张编码。该值没有最大限制，依赖于 Host 和 Device 的内存大小，过大将导致内存溢出，请谨慎设置
- **width**: YUV 文件的宽度
- **height**: YUV 文件的高度
- **frameformat**: YUV 数据的排布格式，目前支持 [NONE: 0, NV12: 1, NV21: 2, YUYV: 3, UYVY: 4, YVYU: 5, VYUY: 6]，不指定时该值将被默认设置为 0

3.5.3 编解码混合用例使用指南

天数智芯针对 ixJPEG 提供编解码混合用例 ixjpegdecenc，支持对输入的 JPEG 图片先进行解码，再进行编码，最终输出处理过的 JPEG 图片。

本小节内容将详尽说明 ixjpegdecenc 可执行文件的使用参数规则，并结合可执行文件源码，对函数接口调用进行说明。

用户使用说明

您可以使用 **./ixjpegdecenc -h** 打印提示信息用于查看 ixjpegdecenc 的使用说明。

ixjpegdecenc 的使用方法是：

```
$ ./ixjpegdecenc -i images_dir [-o output_dir] [-device=device_id][-q quality][-s 420/444] [-fmt  
↪ output_format] [-huf 0]
```

参数具体说明如下：

- **images_dir**: 单个文件的名称或多个文件的路径
- **output_dir**: 保存处理后 JPEG 文件的目录，该目录名与输入文件名保持一致
- **device_id**: 指定需要运行的设备 ID
- **quality**: 设置编码质量参数，支持输入的数值范围是 [0-100]，不指定时该值将被默认设置为 70
- **-s/subsampling**: YUV 文件的格式，目前支持 [444, 420] 两种，不指定时该值将被默认设置为 420
- **output_format**: 指定图片输出格式，支持 [rgb, rgbi, bgr, bgri, yuv]，不指定时该值将被默认设置为 0；使用 yuv 格式时，**-s/subsampling** 仅支持使用 420
- **-huf/huffman**: 设置 huffman 优化系数，不指定时该值将被默认设置为 0

4 ixCODEC

4.1 ixCODEC 库介绍

VPU (Video Processing Unit) 视频硬件解码模块，提供全加速硬件视频解码能力，支持多路同时解码，可以将压缩格式的视频、视频流（可以是文件格式或实时流）通过本解码模块输出为多种 YUV 格式。

本模块可用于多种常见视频压缩格式比特码流的硬件解码，并且完全运行独立于计算/图形引擎。VPU 能够解码压缩格式的视频流并允许开发者将生成的 YUV 帧数据复制到内存，可以使用 CUDA 进行视频后期处理。解码后的视频帧可以用于 GPU 加速推理或由 CUDA 或基于 CPU 的处理器进一步使用加工，也可以给具有图形互操作性的显示器以进行视频播放等。

本模块同时提供用于编程软件 API 与库所支持的 C++/Python API。VPU 模块所使用的函数库是天数智算软件栈内置的 ixCODEC 库，该函数库的默认安装路径为 /usr/local/corex/lib64/。

软件 API 允许开发人员访问 ixCODEC 的视频解码功能，并支持获取帧 buffer。相关流程如下图所示：

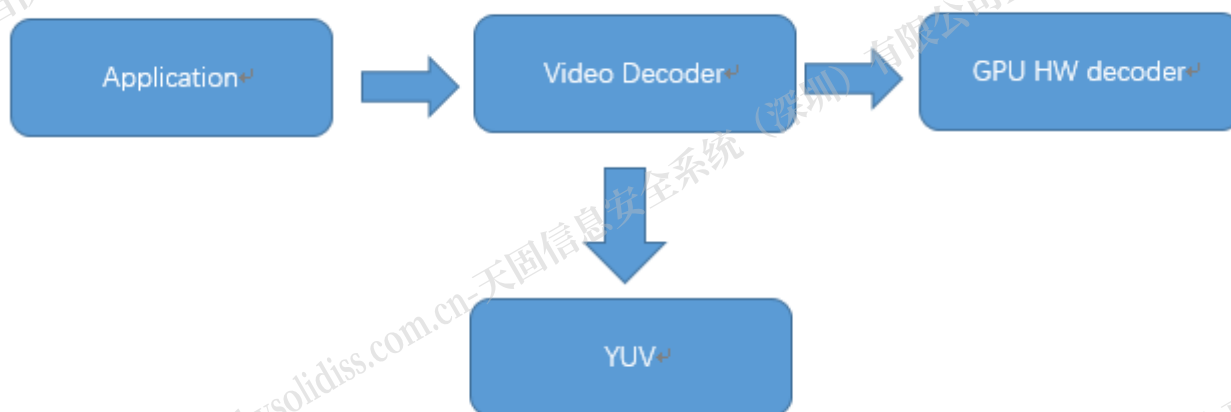


Figure 1: 使用 ixCODEC 库进行视频解码流程

天数智芯 VPU 模块目前支持的格式有：

- H264
- H265
- AVS2
- VP9

天数智算软件栈提供以下 ixCODEC API：

4.2 cuvidCreateDecoder

功能概述

创建一个解码实例。

函数原型

```
CUresult cuvidCreateDecoder(CUcontext cuda_ctx, CUvideodec *hDecoder, CUIDECODECREATEINFO
↳ dec_config);
```

参数说明

- cuda_ctx: cuda Context
- hDecoder: 函数调用后, 返回的解码句柄
- dec_config: 传递的解码参数

返回值

CUresult:

- CUDA_SUCCESS: 成功
- CUDA_ERROR_UNKNOWN 或 CUDA_ERROR_OUT_OF_MEMORY: 失败

4.3 cuvidDestroyDecoder

功能概述

销毁指定的解码实例。

函数原型

```
CUresult cuvidDestroyDecoder(CUvideodec hDecoder)
```

参数说明

- hDecoder: 解码句柄, cuvidCreateDecoder 函数返回的句柄

返回值

CUresult:

- CUDA_SUCCESS: 成功
- CUDA_ERROR_UNKNOWN: 失败

4.4 cuvidDecodePicture

功能概述

解码函数。

函数原型

```
CUresult cuvidDecodePicture(CUvideodec hDecoder, CUVIDPICPARAMS *pPicParams)
```

参数说明

- hDecoder: 解码句柄
- pPicParams: 解码参数的指针

返回值

CUresult:

- CUDA_SUCCESS: 成功
- CUDA_ERROR_NOT_READY: 失败

4.5 cuvidGetDecodeStatus

功能概述

获得解码状态。

函数原型

```
CUresult cuvidGetDecodeStatus(CUvideodec hDecoder, CUVIDGETDECODESTATUS* pDecodeStatus);
```

参数说明

- hDecoder: 解码句柄
- pDecodeStatus: 返回解码状态

返回值

CUresult:

- CUDA_SUCCESS: 成功
- CUDA_ERROR_INVALID_HANDLE: 失败

4.6 cuvidMapVideoFrame

功能概述

获取解码输出帧数据。

函数原型

```
CUresult cuvidMapVideoFrame(CUvideodec hDecoder, unsigned int *pDevPtr, unsigned int  
↪ *pPitch, CUVIDPROCPARAMS * pVPP);
```

参数说明

- hDecoder: 解码句柄
- pDevPtr: 帧 buffer
- pPitch: pitch
- pVPP: 返回其他参数, width, height 等

返回值

CUresult:

- CUDA_SUCCESS: 成功
- CUDA_ERROR_INVALID_HANDLE 或 CUDA_ERROR_NOT_READY: 失败

4.7 cuvidUnmapVideoFrame

功能概述

获取帧结束, 释放资源。

函数原型

```
CUresult cuvidUnmapVideoFrame(CUvideodec hDecoder, unsigned long long DevPtr, int picIdx);
```

参数说明:

- hDecoder: 解码句柄
- pDevPtr: 帧 buffer
- picIdx: 保留参数, 暂时无用

返回值

CUresult:

- CUDA_SUCCESS: 成功
- CUDA_ERROR_INVALID_HANDLE: 失败

4.8 cuvidDecodeGetPictureInfo

功能概述

获取解码后图片的宽高。

函数原型

```
CUresult cuvidDecodeGetPictureInfo(CUvideodec hDecoder, unsigned int * w, unsigned int * h)
```

参数说明

- CUvideodec hDecoder: 解码器 decoder 对象
- unsigned int * w: 图片宽

- unsigned int * h: 图片高

返回值

CUresult:

- CUDA_SUCCESS: 成功
- CUDA_ERROR_INVALID_HANDLE: 解码器 decoder 对象参数为 NULL

4.9 视频解码参考用例使用指南

天数智算软件栈提供了视频解码参考用例 decTestApp。该参考用例是基于简化解码接口函数实现的，呈现形式为可执行文件，您可以在 /root/corex-samples-{v.r.m}-{ARCH}/samples/cudasamples/samples/ 目录下找到。

本小节内容将详尽说明 decTestApp 可执行文件的使用参数规则，并结合可执行文件源码，对函数接口调用进行说明。

4.9.1 使用方法

视频解码用例的使用方法如下：

```
$ ./decTestApp --core-num=[core_number] --instance-num=[instance_number] -n [decode frames] --
  ↪ codec=[stream_type] --input=[input_stream_file] --output=[output_yuv]
```

参数说明

- **--core-num**: 使用的 VPU 总数，支持的数值范围是 [1, 12]，默认为 1，由 instance 数决定具体的 VPU 数
- **--instance-num**: 解码路数，支持的数值范围是 [1, 128]
- **-n**: 解码帧数，不设置则默认解码至无码流为止
- **--codec**: 码流类型，支持：
 - 0: H264
 - 12: HEVC
 - 14: AVS2
- **--input**: 输入的解码码流文件
- **--output**: 输出保存的 YUV 文件

注意事项

- 在使用解码用例时，您需要提前准备好码流文件。
- 在确定跑多少路数时，您需要进行综合考量，可根据芯片型号和内存决定。在内存较小的芯片上因解码内存需求较大不能跑满 128 路。以 1080P 标准码流为例，1 路需要的内存大约为 74M。不同码流编码复杂度不同，所需的显存也不同。

其它参数说明

下列参数为不常用参数，在此列出说明便于您理解：

- **--coreIdx**: 指定运行的 VPU, 默认为 0
- **--bs-size**: 指定 bitstream buffer 的大小, 默认为 2M。需要 2M 对齐, 指定时可以任意设置, 实际值会自动与 2M 对齐
- **--scaler**: 是否开启 scaler, 默认为 0, 即不开启 scaler。如需开启 scaler, 则指定 **--scaler=1**。开启 scaler 后, 需要与 **--sclw** 和 **--sclh** 参数组合使用
- **--sclw**: 设置 scale 的宽, 比如 1080P 的标准宽为 1920; 720P 的标准宽为 1280
- **--sclh**: 设置 scale 的高
- **--bsmode**: 设置码流模式, 默认为 0, 表示中断模式
- **--enable-wtl**: 是否开启 WTL 解码状态, 默认为 0, 即不开启 WTL 解码状态。如需开启 WTL 解码状态, 则指定 **--enable-wtl=1**
- **--wtl-format**: 输出 WTL 的格式, 默认为 0, 表示 YUV 格式
- **--stream-endian**: 流字节存储序, 根据平台及 mem 结构不同判断为大/小端字节序。支持的数值范围为 [16-31], 默认为 31, 默认为小端字节序
- **--frame-endian**: 帧字节存储序, 根据平台及 mem 结构不同判断为大/小端字节序。支持的数值范围为 [16-31], 默认为 31, 默认为小端字节序
- **-fps**: 输出帧率, 一般为 30FPS 或 25FPS。该参数仅为参考, 解码器会根据实际情况输出, 输出的帧率并非一定是设置的值

5 ixBLAS

BLAS (basic linear algebra subroutine) 是一系列基本向量和矩阵运算函数的接口标准。1 级 BLAS 执行标量、矢量和矢量-矢量运算，2 级 BLAS 执行矩阵-矢量运算，3 级 BLAS 执行矩阵-矩阵运算。

BLAS 是高效的、可移植的和广泛可用的，因此它们通常被广泛用于科学计算和工业界，已成为业界标准。天数智算软件栈提供内置的 ixBLAS 库，适配 CUDA 主流版本，包含批处理操作、跨多个 GPU 执行以及混精和半精度运算，并针对天数智芯加速卡进行了高度优化加速。

ixBLAS 库支持 IGIE 中使用的 API 与 INT8 的数据类型。

5.1 ixBLAS API

天数智算软件栈支持以下 BLAS API：

5.1.1 Helper function

```
cublasCreate
cublasDestroy
cublasGetAtomicsMode
cublasGetLoggerCallback
cublasGetMathMode
cublasGetMatrix
cublasGetMatrixAsync
cublasGetPointerMode
cublasGetProperty
cublasGetStream
cublasGetVector
cublasGetVectorAsync
cublasGetVersion
cublasLoggerConfigure
cublasSetAtomicsMode
cublasSetLoggerCallback
cublasSetMathMode
cublasSetMatrix
cublasSetMatrixAsync
cublasSetPointerMode
cublasSetStream
cublasSetVector
cublasSetVectorAsync
```


5.1.2 Level-1 function

```
cublasCaxpy  
cublasCcopy  
cublasCdotc  
cublasCdotu  
cublasCrot  
cublasCrotg  
cublasCscal  
cublasCsrot  
cublasCsscal  
cublasCswap  
cublasIcamax  
cublasIcamin  
cublasIsamax  
cublasIsamin  
cublasSasum  
cublasSaxpy  
cublasScasum  
cublasScnrm2  
cublasScopy  
cublasSdot  
cublasSnrm2  
cublasSrot  
cublasSrotg  
cublasSrotm  
cublasSrotmg  
cublasSscal  
cublasSswap
```

5.1.3 Level-2 function

```
cublasCgbmv  
cublasCgemv  
cublasCgerc  
cublasCgeru  
cublasChbmv  
cublasChemv  
cublasCher  
cublasCher2  
cublasChpmv  
cublasChpr  
cublasChpr2
```

```
cublasCsymv  
cublasCsyr  
cublasCsyr2  
cublasCtbmv  
cublasCtbsv  
cublasCtpmv  
cublasCtpsv  
cublasCtrmv  
cublasCtrsv  
cublasSgbmv  
cublasSgemv  
cublasSger  
cublasSsbmv  
cublasSspmv  
cublasSspr  
cublasSspr2  
cublasSsymv  
cublasSsyr  
cublasSsyr2  
cublasStbmv  
cublasStbsv  
cublasStpmv  
cublasStpsv  
cublasStrmv  
cublasStrsv
```

5.1.4 Level-3 function

```
cublasCgemm  
cublasCgemm3m  
cublasCgemmBatched  
cublasCgemmStridedBatched  
cublasChemm  
cublasCher2k  
cublasCherk  
cublasCherkx  
cublasCsymm  
cublasCsyr2k  
cublasCsyrk  
cublasCsyrkx  
cublasCtrmm  
cublasCtrsm  
cublasCtrsmBatched  
cublasHgemm
```

```
cublasHgemvBatched
cublasHgemvStridedBatched
cublasSgemv
cublasSgemvBatched
cublasSgemvStridedBatched
cublasSsymv
cublasSsymv2k
cublasSsymv
cublasSsymv
cublasStrmm
cublasStrsm
cublasStrsmBatched
```

5.1.5 Extension function

```
cublasAxpvEx
cublasCdgemv
cublasCgemv
cublasCgelsBatched
cublasCgemvEx
cublasCgeqrfBatched
cublasCgetrfBatched
cublasCgetriBatched
cublasCgetrsBatched
cublasCmatinvBatched
cublasCtptr
cublasCtrttp
cublasDotEx
cublasGemmBatchedEx
cublasGemmEx
cublasGemmStridedBatchedEx
cublasScalEx
cublasSdgemv
cublasSgemv
cublasSgelsBatched
cublasSgemvEx
cublasSgeqrfBatched
cublasSgetrfBatched
cublasSgetriBatched
cublasSgetrsBatched
cublasSmatinvBatched
cublasStptr
cublasStrttp
```

5.2 ixBLASLt API

天数智算软件栈支持以下 ixBLASLt API，提供基本矩阵乘和矩阵变换功能的实现支持：

```
cublasLtCreate
cublasLtDestroy
cublasLtGetCudartVersion
cublasLtGetProperty
cublasLtGetVersion
cublasLtMatmul
cublasLtMatmulAlgoCapGetAttribute
cublasLtMatmulAlgoCheck
cublasLtMatmulAlgoConfigGetAttribute
cublasLtMatmulAlgoConfigSetAttribute
cublasLtMatmulAlgoGetHeuristic
cublasLtMatmulAlgoGetIds
cublasLtMatmulAlgoInit
cublasLtMatmulDescCreate
cublasLtMatmulDescDestroy
cublasLtMatmulDescGetAttribute
cublasLtMatmulDescSetAttribute
cublasLtMatmulPreferenceCreate
cublasLtMatmulPreferenceDestroy
cublasLtMatmulPreferenceGetAttribute
cublasLtMatmulPreferenceSetAttribute
cublasLtMatrixLayoutCreate
cublasLtMatrixLayoutDestroy
cublasLtMatrixLayoutGetAttribute
cublasLtMatrixLayoutSetAttribute
cublasLtMatrixTransform
cublasLtMatrixTransformDescCreate
cublasLtMatrixTransformDescDestroy
cublasLtMatrixTransformDescGetAttribute
cublasLtMatrixTransformDescSetAttribute
```

6 ixFFT

天数智算软件栈提供内置的 ixFFT 库，适配 CUDA 主流版本，用于 GPU 加速的快速傅里叶变换实现。FFT 相关 API 旨在为天数智芯加速卡提供高性能，FFTW 相关 API 是作为移植工具提供的，以便使用 FFTW 库的用户能够以最小的工作量开始使用天数智芯加速卡。单卡环境下，天数智算软件栈支持以下 FFT API。

Important

目前天数智算软件栈 FFT 库暂不支持多卡。

6.1 cufft API

```
cufftCreate  
cufftDestroy  
cufftEstimate1d  
cufftEstimate2d  
cufftEstimate3d  
cufftEstimateMany  
cufftExecC2C  
cufftExecC2R  
cufftExecR2C  
cufftGetProperty  
cufftGetSize  
cufftGetSize1d  
cufftGetSize2d  
cufftGetSize3d  
cufftGetSizeMany  
cufftGetVersion  
cufftMakePlan1d  
cufftMakePlan2d  
cufftMakePlan3d  
cufftMakePlanMany  
cufftPlan2d  
cufftPlan3d  
cufftPlanMany  
cufftSetAutoAllocation  
cufftSetStream  
cufftSetWorkArea  
cufftPlan1d
```

6.2 cufftw API

```
fftwf_destroy_plan  
fftwf_execute  
fftwf_execute_dft  
fftwf_execute_dft_c2r  
fftwf_execute_dft_r2c  
fftwf_free  
fftwf_malloc  
fftwf_plan_dft  
fftwf_plan_dft_2d  
fftwf_plan_dft_3d  
fftwf_plan_dft_c2r  
fftwf_plan_dft_c2r_1d  
fftwf_plan_dft_c2r_2d  
fftwf_plan_dft_c2r_3d  
fftwf_plan_dft_r2c  
fftwf_plan_dft_r2c_1d  
fftwf_plan_dft_r2c_2d  
fftwf_plan_dft_r2c_3d  
fftwf_plan_guru_dft  
fftwf_plan_guru_dft_c2r  
fftwf_plan_guru_dft_r2c  
fftwf_plan_many_dft  
fftwf_plan_many_dft_c2r  
fftwf_plan_many_dft_r2c  
fftwf_plan_dft_1d
```

7 ixDNN

天数智算软件栈提供适配 cuDNN v7.6.5 的神经网络基元库 ixDNN。借助该加速库，开发者不必花时间针对天数智芯加速卡进行低层级的 GPU 性能调整，从而可以更专注于神经网络的训练和应用开发。

ixDNN 库支持 IGIE 中使用的 API 与 INT8 的数据类型。

天数智算软件栈支持以下 DNN API：

```

cudnnActivationBackward
cudnnActivationForward
cudnnAddTensor
cudnnBatchNormalizationBackward
cudnnBatchNormalizationBackwardEx
cudnnBatchNormalizationForwardInference
cudnnBatchNormalizationForwardTraining
cudnnBatchNormalizationForwardTrainingEx
cudnnConvolutionBackwardBias
cudnnConvolutionBackwardData
cudnnConvolutionBackwardFilter
cudnnConvolutionBiasActivationForward
cudnnConvolutionForward
cudnnCreate
cudnnCreateActivationDescriptor
cudnnCreateConvolutionDescriptor
cudnnCreateCTCLossDescriptor
cudnnCreateDropoutDescriptor
cudnnCreateFilterDescriptor
cudnnCreateFusedOpsConstParamPack
cudnnCreateFusedOpsPlan
cudnnCreateFusedOpsVariantParamPack
cudnnCreateLRNDescriptor
cudnnCreatePersistentRNNPlan
cudnnCreatePoolingDescriptor
cudnnCreateReduceTensorDescriptor
cudnnCreateRNNDescriptor
cudnnCreateTensorDescriptor
cudnnCTCLoss
cudnnDeriveBNTensorDescriptor
cudnnDestroy
cudnnDestroyActivationDescriptor
cudnnDestroyConvolutionDescriptor
cudnnDestroyCTCLossDescriptor
cudnnDestroyDropoutDescriptor
cudnnDestroyFilterDescriptor
cudnnDestroyFusedOpsConstParamPack
cudnnDestroyFusedOpsPlan
```

cudaDestroyFusedOpsVariantParamPack
cudaDestroyLRNDescriptor
cudaDestroyPersistentRNNPlan
cudaDestroyPoolingDescriptor
cudaDestroyReduceTensorDescriptor
cudaDestroyRNNDescriptor
cudaDestroyTensorDescriptor
cudaDropoutBackward
cudaDropoutForward
cudaDropoutGetReserveSpaceSize
cudaDropoutGetStatesSize
cudaFindConvolutionBackwardDataAlgorithm
cudaFindConvolutionBackwardDataAlgorithmEx
cudaFindConvolutionBackwardFilterAlgorithm
cudaFindConvolutionBackwardFilterAlgorithmEx
cudaFindConvolutionForwardAlgorithm
cudaFindConvolutionForwardAlgorithmEx
cudaGetActivationDescriptor
cudaGetBatchNormalizationBackwardExWorkspaceSize
cudaGetBatchNormalizationForwardTrainingExWorkspaceSize
cudaGetBatchNormalizationTrainingExReserveSpaceSize
cudaGetCallback
cudaGetConvolution2dDescriptor
cudaGetConvolution2dForwardOutputDim
cudaGetConvolutionBackwardDataAlgorithm
cudaGetConvolutionBackwardDataAlgorithm_v7
cudaGetConvolutionBackwardDataAlgorithmMaxCount
cudaGetConvolutionBackwardDataWorkspaceSize
cudaGetConvolutionBackwardFilterAlgorithm
cudaGetConvolutionBackwardFilterAlgorithm_v7
cudaGetConvolutionBackwardFilterAlgorithmMaxCount
cudaGetConvolutionBackwardFilterWorkspaceSize
cudaGetConvolutionForwardAlgorithm
cudaGetConvolutionForwardAlgorithm_v7
cudaGetConvolutionForwardAlgorithmMaxCount
cudaGetConvolutionForwardWorkspaceSize
cudaGetConvolutionMathType
cudaGetConvolutionNdDescriptor
cudaGetConvolutionNdForwardOutputDim
cudaGetCTCLossDescriptor
cudaGetCTCLossDescriptorEx
cudaGetCTCLossWorkspaceSize
cudaGetCudartVersion
cudaGetDropoutDescriptor
cudaGetErrorString
cudaGetFilter4dDescriptor

cudaGetFilterNdDescriptor
cudaGetFusedOpsConstParamPackAttribute
cudaGetFusedOpsVariantParamPackAttribute
cudaGetLRNDescriptor
cudaGetPooling2dDescriptor
cudaGetPooling2dForwardOutputDim
cudaGetPoolingNdDescriptor
cudaGetPoolingNdForwardOutputDim
cudaGetProperty
cudaGetReduceTensorDescriptor
cudaGetReductionIndicesSize
cudaGetRNNDescriptor
cudaGetRNNLinLayerBiasParams
cudaGetRNNLinLayerMatrixParams
cudaGetRNNParamsSize
cudaGetRNNProjectionLayers
cudaGetRNNTrainingReserveSize
cudaGetRNNWorkspaceSize
cudaGetStream
cudaGetTensor4dDescriptor
cudaGetTensorNdDescriptor
cudaGetTensorSizeInBytes
cudaGetVersion
cudaIm2Col
cudaLRNCrossChannelBackward
cudaLRNCrossChannelForward
cudaMakeFusedOpsPlan
cudaPoolingBackward
cudaPoolingForward
cudaReduceTensor
cudaRNNBackwardData
cudaRNNBackwardWeights
cudaRNNForwardInference
cudaRNNForwardTraining
cudaSetActivationDescriptor
cudaSetCallback
cudaSetConvolution2dDescriptor
cudaSetConvolutionGroupCount
cudaSetConvolutionMathType
cudaSetConvolutionNdDescriptor
cudaSetCTCLossDescriptor
cudaSetCTCLossDescriptorEx
cudaSetDropoutDescriptor
cudaSetFilter4dDescriptor
cudaSetFilterNdDescriptor
cudaSetFusedOpsConstParamPackAttribute

```
cudaDnnSetFusedOpsVariantParamPackAttribute
cudaDnnSetLRNDescriptor
cudaDnnSetPersistentRNNPlan
cudaDnnSetPooling2dDescriptor
cudaDnnSetPoolingNdDescriptor
cudaDnnSetReduceTensorDescriptor
cudaDnnSetRNNDescriptor
cudaDnnSetRNNDescriptor_v5
cudaDnnSetRNNDescriptor_v6
cudaDnnSetRNNMatrixMathType
cudaDnnSetRNNProjectionLayers
cudaDnnSetStream
cudaDnnSetTensor4dDescriptor
cudaDnnSetTensor4dDescriptorEx
cudaDnnSetTensorNdDescriptor
cudaDnnSetTensorNdDescriptorEx
cudaDnnSoftmaxBackward
cudaDnnSoftmaxForward
cudaDnnTransformTensor
```

7.1 flashAttn

为了方便高效地支持 flashAttn 算法实现，在 ixDNN 库中新增 flashAttn 相关的 API 接口。flashAttn 与 ixDNN 库中的 multi-head-attention (简称 MHA) 接口不同，MHA 包含 attention 前后 q, k, v 和 out 矩阵的线性变换和 scale dot-product attention (如下图所示)，而 flashAttn 只涉及 scale dot-product attention 计算。

Scaled Dot-Product Attention

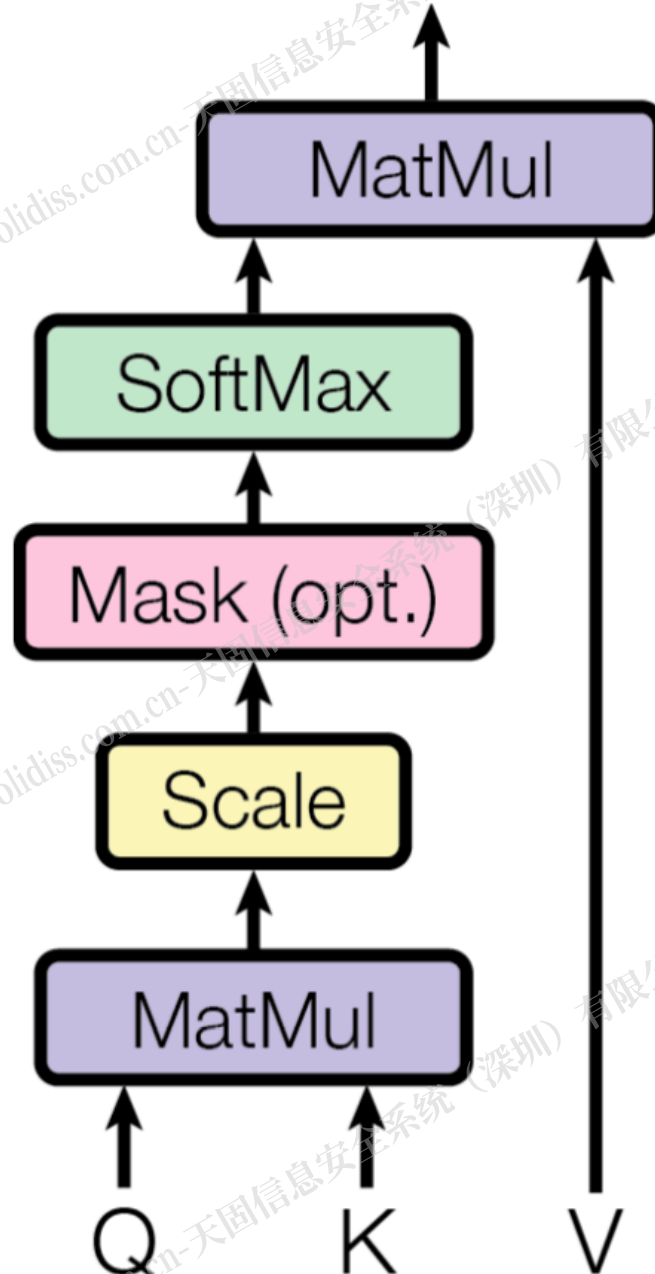


Figure 2: Scaled Dot-Product Attention

7.1.1 相关 API

```

cudnnFlashAttnLayout_t
cudnnFlashAttnConfigInfo
cudnnFlashAttnDescriptor_t
cudnnCreateFlashAttnDescriptor
cudnnDestroyFlashAttnDescriptor
cudnnGetFlashAttnBuffers
cudnnFlashAttnForward
cudnnFlashAttnBackward

```

7.1.2 相关 API 适用范围

- 目前仅支持 HeadDim = 64 和 128
- 目前仅支持 seq_len_src = seq_len_trg = seq_len 且 seq_len % 64 = 0 (seq_len 为 64 的倍数) 的 case
- 目前 HeadDim = 128 且 seq_len % 128 = 64 的 case 暂不支持
- 已支持 bhsd_unpack, bshd_unpack, bshd_qkv_pack, bshd_kv_pack 四种不同的 layout
- 已支持 Multi-Query-Attn 以及 Group-Query-Attn
- 已支持 softmaxMask broadcast 广播机制
- 已支持 is_causal 计算方式
- 已支持 is_causal + alibi 计算方式

7.1.3 API 调用方式

flashAttn API 的调用方式与 ixDNN 库中其他算子调用方式一致。

7.1.4 结构体说明

7.1.4.1 cudnnFlashAttnLayout_t

功能概述

用于指定 flashAttn 的输入输出数据排布。

函数原型

```

typedef enum {
    CUDNN_FATTN_BHSD_UNPACK = 0,
    CUDNN_FATTN_BSHD_UNPACK = 1,
    CUDNN_FATTN_BSHD_QKVPACK = 2,
    CUDNN_FATTN_BSHD_KVPACK = 3,
} cudnnFlashAttnLayout_t;

```

Layout 类型说明 (方向：由外至内)

- CUDNN_FATTN_BHSD_UNPACK: 对应 B(batch), H(head_num), S(seq_len), D(head_dim) 数据排布且 q,k,v tensor unpacked
- CUDNN_FATTN_BSHD_UNPACK: 对应 B(batch), S(seq_len), H(head_num), D(head_dim) 数据排布且 q,k,v tensor unpacked
- CUDNN_FATTN_BSHD_QKVPACK: 对应 B(batch), S(seq_len), H(head_num), D(head_dim) 数据排布且 q,k,v tensor packed
- CUDNN_FATTN_BSHD_KVPACK: 对应 B(batch), S(seq_len), H(head_num), D(head_dim) 数据排布且 k,v tensor packed

以上 4 种 Layout 类型的详细说明, 见 [Layout 说明](#)。

7.1.4.2 cudnnFlashAttnConfigInfo

功能概述

用于指定 flashAttn 的配置信息。

函数原型

```
struct cudnnFlashAttnConfigInfo {
    cudnnFlashAttnLayout_t layout;
    unsigned long long seed;
    float softmax_scale;
    float dropout_prob;
    bool is_causal;
    bool return_softmax_lse;
    bool is_alibi;
    int32_t alibi_mode;
    float* alibi_slopeM;
};
```

Config 信息说明

- layout: 用于指定 flashAttn 输入输出 tensor 的 layout 排布
- seed: 用于指定 flashAttn 开启 dropout 训练时的 dropout 随机数种子 (由于目前在 kernel 内固定了随机数种子, 该参数暂未使用)
- softmax_scale: 用于指定进行 softmax 计算之前的缩放因子
- dropout_prob: 用于指定 flashAttn 开启 dropout 训练时 dropout 丢弃的数据概率
- is_causal: 用于指定 flashAttn 是否在 softmax 计算时开启 is_causal 下三角 Mask 模式
- return_softmax_lse: 用于指定 flashAttn fwd 前向是否返回 flashAttn bwd 反向所需要的 softmax_lse 值
- is_alibi: 用于指定在 is_causal 模式下是否使用 alibi 偏置
- alibi_mode: 用于指定 alibi 偏置矩阵的模式, 目前有两种模式 (alibi_mode = 0; alibi_mode = 1)
- alibi_slopeM: 用于传入 alibi 模式下的 m 矩阵。其是一个 device 侧指针, 指向大小为 head_num_q 的一块 float 数据

7.1.5 函数说明

7.1.5.1 cudnnGetFlashAttnBuffers()

功能概述

用于给以下函数计算 workSpace 空间大小:

- cudnnFlashAttnForward()
- cudnnFlashAttnBackward()

函数原型

```

cudnnStatus_t CUDNNWINAPI
cudnnGetFlashAttnBuffers(
    cudnnHandle_t handle,
    const cudnnFlashAttnDescriptor_t flashAttnDesc,
    const uint32_t batch,
    const uint32_t head_num,
    const uint32_t seq_len_src,
    const uint32_t seq_len_trg,
    const uint32_t head_dim,
    bool isBackward,
    size_t* workspaceSizeInBytes);
    
```

参数说明

- handle: input, 当前 ixDNN/cuDNN 的 context handle
- flashAttnDesc: input, 指向已初始化的 flashAttn descriptor 指针
- batch: input, Tensor 的 batch 维度
- head_num: input, Tensor 的 head_num 维度
- seq_len_src: input, q, o Tensor 的 seq_len 维度
- seq_len_trg: input, k, v Tensor 的 seq_len 维度
- head_dim: input, Tensor 的 head_dim 维度
- isBackward: input, 指定计算 cudnnFlashAttnForward() 或 cudnnFlashAttnBackward() 的 workSpace 空间
 - true: cudnnFlashAttnBackward()
 - false: cudnnFlashAttnForward()
- workspaceSizeInBytes: output, 返回 forward 或者 backward 函数所需的 workSpace 空间大小

7.1.5.2 cudnnFlashAttnForward()

功能概述

用于完成 flashAttn forward 计算流程, 目前支持 4 种不同的 layout 且性能相近。

函数原型

```

cudannStatus_t CUDNNWINAPI
cudannFlashAttnForward(
    cudannHandle_t handle,
    const cudannFlashAttnDescriptor_t flashAttnDesc,
    const cudannFlashAttnConfigInfo& flashAttnInfo,
    const cudannTensorDescriptor_t qDesc,
    const cudannTensorDescriptor_t kDesc,
    const cudannTensorDescriptor_t vDesc,
    const cudannTensorDescriptor_t oDesc,
    const cudannTensorDescriptor_t maskDesc,

    const void* queries,
    const void* keys,
    const void* values,
    const void* softmaxMask,
    const int32_t* qoSeqArray,
    const int32_t* kvSeqArray,
    const int32_t* loWinIdx,
    const int32_t* hiWinIdx,
    const void* dropout_states,
    void* out,
    float* softmax_lse);
    
```

参数说明

- handle: 当前 ixDNN/cuDNN 的 context handle
- flashAttnDesc: input, 指向已初始化的 flashAttn descriptor 指针
- flashAttnInfo: input, 指明 flashAttn 计算所需的配置信息, 详情见 `cudaFlashAttnConfigInfo` 结构体说明
- qDesc: input, 描述 q Tensor 的信息, 包括 dataType, nb_dims, shape, stride
- kDesc: input, 描述 k Tensor 的信息, 包括 dataType, nb_dims, shape, stride
- vDesc: input, 描述 v Tensor 的信息, 包括 dataType, nb_dims, shape, stride
- oDesc: input, 描述 o Tensor 的信息, 包括 dataType, nb_dims, shape, stride
- maskDesc: input, 描述 mask Tensor 的信息, 包括 maskType, mask_nb_dims, maskShape, maskStride, 设置注意事项见下
- queries: input, 输入 q Tensor 在 Device 侧的指针
- keys: input, 输入 k Tensor 在 Device 侧的指针
- values: input, 输入 v Tensor 在 Device 侧的指针
- softmaxMask: input, 输入 softmax Mask 在 Device 侧的指针
- qoSeqArray: input, 长度为 batch 的一维数组, 用于描述每个 batch 有效的 q/o Tensor 的 seqLen 长度
- kvSeqArray: input, 长度为 batch 的一维数组, 用于描述每个 batch 有效的 k/v Tensor 的 seqLen 长度
- loWinIdx: input, 用于描述 softmax Mask 左闭右开区间的左端点
- hiWinIdx: input, 用于描述 softmax Mask 左闭右开区间的右端点
- dropout_states: input, 传入根据 seed 和 dropout_prob 计算得到的随机数 Tensor (当前未使用)
- out: output, 输出 o Tensor 在 Device 侧的指针
- softmax_lse: output, 输出 lse Tensor 在 Device 侧的指针, 仅当 Config 配置信息中 return_softmax_lse = true 时返回有效值

7.1.5.3 cudnnFlashAttnBackward()

功能概述

用于完成 flashAttn backward 计算流程，目前支持 4 种不同的 layout 且性能相近。

函数原型

```

cudnnStatus_t CUDNNWINAPI
cudnnFlashAttnBackward(
    cudnnHandle_t handle,
    const cudnnFlashAttnDescriptor_t flashAttnDesc,
    const cudnnFlashAttnConfigInfo& flashAttnInfo,
    const cudnnTensorDescriptor_t qDesc,
    const cudnnTensorDescriptor_t kDesc,
    const cudnnTensorDescriptor_t vDesc,
    const cudnnTensorDescriptor_t oDesc,
    const cudnnTensorDescriptor_t maskDesc,
    const void* queries,
    const void* keys,
    const void* values,
    const void* out,
    const void* dout,
    const float* softmax_lse,
    const void* softmaxMask,
    const int32_t* qoSeqArray,
    const int32_t* kvSeqArray,
    const int32_t* loWinIdx,
    const int32_t* hiWinIdx,
    const void* dropout_states,
    void* workSpace,
    void* dqueries,
    void* dkeys,
    void* dvalues);

```

参数说明

- handle: 当前 ixDNN/cuDNN 的 context handle
- flashAttnDesc: input, 指向已初始化的 flashAttn descriptor 指针
- flashAttnInfo: input, 指明 flashAttn 计算所需的配置信息，详情见 [cudnnFlashAttnConfigInfo](#) 结构体说明
- qDesc: input, 描述 q Tensor 的信息，包括 dataType, nb_dims, shape, stride
- kDesc: input, 描述 k Tensor 的信息，包括 dataType, nb_dims, shape, stride
- vDesc: input, 描述 v Tensor 的信息，包括 dataType, nb_dims, shape, stride
- oDesc: input, 描述 o Tensor 的信息，包括 dataType, nb_dims, shape, stride
- maskDesc: input, 描述 mask Tensor 的信息，包括 maskType, mask_nb_dims, maskShape, maskStride, 设置注意事项见下

- queries: input, 输入 q Tensor 在 Device 侧的指针
- keys: input, 输入 k Tensor 在 Device 侧的指针
- values: input, 输入 v Tensor 在 Device 侧的指针
- out: input, 前向计算得到的 output Tensor 在 Device 侧的指针
- dout: input, output 对应梯度 Tensor: dout 的 Device 侧指针
- softmax_lse: input, 前向计算得到的 softmax_lse 值在 Device 侧的指针, 用于存放前向进行 softmax 计算时保留的中间结果
- softmaxMask: input, 输入 softmax Mask 在 Device 侧的指针
- qoSeqArray: input, 长度为 batch 的一维数组, 用于描述每个 batch 有效的 q/o Tensor 的 seqLen 长度
- kvSeqArray: input, 长度为 batch 的一维数组, 用于描述每个 batch 有效的 k/v Tensor 的 seqLen 长度
- loWinIdx: input, 用于描述 softmax Mask 左闭右开区间的左端点
- hiWinIdx: input, 用于描述 softmax Mask 左闭右开区间的右端点
- dropout_states: input, 传入根据 seed 和 dropout_prob 计算得到的随机数 Tensor (当前未使用)
- workSpace: input, 当前函数进行计算时所需的临时空间, 大小由 cudnnGetFlashAttnBuffers() 函数获得
- dqueries: output, 输入 q Tensor 在进行反向计算时得到的 dq 梯度输出
- dkeys: output, 输入 k Tensor 在进行反向计算时得到的 dk 梯度输出
- dvalues: output, 输入 v Tensor 在进行反向计算时得到的 dv 梯度输出

7.1.6 接口调用说明

7.1.6.1 不同 Layout 调用说明

目前该 flashAttn 函数接口能够支持 4 种不同的 layout, 不同 layout 调用方式如下:

首先, 设置 cudnnFlashAttnConfigInfo 配置结构体种 layout 成员变量为对应所需的 layout。其次, 不同 layout 设置不同的 Tensor descriptor:

- CUDNN_FATTN_BHSD_UNPACK:
 - qShape: {batch, head_num_q, seq_len_src, head_dim}; qStride: {head_num_q * seq_len_src * head_dim, seq_len_src * head_dim, head_dim, 1}
 - kShape: {batch, head_num_kv, seq_len_trg, head_dim}; kStride: {head_num_kv * seq_len_trg * head_dim, seq_len_trg * head_dim, head_dim, 1}
 - vShape: {batch, head_num_kv, seq_len_trg, head_dim}; vStride: {head_num_kv * seq_len_trg * head_dim, seq_len_trg * head_dim, head_dim, 1}
 - oShape: {batch, head_num_q, seq_len_src, head_dim}; oStride: {head_num_q * seq_len_src * head_dim, seq_len_src * head_dim, head_dim, 1}
- CUDNN_FATTN_BSHD_UNPACK:
 - qShape: {batch, seq_len_src, head_num_q, head_dim}; qStride: {seq_len_src * head_num_q * head_dim, head_num_q * head_dim, head_dim, 1}
 - kShape: {batch, seq_len_trg, head_num_kv, head_dim}; kStride: {seq_len_trg * head_num_kv * head_dim, head_num_kv * head_dim, head_dim, 1}
 - vShape: {batch, seq_len_trg, head_num_kv, head_dim}; vStride: {seq_len_trg * head_num_kv * head_dim, head_num_kv * head_dim, head_dim, 1}
 - oShape: {batch, seq_len_src, head_num_q, head_dim}; oStride: {seq_len_src * head_num_q * head_dim, head_num_q * head_dim, head_dim, 1}

- CUDNN_FATTN_BSHD_QKVPACK:
 - qShape: {batch, seq_len_src, head_num_q, head_dim}; qStride: {seq_len_src * (head_num_q + head_num_kv * 2) * head_dim, (head_num_q + head_num_kv * 2) * head_dim, head_dim, 1}
 - kShape: {batch, seq_len_trg, head_num_kv, head_dim}; kStride: {seq_len_trg * (head_num_q + head_num_kv * 2) * head_dim, (head_num_q + head_num_kv * 2) * head_dim, head_dim, 1}
 - vShape: {batch, seq_len_trg, head_num_kv, head_dim}; vStride: {seq_len_trg * (head_num_q + head_num_kv * 2) * head_dim, (head_num_q + head_num_kv * 2) * head_dim, head_dim, 1}
 - oShape: {batch, seq_len_src, head_num_q, head_dim}; oStride: {seq_len_src * head_num * head_dim, head_num * head_dim, head_dim, 1}
- CUDNN_FATTN_BSHD_KVPACK:
 - qShape: {batch, seq_len_src, head_num_q, head_dim}; qStride: {seq_len_src * head_num_q * head_dim, head_num_q * head_dim, head_dim, 1}
 - kShape: {batch, seq_len_trg, head_num_kv, head_dim}; kStride: {seq_len_trg * 2 * head_num_kv * head_dim, 2 * head_num_kv * head_dim, head_dim, 1}
 - vShape: {batch, seq_len_trg, head_num_kv, head_dim}; vStride: {seq_len_trg * 2 * head_num_kv * head_dim, 2 * head_num_kv * head_dim, head_dim, 1}
 - oShape: {batch, seq_len_src, head_num_q, head_dim}; oStride: {seq_len_src * head_num_q * head_dim, head_num_q * head_dim, head_dim, 1}

7.1.6.2 softmax 多种 Mask 接口调用说明

目前接口支持 3 种方式对 softmax 进行 Mask 操作

- is_causal: 优先级最高, 通过 cudnnFlashAttnConfigInfo 配置结构体中 is_causal 成员变量控制。当 is_causal 设置为 true 时, 其他两种 Mask 方式失效; 自由度最低, 只能使用下三角 Mask 进行计算
- loWinIdx/hiWinIdx: 自由度稍高, 能够支持 [loWinIdx, hiWinIdx) 形式的 Mask
- softmaxMask: 优先级最低, 当 cudnnFlashAttnConfigInfo 配置结构体中 is_causal 变量设置为 False 且 loWinIdx/hiWinIdx 均为 nullptr 时生效。自由度最高, 能够支持任意形式的 Mask

Mask 配置及生效情况见下表:

Mask 配置	Mask 配置	Mask 配置	生效情况
is_causal	IoWin/hiWin	softmaxMask	/
true	-----	-----	仅 is_causal 生效, softmax 使用下三角 Mask 进行计算
false	nullptr	nullptr	仅 is_causal 生效, softmax 不作任何 Mask, 进行计算
false	非 nullptr	-----	is_causal 不生效, loWin/hiWin 生效, softmax 使用 [loWin, hiWin) 进行 Mask 计算

Mask 配置	Mask 配置	Mask 配置	生效情况
false	nullptr	非 nullptr	仅 softmaxMask 生效, softmax 使用传入的 Mask 矩阵进行 Mask 计算

softmaxMask 接口详细说明

功能说明:

- softmaxMask 为四维 Tensor, 其 shape 为: [batch, head_num, seq_len_src, seq_len_trg]
- softmaxMask 支持 Broadcast 操作, 即: 以上对应四个维度中, 任意维度为 1 时支持广播为对应维度
- softmaxMask 支持两种 Mask 类型:
 - Mask 值为 int 类型, 存放 0/1 的数据, 当 Mask 的值为 1 时进行掩码, 值为 0 时不进行掩码
 - Mask 值为 float 类型, 存放浮点数据, 此时 load mask 的值与对应位置处的结果进行 eltwise add 操作, 从而得到最终结果

性能说明:

由于使用 softmaxMask 接口时, 需要额外 load mask tensor 且消耗额外的 vrf 存放 Mask, 因此相较于前 3 种性能会有相应的下降。各种不同的 Mask shape 下性能对比如下表所示:

kernel 模式	MaskShape	性能情况
BROADCAST_ALL	b, h, 1, 1 b, 1, 1, 1 1, h, 1, 1 1, 1, 1, 1	Mask 情况下最优
BROADCAST_Q	b, h, 1, s b, 1, 1, s 1, h, 1, s 1, 1, 1, s	Mask 情况下次优
BROADCAST_K	b, h, s, 1 b, 1, s, 1 1, h, s, 1 1, 1, s, 1	Mask 情况下次优
BROADCAST_NON E	b, h, s, s b, 1, s, s 1, h, s, s 1, 1, s, s	Mask 情况下最劣

由上表可以看到，如果对于性能敏感，softmax 多种 Mask 接口使用的性能优先级为：is_causal > winIdx > softmaxMask-BROADCAST_ALL > BROADCAST_Q = BROADCAST_K > BROADCAST_NONE。

如果从接口功能来看，softmax 多种 Mask 接口使用的灵活度优先级为：softmaxMask > winIdx > is_causal。

Important

由于 flashAttn kernel 模板参数过多导致编译时间过长，目前注释掉 BROADCAST_ALL 和 BROADCAST_K 两种 Mask 广播方式以及 Mask 为 float 类型的方式。

7.1.7 Layout 说明

目前增加了 4 种不同 layout 的支持后，暂不支持 (head_dim = 128 且 seqlen % 128 = 64) 的 case。下面 4 种 layout 展示，均以行优先的方式存储。

7.1.7.1 CUDNN_FATTN_BHSD_UNPACK

q,k,v,o Tensor 数据排布如下图所示：

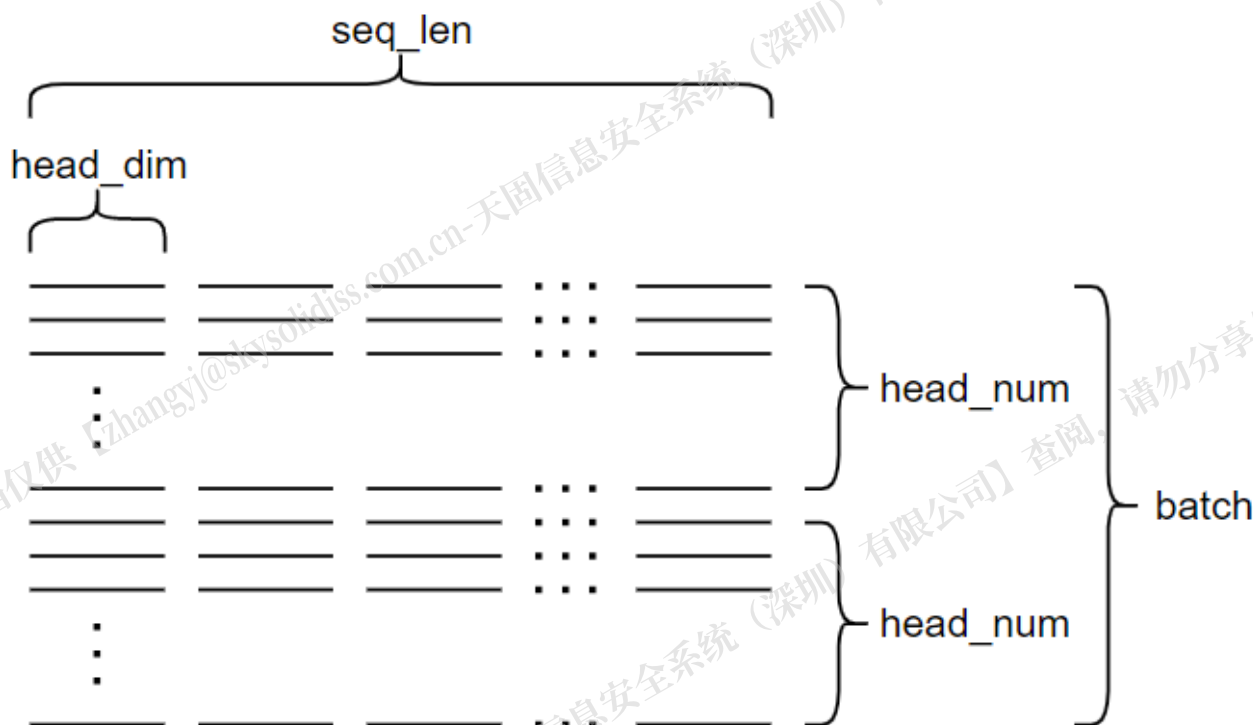


Figure 3: q,k,v,o Tensor 数据排布

softmax_lse 数据排布如下图所示：

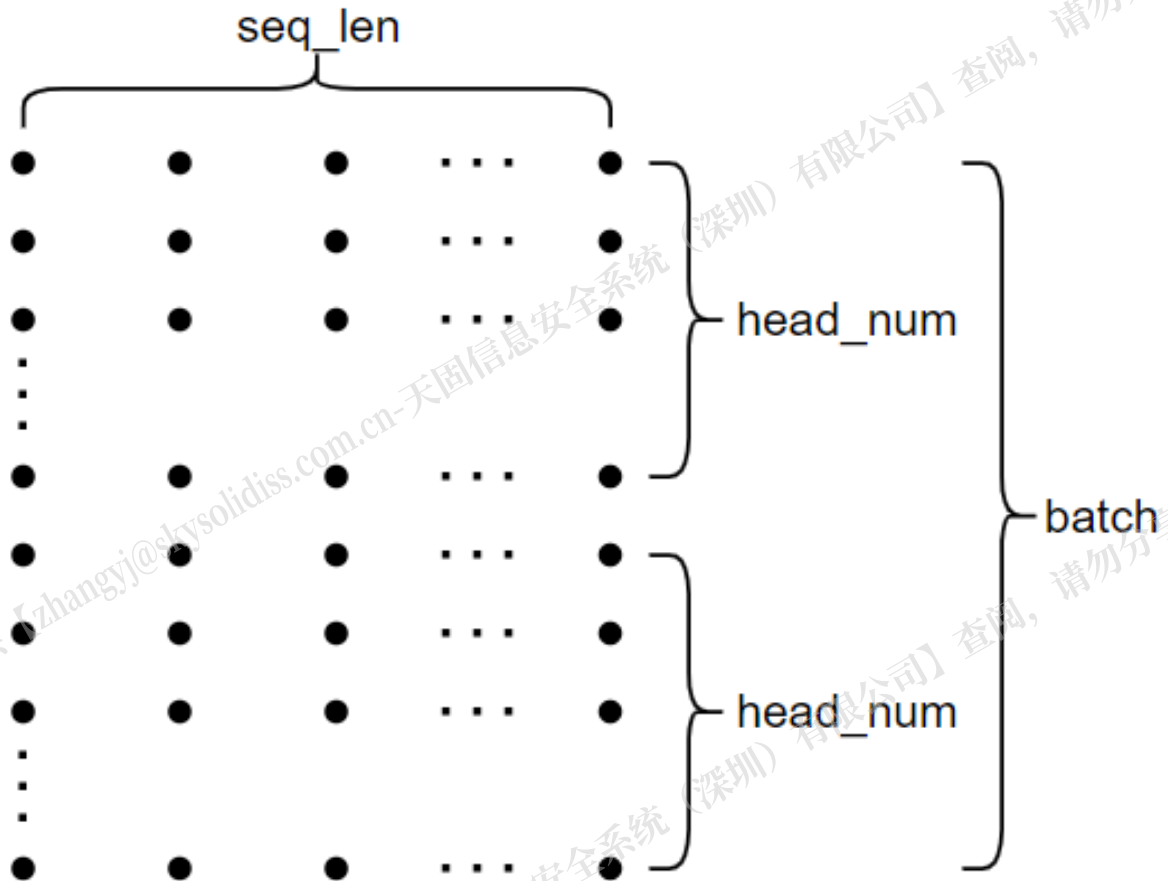


Figure 4: softmax_lse 数据排布

7.1.7.2 CUDNN_FATTN_BSHD_UNPACK

q,k,v,o Tensor 数据排布如下图所示：

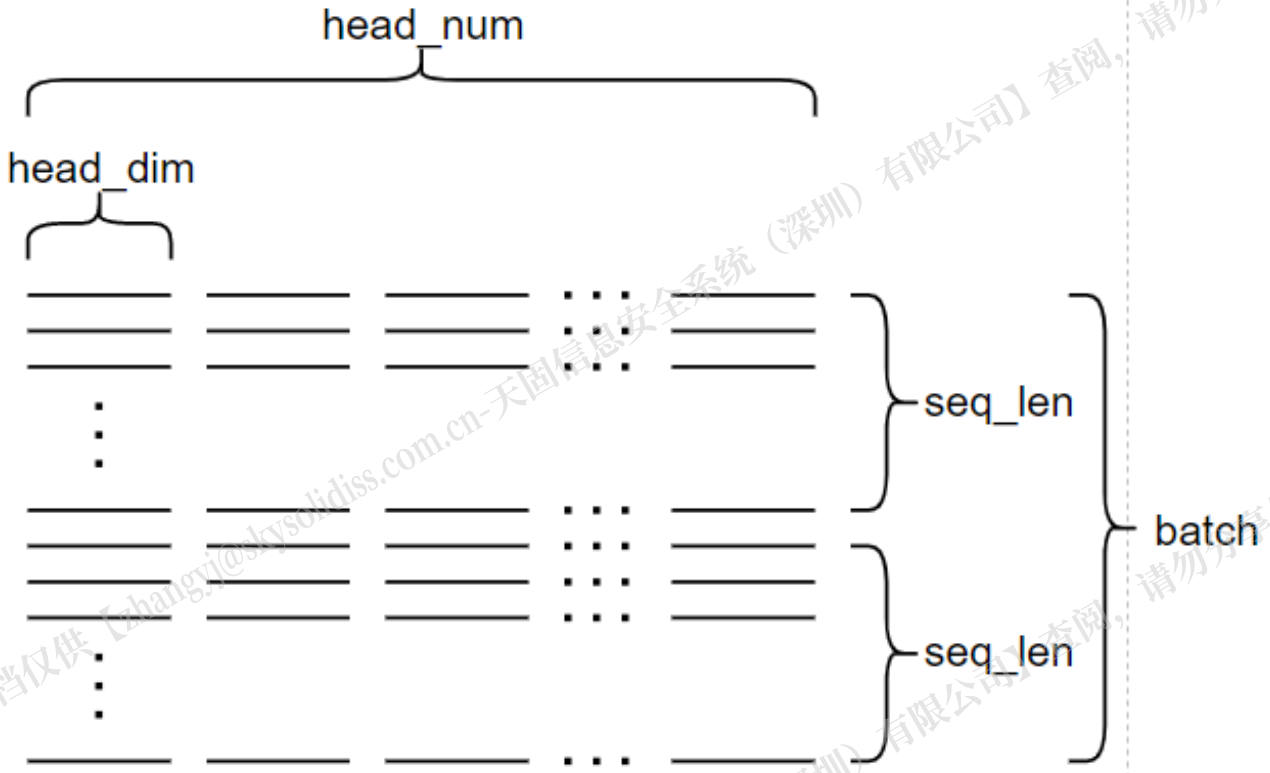


Figure 5: q,k,v,o Tensor 数据排布

softmax_lse 数据排布如下图所示：

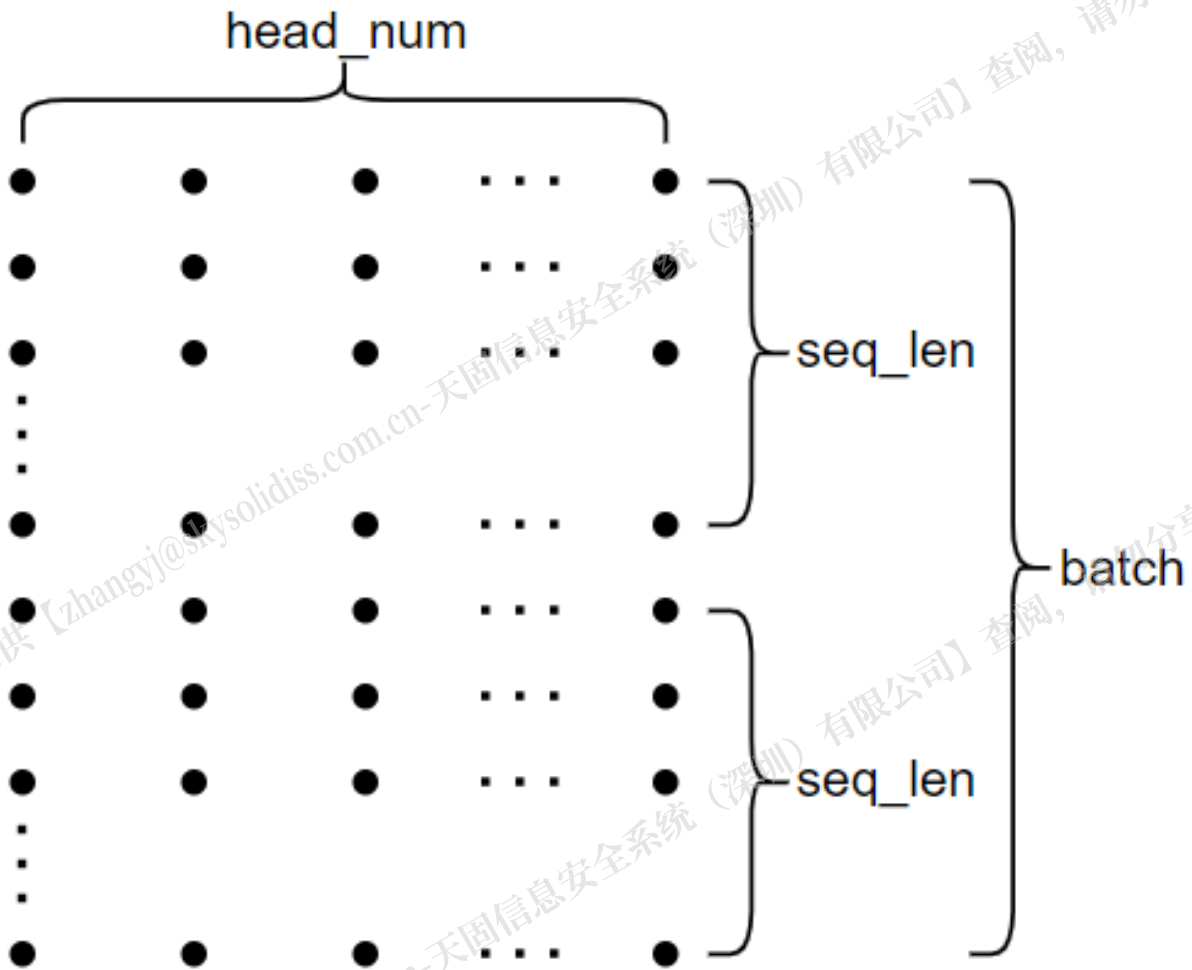


Figure 6: softmax_lse 数据排布

7.1.7.3 CUDNN_FATTN_BSHD_QKVPACK

`o/do/softmax_lse` 数据排布同 `CUDNN_FATTN_BSHD_UNPACK` 中 `bshd_unpack` 数据排布一致。

`q,k,v` 和 `dq,dk,dv` 数据排布如下图所示：

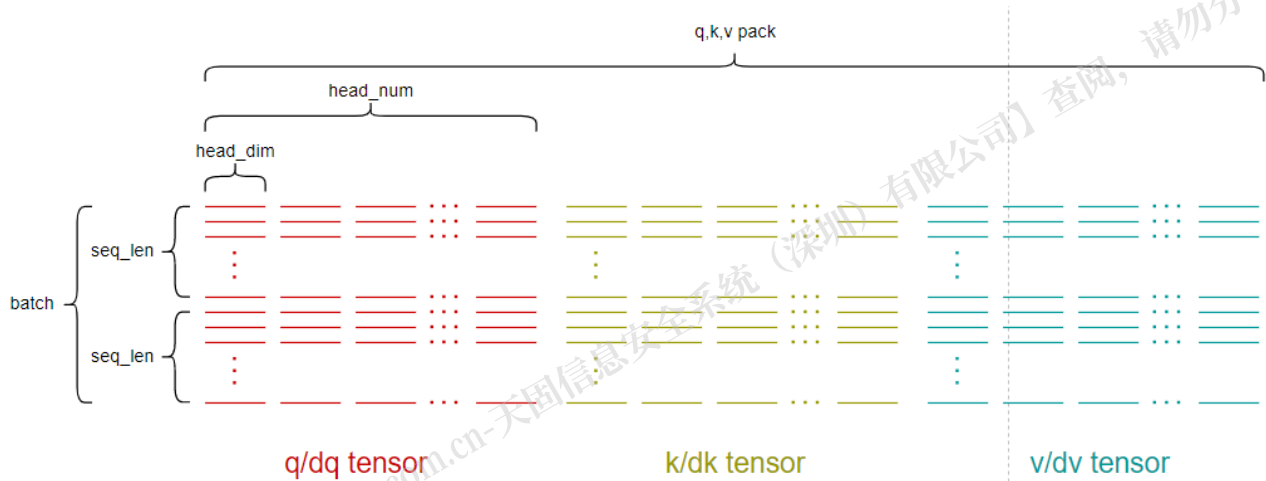


Figure 7: q,k,v 和 dq,dk,dv 数据排布

7.1.7.4 CUDNN_FATTN_BSHD_KVPACK

q/dq, o/do, softmax_lse 数据排布同CUDNN_FATTN_BSHD_UNPACK 中 bshd_unpack 数据排布一致。

k,v 和 dk,dv 数据排布如下所示：

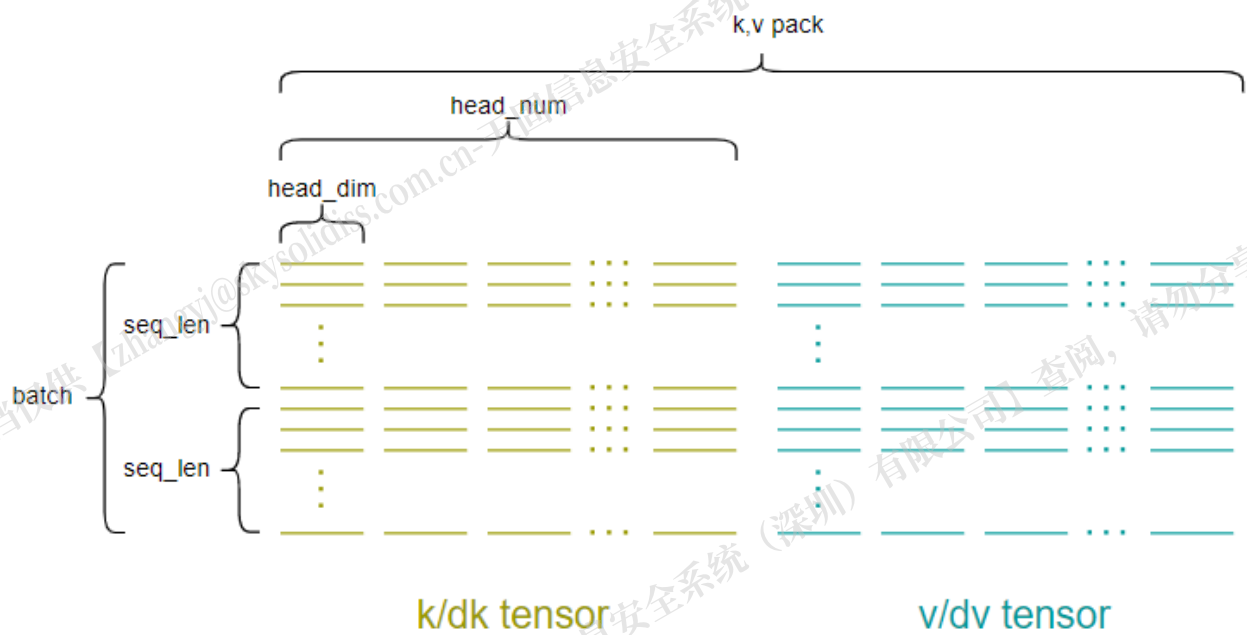


Figure 8: k,v 和 dk,dv 数据排布

7.1.8 group-query-attn 支持

Note

multi-query-attn(mqa) 为 group_query-attn(gqa) 在 head_num_kv == 1 时的特例。

GQA 算法中, k,v Tensor 的 head_num_kv 为 q Tensor 的 head_num_q 的因子, 即 $\text{head_num_q} \% \text{head_num_kv} == 0$ 。因此, 在进行计算时, 需要将 k,v 输入的 head_num_kv repeat 拷贝为 head_num_q, 在输出时要将计算得到的 dk,dv 的 head_num_q reduce 规约为 head_num_kv。

当前 flashAttn 算子支持 GQA 后的输入输出 layout 对应情况

算法	输入 layout	gqa 模式	输出 layout	说明
flashAttn fwd	bhsd_unpack	非 gqa	bhsd	对于前向输出 o, 其 tensor shape 与 q shape 一致, 且 q shape 不受 gqa 的影响, head_num 固定为 hn_q, 所以前向 8 种情况下对应的输出 layout 与输入 q 均一致
	bshd_unpack	gqa	bhsd	
	bshd_qkv_pack	非 gqa	bshd bshd	
	bshd_kv	gqa	bshd	
	_pack	非 gqa	bshd	
		gqa	bshd	
		非 gqa	bshd	
		gqa		
flashAttn bwd	bhsd_unpack	非 gqa	bhsd_unpack	对于反向输出 dq,dk,dv, 在非 gqa 模式下, 输出 layout 与输入 q,k,v layout 一致; 但在 gqa 模式下, 由于商定交给框架去做 dk,dv 的规约操作, 而框架 (PyTorch) 无法支持跨 stride 的规约计算, 因此 dq,dk,dv 的 layout 均为对应输入 layout 的 unpack 模式
	bshd_unpack	gqa	bhsd_unpack	
	bshd_qkv_pack	非 gqa	bshd_unpack	
	bshd_kv	gqa	bshd_qkv_pack	
	_pack	非 gqa	bshd_unpack	
		gqa	bshd_kv	
		非 gqa	_pack	
		gqa	bshd_unpack	

目前已知不同的 repeat 方式

示例 1:

head_num_q: [0,1,2,3,4,5]

head_num_kv: [0,1] $\square$ head_num_q: [0,0,0,1,1,1]

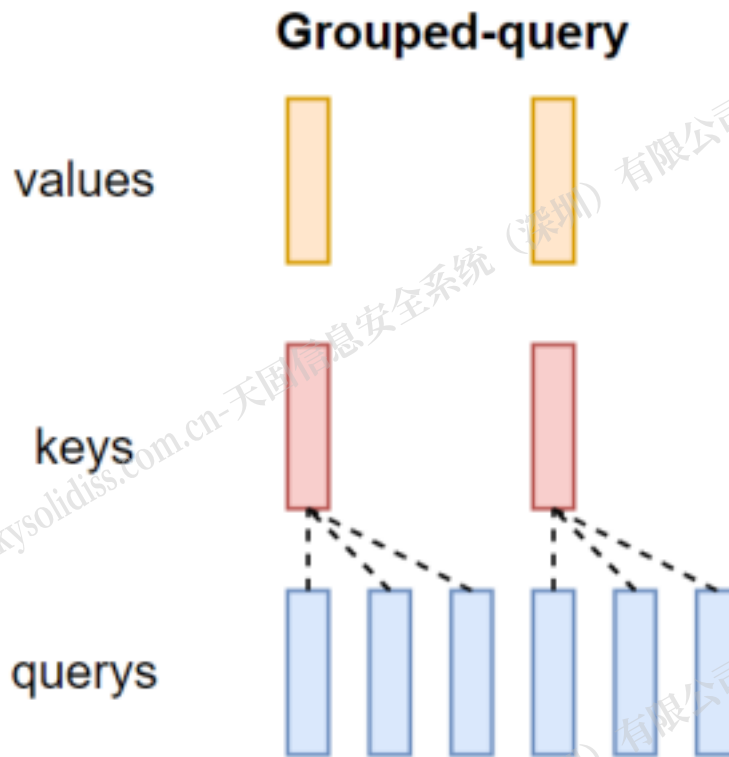


Figure 9: 示例 1

当前 flashAttn 所支持的 repeat 方式。

示例 2:

head_num_q: [0,1,2,3,4,5]

head_num_kv: [0,1] □ head_num_q: [0,1,0,1,0,1]

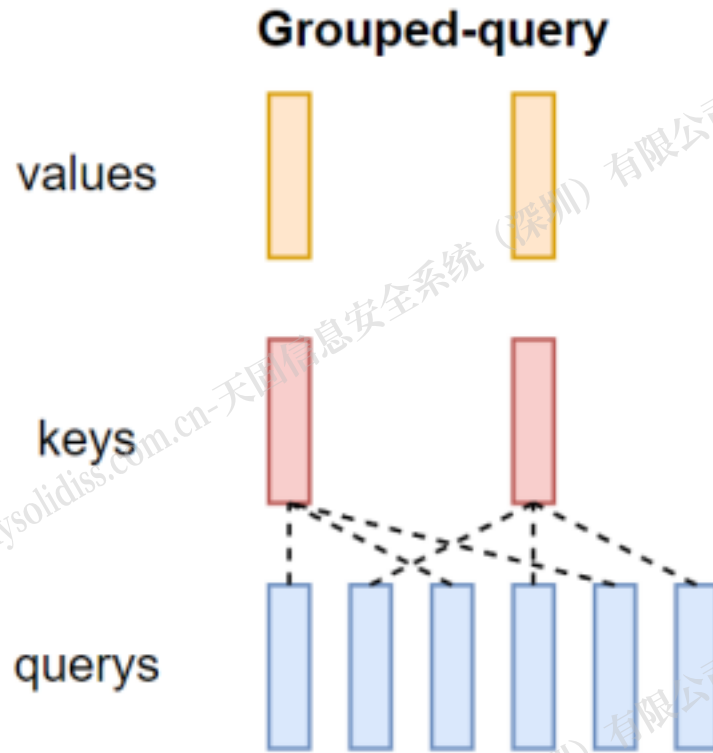


Figure 10: 示例 2

当前 flashAttn 暂未支持该 repeat 方式，待后续支持。

目前已知的 reduce 方式 (目前 flashAttn 算子不做规约操作，交由框架去实现)

示例：

head_num_q: [0,1,2,3,4,5]

head_num_kv: [0,0,0,1,1,1] □ head_num_kv: [0,1]

需由框架或上层调用 flashAttn 接口处理完成上述的 reduce 规约操作。

7.1.9 新增 Alibi 支持

7.1.9.1 Alibi 计算流程说明

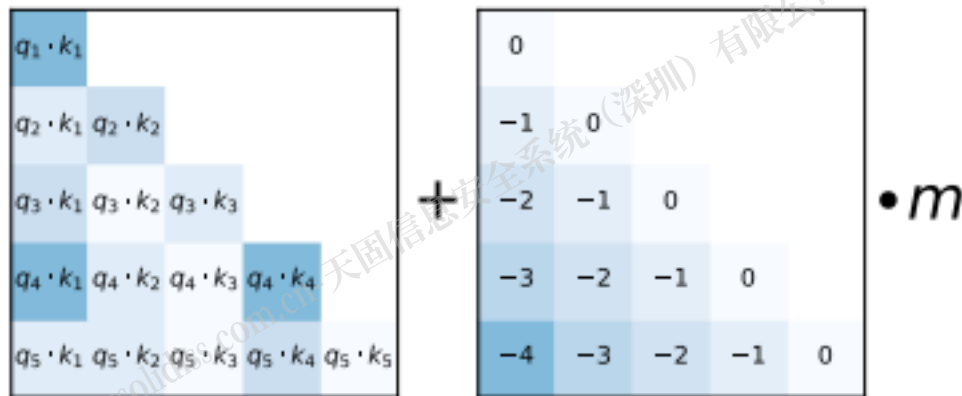


Figure 11: Alibi 计算流程

计算 MaskedSoftmax:

- 原 flashAttnFwd: $\text{softmax}(Q \cdot (K^T) \cdot \text{scale})$
- flashAttnFwd_with_alibi: $\text{softmax}(Q \cdot (K^T) \cdot \text{scale} + \text{BiasMatrix} \cdot m)$

即, Alibi 功能相较于之前, 在完成第一步 Q 与 K 的 gemm 以及 scale 计算之后, 需要额外加上一个偏置矩阵, 之后再进行 softmax 计算。

7.1.9.2 Alibi 偏置矩阵

Alibi 偏置矩阵 = $\text{BiasMatrix} \cdot m$, 其中 m 为长度 head_num_q 大小的浮点数组 (即相同的 head_num_q 维度共享同一个 m 值), BiasMatrix 则是一个 $\text{seq_len_src} \cdot \text{seq_len_trg}$ 大小的有规律矩阵, 目前支持的两种 BiasMatrix 类型如下:

1. $\text{alibi_mode} = 0$

BiasMatrix 中数值计算公式:

- $\text{row} \geq \text{col}$, $\text{data}[\text{row}][\text{col}] = \text{col} - \text{row}$
- $\text{row} < \text{col}$, $\text{data}[\text{row}][\text{col}] = 0$

即为一个下三角矩阵, 其具体效果如下图所示:

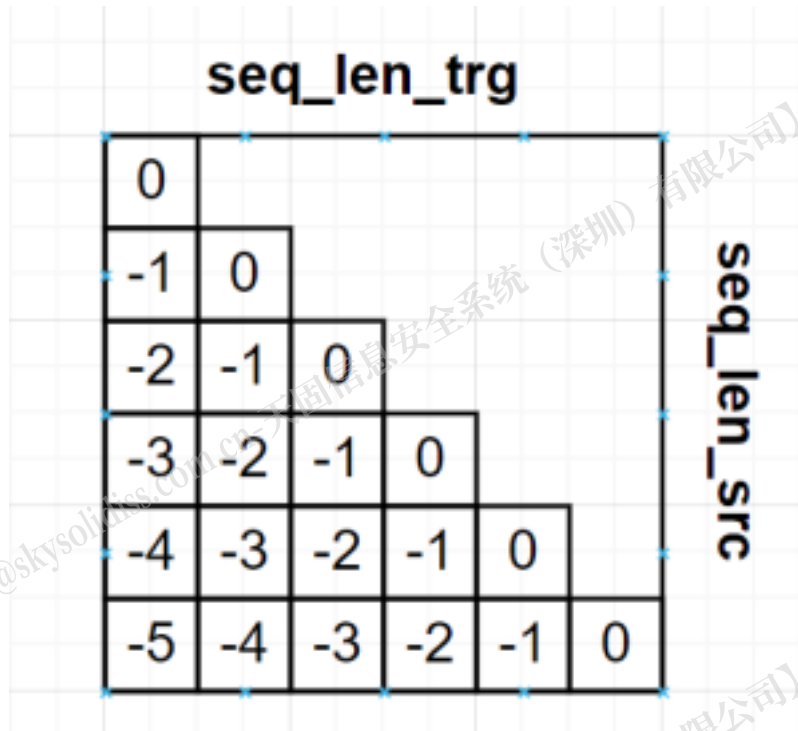


Figure 12: 下三角矩阵效果

2. alibi_mode = 1

BiasMatrix 中数值计算公式:

- row $\geq$ col, data[row][col] = $-\text{sqrt}(\text{row} - \text{col})$
- row < col, data[row][col] = 0

即为一个下三角矩阵, 其具体效果如图:

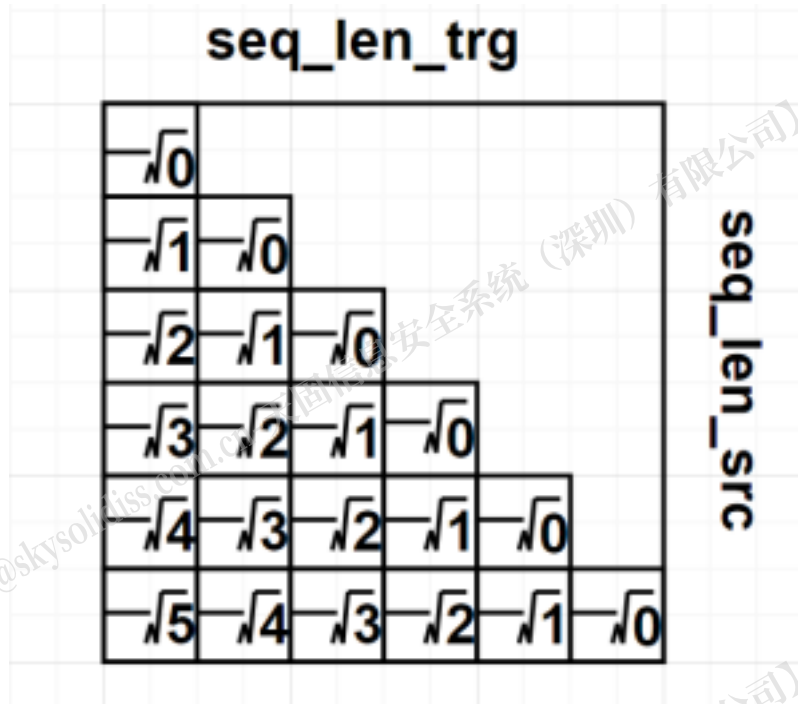


Figure 13: 下三角矩阵效果

Note

由于前端框架需求，目前将 Alibi 偏置矩阵 hardcode，对应 slopeM 不使用接口传入的值，而是在 kernel 中 hardcode。

slopeM 数组 hardcode 逻辑：

```
uint32_t hn_ = exp2(floor(log2(head_num)));
float slopeM = (hn < hn_) ? powf(powf(2, -8.0f / hn_), hn + 1) : powf(powf(2, -4.0f / hn_), 2 *
    ↪ (hn - hn_) + 1);
```

示例：

head_num = 10 □ hn_ = 8, slopeM[8] = { $2^{(-1)}$, $2^{(-2)}$, $2^{(-3)}$, $2^{(-4)}$, $2^{(-5)}$, $2^{(-6)}$, $2^{(-7)}$, $2^{(-8)}$, $2^{(-1/2)}$, $2^{(-3/2)}$ }

7.1.10 flashAttn 接口调用示例

天数智芯分别提供了 flashAttn fwd 前向和 flashAttn bwd 反向的接口调用示例，请向您的应用工程师获取示例文件。

8 ixRAND

天数智算软件栈提供适配 cuRAND v10.1.2 的随机数库。借助 ixRAND 加速库，开发者可以利用 GPU 的并行能力快速大量生成所需分布的伪随机数。

天数智算软件栈支持以下 RAND API：

```
curandCreateGenerator
curandCreateGeneratorHost
curandDestroyGenerator
curandGenerate
curandGenerateLogNormal
curandGenerateLongLong
curandGenerateNormal
curandGenerateSeeds
curandGenerateUniform
curandGetDirectionVectors32
curandGetDirectionVectors64
curandGetScrambleConstants32
curandGetScrambleConstants64
curandGetVersion
curandSetGeneratorOffset
curandSetGeneratorOrdering
curandSetPseudoRandomGeneratorSeed
curandSetQuasiRandomGeneratorDimensions
curandSetStream
```

以上 API 保证一样的参数重新跑后生成的伪随机数和上次完全一致。curandSetGeneratorOrdering 只支持默认顺序。

9 ixCCL

ixCCL 是天数版的 GPU 集合通信库，API 接口兼容 NCCL v2.14。

9.1 功能说明

```
ncclAllReduce
ncclBroadcast/ncclBcast
ncclReduce
ncclAllGather
ncclReduceScatter
ncclSend
ncclRecv
```

NCCL 相关集合操作可参照 [NCCL Collective Operations](#)。

ixCCL 支持 ncclRedOp_t 的规约计算类型，具体包括: ncclSum, ncclProd, ncclMin, ncclMax, ncclAvg。

ixCCL 支持 One GPU per Process or Thread 和 Multiple GPUs per Thread 两种并行化模式。

9.2 支持的多机通信协议

目前 IXCCL 多机之间的通信支持 InfiniBand 网卡和以太网卡设备。

9.2.1 方式一：使用以太网卡

若当前环境内没有 InfiniBand 网卡或驱动，IXCCL 会使用以太网卡在多机之间传输数据，否则 IXCCL 会优先使用 InfiniBand RDMA。

9.2.2 方式二：使用 InfiniBand 网卡 RDMA 通信

您需要完成以下步骤以使用 InfiniBand RDMA 通信：

1. 确保在安装天数智算软件栈之前先安装 Mellanox InfiniBand RDMA driver：

```
# Mellanox InfiniBand RDMA driver download:
# https://www.mellanox.com/products/infiniband-drivers/linux/mlnx_ofed
# Driver package: MLNX_OFED_LINUX-5.7-1.0.2.0-ubuntu18.04-x86_64.tgz

# Install steps:
tar xvf MLNX_OFED_LINUX-5.7-1.0.2.0-ubuntu18.04-x86_64.tgz
```



```
cd MLNX_OFED_LINUX-5.7-1.0.2.0-ubuntu18.04-x86_64/
apt install python tcl tk
./mlnxofedinstall --force --vma --without-fw-update
systemctl enable openibd
systemctl restart openibd

# Check RDMA device status:
ibv_devices
ibv_devinfo
```

2. 安装天数智算软件栈，详见《软件栈安装指南》。
3. 配置网络接口 NCCL_SOCKET_IFNAME。
4. 您可以通过环境变量 NCCL_IB_HCA 来指定使用特定的 InfiniBand 设备，例如：mlx5_0。

9.3 PyTorch 下指定通信后端

针对 PyTorch 分布式通信，天数智芯支持 NCCL 和 Gloo 两种通讯后端：

- NCCL：有两种实现方式：
 - ixCCCL：此实现方式为 NCCL 通讯后端的默认设置，功能更全面。
 - GLOOGPU：如需使用 GLOOGPU 实现方式，设置环境变量 USE_GLOOGPU=1。
- Gloo：天数智芯适配版 PyTorch 对 Gloo 后端支持的算子和官方 torch 一致。

NCCL 和 Gloo 通讯后端的支持详情如下表所示：

Backend	NCCL (ixCCL)		Gloo	
Device	CPU	GPU	CPU	GPU
send	不支持	不支持	支持	不支持
recv	不支持	不支持	支持	不支持
broadcast	不支持	支持	支持	支持
all_reduce	不支持	支持 [1]	支持	支持
reduce	不支持	支持 [1]	支持	不支持
all_gather	不支持	支持	支持	不支持
gather	不支持	不支持	支持	不支持
scatter	不支持	不支持	支持	不支持
reduce_scatter	不支持	支持 [1]	不支持	不支持
all_to_all	不支持	支持	不支持	不支持
barrier	不支持	支持	支持	不支持

注：[1] all_reduce、reduce 和 reduce_scatter 支持以下 ReduceOp：

- ReduceOp::SUM
- ReduceOp::PRODUCT
- ReduceOp::MIN
- ReduceOp::MAX
- ReduceOp::AVG

9.3.1 通信后端使用方式

9.3.1.1 指定通信后端

以在天数适配版 PyTorch 为例，指定通信后端（GLOO 或 NCCL）即可使用天数适配版 PyTorch 支持的集合通信操作。

例如使用 NCCL：

```
torch.distributed.init_process_group("nccl", ...)
```

完整示例如下：

```
import torch
import torch.multiprocessing as mp
import torch.nn as nn
import torch.optim as optim
from torch.nn.parallel import DistributedDataParallel as DDP

def example(rank, world_size):
    # create default process group
    torch.distributed.init_process_group("nccl", init_method='tcp://localhost:8888', rank=rank,
    ↪ world_size=world_size)
    # create local model
    model = nn.Linear(10, 10).to(rank)
    # construct DDP model
    ddp_model = DDP(model, device_ids=[rank])
    # define loss function and optimizer
    loss_fn = nn.MSELoss()
    optimizer = optim.SGD(ddp_model.parameters(), lr=0.001)

    # forward pass
    outputs = ddp_model(torch.randn(20, 10).to(rank))
    labels = torch.randn(20, 10).to(rank)
    # backward pass
    loss_fn(outputs, labels).backward()
    # update parameters
    optimizer.step()
```

```
def main():
    world_size = 2
    mp.spawn(example,
              args=(world_size,),
              nprocs=world_size,
              join=True)

if __name__ == "__main__":
    main()
```

9.3.1.2 多机多卡：配置网络接口

对于多机多卡的分布式训练，默认会自动查找正确网络接口来使用。

如果自动查找的网络接口不正确，您可设置以下环境变量来指定网络接口：

通讯后端	环境变量
Gloo	GLOO_SOCKET_IFNAME
NCCL(IXCCL)	NCCL_SOCKET_IFNAME
NCCL(GLOOGPU)	GLOO_SOCKET_IFNAME

例如，为 NCCL(GLOOGPU)，即 NCCL 后端的 GLOOGPU 实现，指定网络接口：

```
export GLOO_SOCKET_IFNAME=<socket>
```

您可以通过 ifconfig 命令查看网络接口的名称。

10 ixSPARSE

天数智算软件栈提供适配 cuSPARSE 的线性代数运算高性能库 ixSPARSE。该函数库提供了一系列用于处理稀疏矩阵的线性代数工具，基于 CUDA Runtime API 开发，是天数智算软件栈对 CUDA Toolkit 10.2 适配的一部分。

ixSPARSE 函数库功能主要分为以下 4 部分：

- L1 functions, 稀疏向量和稠密向量的运算
- L2 functions, 稀疏矩阵和稠密向量的运算
- L3 functions, 稀疏矩阵和稠密矩阵的运算
- Conversion, 各种形式的矩阵之间的转换

天数智算软件栈支持以下 SPARSE API。

10.1 Management Functions

```
cusparseCreate
cusparseDestroy
cusparseGetPointerMode
cusparseSetPointerMode
```

10.2 Helper Functions

```
cusparseCreateMatDescr
cusparseDestroyMatDescr
cusparseCreateMatInfo
cusparseDestroyMatInfo
cusparseSetMatIndexBase
cusparseGetMatIndexBase
cusparseSetMatType
cusparseGetMatType
cusparseSetMatFillMode
cusparseGetMatFillMode
cusparseSetMatDiagType
cusparseGetMatDiagType
```

10.3 L1 Functions

```
cusparseSaxpyi  
cusparseSdoti  
cusparseSgthr  
cusparseSgthrz  
cusparseSroti  
cusparseSsctr
```

11 CUB

天数智算软件栈适配 CUB v1.8.0，支持所有的 CUB API，可提供以下功能：

- 并行原语
 - Warp 范围的集合原语
 - * Warp 范围的协作前缀和、规约等
 - Block 范围的集合原语
 - * 协作 I/O、排序、扫描、规约、直方图等
 - * 适配任意数目的 Thread block 和任意数据类型
 - Device 范围原语
 - * 并行排序、前缀和、规约、直方图等
- 工具
 - 高级迭代器
 - Thread 和 Thread Block I/O
 - PTX 内嵌函数
 - Device, Kernel 和存储管理

12 THRUST

天数智算软件栈适配 THRUST v1.9.7，只是所有从 Host 端调用的 API，包括以下功能：

- Memory Management
 - Allocators
 - Memory Resources
- Iterators
 - Fancy Iterators
 - Iterator Tags
- Algorithms
 - Searching
 - Copying
 - Reductions
 - Merging
 - Reordering
 - Prefix Sums
 - Set Operations
 - Sorting
 - Transformations
- Numerics
 - Complex Numbers
- Containers
- Parallel Execution Policies
- Function Objects
 - Function Object Adaptors
 - Predefined Function Objects
 - Placeholder Objects
- Container Classes
 - Host Containers
- Utility
 - Pair
 - Swap
 - Tuple
 - Type Traits
- Random Number Generation
 - Random Number Engine Adaptor Class Templates
 - Random Number Engine Class Template
 - Random Number Distributions Class Templates
 - Random Number Engines with Predefined Parameters

- System
 - Systems
 - System Diagnostics

有关 THRUST 函数详情，请参考 [THRUST](#) 和 [Nvidia THRUST](#)。

13 Driver API

天数智算软件栈支持以下 API:

```
cuCtxAttach
cuCtxCreate_v2
cuCtxDestroy_v2
cuCtxDetach
cuCtxDisablePeerAccess
cuCtxEnablePeerAccess
cuCtxGetApiVersion
cuCtxGetCacheConfig
cuCtxGetCurrent
cuCtxGetDevice
cuCtxGetFlags
cuCtxGetLimit
cuCtxGetSharedMemConfig
cuCtxGetStreamPriorityRange
cuCtxPopCurrent_v2
cuCtxPushCurrent_v2
cuCtxSetCacheConfig
cuCtxSetCurrent
cuCtxSetLimit
cuCtxSynchronize
cuDeviceCanAccessPeer
cuDeviceComputeCapability
cuDeviceGet
cuDeviceGetAttribute
cuDeviceGetByPCIBusId
cuDeviceGetCount
cuDeviceGetName
cuDeviceGetP2PAttribute
cuDeviceGetPCIBusId
cuDeviceGetProperties
cuDeviceGetUuid
cuDevicePrimaryCtxGetState
cuDevicePrimaryCtxRelease
cuDevicePrimaryCtxReset
cuDevicePrimaryCtxRetain
cuDevicePrimaryCtxSetFlags
cuDeviceTotalMem_v2
cuDriverGetVersion
cuEventCreate
cuEventDestroy_v2
cuEventElapsedTime
```

cuEventQuery
cuEventRecord
cuEventRecord_ptsz
cuEventSynchronize
cuFuncGetAttribute
cuFuncSetAttribute
cuFuncSetCacheConfig
cuGetErrorName
cuGetErrorString
cuGetExportTable
cuGraphAddChildGraphNode
cuGraphAddDependencies
cuGraphAddEmptyNode
cuGraphAddHostNode
cuGraphAddKernelNode
cuGraphAddMemcpyNode
cuGraphAddMemsetNode
cuGraphChildGraphNodeGetGraph
cuGraphClone
cuGraphCreate
cuGraphDestroy
cuGraphDestroyNode
cuGraphExecDestroy
cuGraphExecHostNodeSetParams
cuGraphExecKernelNodeSetParams
cuGraphExecMemcpyNodeSetParams
cuGraphExecMemsetNodeSetParams
cuGraphExecUpdate
cuGraphGetEdges
cuGraphGetNodes
cuGraphGetRootNodes
cuGraphHostNodeGetParams
cuGraphHostNodeSetParams
cuGraphInstantiate
cuGraphKernelNodeGetParams
cuGraphKernelNodeSetParams
cuGraphLaunch
cuGraphMemcpyNodeGetParams
cuGraphMemcpyNodeSetParams
cuGraphMemsetNodeGetParams
cuGraphMemsetNodeSetParams
cuGraphNodeFindInClone
cuGraphNodeGetDependencies
cuGraphNodeGetDependentNodes
cuGraphNodeGetType
cuGraphRemoveDependencies

cuInit
cuLaunchHostFunc
cuLaunchHostFunc_ptsz
cuLaunchKernel
cuLaunchKernel_ptsz
cuMemAlloc_v2
cuMemAllocHost_v2
cuMemAllocPitch_v2
cuMemcpy
cuMemcpy_ptds
cuMemcpy2D_v2
cuMemcpy2D_v2_ptds
cuMemcpy2DAsync_v2
cuMemcpy2DAsync_v2_ptsz
cuMemcpy2DUnaligned_v2
cuMemcpy2DUnaligned_v2_ptds
cuMemcpy3D_v2
cuMemcpy3D_v2_ptds
cuMemcpy3DAsync_v2
cuMemcpy3DAsync_v2_ptsz
cuMemcpy3DPeer
cuMemcpy3DPeer_ptds
cuMemcpy3DPeerAsync
cuMemcpy3DPeerAsync_ptsz
cuMemcpyAsync
cuMemcpyAsync_ptsz
cuMemcpyDtoD_v2
cuMemcpyDtoD_v2_ptds
cuMemcpyDtoDAsync_v2
cuMemcpyDtoDAsync_v2_ptsz
cuMemcpyDtoH_v2
cuMemcpyDtoH_v2_ptds
cuMemcpyDtoHAsync_v2
cuMemcpyDtoHAsync_v2_ptsz
cuMemcpyHtoD_v2
cuMemcpyHtoD_v2_ptds
cuMemcpyHtoDAsync_v2
cuMemcpyHtoDAsync_v2_ptsz
cuMemcpyPeer
cuMemcpyPeer_ptds
cuMemcpyPeerAsync
cuMemcpyPeerAsync_ptsz
cuMemFree_v2
cuMemFreeHost
cuMemGetAddressRange_v2
cuMemGetInfo_v2

cuMemHostAlloc
cuMemHostGetDevicePointer_v2
cuMemHostGetFlags
cuMemHostRegister_v2
cuMemHostUnregister
cuMemsetD16_v2
cuMemsetD16_v2_ptds
cuMemsetD16Async
cuMemsetD16Async_ptsz
cuMemsetD2D16_v2
cuMemsetD2D16_v2_ptds
cuMemsetD2D16Async
cuMemsetD2D16Async_ptsz
cuMemsetD2D32_v2
cuMemsetD2D32_v2_ptds
cuMemsetD2D32Async
cuMemsetD2D32Async_ptsz
cuMemsetD2D8_v2
cuMemsetD2D8_v2_ptds
cuMemsetD2D8Async
cuMemsetD2D8Async_ptsz
cuMemsetD32_v2
cuMemsetD32_v2_ptds
cuMemsetD32Async
cuMemsetD32Async_ptsz
cuMemsetD8_v2
cuMemsetD8_v2_ptds
cuMemsetD8Async
cuMemsetD8Async_ptsz
cuModuleGetFunction
cuModuleGetGlobal_v2
cuModuleLoad
cuModuleLoadData
cuModuleLoadDataEx
cuModuleLoadFatBinary
cuModuleUnload
cuOccupancyMaxActiveBlocksPerMultiprocessor
cuOccupancyMaxActiveBlocksPerMultiprocessorWithFlags
cuOccupancyMaxPotentialBlockSizeWithFlag
cuPointerGetAttribute
cuPointerGetAttributes
cuPointerSetAttribute
cuStreamAddCallback
cuStreamAddCallback_ptsz
cuStreamBeginCapture
cuStreamCreate

```
cuStreamCreateWithPriority
cuStreamDestroy_v2
cuStreamEndCapture
cuStreamGetCaptureInfo
cuStreamGetCtx
cuStreamGetCtx_ptsz
cuStreamGetFlags
cuStreamGetFlags_ptsz
cuStreamGetPriority
cuStreamGetPriority_ptsz
cuStreamIsCapturing
cuStreamQuery
cuStreamQuery_ptsz
cuStreamSynchronize
cuStreamSynchronize_ptsz
cuStreamWaitEvent
cuStreamWaitEvent_ptsz
cuStreamWaitValue32
cuStreamWaitValue32_ptsz
cuStreamWaitValue64
cuStreamWaitValue64_ptsz
cuStreamWriteValue32
cuStreamWriteValue32_ptsz
cuStreamWriteValue64
cuStreamWriteValue64_ptsz
cuThreadExchangeStreamCaptureMode
```

14 Runtime API

天数智算软件栈支持以下 API:

```
__cudaInitModule
__cudaPopCallConfiguration
__cudaPushCallConfiguration
__cudaRegisterFatBinary
__cudaRegisterFatBinaryEnd
__cudaRegisterFunction
__cudaRegisterVar
__cudaUnregisterFatBinary
cudaChooseDevice
cudaConfigureCall
cudaDeviceCanAccessPeer
cudaDeviceDisablePeerAccess
cudaDeviceEnablePeerAccess
cudaDeviceGetAttribute
cudaDeviceGetByPCIBusId
cudaDeviceGetCacheConfig
cudaDeviceGetLimit
cudaDeviceGetP2PAttribute
cudaDeviceGetPCIBusId
cudaDeviceGetSharedMemConfig
cudaDeviceGetStreamPriorityRange
cudaDeviceReset
cudaDeviceSetCacheConfig
cudaDeviceSetLimit
cudaDeviceSynchronize
cudaDriverGetVersion
cudaEventCreate
cudaEventCreateWithFlags
cudaEventDestroy
cudaEventElapsedTime
cudaEventQuery
cudaEventRecord
cudaEventRecord_ptsz
cudaEventSynchronize
cudaFree
cudaFreeHost
cudaFuncGetAttributes
cudaFuncSetAttribute
cudaFuncSetCacheConfig
cudaGetDevice
cudaGetDeviceCount
```

cudaGetDeviceFlags
cudaGetDeviceProperties
cudaGetErrorName
cudaGetErrorString
cudaGetLastError
cudaGraphAddChildGraphNode
cudaGraphAddDependencies
cudaGraphAddEmptyNode
cudaGraphAddHostNode
cudaGraphAddKernelNode
cudaGraphAddMemcpyNode
cudaGraphAddMemsetNode
cudaGraphChildGraphNodeGetGraph
cudaGraphClone
cudaGraphCreate
cudaGraphDestroy
cudaGraphDestroyNode
cudaGraphExecDestroy
cudaGraphExecHostNodeSetParams
cudaGraphExecKernelNodeSetParams
cudaGraphExecMemcpyNodeSetParams
cudaGraphExecMemsetNodeSetParams
cudaGraphExecUpdate
cudaGraphGetEdges
cudaGraphGetNodes
cudaGraphGetRootNodes
cudaGraphHostNodeGetParams
cudaGraphHostNodeSetParams
cudaGraphInstantiate
cudaGraphKernelNodeGetParams
cudaGraphKernelNodeSetParams
cudaGraphLaunch
cudaGraphMemcpyNodeGetParams
cudaGraphMemcpyNodeSetParams
cudaGraphMemsetNodeGetParams
cudaGraphMemsetNodeSetParams
cudaGraphNodeFindInClone
cudaGraphNodeGetDependencies
cudaGraphNodeGetDependentNodes
cudaGraphNodeGetType
cudaGraphRemoveDependencies
cudaHostAlloc
cudaHostGetDevicePointer
cudaHostGetFlags
cudaHostRegister
cudaHostUnregister

```
cudaIpcCloseMemHandle
cudaIpcGetEventHandle
cudaIpcGetMemHandle
cudaIpcOpenEventHandle
cudaIpcOpenMemHandle
cudaLaunch
cudaLaunchHostFunc
cudaLaunchHostFunc_ptsz
cudaLaunchKernel
cudaLaunchKernel_ptsz
cudaMalloc
cudaMallocHost
cudaMallocPitch
cudaMemcpy
cudaMemcpy_ptds
cudaMemcpy2D
cudaMemcpy2D_ptds
cudaMemcpy2DAsync
cudaMemcpy2DAsync_ptsz
cudaMemcpy3D
cudaMemcpy3D_ptds
cudaMemcpy3DAsync
cudaMemcpy3DAsync_ptsz
cudaMemcpy3DPeer
cudaMemcpy3DPeer_ptds
cudaMemcpy3DPeerAsync
cudaMemcpy3DPeerAsync_ptsz
cudaMemcpyAsync
cudaMemcpyAsync_ptsz
cudaMemcpyFromSymbol
cudaMemcpyFromSymbol_ptds
cudaMemcpyFromSymbolAsync
cudaMemcpyFromSymbolAsync_ptsz
cudaMemcpyPeer
cudaMemcpyPeerAsync
cudaMemcpyToSymbol
cudaMemcpyToSymbol_ptds
cudaMemcpyToSymbolAsync
cudaMemcpyToSymbolAsync_ptsz
cudaMemGetInfo
cudaMemset
cudaMemset_ptds
cudaMemset2D
cudaMemset2D_ptds
cudaMemset2DAsync
cudaMemset2DAsync_ptsz
```


cudaMemsetAsync
cudaMemsetAsync_ptsz
cudaOccupancyMaxActiveBlocksPerMultiprocessor
cudaOccupancyMaxActiveBlocksPerMultiprocessorWithFlags
cudaPeekAtLastError
cudaPointerGetAttributes
cudaRuntimeGetVersion
cudaSetDevice
cudaSetDeviceFlags
cudaSetupArgument
cudaStreamAddCallback
cudaStreamAddCallback_ptsz
cudaStreamAttachMemAsync
cudaStreamBeginCapture
cudaStreamCreate
cudaStreamCreateWithFlags
cudaStreamCreateWithPriority
cudaStreamDestroy
cudaStreamEndCapture
cudaStreamGetCaptureInfo
cudaStreamGetFlags
cudaStreamGetFlags_ptsz
cudaStreamGetPriority
cudaStreamGetPriority_ptsz
cudaStreamIsCapturing
cudaStreamQuery
cudaStreamQuery_ptsz
cudaStreamSynchronize
cudaStreamSynchronize_ptsz
cudaStreamWaitEvent
cudaStreamWaitEvent_ptsz
cudaThreadExchangeStreamCaptureMode
cudaThreadSynchronize

15 Math API

天数智算软件栈支持以下 API:

15.1 Integer Function

```
abs  
labs  
llabs  
llmax  
llmin  
max_i_i  
max_i_ui  
max_l_l  
max_l_ul  
max_ll_ll  
max_ll_ull  
max_ui_i  
max_ui_ui  
max_ul_l  
max_ul_ul  
max_ull_ll  
max_ull_ull  
min_i_i  
min_i_ui  
min_l_l  
min_l_ul  
min_ll_ll  
min_ll_ull  
min_ui_i  
min_ui_ui  
min_ul_l  
min_ul_ul  
min_ull_ll  
min_ull_ull  
ullmax  
ullmin  
umax  
umin
```

15.2 Integer Intrinsic

```
__brev  
__brevll  
__byte_perm  
__clz  
__clzll  
__ffs  
__ffsll  
__funnelshift_l  
__funnelshift_lc  
__funnelshift_r  
__funnelshift_rc  
__hadd  
__mul24  
__mul64hi  
__mulhi  
__popc  
__popc11  
__rhadd  
__uhadd  
__umul24  
__umul64hi  
__umulhi  
__urhadd  
__usad
```

15.3 Float Function

```
acosf  
acoshf  
asinf  
asinhf  
atan2f  
atanf  
atanhf  
cbrtf  
ceilf  
copysignf  
cosf  
coshf  
cospif
```

cyl_bessel_i0f
cyl_bessel_i1f
erfcf
erfcinvf
erfcxf
erff
erfinvf
exp10f
exp2f
expf
expm1f
fabsf
fdimf
fdividef
floorf
fmaf
fmaxf
fminf
fmodf
frexpf
hypotf
ilogbf
isfinite
isinf
isnan
j0f
j1f
jnf
ldexpf
lgammaf
llrintf
llroundf
log10f
log1pf
log2f
logbf
logf
lrintf
lroundf
max
min
modff
nearbyintf
nextafterf
norm3df
norm4df

```
normcdf  
normcdfinv  
normf  
powf  
rcbrtf  
remainderf  
remquof  
rhypotf  
rintf  
rnorm3df  
rnorm4df  
rnormf  
roundf  
rsqrtf  
scalblnf  
scalbnf  
signbit  
sincosf  
sincospif  
sinf  
sinhf  
sinpif  
sqrtf  
tanf  
tanhf  
tgammaf  
truncf  
y0f  
y1f  
ynf
```

15.4 Float Intrinsic

```
__cosf  
__exp10f  
__expf  
__fadd_rd  
__fadd_rn  
__fadd_ru  
__fadd_rz  
__fdiv_rd  
__fdiv_rn  
__fdiv_ru
```

```
__fdiv_rz
__fdivdef
__fmaf_rd
__fmaf_rn
__fmaf_ru
__fmaf_rz
__fmul_rd
__fmul_rn
__fmul_ru
__fmul_rz
__frcp_rd
__frcp_rn
__frcp_ru
__frcp_rz
__frsqrtn_rn
__fsqrt_rd
__fsqrt_rn
__fsqrt_ru
__fsqrt_rz
__fsub_rd
__fsub_rn
__fsub_ru
__fsub_rz
__log10f
__log2f
__logf
__powf
__saturatef
__sincosf
__sinf
__tanf
```

15.5 Type Cast

```
__float_as_int
__float_as_uint
__float2int_rd
__float2int_rn
__float2int_ru
__float2int_rz
__float2ll_rd
__float2ll_rn
__float2ll_ru
```

```
__float2ll_rz
__float2uint_rd
__float2uint_rn
__float2uint_ru
__float2uint_rz
__float2ull_rd
__float2ull_rn
__float2ull_ru
__float2ull_rz
__int_as_float
__int2float_rd
__int2float_rn
__int2float_ru
__int2float_rz
__ll2float_rd
__ll2float_rn
__ll2float_ru
__ll2float_rz
__uint_as_float
__uint2float_rd
__uint2float_rn
__uint2float_ru
__uint2float_rz
__ull2float_rd
__ull2float_rn
__ull2float_ru
__ull2float_rz
```

15.6 Half Function

```
hceil
hcos
hexp
hexp10
hexp2
hfloor
hlog
hlog10
hlog2
hrcp
hrint
hrsqr
hsin
```

```
hsqrt  
htrunc
```

15.7 Half Arithmetic

```
__habs  
__hadd_half  
__hadd_sat  
__hdiv  
__hfma  
__hfma_sat  
__hmul  
__hmul_sat  
__hneg  
__hsub  
__hsub_sat
```

15.8 Half Comparison

```
__heq  
__hequ  
__hge  
__hgeu  
__hgt  
__hgtu  
__hisinf  
__hisnan  
__hle  
__hleu  
__hlt  
__hltu  
__hne  
__hneu
```

15.9 Half Precision Conversion and Data Movement


```

__float2half2_rn
__float2half
__float2half_rd
__float2half_rn
__float2half_ru
__float2half_rz
__float2half2_rn
__floats2half2_rn
__half_as_short
__half_as_ushort
__half2float2
__half2float
__half2half2
__half2int_rd
__half2int_rn
__half2int_ru
__half2int_rz
__half2ll_rd
__half2ll_rn
__half2ll_ru
__half2ll_rz
__half2short_rd
__half2short_rn
__half2short_ru
__half2short_rz
__half2uint_rd
__half2uint_rn
__half2uint_ru
__half2uint_rz
__half2ull_rd
__half2ull_rn
__half2ull_ru
__half2ull_rz
__half2ushort_rd
__half2ushort_rn
__half2ushort_ru
__half2ushort_rz
__halves2half2
__high2float
__high2half
__high2half2
__highs2half2
__int2half_rd
__int2half_rn
__int2half_ru
__int2half_rz

```

```
__ll2half_rd
__ll2half_rn
__ll2half_ru
__ll2half_rz
__low2float
__low2half
__low2half2
__lowhigh2highlow
__lows2half2
__short_as_half
__short2half_rd
__short2half_rn
__short2half_ru
__short2half_rz
__uint2half_rd
__uint2half_rn
__uint2half_ru
__uint2half_rz
__ull2half_rd
__ull2half_rn
__ull2half_ru
__ull2half_rz
__ushort_as_half
__ushort2half_rd
__ushort2half_rn
__ushort2half_ru
__ushort2half_rz
```

15.10 Half2 Function

```
h2ceil
h2cos
h2exp
h2exp10
h2exp2
h2floor
h2log
h2log10
h2log2
h2rcp
h2rint
h2rsqrt
h2sin
```

h2sqrt
h2trunc

15.11 Half2 Arithmetic

__h2div
__habs2
__hadd2
__hadd2_sat
__hfma2
__hfma2_sat
__hmul2
__hmul2_sat
__hneg2
__hsub2
__hsub2_sat

15.12 Half2 Comparison

__hbeq2
__hbequ2
__hbge2
__hbgeu2
__hbg2
__hbg2u2
__hble2
__hbleu2
__hblt2
__hbltu2
__hbne2
__hbneu2
__heq2
__hequ2
__hge2
__hgeu2
__hgt2
__hgtu2
__hisnan2
__hle2
__hleu2

```
__hlt2  
__hltu2  
__hne2  
__hneu2
```

此文档仅供【zhangyj@skysolidiss.com.cn-天固信息安全系统（深圳）有限公司】查阅，请勿分享他人

16 CV-CUDA

CV-CUDA 是基于 GPU 的图像预处理加速库，天数智算软件栈兼容 CV-CUDA v0.4，并支持以下 CV-CUDA 操作：

Advanced Color Format Conversions

AverageBlur

BilateralFilter

Bounding Box

Box Blurring

CenterCrop

ChannelReorder

Color_Twist

Composite

Conv2D

CopyMakeBorder

CustomCrop

CvtColor

DataTypeConvert

Erase

Flip

GammaContrast

Gaussian

Gaussian Noise

Histogram

Histogram Equalizer

Joint Bilateral Filter

Laplacian

MedianBlur

MinArea Rect

MinMaxLoc

Morphology

Morphology (close)

Morphology (open)

Non-max Suppression

Normalize

PadStack

PillowResize

RandomResizedCrop

Reformat

Remap

Resize

Rotate

SIFT

WarpAffine

WarpPerspective

此文档仅供【zhangyj@skysolidiss.com.cn-天固信息安全系统（深圳）有限公司】查阅，请勿分享他人

17 商标声明

- 天数智芯、天数智芯 logo、Iluvatar CoreX 等商标、标识、组合商标为上海天数智芯半导体有限公司之注册商标或商标，受法律保护。
- 除了天数智芯的注册商标外，本内容中使用的其他产品名称及标志仅用于识别目的，该等名称及标志可能是归属于其各自公司的商标。我们否认对该等名称及标志的所有权利。
- CentOS 标识为 Red Hat 公司的商标。
- Docker 为 Docker 公司在美国和其他国家的商标或注册商标。
- Linux 为 Linus Torvalds 在美国和其它国家的注册商标。
- NVIDIA 和 CUDA 为 NVIDIA 公司在美国和/或其它国家的商标和/或注册商标。
- PyTorch 为 Facebook 公司的商标。
- TensorFlow 为 Google 公司的商标。
- Ubuntu 为 Canonical 公司的注册商标。