曦云® 系列通用计算 GPU

mcDNN API 参考

CSRD-23014-020-F3_V03　　2024-03-15

沐曦专有和三级保密信息　　本文档受 NDA 管控

声明

更新记录

版本	日期	更新说明
V03	2024-03-15	新增曦云®系列 GPU 产品信息
V02	2023-12-29	描述性修改及格式修正
V01	2023-10-16	正式版本首次发布

目录

1 简介

mcDNN 是沐曦提供的深度神经网络（Deep Neural Network，DNN）库。其基于上下文的 API 可轻松实现多线程以及与 MXMACA® 流的互操作性 (可选)。mcDNN 库以及此 API 文档拆分为以下库：

mcdnn_ops_infer

此实体包含与 mcDNN 上下文创建和销毁，张量描述符管理，张量实用程序函数以及常见机器学习算法的推理部分相关的函数，例如批量归一化（Batch Normalization，BN），softmax，丢弃（Dropout）等。

mcdnn_ops_train

此实体包含常用的训练函数算法，例如批量归一化，softmax，丢弃等。mcdnn_ops_train 库依赖于 mcdnn_ops_infer。

mcdnn_cnn_infer

此实体包含在推理时需要用到的卷积神经网络相关的所有函数。mcdnn_cnn_infer 库依赖于 mcdnn_ops_infer。

mcdnn_cnn_train

此实体包含在训练时需要用到的卷积神经网络相关的所有函数。mcdnn_cnn_train 库依赖于 mcdnn_ops_infer，mcdnn_ops_train 和 mcdnn_cnn_infer。

mcdnn_adv_infer

此实体包含所有其他功能与算法。包括循环神经网络（Rerrent Neural Network，RNN），CTC 损失（CTC Loss）和多头注意力机制（Multi-Head Attention）。mcdnn_adv_infer 库依赖于 mcdnn_ops_infer。

mcdnn_adv_train

此实体包含对应 mcdnn_adv_infer 的训练相关功能与算法。mcdnn_adv_train 库依赖于 mcdnn_ops_infer，mcdnn_ops_train 和 mcdnn_adv_infer。

mcdnnBackend

此实体包含 mcDNN 后端描述符有效类型列表，有效属性列表，有效属性值的子集以及每个后端描述符类型及其属性的完整描述。

mcdnn

这是一个在应用层和 mcDNN 代码之间的可选中介层（Shim Layer）。此层在运行时为 API 打开正确的库。

2 mcdnn_ops_infer

此实体包含与 mcDNN 上下文创建和销毁，张量描述符管理，张量实用程序函数以及常见机器学习算法的推理部分相关的函数，例如批量归一化，softmax，丢弃等。

2.1 数据类型参考

2.1.1 不透明结构类型的指针

这些是指向 mcdnn_ops_infer 中不透明结构类型的指针。

2.1.1.1 mcdnnActivationDescriptor_t

mcdnnActivationDescriptor_t 是指向不透明结构的指针，该结构包含激活操作的说明。mcdnnCreateActivationDescriptor() 用于创建实例，且必须用 mcdnnSetActivationDescriptor() 来初始化此实例。

2.1.1.2 mcdnnCTCLossDescriptor_t

mcdnnCTCLossDescriptor_t 是指向不透明结构的指针，该结构包含 CTC 损失操作的说明。mcdnnCreateCTCLossDescriptor() 用于创建实例，mcdnnSetCTCLossDescriptor() 用于初始化此实例，mcdnnDestroyCTCLossDescriptor() 用于销毁此实例。

2.1.1.3 mcdnnDropoutDescriptor_t

mcdnnDropoutDescriptor_t 是指向不透明结构的指针，该结构包含丢弃操作的说明。mcdnnCreateDropoutDescriptor() 用于创建实例，mcdnnSetDropoutDescriptor() 用于初始化此实例，mcdnnDestroyDropoutDescriptor() 用于销毁此实例，mcdnnGetDropoutDescriptor() 用于查询已初始化实例的字段，mcdnnRestoreDropoutDescriptor() 用于将实例恢复到之前的已保存关闭状态。

2.1.1.4 mcdnnFilterDescriptor_t

mcdnnFilterDescriptor_t 是指向不透明结构的指针，该结构包含卷积核（Filter）数据集的说明。mcdnnCreateFilterDescriptor() 用于创建实例，且必须用 mcdnnSetFilter4dDescriptor() 或 mcdnnSetFilterNdDescriptor() 来初始化此实例。

2.1.1.5 mcdnnHandle_t

mcdnnHandle_t 是指向不透明结构的指针，该结构包含 mcDNN 库的上下文。mcDNN 库上下文必须使用 mcdnnCreate() 创建，并且返回的句柄必须传入所有后续库函数调用。需要在结束时使用 mcdnnDestroy() 销毁此上下文。上下文仅与一个 GPU 设备关联，即当前调用 mcdnnCreate() 时的设备。但是，在同一个 GPU 设备上可以创建多个上下文。

2.1.1.6 mcdnnLRNDescriptor_t

mcdnnLRNDescriptor_t 是指向不透明结构的指针，该结构包含局部响应归一化（Local Response Normalization，LRN）的参数。mcdnnCreateLRNDescriptor() 用于创建实例，且必须用 mcdnnSetLRNDescriptor() 来初始化此实例。

2.1.1.7 mcdnnOpTensorDescriptor_t

mcdnnOpTensorDescriptor_t 用作 mcdnnOpTensor() 的参数，是指向不透明结构的指针，该结构包含 Tensor Core 操作的说明。mcdnnCreateOpTensorDescriptor() 用于创建实例，且必须用 mcdnnSetOpTensorDescriptor() 来初始化此实例。

2.1.1.8 mcdnnPoolingDescriptor_t

mcdnnPoolingDescriptor_t 是指向不透明结构的指针，该结构包含池化（Pooling）操作的说明。mcdnnCreatePoolingDescriptor() 用于创建实例，且必须用 mcdnnSetPoolingNdDescriptor() 或 mcdnnSetPooling2dDescriptor() 来初始化此实例。

2.1.1.9 mcdnnReduceTensorDescriptor_t

mcdnnReduceTensorDescriptor_t 用作 mcdnnReduceTensor() 的参数，是指向不透明结构的指针，该结构包含张量归约（Tensor Reduction）操作的说明。mcdnnCreateReduceTensorDescriptor() 用于创建实例，且必须用 mcdnnSetReduceTensorDescriptor() 来初始化此实例。

2.1.1.10 mcdnnSpatialTransformerDescriptor_t

mcdnnSpatialTransformerDescriptor_t 是指向不透明结构的指针，该结构包含空间转换操作的说明。mcdnnCreateSpatialTransformerDescriptor() 用于创建实例，mcdnnSetSpatialTransformerNdDescriptor() 用于初始化此实例，mcdnnDestroySpatialTransformerDescriptor() 用于销毁此实例。

2.1.1.11 mcdnnTensorDescriptor_t

mcdnnTensorDescriptor_t 是指向不透明结构的指针，该结构包含通用的 N 维数据集的说明。mcdnnCreateTensorDescriptor() 用于创建实例，且必须用 mcdnnSetTensorNdDescriptor()、mcdnnSetTensor4dDescriptor() 或 mcdnnSetTensor4dDescriptorEx() 来初始化此实例。

2.1.1.12 mcdnnTensorTransformDescriptor_t

mcdnnTensorTransformDescriptor_t 是指向不透明结构的指针，该结构包含张量转换的说明。mcdnnCreateTensorTransformDescriptor() 用于创建实例，mcdnnDestroyTensorTransformDescriptor() 用于销毁已创建的实例。

2.1.2 枚举类型

这些是 mcdnn_ops_infer 中的枚举类型。

2.1.2.1 mcdnnActivationMode_t

mcdnnActivationMode_t 是一种枚举类型，用于选择 mcdnnActivationForward()、mcdnnActivationBackward() 和 mcdnnConvolutionBiasActivationForward() 中使用的神经元激活 (Neuron Activation) 函数。

值

MCDNN_ACTIVATION_SIGMOID

选择 sigmoid 函数。

MCDNN_ACTIVATION_RELU

选择线性整流（ReLU）函数。

MCDNN_ACTIVATION_TANH

选择双曲正切 (tanh) 函数。

MCDNN_ACTIVATION_CLIPPED_RELU

选择裁剪 ReLU（Clipped ReLU）函数。

MCDNN_ACTIVATION_ELU

选择指数线性 (ELU) 函数。

MCDNN_ACTIVATION_IDENTITY

选择恒等函数（Identity Function），用于绕过 mcdnnConvolutionBiasActivationForward() 中的激活步骤。（mcdnnConvolutionBiasActivationForward() 函数必须使用 MCDNN_CONVOLUTION_FWD_ALGO_IMPLICIT_PRECOMP_GEMM。）不能与 mcdnnActivationForward() 或 mcdnnActivationBackward() 一起使用。

MCDNN_ACTIVATION_SWISH

选择 swish 函数。

2.1.2.2 mcdnnBatchNormMode_t

mcdnnBatchNormMode_t 是一种枚举类型，用于指定 mcdnnBatchNormalizationForwardInference()、mcdnnBatchNormalizationForwardTraining()、mcdnnBatchNormalizationBackward() 和 mcdnnDeriveBNTensorDescriptor() 函数中的操作模式。

值

MCDNN_BATCHNORM_PER_ACTIVATION

每次激活时执行规一化。该模式用于非卷积网络层之后。在这种模式下，bnBias 和 bnScale 的张量维度以及 mcdnnBatchNormalization 函数中使用的参数均为 1xCxHxW。

MCDNN_BATCHNORM_SPATIAL

在 N+ 空间维度上执行归一化。该模式用于卷积层 (需要空间不变) 之后。在这种模式下，bnBias 和 bnScale 的张量维度为 1xCx1x1。

MCDNN_BATCHNORM_SPATIAL_PERSISTENT

该模式与 MCDNN_BATCHNORM_SPATIAL 类似，但在某些任务中可以更快。可以为以下 MCDNN_DATA_FLOAT 和 MCDNN_DATA_HALF 类型的规一化 API 调用选择优化路径:mcdnnBatchNormalizationForwardTraining() 和 mcdnnBatchNormalizationBackward()。对于 mcdnnBatchNormalizationBackward()，savedMean 和 savedInvVariance 参数不能为 NULL。

本节的以下部分仅适用于 NCHW 模式：此模式可以使用具有确定性的原子整数缩减（Scaled Atomic Integer Reduction），但对输入数据范围增加了更多限制。当出现数值溢出时，算法可能会在输出缓冲区中生成 NaN-s 或 Inf-s(无穷)。

当输入数据中存在 Inf-s/NaN-s 时，此模式下的输出与纯浮点实现中的输出相同。

对于有限但非常大的输入值，由于较低的动态范围，算法可能会更频繁地遇到溢出并发出 Inf-s/NaN-s，而 MCDNN_BATCHNORM_SPATIAL 会生成有限的输出值。用户可以调用 mcdnnQueryRuntimeError() 来检查在此模式下是否发生了数值溢出。

2.1.2.3 mcdnnBatchNormOps_t

mcdnnBatchNormOps_t 是一种枚举类型，用于指定 mcdnnGetBatchNormalizationForwardTrainingExWorkspaceSize()、mcdnnBatchNormalizationForwardTrainingEx()、mcdnnGetBatchNormalizationBackwardExWorkspaceSize()、mcdnnBatchNormalizationBackwardEx() 和 mcdnnGetBatchNormalizationTrainingExReserveSpaceSize() 函数中的操作模式。

值

MCDNN_BATCHNORM_OPS_BN

每次激活仅执行批量归一化。

MCDNN_BATCHNORM_OPS_BN_ACTIVATION

先执行批量归一化，然后执行激活。

MCDNN_BATCHNORM_OPS_BN_ADD_ACTIVATION

先执行批量归一化，然后执行元素级加法（Element-Wise Addition），再执行激活。

2.1.2.4 mcdnnCTCLossAlgo_t

mcdnnCTCLossAlgo_t 是一种枚举类型，暴露可用于执行 CTC 损失操作的不同算法。

值

MCDNN_CTC_LOSS_ALGO_DETERMINISTIC

保证结果可重现。

MCDNN_CTC_LOSS_ALGO_NON_DETERMINISTIC

不保证结果可重现。

2.1.2.5 mcdnnDataType_t

mcdnnDataType_t 是一种枚举类型，用于表明张量描述符或卷积核描述符引用的数据类型。

值

MCDNN_DATA_FLOAT

32-bit 单精度浮点数据。

MCDNN_DATA_DOUBLE

64-bit 双精度浮点数据。

MCDNN_DATA_HALF

16-bit 浮点数据。

MCDNN_DATA_INT8

8-bit 带符号的整数数据。

MCDNN_DATA_INT32

32-bit 带符号的整数数据。

MCDNN_DATA_INT8x4

32-bit 元素数据，每个元素由 4 个 8-bit 带符号的整数组成。此数据类型仅支持张量格式 MCDNN_TENSOR_NCHW_VECT_C。

MCDNN_DATA_UINT8

8-bit 无符号的整数数据。

MCDNN_DATA_UINT8x4

32-bit 元素数据，每个元素由 4 个 8-bit 无符号的整数组成。此数据类型仅支持张量格式 MCDNN_TENSOR_NCHW_VECT_C。

MCDNN_DATA_INT8x32

32 元素向量数据，每个元素是一个 8-bit 带符号的整数。此数据类型仅支持张量格式 MCDNN_TENSOR_NCHW_VECT_C。此外，此数据类型只能与 algo 1 一起使用，即 MCDNN_CONVOLUTION_FWD_ALGO_IMPLICIT_PRECOMP_GEMM。更多信息，参见 mcdnnConvolutionFwdAlgo_t。

MCDNN_DATA_BFLOAT16

16-bit 数据，有 7 个 mantissa 位，8 个指数位和 1 个符号位。

MCDNN_DATA_INT64

64-bit 带符号的整数数据。

MCDNN_DATA_BOOLEAN

布尔型数据（bool）。

请注意，对于 MCDNN_TYPE_BOOLEAN 类型，元素应 “压缩 (Packed)”：即一个字节包含 8 个 MCDNN_TYPE_BOOLEAN 类型的元素。此外，在每个字节中，元素从最低有效位（Least Significant Bit）到最高有效位（Most Significant Bit）进行索引。例如，包含 01001111 的 1 维张量有 8 个元素，元素 0 到 3 的值为 1，元素 4 和 5 的值为 0，元素 6 的值为 1，元素 7 的值为 0。

具有 8 个以上元素的张量需要使用更多字节，其中顺序也从最低有效位到最高有效位。请注意，MXMACA 是小端序（Little-Endian）模式，即最低有效位的内存地址较低。例如，16 个元素 01001111 11111100，元素 0 到 3 的值为 1，元素 4 和 5 的值为 0，元素 6 的值为 1，元素 7 的值为 0，元素 8 和 9 的值为 0，元素 10 到 15 的值为 1。

2.1.2.6 mcdnnDeterminism_t

mcdnnDeterminism_t 是一种枚举类型，用于表明计算的结果是否具有确定性 (可重现)。

值

MCDNN_NON_DETERMINISTIC

不保证结果可重现。

MCDNN_DETERMINISTIC

保证结果可重现。

2.1.2.7 mcdnnDivNormMode_t

mcdnnDivNormMode_t 是一种枚举类型，用于表明 mcdnnDivisiveNormalizationForward() 和 mcdnnDivisiveNormalizationBackward() 中的操作模式。

值

MCDNN_DIVNORM_PRECOMPUTED_MEANS

均值张量数据指针应包含用户预计算得到的均值或其他内核卷积值。均值指针也可以是 NULL，在这种情况下，均值指针被视为是用零填充的。相当于空间 LRN。

注解: 在反向传递中，均值被视为独立输入，且会独立计算均值梯度。在此模式下，要在整个局部对比度归一化（Local Contrastive Normalization，LCN）计算图上产生净梯度（Net Gradient），destDiffMeans 结果应通过用户的均值层 (可使用平均池化实现) 反向传播，并添加到 mcdnndivisionNormalizationBackward() 生成的 destDiffData 张量中。

2.1.2.8 mcdnnErrQueryMode_t

mcdnnErrQueryMode_t 是一种枚举类型，传入 mcdnnQueryRuntimeError() 以选择远程计算核错误查询模式。

值

MCDNN_ERRQUERY_RAWCODE

无论计算核完成状态如何，都读取错误存储位置。

MCDNN_ERRQUERY_NONBLOCKING

报告 mcDNN 句柄的用户流中所有任务是否已完成。如果未完全完成，报告远程计算核错误代码。

MCDNN_ERRQUERY_BLOCKING

等待用户流中的所有任务完成，然后再报告远程计算核错误代码。

2.1.2.9 mcdnnFoldingDirection_t

mcdnnFoldingDirection_t 是用于选择折叠方向的枚举类型。更多信息，参见 mcdnnTensorTransformDescriptor_t。

数据成员（Data Members）

MCDNN_TRANSFORM_FOLD

选择折叠。

MCDNN_TRANSFORM_UNFOLD

选择展开。

2.1.2.10 mcdnnIndicesType_t

mcdnnIndicesType_t 是一种枚举类型，用于表明 mcdnnReduceTensor() 要计算的索引的数据类型。此枚举类型用作 mcdnnReduceTensorDescriptor_t 描述符的字段。

值

MCDNN_32BIT_INDICES

计算无符号 int 索引。

MCDNN_64BIT_INDICES

计算无符号长索引。

MCDNN_16BIT_INDICES

计算无符号短索引。

MCDNN_8BIT_INDICES

计算无符号字符索引。

2.1.2.11 mcdnnLRNMode_t

mcdnnLRNMode_t 是一种枚举类型，用于表明 mcdnnLRNCrossChannelForward() 和 mcdnnLRN-CrossChannelBackward() 中的操作模式。

值

MCDNN_LRN_CROSS_CHANNEL_DIM1 LRN 计算在张量的 dimA[1] 维中执行。

2.1.2.12 mcdnnMathType_t

mcdnnMathType_t 是一种枚举类型，用于表明在给定库函数中是否允许使用 Tensor Core 操作。

值

MCDNN_DEFAULT_MATH

在曦云系列 GPU 设备上，支持 Tensor Core TF32 操作。

MCDNN_TENSOR_OP_MATH

允许使用 Tensor Core 操作，但不会主动在张量上执行数据类型向下转换来使用 Tensor Core。

MCDNN_TENSOR_OP_MATH_ALLOW_CONVERSION

允许使用 Tensor Core 操作，并将在张量上主动执行数据类型向下转换，以使用 Tensor Core。

MCDNN_FMA_MATH

仅限于使用乘积累加（Fused Multiply–accumulate，FMA）指令的内核。

2.1.2.13 mcdnnNanPropagation_t

mcdnnNanPropagation_t 是一种枚举类型，用于表明给定函数是否要传播 NaN 数。此枚举类型用作 mcdnnActivationDescriptor_t 描述符和 mcdnnPoolingDescriptor_t 描述符的字段。

值

MCDNN_NOT_PROPAGATE_NAN

不传播 NaN 数。

MCDNN_PROPAGATE_NAN

传播 NaN 数。

2.1.2.14 mcdnnNormAlgo_t

mcdnnNormalAlgo_t 是一种枚举类型，用于指定执行归一化操作的算法。

值

MCDNN_NORM_ALGO_STANDARD

执行标准归一化。

MCDNN_NORM_ALGO_PERSIST

该模式与 MCDNN_NORM_ALGO_STANDARD 类似，但仅支持 MCDNN_NORM_PER_CHANNEL，且在某些任务中可以更快。

可以为以下 MCDNN_DATA_FLOAT 和 MCDNN_DATA_HALF 类型的规一化 API 调用选择优化路径:mcdnnNormalizationForwardTraining() 和 mcdnnNormalizationBackward()。对于 mcdnnNormalizationBackward()，savedMean 和 savedInvVariance 参数不能为 NULL。

本节的以下部分仅适用于 NCHW 模式：此模式可以使用具有确定性的原子整数缩减（Scaled Atomic Integer Reduction），但对输入数据范围增加了更多限制。当发生数值溢出时，算法可能会在输出缓冲区中生成 NaN-s 或 Inf-s(无穷)。

当输入数据中存在 Inf-s/NaN-s 时，此模式下的输出与纯浮点实现中的输出相同。

对于有限但非常大的输入值，由于较低的动态范围，算法可能会更频繁地遇到溢出并发出 Inf-s/NaN-s，而 MCDNN_NORM_ALGO_STANDARD 会生成有限的输出值。用户可以调用 mcdnnQueryRuntimeError() 来检查在此模式下是否发生了数值溢出。

2.1.2.15 mcdnnNormMode_t

mcdnnNormMode_t 是一种枚举类型，用于表明 mcdnnNormalizationForwardInference()、mcdnnNormalizationForwardTraining()、mcdnnBatchNormalizationBackward()、mcdnnGetNormalizationForwardTrainingWorkspaceSize()、mcdnnGetNormalizationBackwardWorkspaceSize() 和 mcdnnGetNormalizationTrainingReserveSpaceSize() 中的操作模式。

值

MCDNN_NORM_PER_ACTIVATION

每次激活时执行规一化。该模式用于非卷积网络层之后。在这种模式下，normBias 和 normScale 的张量维度以及 mcdnnNormalization 函数中使用的参数均为 1xCxHxW。

MCDNN_NORM_PER_CHANNEL

在 N+ 空间维度的每一通道上执行归一化。该模式用于卷积层 (需要空间不变) 之后。在这种模式下，normBias 和 normScale 的张量维度为 1xCx1x1。

2.1.2.16 mcdnnNormOps_t

mcdnnNormOps_t 是一种枚举类型，用于表明 mcdnnGetNormalizationForwardTrainingWorkspaceSize()、mcdnnNormalizationForwardTraining()、mcdnnGetNormalizationBackwardWorkspaceSize()、mcdnnNormalizationBackward() 和 mcdnnGetNormalizationTrainingReserveSpaceSize() 中的操作模式。

值

MCDNN_NORM_OPS_NORM

仅执行归一化。

MCDNN_NORM_OPS_NORM_ACTIVATION

先执行归一化，然后执行激活。

MCDNN_NORM_OPS_NORM_ADD_ACTIVATION

先执行归一化，然后执行元素级加法，再执行激活。

2.1.2.17 mcdnnOpTensorOp_t

mcdnnOpTensorOp_t 是一种枚举类型，用于表明 mcdnnOpTensor() 要使用的 Tensor Core 操作。此枚举类型用作 mcdnnOpTensorDescriptor_t 描述符的字段。

值

MCDNN_OP_TENSOR_ADD

要执行的操作是加法（Addition）。

MCDNN_OP_TENSOR_MUL

要执行的操作是乘法（Multiplication）。

MCDNN_OP_TENSOR_MIN

要执行的操作是最小值比较 (Minimum Comparison)。

MCDNN_OP_TENSOR_MAX

要执行的操作是最大值比较 (Maximum Comparison)。

MCDNN_OP_TENSOR_SQRT

要执行的操作是平方根（Square Root），仅在 A 张量上执行。

MCDNN_OP_TENSOR_NOT

要执行的操作是否定（Negation），仅在 A 张量上执行。

2.1.2.18 mcdnnPoolingMode_t

mcdnnPoolingMode_t 是一种枚举类型，传入 mcdnnSetPooling2dDescriptor() 以选择 mcdnnPoolingForward() 和 mcdnnPoolingBackward() 要使用的池化方式。

值

MCDNN_POOLING_MAX

使用池化窗口中的最大值。

MCDNN_POOLING_AVERAGE_COUNT_INCLUDE_PADDING

对池化窗口中的值求平均值。用于计算平均值的元素数包括位于填充区域中的空间位置。

MCDNN_POOLING_AVERAGE_COUNT_EXCLUDE_PADDING

对池化窗口中的值求平均值。用于计算平均值的元素数不包括位于填充区域中的空间位置。

MCDNN_POOLING_MAX_DETERMINISTIC

使用池化窗口中的最大值。使用的算法具有确定性。

2.1.2.19 mcdnnReduceTensorIndices_t

mcdnnReduceTensorIndices_t 是一种枚举类型，用于表明 mcdnnReduceTensor() 是否需要计算索引。此枚举类型用作 mcdnnReduceTensorDescriptor_t 描述符的字段。

值

MCDNN_REDUCE_TENSOR_NO_INDICES

不计算索引。

MCDNN_REDUCE_TENSOR_FLATTENED_INDICES

计算索引。计算得到的索引是相关且平展的。

2.1.2.20 mcdnnReduceTensorOp_t

mcdnnReduceTensorOp_t 是一种枚举类型，用于表明 mcdnnReduceTensor() 要使用的 Tensor Core 操作。此枚举类型用作 mcdnnReduceTensorDescriptor_t 描述符的字段。

值

MCDNN_REDUCE_TENSOR_ADD

要执行的操作是加法（Addition）。

MCDNN_REDUCE_TENSOR_MUL

要执行的操作是乘法（Multiplication）。

MCDNN_REDUCE_TENSOR_MIN

要执行的操作是最小值比较 (Minimum Comparison)。

MCDNN_REDUCE_TENSOR_MAX

要执行的操作是最大值比较 (Maximum Comparison)。

MCDNN_REDUCE_TENSOR_AMAX

要执行的操作是最大绝对值比较。

MCDNN_REDUCE_TENSOR_AVG

要执行的操作是求均值（Averaging)。

MCDNN_REDUCE_TENSOR_NORM1

要执行的操作是绝对值相加。

MCDNN_REDUCE_TENSOR_NORM2

要执行的操作是求平方和的平方根。

MCDNN_REDUCE_TENSOR_MUL_NO_ZEROS

要执行的操作是乘法，不包括值为零的元素。

2.1.2.21 mcdnnRNNAlgo_t

mcdnnRNNAlgo_t 是一种枚举类型，用于指定 mcdnnRNNForwardInference()，mcdnnRNNForward-Train()，mcdnnRNNBackwardData() 和 mcdnnRNNBackwardWeights() 函数中使用的算法。

值

MCDNN_RNN_ALGO_STANDARD

每个 RNN 层作为一个操作序列执行。该算法在各种网络参数中具有强大的性能。

MCDNN_RNN_ALGO_PERSIST_STATIC

使用持久内核（Persistent Kernel）方法执行网络的循环部分。当输入张量的第一个维度很小 (即小批量，Minibatch) 时，这种方法会很快。

MCDNN_RNN_ALGO_PERSIST_DYNAMIC

使用持久内核（Persistent Kernel）方法执行网络的循环部分。当输入张量的第一个维度很小 (即小批量，Minibatch) 时，这种方法会很快。使用 MCDNN_RNN_ALGO_PRESERT_DYNAMIC 时，会在运行时准备持久内核，并能够使用网络和运行中 GPU 的特定参数进行优化。因此，在使用 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，必须执行一次性计划准备阶段。然后，可以使用相同的模型参数重复调用这些计划。使用 MCDNN_RNN_ALGO_PRESERT_DYNAMIC 时，所支持的隐藏单元（Hidden Unit）最大数量限制远高于使用 MCDNN_RNN_ALGO_PRESERT_STATIC 时的限制，但是如果超过 MCDNN_RNN_ALGO_PRESERT_STATIC 支持的最大值，吞吐量可能会大大降低。即便如此，在某些情况下，此方法仍优于 MCDNN_RNN_ALGO_STANDARD。

2.1.2.22 mcdnnSamplerType_t

mcdnnSamplerType_t 是一种枚举类型，传入 mcdnnSetSpatialTransformerNdDescriptor() 以选择 mcdnnSpatialTfSamplerForward() 和 mcdnnSpatialTfSamplerBackward() 要使用的采样器（Sam-pler）类型。

值

MCDNN_SAMPLER_BILINEAR

选择双线性采样器（Bilinear Sampler）。

2.1.2.23 mcdnnSeverity_t

mcdnnSeverity_t 是一种枚举类型，传入自定义回调函数以记录用户可以设置的日志。此枚举描述项目的严重性级别，因此自定义的记录回调可能会有不同的反应。

值

MCDNN_SEV_FATAL 此值表示 mcDNN 发出一条致命错误。

MCDNN_SEV_ERROR 此值表示 mcDNN 发出一条一般错误。

MCDNN_SEV_WARNING 此值指示 mcDNN 发出一条警告。

MCDNN_SEV_INFO 此值表示 mcDNN 发出一条信息（例如，API 日志）。

2.1.2.24 mcdnnSoftmaxAlgorithm_t

mcdnnSoftmaxAlgorithm 用于选择 mcdnnSoftmaxForward() 和 mcdnnSoftmaxBackward() 中使用的 softmax 函数的实现。

值

MCDNN_SOFTMAX_FAST

此实现应用简单的 softmax 运算。

MCDNN_SOFTMAX_ACCURATE

此实现将 softmax 输入域的每个点按其最大值进行缩放，以避免 softmax 求值中可能存在的浮点溢出。

MCDNN_SOFTMAX_LOG

此条目执行日志 softmax 运算，按 MCDNN_SOFTMAX_ACCURATE 中的比例缩放输入域中的每个点来避免溢出。

2.1.2.25 mcdnnSoftmaxMode_t

mcdnnSoftmaxMode_t 用于选择 mcdnnSoftmaxForward() 和 mcdnnSoftmaxBackward() 正在计算其结果的数据。

值

MCDNN_SOFTMAX_MODE_INSTANCE

softmax 运算根据 C、H、W 维度上的图像（N）进行计算。

MCDNN_SOFTMAX_MODE_CHANNEL

softmax 运算根据维度 C 上每个图像（N）的空间位置（H,W）计算。

2.1.2.26 mcdnnStatus_t

mcdnnStatus_t 是用于返回函数状态的枚举类型。所有 mcDNN 库函数返回其状态，状态可以是以下值之一：

值

MCDNN_STATUS_SUCCESS

运算已成功完成。

MCDNN_STATUS_NOT_INITIALIZED

未正确初始化 mcDNN 库。当调用 mcdnnCreate() 失败或在调用另一个 mcDNN 函数之前未调用 mcdnnCreate() 时，通常会返回此错误。前一种情况，通常是由于 mcdnnCreate() 调用的 MXMACA 运行时 API 中存在错误或硬件安装中存在错误。

MCDNN_STATUS_ALLOC_FAILED

mcDNN 库中的资源分配失败。这通常是由内部 mcMalloc() 故障引起的。要解决此问题，在函数调用之前，应尽可能多地释放先前分配的内存。

MCDNN_STATUS_BAD_PARAM

传入函数的值或参数不正确。要解决此问题，请确保所有要传入的参数都具有有效值。

MCDNN_STATUS_ARCH_MISMATCH

当前 GPU 设备不支持此函数需要的功能。

MCDNN_STATUS_MAPPING_ERROR

访问 GPU 内存空间失败，这通常是由于绑定纹理（Texture）失败导致的。要解决此问题，在函数调用之前，取消绑定所有已绑定的纹理。否则，这可能表示库中存在内部错误。

MCDNN_STATUS_EXECUTION_FAILED

GPU 程序执行失败。这通常是由于无法在 GPU 上启动某些 mcDNN 内核导致的，内核启动失败可能是多种原因造成的。要解决此问题，请检查硬件，正确版本的驱动程序以及 mcDNN 库是否已正确安装。否则，这可能表示库中存在内部错误。

MCDNN_STATUS_INTERNAL_ERROR

mcDNN 内部操作失败。

MCDNN_STATUS_NOT_SUPPORTED

mcDNN 目前不支持请求的功能。

MCDNN_STATUS_LICENSE_ERROR

请求的功能需要一些许可，并且在尝试检查当前许可时检测到错误。

MCDNN_STATUS_RUNTIME_PREREQUISITE_MISSING

在预定义的搜索路径中找不到 mcDNN 所需的运行时库。

MCDNN_STATUS_RUNTIME_IN_PROGRESS

用户流中的某些任务未完成。

MCDNN_STATUS_RUNTIME_FP_OVERFLOW

GPU 内核执行期间出现数值溢出。

2.1.2.27 mcdnnTensorFormat_t

mcdnnTensorFormat_t 是一种枚举类型，用于 mcdnnSetTensor4dDescriptor() 创建具有预定义布局的张量。

值

MCDNN_TENSOR_NCHW

此张量格式指定数据按以下顺序排列：批大小（Batch Size），特征图，行，列。步幅（Stride）的隐式定义方式是：数据在内存中是连续的，图像，特征图，行和列之间没有填充；列是内部维度，图像是最外侧维度。

MCDNN_TENSOR_NHWC

此张量格式指定数据按以下顺序排列：批大小，行，列，特征图。步幅的隐式定义方式是：数据在内存中是连续的，图像，行，列和特征图之间没有填充；特征图是内部维度，图像是最外侧维度。

MCDNN_TENSOR_NCHW_VECT_C

此张量格式指定数据按以下顺序排列：批大小（Batch Size），特征图，行，列。但是，张量的每个元素都是一个多特征图的向量。向量的长度由张量的数据类型携带。步幅（Stride）的隐式定义方式是：数据在内存中是连续的，图像，特征图，行和列之间没有填充；列是内部维度，图像是最外侧维度。此格式仅在以下张量数据类型中支持：MCDNN_DATA_INT8x4、MCDNN_DATA_INT8x32 和 MCDNN_DATA_UINT8x4。MCDNN_Tensor_NCH_VECT_C 也可以按以下方式解析：NCHW INT8x32 格式实际上是 N x (C/32) x H x W x 32（32 Cs/W），类似于 NCHW INT8x4 格式是 N x (C/4) x H x W x 4（4 Cs/W）。因此，VECT_C 名称-每个 W 都是一个 Cs 的向量 (4 或 32)。

2.2 API 参考

2.2.1 API 函数

以下为 mcdnn_ops_infer.h 中的 API 函数。

2.2.1.1 mcdnnActivationForward()

此函数以逐元素（Element-Wise）的方式在每个输入值上应用指定的神经元激活函数。

```
mcdnnStatus_t mcdnnActivationForward(mcdnnHandle_t handle,
  mcdnnActivationDescriptor_t activationDesc,
  const void *alpha,
  const mcdnnTensorDescriptor_t xDesc,
  const void *x,
  const void *beta,
  const mcdnnTensorDescriptor_t yDesc,
  void *y)
```

该函数支持就地操作（In-place Operation），这意味着 xData 和 yData 指针可以相等。但是，这要求 xDesc 描述符和 yDesc 描述符必须相同（特别是输入和输出的步幅必须匹配，才能支持就地操作）。所有张量格式都支持 4 维和 5 维。但是，当 xDesc 和 yDesc 的步幅相等且都为 HW-packed 时，可以获得最佳性能。对于超过 5 维的张量，必须压缩其空间维度。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

activationDesc

输入。激活描述符。更多信息，参见 mcdnnActivationDescriptor_t。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将计算结果与输出层中的先验值混合，如下所示：dstValue = alpha[0]*result + beta[0]*priorDstValue。

xDesc

输入。已初始化的输入张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存。

yDesc

输入。已初始化的输出张量描述符的句柄。

y

输出。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- mode 参数具有无效的枚举值。
- 输入张量和输出张量的 n，c，h，w 维度不同。
- 输入张量和输出张量的数据类型不同。
- 输入张量和输出张量的 nStride，cStride，hStride，wStride 步幅不同，并使用就地操作（即 x 和 y 指针相等）。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.2 mcdnnAddTensor()

该函数将一个 bias 张量的缩放值添加到另一个张量。bias 张量 A 的每个维度必须与目标张量 C 的相应维度匹配，或者必须等于 1。在后一种情况下，bias 张量中维度的值将与张量 C 中对应的值混合。

```
mcdnnStatus_t mcdnnAddTensor(
    mcdnnHandle_t handle,
    const void *alpha,
    const mcdnnTensorDescriptor_t aDesc,
    const void *A,
    const void *beta,
    const mcdnnTensorDescriptor_t cDesc,
    void *C)
```

仅支持 4D 和 5D 张量。如果超过这些维度，则不支持此函数。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将源数值与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*srcValue + beta[0]*priorDstValue。

aDesc

输入。已初始化的张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

A

输入。指针，指向 aDesc 描述符描述的张量数据。

cDesc

输入。已初始化的张量描述符的句柄。

C

输入/输出。指针，指向 cDesc 描述符描述的张量数据。

返回值

MCDNN_STATUS_SUCCESS

此函数执行成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

bias 张量的维度是指与输出张量维度不兼容的数据量，或者两个张量描述符的 dataType 不同。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.3 mcdnnBatchNormalizationForwardInference()

此函数在推理阶段执行正向 BN 层计算。对于 BN 层，可参见 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift。

```
mcdnnStatus_t mcdnnBatchNormalizationForwardInference(
    mcdnnHandle_t                handle,
    mcdnnBatchNormMode_t         mode,
    const void                  *alpha,
    const void                  *beta,
    const mcdnnTensorDescriptor_t xDesc,
    const void                  *x,
    const mcdnnTensorDescriptor_t yDesc,
    void                        *y,
    const mcdnnTensorDescriptor_t bnScaleBiasMeanVarDesc,
    const void                  *bnScale,
    const void                  *bnBias,
    const void                  *estimatedMean,
    const void                  *estimatedVariance,
    double                       epsilon)
```

仅支持 4D 和 5D 张量。此函数执行的输入转换定义如下：y = beta*y + alpha * (bnBias + (bnScale * (x-estimatedMean)/sqrt(epsilon + estimatedVariance))。

有关训练阶段的信息，请参见 mcdnnBatchNormalizationForwardTraining()。当所有 x 和 dx 都使用 HW-packed 张量时，可以获得更佳的性能。有关此函数中使用的参数的辅助张量描述符生成信息，请参见 mcdnnDeriveBNTensorDescriptor()。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式（spatial 或 per-activation）。更多信息，参见 mcdnnBatchNormMode_t。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将层输出值与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

xDesc, yDesc

输入。已初始化的张量描述符的句柄。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存，用于层的 x 输入数据。

y

输入/输出。数据指针，指向与张量描述符 yDesc 关联的 GPU 内存，用于 BN 层的 y 输出。

bnScaleBiasMeanVarDesc, bnScale, bnBias

输入。设备内存中用于 BN scale 和 bias 参数的张量描述符和指针。

estimatedMean, estimatedVariance

输入。均值和方差张量（与 bias 和 scale 张量具有相同的描述符）。在 mcdnnBatchNormalizationForwardTraining() 调用的训练阶段累积的 resultRunningMean 和 resultRunningVariance，此处作为输入传入。

epsilon

输入。BN 公式中使用的 Epsilon 值。其值应等于或大于 mcdnn.h 中为 MCDNN_BN_MIN_EPSILON 定义的值。

支持的配置

此函数支持多种描述符的以下数据类型组合。

数据类型配置	xDesc	bnScaleBias MeanVarDesc	alpha, beta	yDesc
IN T8_CONFIG	MCDNN_ DATA_INT8	MCDNN_D ATA_FLOAT	MCDNN_D ATA_FLOAT	MCDNN_ DATA_INT8
PSEUDO_HA LF_CONFIG	MCDNN_ DATA_HALF	MCDNN_D ATA_FLOAT	MCDNN_D ATA_FLOAT	MCDNN_ DATA_HALF
FLO AT_CONFIG	MCDNN_D ATA_FLOAT	MCDNN_D ATA_FLOAT	MCDNN_D ATA_FLOAT	MCDNN_D ATA_FLOAT
DOUB LE_CONFIG	MCDNN_DA TA_DOUBLE	MCDNN_DA TA_DOUBLE	MCDNN_DA TA_DOUBLE	MCDNN_DA TA_DOUBLE
BFLOAT 16_CONFIG	MCDNN_ DATA _BFLOAT16	MCDNN_D ATA_FLOAT	MCDNN_D ATA_FLOAT	MCDNN_ DATA _BFLOAT16

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- alpha，beta，x，y，bnScale，bnBias，estimatedMean，estimatedInvVariance 中任一指针为 NULL。
- xDesc 或 yDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量。）
- spatial 模式下，4D 张量中 bnScaleBiasMeanVarDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- epsilon 值小于 MCDNN_BN_MIN_EPSILON。
- xDesc 和 yDesc 的维度或数据类型不匹配。

2.2.1.4 mcdnnCopyAlgorithmDescriptor()

2.2.1.5 mcdnnCreate()

该函数用于初始化 mcDNN 库并创建一个不透明结构的句柄，该结构存放 mcDNN 库上下文。该函数在主机和设备上分配硬件资源，必须在调用其他任何 mcDNN 库之前对其进行调用。

```
mcdnnStatus_t mcdnnCreate(mcdnnHandle_t *handle)
```

mcDNN 库句柄与当前 MXMACA 设备（上下文）绑定。要在多个设备上使用库，需要为每个设备创建一个 mcDNN 句柄。对于给定设备，可以创建多个具有不同配置的 mcDNN 句柄（例如，不同的当前 MXMACA 流）。由于 mcdnnCreate() 分配一些内部资源，因此通过调用 mcdnnDestroy() 来释放这些资源将隐式调用 mcDeviceSynchronize()；因此，建议的最佳做法是在 performance-critical 代码路径之外调用 mcdnnCreate/mcdnnDestroy。对于使用相同设备但来自不同线程的多线程应用，推荐使用的编程模型是为每个线程创建一个（或几个）mcDNN 句柄，并在线程的整个生命周期中使用该 mcDNN 句柄。

参数

handle

输出。指向将地址存储到分配的 mcDNN 句柄的指针。更多信息，参见 mcdnnHandle_t。

返回值

MCDNN_STATUS_BAD_PARAM

提供的输入指针无效（NULL）。

MCDNN_STATUS_NOT_INITIALIZED

未找到兼容的 GPU，未安装或禁用 MXMACA 驱动程序，MXMACA 运行时 API 初始化失败。

MCDNN_STATUS_ARCH_MISMATCH

MetaX GPU 架构太旧。

MCDNN_STATUS_ALLOC_FAILED

主机内存分配失败。

MCDNN_STATUS_INTERNAL_ERROR

MXMACA 资源分配失败。

MCDNN_STATUS_LICENSE_ERROR

mcDNN 许可验证失败（仅当启用此功能时）。

MCDNN_STATUS_SUCCESS

已成功创建 mcDNN 句柄。

2.2.1.6 mcdnnCreateActivationDescriptor()

```
mcdnnStatus_t mcdnnCreateActivationDescriptor(
    mcdnnActivationDescriptor_t    *activationDesc)
```

此函数通过分配保存激活描述符不透明结构所需内存的方式，创建激活描述符对象。更多信息，参见 mcdnnActivationDescriptor_t。

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

2.2.1.7 mcdnnCreateAlgorithmDescriptor()

此函数通过分配保存算法描述符不透明结构所需内存的方式，创建算法描述符对象。

```
mcdnnStatus_t mcdnnCreateAlgorithmDescriptor(
    mcdnnAlgorithmDescriptor_t *algoDesc)
```

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

2.2.1.8 mcdnnCreateAlgorithmPerformance()

此函数通过分配保存多个算法性能不透明结构所需内存的方式，创建多个算法性能对象。

```
mcdnnStatus_t mcdnnCreateAlgorithmPerformance(
    mcdnnAlgorithmPerformance_t *algoPerf,
    int numberToCreate)
```

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

2.2.1.9 mcdnnCreateDropoutDescriptor()

```
mcdnnStatus_t mcdnnCreateDropoutDescriptor(
    mcdnnDropoutDescriptor_t *dropoutDesc)
```

此函数通过分配保存丢弃描述符不透明结构所需内存的方式，创建通用丢弃描述符对象。更多信息，参见 mcdnnDropoutDescriptor_t。

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

2.2.1.10 mcdnnCreateFilterDescriptor()

此函数通过分配保存卷积核描述符不透明结构所需内存的方式，创建卷积核描述符对象。更多信息，参见 mcdnnFilterDescriptor_t。

```
mcdnnStatus_t mcdnnCreateFilterDescriptor(
    mcdnnFilterDescriptor_t *filterDesc)
```

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

2.2.1.11 mcdnnCreateLRNDescriptor()

此函数分配内存，用于存储 LRN 和 DivisionNormalization 层操作所需的数据。并返回一个描述符，用于后续层正向和反向调用。

```
mcdnnStatus_t mcdnnCreateLRNDescriptor(
    mcdnnLRNDescriptor_t *poolingDesc)
```

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

2.2.1.12 mcdnnCreateOpTensorDescriptor()

此函数用于创建张量逐点数学运算（Pointwise Math）描述符。更多信息，参见 mcdnnOpTensorDescriptor_t。

```
mcdnnStatus_t mcdnnCreateOpTensorDescriptor(
    mcdnnOpTensorDescriptor_t* opTensorDesc)
```

参数

opTensorDesc

输出。指针，指向保存张量逐点数学运算（如加法，乘法等）描述的结构。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

MCDNN_STATUS_BAD_PARAM

传入函数的张量逐点数学运算描述符无效。

MCDNN_STATUS_ALLOC_FAILED

此张量逐点数学运算描述符的内存分配失败。

2.2.1.13 mcdnnCreatePoolingDescriptor()

```
mcdnnStatus_t mcdnnCreatePoolingDescriptor(
    mcdnnPoolingDescriptor_t    *poolingDesc)
```

此函数通过分配保存池化描述符不透明结构所需内存的方式，创建池化描述符对象。

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

2.2.1.14 mcdnnCreateReduceTensorDescriptor()

此函数通过分配保存归约张量描述符不透明结构所需内存的方式，创建归约张量描述符对象。

```
mcdnnStatus_t mcdnnCreateReduceTensorDescriptor(
    mcdnnReduceTensorDescriptor_t* reduceTensorDesc)
```

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

MCDNN_STATUS_BAD_PARAM

reduceTensorDesc 是一个 NULL 指针。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

2.2.1.15 mcdnnCreateSpatialTransformerDescriptor()

此函数通过分配保存空间转换描述符不透明结构所需内存的方式，创建通用空间转换描述符对象。

```
mcdnnStatus_t mcdnnCreateSpatialTransformerDescriptor(
    mcdnnSpatialTransformerDescriptor_t *stDesc)
```

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

2.2.1.16 mcdnnCreateTensorDescriptor()

```
mcdnnStatus_t mcdnnCreateTensorDescriptor(
mcdnnTensorDescriptor_t *tensorDesc)
```

此函数通过分配保存通用张量描述符不透明结构所需内存的方式，创建通用张量描述符对象。数据初始化为全零。

参数

tensorDesc

输入。指向指针的指针，其中分配的张量描述符对象的地址应减少张量。

返回值

MCDNN_STATUS_BAD_PARAM

输入参数无效。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

MCDNN_STATUS_SUCCESS

已成功创建对象。

2.2.1.17 mcdnnCreateTensorTransformDescriptor()

此函数通过分配保存通用张量转换描述符不透明结构所需内存的方式，创建张量转换描述符对象。张量数据初始化为全零。使用 mcdnnSetTensorTransformDescriptor() 函数初始化此函数创建的描述符。

```
mcdnnStatus_t mcdnnCreateTensorTransformDescriptor(
 mcdnnTensorTransformDescriptor_t *transformDesc);
```

参数

transformDesc

输出。指向未初始化张量转换描述符的指针。

返回值

MCDNN_STATUS_SUCCESS

已成功创建描述符对象。

MCDNN_STATUS_BAD_PARAM

transformDesc 为 NULL。

MCDNN_STATUS_ALLOC_FAILED

内存分配失败。

2.2.1.18 mcdnnDeriveBNTensorDescriptor()

该函数从层的 x 数据描述符中获取 BN scale，invVariance，bnBias 和 bnScale 子张量的辅助张量描述符。

```
mcdnnStatus_t mcdnnDeriveBNTensorDescriptor(
    mcdnnTensorDescriptor_t          derivedBnDesc,
    const mcdnnTensorDescriptor_t    xDesc,
    mcdnnBatchNormMode_t             mode)
```

此函数创建的张量描述符的使用，同 mcdnnBatchNormalizationForwardInference() 和 mcdnnBatchNormalizationForwardTraining() 函数中的 bnScaleBiasMeanVarDesc 参数，以及 mcdnnBatchNormalizationBackward() 函数中的 bnScaleBiasDiffDesc 参数。得到的结果维度：

- 在 BATCHNORM_MODE_SPATIAL 模式下，4D 张量中为 1xCx1x1，5D 张量中为 1xCx1x1x1。目前不支持 5D。
- 在 BATCHNORM_MODE_PER_ACTIVATION 模式下，4D 张量中为 1xCxHxW，5D 张量中为 1xCxDxHxW。目前不支持 5D。

对于 HALF 输入数据类型，生成的张量描述符具有 FLOAT 类型。对于其他数据类型，则具有与输入数据相同的类型。

注解: 仅支持 4D 和 5D 张量。应首先使用 mcdnnCreateTensorDescriptor() 创建 deredBnDesc。xDesc 是层的 x 数据描述符，在调用此函数之前，必须对 xDesc 设置正确的维度。

参数

derivedBnDesc

输出。已创建的张量描述符的句柄。

xDesc

输入。已创建和初始化的层的 x 数据描述符句柄。

mode

输入。BN 层操作模式。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_BAD_PARAM

BN 模式无效。

2.2.1.19 mcdnnDeriveNormTensorDescriptor()

该函数从层的 x 数据描述符和 norm 模式获取归一化，均值，invariance，normBias 和 normScale 子张量的描述符。归一化，均值和 invariance 共享一个描述符，normBias 和 normScale 共享一个描述符。

```
mcdnnStatus_t MCDNNWINAPI
mcdnnDeriveNormTensorDescriptor(mcdnnTensorDescriptor_t
↪derivedNormScaleBiasDesc,
    mcdnnTensorDescriptor_t derivedNormMeanVarDesc,
    const mcdnnTensorDescriptor_t xDesc,
    mcdnnNormMode_t mode,
    int groupCnt)
```

此函数创建的张量描述符的使用，同 mcdnnNormalizationForwardInference() 和 mcdnnNormalizationForwardTraining() 函数中的 normScaleBiasDesc 或 normMeanVarDesc 参数，以及 mcdnnNormalizationBackward() 函数中的 dNormScaleBiasDesc 和 normMeanVarDesc 参数。

得到的结果维度：

- 在 MCDNN_NORM_PER_ACTIVATION 模式下，4D 张量中为 1xCx1x1，5D 张量中为 1xCx1x1x1
- 在 MCDNN_NORM_PER_CHANNEL 模式下，4D 张量中为 1xCxHxW，5D 张量中为 1xCxDxHxW

对于 HALF 输入数据类型，生成的张量描述符具有 FLOAT 类型。对于其他数据类型，则具有与输入数据相同的类型。

- 仅支持 4D 和 5D 张量。
- 应首先使用 mcdnnCreateTensorDescriptor() 创建 derivedNormScaleBiasDesc 和 derivedNormMeanVarDesc。
- xDesc 是层的 x 数据描述符，在调用此函数之前，必须对 xDesc 设置正确的维度。

参数

derivedNormScaleBiasDesc

输出。已创建的张量描述符的句柄。

derivedNormMeanVarDesc

输出。已创建的张量描述符的句柄。

xDesc

输入。已创建和初始化的层的 x 数据描述符句柄。

mode

输入。BN 层操作模式。

groupCnt

输入。分组卷积数。当前仅支持 1。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_BAD_PARAM

BN 模式无效。

2.2.1.20 mcdnnDestroy()

```
mcdnnStatus_t mcdnnDestroy(mcdnnHandle_t handle)
```

此函数释放 mcDNN 句柄使用的资源。此函数通常是使用 mcDNN 句柄的特定句柄进行的最后一次调用。由于 mcdnnCreate() 分配一些内部资源，因此通过调用 mcdnnDestroy() 来释放这些资源将隐式调用 macaDeviceSynchronize；因此，建议的最佳做法是在 performance-critical 代码路径之外调用 mcdnnCreate/mcdnnDestroy。

参数

handle

输入。要销毁的 mcDNN 句柄

返回值

MCDNN_STATUS_SUCCESS

mcDNN 上下文已销毁成功。

MCDNN_STATUS_BAD_PARAM

提供的指针无效（NULL）。

2.2.1.21 mcdnnDestroyActivationDescriptor()

```
mcdnnStatus_t mcdnnDestroyActivationDescriptor(
    mcdnnActivationDescriptor_t activationDesc)
```

此函数用于销毁已创建的激活描述符对象。

返回值

MCDNN_STATUS_SUCCESS

对象已销毁成功。

2.2.1.22 mcdnnDestroyAlgorithmDescriptor()

此函数用于销毁已创建的算法描述符对象。

```
mcdnnStatus_t mcdnnDestroyAlgorithmDescriptor(
    mcdnnActivationDescriptor_t algorithmDesc)
```

返回值

MCDNN_STATUS_SUCCESS

对象已销毁成功。

2.2.1.23 mcdnnDestroyAlgorithmPerformance()

此函数用于销毁已创建的算法描述符对象。

```
mcdnnStatus_t mcdnnDestroyAlgorithmPerformance(
    mcdnnAlgorithmPerformance_t algoPerf)
```

返回值

MCDNN_STATUS_SUCCESS

对象已销毁成功。

2.2.1.24 mcdnnDestroyDropoutDescriptor()

```
mcdnnStatus_t mcdnnDestroyDropoutDescriptor(
    mcdnnDropoutDescriptor_t dropoutDesc)
```

此函数用于销毁已创建的丢弃描述符对象。

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

2.2.1.25 mcdnnDestroyFilterDescriptor()

此函数用于销毁卷积核描述符对象。

```
mcdnnStatus_t mcdnnDestroyFilterDescriptor(
    mcdnnFilterDescriptor_t filterDesc)
```

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

2.2.1.26 mcdnnDestroyLRNDescriptor()

此函数用于销毁已创建的 LRN 描述符对象。

```
mcdnnStatus_t mcdnnDestroyLRNDescriptor(
    mcdnnLRNDescriptor_t lrnDesc)
```

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

2.2.1.27 mcdnnDestroyOpTensorDescriptor()

此函数用于删除张量逐点数学运算描述符对象。

```
mcdnnStatus_t mcdnnDestroyOpTensorDescriptor(
    mcdnnOpTensorDescriptor_t opTensorDesc)
```

参数

opTensorDesc

输入。指针，指向保存了要删除的张量逐点数学运算描述符的结构。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

2.2.1.28 mcdnnDestroyPoolingDescriptor()

```
mcdnnStatus_t mcdnnDestroyPoolingDescriptor(
    mcdnnPoolingDescriptor_t poolingDesc)
```

此函数用于销毁已创建的池化描述符对象。

返回值

MCDNN_STATUS_SUCCESS

对象已销毁成功。

2.2.1.29 mcdnnDestroyReduceTensorDescriptor()

此函数用于销毁已创建的归约张量描述符对象。当输入指针为 NULL 时，该函数不执行销毁操作。

```
mcdnnStatus_t mcdnnDestroyReduceTensorDescriptor(
    mcdnnReduceTensorDescriptor_t tensorDesc)
```

参数

tensorDesc

输入。指向要销毁的归约张量描述符对象的指针。

返回值

MCDNN_STATUS_SUCCESS

对象已销毁成功。

2.2.1.30 mcdnnDestroySpatialTransformerDescriptor()

此函数用于销毁已创建的空间转换描述符对象。

```
mcdnnStatus_t mcdnnDestroySpatialTransformerDescriptor(
    mcdnnSpatialTransformerDescriptor_t stDesc)
```

返回值

MCDNN_STATUS_SUCCESS

对象已销毁成功。

2.2.1.31 mcdnnDestroyTensorDescriptor()

```
mcdnnStatus_t mcdnnDestroyTensorDescriptor(mcdnnTensorDescriptor_t
↪tensorDesc)
```

此函数用于销毁已创建的张量描述符对象。当输入指针为 NULL 时，该函数不执行销毁操作。

参数

tensorDesc

输入。指向要销毁的张量描述符对象的指针。

返回值

MCDNN_STATUS_SUCCESS

对象已销毁成功。

2.2.1.32 mcdnnDestroyTensorTransformDescriptor()

```
mcdnnStatus_t mcdnnDestroyTensorTransformDescriptor(
 mcdnnTensorTransformDescriptor_t transformDesc);
```

此函数用于销毁已创建的张量转换描述符。

参数

transformDesc

输入。要销毁的张量转换描述符。

返回值

MCDNN_STATUS_SUCCESS

描述符已销毁成功。

2.2.1.33 mcdnnDivisiveNormalizationForward()

该函数执行正向空间 DivisiveNormalization 层计算。它将层中的每个值除以其空间近邻的标准偏差。请注意，DivisiveNormalization 仅实现计算的 x/max(c, sigma_x) 部分，其中 sigma_x 是 x 的空间邻域上的方差。

```
mcdnnStatus_t mcdnnDivisiveNormalizationForward(
    mcdnnHandle_t handle,
    mcdnnLRNDescriptor_t normDesc,
    mcdnnDivNormMode_t mode,
    const void *alpha,
    const mcdnnTensorDescriptor_t xDesc,
    const void *x,
    const void *means,
    void *temp,
    void *temp2,
    const void *beta,
    const mcdnnTensorDescriptor_t yDesc,
    void *y)
```

完整的 LCN 计算可以作为两步进程实现：x_m = x-mean(x)；y = x_m/max(c, sigma(x_m))；

x-mean(x) 通常被称为计算的“减法归一化（Subtractive Normalization）”部分，可以使用 mcDNN 平均池化层实现，然后调用 addTensor。

注解： 支持的张量格式为 NCHW（适用于 4D）和 NCDHW（适用于 5D），具有任意非重叠非负值步幅。仅支持 4D 和 5D 张量。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

normDesc

输入。已初始化的 LRN 参数描述符的句柄。此描述符用于 LRN 和 DivisiveNormalization 层。

divNormMode

输入。DivisiveNormalization 层操作模式。目前仅支持实现 MCDNN_DIVNORM_PRECOMPUTED_MEANS。归一化是使用用户要预计算的均值输入张量来执行的。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将层输出值与目标张量中的先验值混合，如下所示：

```
dstValue = alpha[0]*resultValue + beta[0]*priorDstValue
```

xDesc, yDesc

输入。输入和输出张量的张量描述符对象。请注意，x、means、temp、temp2 张量共享 xDesc。

x

输入。设备内存中的输入张量数据指针。

means

输入。设备内存中的输入均值张量数据指针。请注意，此张量可以为 NULL（在这种情况下，其值在计算过程中假定为零）。此张量也可以不包含均值，其值可以为任何值，常用的变量是具有归一化正态内核（如高斯）卷积的结果。

temp, temp2

Workspace. 设备内存中的临时张量。这些张量用于在正向传递过程中计算中间值。但不需要保留为从正向传递到反向传递的输入。正向和反向传递都使用 xDesc 作为其描述符。

y

输出。设备内存中的指针，指向用于正向 DivisiveNormalization 计算的张量。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- x，y，temp 和 temp2 中任一张量指针为 NULL。
- 输入张量或输出张量的维数不在 [4,5] 范围内。
- 任意两个输入或输出张量之间的维度不匹配。
- 对于指针 x == y 时的就地计算，输入数据和输出数据张量之间的步幅不匹配。
- Alpha 或 beta 指针为 NULL。
- LRN 描述符参数超出其有效范围。
- 任意张量步幅均为负的。

MCDNN_STATUS_UNSUPPORTED

该函数不支持已提供的配置，例如，不支持输入和输出张量之间的步幅不匹配（同一维度下）。

2.2.1.34 mcdnnDropoutForward()

```
mcdnnStatus_t mcdnnDropoutForward(
    mcdnnHandle_t handle,
    const mcdnnDropoutDescriptor_t dropoutDesc,
    const mcdnnTensorDescripter_t xdesc,
    const void *x,
    const mcdnnTensorDescripter_t ydesc,
    void *y,
    void *reserveSpace,
    size_t reserveSpaceSizeInBytes)
```

此函数以 x 执行正向丢弃操作，以 y 返回结果。如果 dropout 用作 mcdnnSetDropoutDescriptor() 的参数，则 x 被丢弃的值将替换为 0，其余部分将按 1/(1-dropout) 进行缩放。此函数不应与使用相同状态的另一个 mcdnnDropoutForward() 函数同时运行。

注解： 对于 fully packed 的张量，性能更好，并且该函数不能在推理过程中调用。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

dropoutDesc

输入。已创建的丢弃描述符对象。

xDesc

输入。已初始化的张量描述符的句柄。

x

输入。指针，指向 xDesc 描述符描述的张量数据。

yDesc

输入。已初始化的张量描述符的句柄。

y

输出。指针，指向 yDesc 描述符描述的张量数据。

reserveSpace

输出。指针，指向此函数使用的用户分配的 GPU 内存。reserveSpace 的内容不会在 mcdnnDropoutForward() 和 mcdnnDropoutBackward() 调用之间发生变化。

reserveSpaceSizeInBytes

输入。reserveSpace 的大小。

返回值

MCDNN_STATUS_SUCCESS

调用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 输入张量和输出张量的元素数不同。
- 输入张量和输出张量的数据类型不同。
- 输入张量和输出张量的步幅不同，并使用就地操作（即 x 和 y 指针相等）。
- 提供的 reserveSpaceSizeInBytes 小于 mcdnnDropoutGetReserveSpaceSize() 返回的值。
- 没有在带有非 NULL 状态参数的 dropoutDesc 上调用 mcdnnSetDropoutDescriptor()。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.35 mcdnnDropoutGetReserveSpaceSize()

```
mcdnnStatus_t mcdnnDropoutGetReserveSpaceSize(
    mcdnnTensorDescripter_t xDesc,
    size_t *sizeInBytes)
```

此函数用于查询使用 xDesc 给出的输入维度运行丢弃所需的预留量。同一预留空间将传入 mcdnnDropoutForward() 和 mcdnnDropoutBackward()，并且其内容在 mcdnnDropoutForward() 和 mcdnnDropoutBackward() 调用之间保持不变。

参数

xDesc 输入。已初始化的张量描述符的句柄，描述丢弃操作的输入。

sizeInBytes

输出。使用 xDesc 指定的输入张量描述符运行丢弃所需的 GPU 内存量，作为预留空间。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

2.2.1.36 mcdnnDropoutGetStatesSize()

```
mcdnnStatus_t mcdnnDropoutGetStatesSize(
    mcdnnHandle_t handle,
    size_t *sizeInBytes)
```

此函数用于查询存储 mcdnnDropoutForward() 函数使用的随机数生成器状态所需的空间量。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

sizeInBytes

输出。存储随机数生成器状态所需的 GPU 内存量。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

2.2.1.37 mcdnnGetActivationDescriptor()

```
mcdnnStatus_t mcdnnGetActivationDescriptor(
    const mcdnnActivationDescriptor_t   activationDesc,
    mcdnnActivationMode_t              *mode,
    mcdnnNanPropagation_t              *reluNanOpt,
    double                             *coef)
```

此函数用于查询已初始化的通用激活描述符对象。

参数

activationDesc

输入。已创建的激活描述符的句柄。

mode

输出。指定激活模式的枚举。

reluNanOpt

输出。指定 NaN 传播模式的枚举。

coef

输出。浮点数，用于在激活模式设置为 MCDNN_ACTIVATION_CHIPPED_RELU 时指定裁剪阈值，或在激活模式设置为 MCDNN_ACTIVATION_ELU 时指定 alpha 系数。

返回值

MCDNN_STATUS_SUCCESS

对象查询成功。

2.2.1.38 mcdnnGetActivationDescriptorSwishBeta()

```
mcdnnStatus_t mcdnnGetActivationDescriptorSwishBeta(
    mcdnnActivationDescriptor_t activationDesc,
    double* swish_beta)
```

此函数用于查询当前为 Swish 激活设置的 beta 参数。目前不支持。

参数

activationDesc

输入。已创建的激活描述符的句柄。

swish_beta

输出。指针，指向接收当前配置的 Swish beta 参数的双精度值。

返回值

MCDNN_STATUS_SUCCESS

beta 参数查询成功。

MCDNN_STATUS_BAD_PARAM

activationDesc 或 swish_beta 中任一个为 NULL。

2.2.1.39 mcdnnGetAlgorithmDescriptor()

此函数用于查询已初始化的通用算法描述符对象。

```
mcdnnStatus_t mcdnnGetAlgorithmDescriptor(
    const mcdnnAlgorithmDescriptor_t algoDesc,
    mcdnnAlgorithm_t *algorithm)
```

参数

algorithmDesc

输入。已创建的算法描述符的句柄。

algorithm

输入。指定算法的结构。

返回值

MCDNN_STATUS_SUCCESS

对象查询成功。

2.2.1.40 mcdnnGetAlgorithmPerformance()

此函数用于查询已初始化的通用算法性能对象。

```
mcdnnStatus_t mcdnnGetAlgorithmPerformance(
    const mcdnnAlgorithmPerformance_t algoPerf,
    mcdnnAlgorithmDescriptor_t* algoDesc,
    mcdnnStatus_t* status,
    float* time,
    size_t* memory)
```

参数

algoPerf

输入/输出。已创建的算法性能对象的句柄。

algoDesc

输出。性能结果描述的算法描述符。

status

输出。运行 algoDesc 算法返回的 mcDNN 状态。

timecoef

输出。GPU 运行 algoDesc 算法消耗的时间。

memory

输出。运行 algoDesc 算法所需的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

对象查询成功。

2.2.1.41 mcdnnGetAlgorithmSpaceSize()

此函数用于查询调用 mcdnnSaveAlgorithm() 所需的主机内存量，类似于使用获取工作空间大小的函数查询所需的设备内存量。

```
mcdnnStatus_t mcdnnGetAlgorithmSpaceSize(
    mcdnnHandle_t handle,
    mcdnnAlgorithmDescriptor_t algoDesc,
    size_t* algoSpaceSizeInBytes)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

algoDesc

输入。已创建的算法描述符。

algoSpaceSizeInBytes

输出。作为工作空间所需的主机内存量，以便能够从指定的 algoDesc 中保存元数据。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

至少有一个参数为 NULL。

2.2.1.42 mcdnnGetCallback()

此函数用于查询 mcDNN 错误报告功能的内部状态。

```
mcdnnStatus_t mcdnnGetCallback(
    unsigned mask,
    void **udata,
    mcdnnCallback_t fptr)
```

参数

mask

输出。指针，指向输出当前内部错误报告消息位掩码的地址。

udata

输出。指针，指向当前内部存储 udata 的地址。

fptr

输出。指针，指向当前内部存储回调函数指针的地址。使用内置的默认回调函数时，将输出 NULL。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

任意输入参数为 NULL。

2.2.1.43 mcdnnGetMacartVersion()

给定的同一版本 mcDNN 库可以根据不同的 MXMACA 工具包版本进行编译。此函数返回编译当前使用的 mcDNN 库所依据的 MXMACA 工具包版本。

2.2.1.44 mcdnnGetDropoutDescriptor()

```
mcdnnStatus_t mcdnnGetDropoutDescriptor(
    mcdnnDropoutDescriptor_t dropoutDesc,
    mcdnnHandle_t handle,
    float *dropout,
    void **states,
    unsigned long long *seed)
```

此函数用于查询已初始化的丢弃描述符的字段。

参数

dropoutDesc

输入。已初始化的丢弃描述符。

handle

输入。已创建的 mcDNN 上下文的句柄。

dropout

输出。在丢弃层输入值设置为 0 的概率。

states

输出。指向用户分配的 GPU 内存的指针，该内存将保存随机数生成器状态。

seed

输出。用于初始化随机数生成器状态的种子（Seed）。

返回值

MCDNN_STATUS_SUCCESS

调用成功。

MCDNN_STATUS_BAD_PARAM

一个或多个参数是无效指针。

2.2.1.45 mcdnnGetErrorString()

此函数用于将 mcDNN 状态代码转换为空字符结尾的（ASCIIZ）静态字符串。例如，当输入参数为 MCDNN_STATUS_SUCCESS 时，返回的字符串为 MCDNN_STATUS_SUCCESS。将无效的状态值传入此函数时，返回的字符串为 MCDNN_UNKNOWN_STATUS。

```
const char * mcdnnGetErrorString(mcdnnStatus_t status)
```

参数

status

输入。mcDNN 枚举状态代码。

返回值

指针，指向带有状态名称的空字符结尾字符串。

2.2.1.46 mcdnnGetFilter4dDescriptor()

此函数用于查询已初始化的 Filter4d 描述符对象的参数。

```
mcdnnStatus_t mcdnnGetFilter4dDescriptor(
    const mcdnnFilterDescriptor_t filterDesc,
    mcdnnDataType_t *dataType,
    mcdnnTensorFormat_t *format,
    int *k,
    int *c,
    int *h,
    int *w)
```

参数

filterDesc

输入。已创建的卷积核描述符的句柄。

datatype

输出。数据类型。

format

输出。格式类型。

k

输出。输出特征图的数量。

c

输出。输入特征图的数量。

h

输出。每个卷积核的高度。

w

输出。每个卷积核的宽度。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

2.2.1.47 mcdnnGetFilterNdDescriptor()

此函数用于查询已初始化的 FilterNd 描述符对象。

```
mcdnnStatus_t mcdnnGetFilterNdDescriptor(
    const mcdnnFilterDescriptor_t wDesc,
    int nbDimsRequested,
    mcdnnDataType_t *dataType,
    mcdnnTensorFormat_t *format,
    int *nbDims,
    int filterDimA[])
```

参数

wDesc

输入。已初始化的卷积核描述符的句柄。

nbDimsRequested

输入。期望卷积核描述符的维度。也是 filterDimA 数组能够保存结果所需的最小空间。

datatype

输出。数据类型。

format

输出。格式类型。

nbDims

输出。卷积核的实际维度。

filterDimA

输出。至少包含 nbDimsRequested 维度的数组，由提供的卷积核描述符中的 filter 参数填充。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

nbDimsRequested 参数为负值。

2.2.1.48 mcdnnGetFilterSizeInBytes()

该函数返回内存中关于给定描述符的卷积核张量的大小。它可以用来了解分配用于保存卷积核张量的 GPU 内存量。

```
mcdnnStatus_t
mcdnnGetFilterSizeInBytes(const mcdnnFilterDescriptor_t filterDesc, size_t
↪*size) ;
```

参数

filterDesc

输入。已初始化的卷积核描述符的句柄。

size

输出。保存张量所需的 GPU 内存大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

filterDesc 有效。

MCDNN_STATUS_BAD_PARAM

filerDesc 无效。

2.2.1.49 mcdnnGetLRNDescriptor()

此函数用于检索存储在已初始化的 LRN 描述符对象中的值。

```
mcdnnStatus_t mcdnnGetLRNDescriptor(
    mcdnnLRNDescriptor_t normDesc,
    unsigned *lrnN,
    double *lrnAlpha,
    double *lrnBeta,
    double *lrnK)
```

参数

normDesc

输入。已创建的 LRN 描述符的句柄。

lrnN, lrnAlpha, lrnBeta, lrnK

输出。指针，用于接收存储在描述符对象中的参数值。更多信息，参见 mcdnnSetLRNDescriptor()。这些指针中的任意一个都可以为 NULL（不会为相应的参数返回任何值）。

返回值

MCDNN_STATUS_SUCCESS

函数已成功完成。

2.2.1.50 mcdnnGetOpTensorDescriptor()

该函数返回传入的张量逐点数学运算描述符的配置。

```
mcdnnStatus_t mcdnnGetOpTensorDescriptor(
    const mcdnnOpTensorDescriptor_t opTensorDesc,
    mcdnnOpTensorOp_t *opTensorOp,
    mcdnnDataType_t *opTensorCompType,
    mcdnnNanPropagation_t *opTensorNanOpt)
```

参数

opTensorDesc

输入。传入张量逐点数学运算描述符以获取配置。

opTensorOp

输出。指针，指向与此张量逐点数学运算描述符关联的张量逐点数学运算类型。

opTensorCompType

输出。指针，指向与此张量逐点数学运算描述符关联的 mcDNN 数据类型。

opTensorNanOpt

输出。指针，指向与此张量逐点数学运算描述符关联的 NaN 传播选项。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

MCDNN_STATUS_BAD_PARAM

传入的输入张量逐点数学运算描述符无效。

2.2.1.51 mcdnnGetPooling2dDescriptor()

```
mcdnnStatus_t mcdnnGetPooling2dDescriptor(
    const mcdnnPoolingDescriptor_t    poolingDesc,
    mcdnnPoolingMode_t                *mode,
    mcdnnNanPropagation_t             *maxpoolingNanOpt,
    int                               *windowHeight,
    int                               *windowWidth,
    int                               *verticalPadding,
    int                               *horizontalPadding,
    int                               *verticalStride,
    int                               *horizontalStride)
```

此函数用于查询已创建的 2D 池化描述符对象。

参数

poolingDesc

输入。已创建的池化描述符的句柄。

mode

输出。指定池化模式的枚举。

maxpoolingNanOpt

输出。指定 NaN 传播模式的枚举。

windowHeight

输出。池化窗口的高度。

windowWidth

输出。池化窗口的宽度。

verticalPadding

输出。垂直填充的大小。

horizontalPadding

输出。水平填充的大小。

verticalStride

输出。池化垂直步幅。

horizontalStride

输出。池化水平步幅。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

2.2.1.52 mcdnnGetPooling2dForwardOutputDim()

```
mcdnnStatus_t mcdnnGetPooling2dForwardOutputDim(
    const mcdnnPoolingDescriptor_t      poolingDesc,
    const mcdnnTensorDescriptor_t       inputDesc,
    int                                *outN,
    int                                *outC,
    int                                *outH,
    int                                *outW)
```

该函数用于提供应用 2D 池化后张量的输出维度。输出图像的各维度 h 和 w 计算如下：

```
outputDim = 1 + (inputDim + 2*padding - windowDim)/poolingStride;
```

参数

poolingDesc

输入。已初始化的池化描述符的句柄。

inputDesc

输入。已初始化的输入张量描述符的句柄。

N

输出。输出的图像数。

C

输出。输出的通道（channel）数。

H

输出。输出的图像高度。

W

输出。输出的图像宽度。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 尚未初始化 poolingDesc。
- poolingDesc 或 inputDesc 具有无效的维数（分别需要为 2 和 4）。

2.2.1.53 mcdnnGetPoolingNdDescriptor()

```
mcdnnStatus_t mcdnnGetPoolingNdDescriptor(
    const mcdnnPoolingDescriptor_t      poolingDesc,
    int                                 nbDimsRequested,
    mcdnnPoolingMode_t                 *mode,
    mcdnnNanPropagation_t              *maxpoolingNanOpt,
    int                                *nbDims,
    int                                 windowDimA[],
    int                                 paddingA[],
    int                                 strideA[])
```

此函数用于查询已初始化的通用池化描述符对象。

参数

poolingDesc

输入。已创建的池化描述符的句柄。

nbDimsRequested

输入。期望池化描述符的维度。也是 windowDimA、paddingA、strideA 数组能够保存结果所需的最小空间。

mode

输出。指定池化模式的枚举。

maxpoolingNanOpt

输入。指定 NaN 传播模式的枚举。

nbDims

输出。池化描述符的实际维度。

windowDimA

输出。至少包含 nbDimsRequested 维度的数组，由提供的池化描述符中的 window 参数填充。

paddingA

输出。至少包含 nbDimsRequested 维度的数组，由提供的池化描述符中的 padding 参数填充。

strideA

输出。至少包含 nbDimsRequested 维度的数组，由提供的池化描述符中的 stride 参数填充。

返回值

MCDNN_STATUS_SUCCESS

对象查询成功。

MCDNN_STATUS_NOT_SUPPORTED

参数 nbDimsRequested 大于 MCDNN_DIM_MAX。

2.2.1.54 mcdnnGetPoolingNdForwardOutputDim()

```
mcdnnStatus_t mcdnnGetPoolingNdForwardOutputDim(
    const mcdnnPoolingDescriptor_t  poolingDesc,
    const mcdnnTensorDescriptor_t   inputDesc,
    int                             nbDims,
    int                             outDimA[])
```

该函数用于提供应用 Nd 池化后张量的输出维度。输出张量的 (nbDims-2)-D 图像的每个维度计算如下：

```
outputDim = 1 + (inputDim + 2*padding - windowDim)/poolingStride;
```

参数

poolingDesc

输入。已初始化的池化描述符的句柄。

inputDesc

输入。已初始化的输入张量描述符的句柄。

nbDims

输入。要应用池化的维数。

outDimA

输出。nbDims 输出维度数组。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 尚未初始化 poolingDesc。
- nbDims 的值与 poolingDesc 和 inputDesc 的维数不一致。

2.2.1.55 mcdnnGetProperty()

此函数用于将 mcDNN 库版本号的特定部分写入提供的主机存储中。

```
mcdnnStatus_t mcdnnGetProperty(
    libraryPropertyType type,
    int *value)
```

参数

type

输入。指示函数报告 mcDNN 主要版本，次要版本或补丁级别数值的枚举类型。

value

输出。应写入版本信息的主机指针。

返回值

MCDNN_STATUS_INVALID_VALUE

type 参数的值无效。

MCDNN_STATUS_SUCCESS

版本信息已成功存储在提供的地址中。

2.2.1.56 mcdnnGetReduceTensorDescriptor()

此函数用于查询已初始化的归约张量描述符对象。

```
mcdnnStatus_t mcdnnGetReduceTensorDescriptor(
    const mcdnnReduceTensorDescriptor_t reduceTensorDesc,
    mcdnnReduceTensorOp_t *reduceTensorOp,
    mcdnnDataType_t *reduceTensorCompType,
    mcdnnNanPropagation_t *reduceTensorNanOpt,
    mcdnnReduceTensorIndices_t *reduceTensorIndices,
    mcdnnIndicesType_t *reduceTensorIndicesType)
```

参数

reduceTensorDesc

输入。指向已初始化的归约张量描述符对象的指针。

reduceTensorOp

输出。指定归约张量操作的枚举。

reduceTensorCompType

输出。指定归约计算数据类型的枚举。

reduceTensorNanOpt

输出。指定 NaN 传播模式的枚举。

reduceTensorIndices

输出。指定归约张量索引的枚举。

reduceTensorIndicesType

输出。指定归约张量索引类型的枚举。

返回值

MCDNN_STATUS_SUCCESS

对象查询成功。

MCDNN_STATUS_BAD_PARAM

ReduceTensorDesc 为 NULL。

2.2.1.57 mcdnnGetReductionIndicesSize()

这是一个辅助函数，用于返回要传入归约（给定输入和输出张量）的最小索引空间。

```
mcdnnStatus_t mcdnnGetReductionIndicesSize(
    mcdnnHandle_t handle,
    const mcdnnReduceTensorDescriptor_t reduceDesc,
    const mcdnnTensorDescriptor_t aDesc,
    const mcdnnTensorDescriptor_t cDesc,
    size_t *sizeInBytes)
```

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

reduceDesc

输入。指向已初始化的归约张量描述符对象的指针。

aDesc

输入。指向输入张量描述符的指针。

cDesc

输入。指向输出张量描述符的指针。

sizeInBytes

输出。要传入归约的最小索引空间。

返回值

MCDNN_STATUS_SUCCESS

成功返回索引空间大小。

2.2.1.58 mcdnnGetReductionWorkspaceSize()

这是一个辅助函数，用于返回要传入归约（给定输入和输出张量）的最小工作空间。

```
mcdnnStatus_t mcdnnGetReductionWorkspaceSize(
    mcdnnHandle_t handle,
    const mcdnnReduceTensorDescriptor_t reduceDesc,
    const mcdnnTensorDescriptor_t aDesc,
    const mcdnnTensorDescriptor_t cDesc,
    size_t *sizeInBytes)
```

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

reduceDesc

输入。指向已初始化的归约张量描述符对象的指针。

aDesc

输入。指向输入张量描述符的指针。

cDesc

输入。指向输出张量描述符的指针。

sizeInBytes

输出。要传入归约的最小索引空间。

返回值

MCDNN_STATUS_SUCCESS

成功返回工作空间大小。

2.2.1.59 mcdnnGetStream()

此函数检索在 mcDNN 句柄中编程的用户 MXMACA 流。未在 mcDNN 句柄中设置用户的 MXMACA 流时，此函数将报告 nullstream。

```
mcdnnStatus_t mcdnnGetStream(
    mcdnnHandle_t handle,
    mcStream_t *streamId)
```

参数

handle

输入。指向 mcDNN 句柄的指针。

streamID

输出。存储来自 mcDNN 句柄的当前 MXMACA 流的指针。

返回值

MCDNN_STATUS_BAD_PARAM

无效（NULL）句柄。

MCDNN_STATUS_SUCCESS

已成功检索流标识符。

2.2.1.60 mcdnnGetTensor4dDescriptor()

此函数用于查询已初始化的 Tensor4d 描述符对象的参数。

```
mcdnnStatus_t mcdnnGetTensor4dDescriptor(
    const mcdnnTensorDescriptor_t tensorDesc,
    mcdnnDataType_t *dataType,
    int *n,
    int *c,
    int *h,
    int *w,
    int *nStride,
    int *cStride,
    int *hStride,
    int *wStride)
```

参数

tensorDesc

输入。已初始化的张量描述符的句柄。

datatype

输出。数据类型。

n

输出。图像数量。

c

输出。每个图像的特征图数量。

h

输出。每个特征图的高度。

w

输出。每个特征图的宽度。

nStride

输出。两个连续图像之间的步幅。

cStride

输出。两个连续特征图之间的步幅。

hStride

输出。两个连续行之间的步幅。

wStride

输出。两个连续列之间的步幅。

返回值

MCDNN_STATUS_SUCCESS

操作成功。

2.2.1.61 mcdnnGetTensorNdDescriptor()

此函数用于检索存储在已初始化的 TensorNd 描述符对象中的值。

```
mcdnnStatus_t mcdnnGetTensorNdDescriptor(
    const mcdnnTensorDescriptor_t tensorDesc,
    int nbDimsRequested,
    mcdnnDataType_t *dataType,
    int *nbDims,
    int dimA[],
    int strideA[])
```

参数

tensorDesc

输入。已初始化的张量描述符的句柄。

nbDimsRequested

输入。从给定张量描述符中提取的维数。它也是 dimA 和 strideA 数组的最小尺寸。如果该数字大于得到的 nbDims[0]，则只返回 nbDims[0] 个维度。

datatype

输出。数据类型。

nbDims

输出。张量的实际维数将以 nbDims[0] 返回。

dimA

输出。至少包含 nbDimsRequested 维度的数组，由提供的张量描述符的维度填充。

strideA

输出。至少包含 nbDimsRequested 维度的数组，由提供的张量描述符的步幅填充。

返回值

MCDNN_STATUS_SUCCESS

已成功返回结果。

MCDNN_STATUS_BAD_PARAM

tensorDesc 或 nbDims 指针为 NULL。

2.2.1.62 mcdnnGetTensorSizeInBytes()

该函数返回内存中关于给定描述符的张量的大小。它可以用来了解分配用于保存该张量的 GPU 内存量。

```
mcdnnStatus_t mcdnnGetTensorSizeInBytes(
    const mcdnnTensorDescriptor_t tensorDesc,
    size_t *size)
```

参数

tensorDesc

输入。已初始化的张量描述符的句柄。

size

输出。保存张量所需的 GPU 内存大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

已成功返回结果。

2.2.1.63 mcdnnGetTensorTransformDescriptor()

此函数用于返回存储在已初始化的张量转换描述符中的值。

```
mcdnnStatus_t mcdnnGetTensorTransformDescriptor(
    mcdnnTensorTransformDescriptor_t transformDesc,
    uint32_t nbDimsRequested,
    mcdnnTensorFormat_t *destFormat,
    int32_t padBeforeA[],
    int32_t padAfterA[],
    uint32_t foldA[],
    mcdnnFoldingDirection_t *direction);
```

参数

transformDesc

输入。已初始化的张量转换描述符。

nbDimsRequested

输入。需要的维数。更多信息，参见张量描述符。

destFormat

输出。将返回的转换格式。

padBeforeA[]

输出。用每个维度前添加的 padding 量填充的数组。此 padBeforeA[] 参数的维度等于 nbDimsRequested。

padAfterA[]

输出。用每个维度后添加的 padding 量填充的数组。此 padAfterA[] 参数的维度等于 nbDimsRequested。

foldA[]

输出。用每个空间维度的 folding 参数填充的数组。foldA[] 数组的维度为 nbDimsRequested-2。

direction

输出。选择折叠或展开的设置。更多信息，参见 mcdnnFoldingDirection_t。

返回值

MCDNN_STATUS_SUCCESS

已成功获取结果。

MCDNN_STATUS_BAD_PARAM

transformDesc 为 NULL，或 nbDimsRequested 小于 3 或大于 MCDNN_DIM_MAX。

2.2.1.64 mcdnnGetVersion()

此函数返回 mcDNN 库的版本号。它返回 mcdnn.h 头文件中定义的 MCDNN_VERSION。该函数可用于动态标识应用程序当前使用的 mcDNN 库。定义的 MCDNN_VERSION 可用于使用条件编译语句将同一应用程序链接到不同的 mcDNN 版本。

```
size_t mcdnnGetVersion()
```

2.2.1.65 mcdnnInitTransformDest()

此函数用于初始化并返回张量转换操作的目标张量描述符 destDesc。完成初始化所需的参数，请参见 mcdnnTensorDescriptor_t。

```
mcdnnStatus_t mcdnnInitTransformDest(
    const mcdnnTensorTransformDescriptor_t transformDesc,
    const mcdnnTensorDescriptor_t srcDesc,
    mcdnnTensorDescriptor_t destDesc,
    size_t *destSizeInBytes);
```

注解： 返回的张量描述符会被压缩。

参数

transformDesc

输入。已初始化的张量转换描述符的句柄。

srcDesc

输入。已初始化的张量描述符的句柄。

destDesc

输出。要初始化并返回的张量描述符的句柄。

destSizeInBytes

输出。保存新张量大小（以字节为单位）的指针。

返回值

MCDNN_STATUS_SUCCESS

张量描述符已初始化成功。

MCDNN_STATUS_BAD_PARAM

srcDesc 或 destDesc 为 NULL，或者张量描述符的 nbDims 不正确。更多信息，参见张量描述符。

MCDNN_STATUS_NOT_SUPPORTED

提供的配置不是 4D。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.66 mcdnnLRNCrossChannelForward()

该函数执行正向 LRN 层计算。

```
mcdnnStatus_t mcdnnLRNCrossChannelForward(
    mcdnnHandle_t handle,
    mcdnnLRNDescriptor_t normDesc,
    mcdnnLRNMode_t lrnMode,
    const void *alpha,
    const mcdnnTensorDescriptor_t xDesc,
    const void *x,
    const void *beta,
    const mcdnnTensorDescriptor_t yDesc,
    void *y)
```

注解： 支持的格式包括：positive-strided，NCHW 和 NHWC（适用于 4D x 和 y），以及 5D 中仅支持 NCDHW DHW-packed（适用于 x 和 y）。仅支持非重叠 4D 和 5D 张量。使用 NCHW 布局，性能更优。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

normDesc

输入。已初始化的 LRN 参数描述符的句柄。

lrnMode

输入。LRN 层操作模式。目前仅支持实现 MCDNN_LRN_CROSS_CHANNEL_DIM1。归一化在张量的 dimA[1] 维中执行。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将层输出值与目标张量中的先验值混合，如下所示：

```
dstValue = alpha[0]*resultValue + beta[0]*priorDstValue
```

xDesc, yDesc

输入。输入和输出张量的张量描述符对象。

x

输入。设备内存中的输入张量数据指针。

y

输出。设备内存中的输出张量数据指针。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- x 和 y 中任一张量指针为 NULL。
- 输入张量维数小于或等于 2。
- LRN 描述符参数超出其有效范围。
- 任一张量参数为 5D，但不是 NCDHW DHW-packed 格式。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。不受支持的配置示例如下：

- 任意输入张量数据类型都与输出张量数据类型不同。
- X 和 y 张量维度不匹配。
- 任意张量参数步幅均为负的。

2.2.1.67 mcdnnNormalizationForwardInference()

此函数在推理阶段执行正向归一化层计算。关于 per-channel 归一化层，可参见 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift。

```
mcdnnStatus_t
mcdnnNormalizationForwardInference(mcdnnHandle_t handle,
    mcdnnNormMode_t mode,
    mcdnnNormOps_t normOps,
    mcdnnNormAlgo_t algo,
    const void *alpha,
    const void *beta,
    const mcdnnTensorDescriptor_t xDesc,
    const void *x,
    const mcdnnTensorDescriptor_t normScaleBiasDesc,
    const void *normScale,
    const void *normBias,
    const mcdnnTensorDescriptor_t normMeanVarDesc,
    const void *estimatedMean,
    const void *estimatedVariance,
    const mcdnnTensorDescriptor_t zDesc,
    const void *z,
    mcdnnActivationDescriptor_t activationDesc,
    const mcdnnTensorDescriptor_t yDesc,
    void *y,
    double epsilon,
    int groupCnt);
```

仅支持 4D 和 5D 张量。此函数执行的输入转换定义如下：

```
y = beta*y + alpha *[normBias + (normScale * (x-estimatedMean)/
 ↪sqrt(epsilon + estimatedVariance)]
```

在训练，反向传播和推理过程中的 epsilon 值必须相同。有关训练阶段的信息，请参见 mcdnnNormalizationForwardTraining()。当所有 x 和 y 都使用 HW-packed 张量时，可以获得更高的性能。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式（per-channel 或 per-activation）。更多信息，参见 mcdnnNormMode_t。

normOps

输入。post-operative 模式。目前不支持 MCDNN_NORM_OPS_NORM_ACTIVATION 和 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION。

algo

输入。要执行的算法。更多信息，参见 mcdnnNormAlgo_t。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将层输出值与目标张量中的先验值混合，如下所示：

```
dstValue = alpha[0]*resultValue + beta[0]*priorDstValue
```

xDesc, yDesc

输入。已初始化的张量描述符的句柄。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存，用于层的 x 输入数据。

y

输出。数据指针，指向与张量描述符 yDesc 关联的 GPU 内存，用于归一化层的 y 输出。

zDesc, z

输入。在激活之前，设备内存中用于归一化运算结果残差相加（Residual Addition）的张量描述符和指针。zDesc 和 z 是可选的，仅当 normOps 为 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION 时使用，否则用户可以传入 NULL。使用时，z 的维度应与 x 和最终输出 y 的维度完全相同。更多信息，参见 mcdnnTensorDescriptor_t。由于仅支持 normOps 为 MCDNN_NORM_OPS_NORM，目前可以将其设置为 NULL。

normScaleBiasDesc, normScale, normBias

输入。设备内存中用于归一化 scale 和 bias 参数的张量描述符和指针。

normMeanVarDesc, estimatedMean, estimatedVariance

输入。均值和方差张量及其张量描述符。在 mcdnnNormalizationForwardTraining() 调用的训练阶段累积的 estimatedMean 和 estimatedVariance 输入，此处作为输入传入。

activationDesc

输入。激活操作的描述符。

当 normOps 输入设置为 MCDNN_NORM_OPS_NORM_ACTIVATION 或 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION 时，将使用此激活，否则用户可以传入 NULL。由于仅支持 normOps 为 MCDNN_NORM_OPS_NORM，目前可以将其设置为 NULL。

epsilon

输入。归一化公式中使用的 Epsilon 值。其值应等于或大于零。

groupCnt

输入。分组卷积数。当前仅支持 1。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

不支持选择的计算或数据类型，或者选择了未知的算法类型。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- alpha，beta，x，y，normScale，normBias，estimatedMean 和 estimatedInvVariance 中任一指针为 NULL。
- xDesc 或 yDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。
- per-channel 模式下，4D 张量中 normScaleBiasDesc 和 normMeanVarDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- epsilon 的值小于 0。
- xDesc 和 yDesc 的维度或数据类型不匹配。

MCDNN_STATUS_NOT_SUPPORTED

选择了浮点以外的计算或数据类型，或者选择了未知算法类型。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.68 mcdnnOpsInferVersionCheck()

此函数是一系列相应函数中的第一个函数，这些函数检查不同模块的 DLL 文件之间的库版本是否一致。

```
mcdnnStatus_t mcdnnOpsInferVersionCheck(void)
```

返回值

MCDNN_STATUS_SUCCESS

此 DLL 文件的版本与它所依赖的 mcDNN DLL 一致。

MCDNN_STATUS_VERSION_MISMATCH

此 DLL 文件的版本与它所依赖的 mcDNN DLL 的版本不匹配。

2.2.1.69 mcdnnOpTensor()

该函数实现了公式 C = op(alpha1[0] * A, alpha2[0] * B) + beta[0] * C，给定张量 A、B、C 以及缩放系数 alpha1、alpha2、beta。要使用的 op 由描述符 mcdnnOpTensorDescriptor_t 表示，即 opTensorDesc 的类型。当前支持的 op 由 mcdnnOpTensorOp_t 枚举列出。

```
mcdnnStatus_t mcdnnOpTensor(
    mcdnnHandle_t handle,
    const mcdnnOpTensorDescriptor_t opTensorDesc,
    const void *alpha1,
    const mcdnnTensorDescriptor_t aDesc,
    const void *A,
    const void *alpha2,
    const mcdnnTensorDescriptor_t bDesc,
    const void *B,
    const void *beta,
    const mcdnnTensorDescriptor_t cDesc,
    void *C)
```

输入和目标张量有以下限制：

- 输入张量 A 的每个维度必须与目标张量 C 的相应维度匹配，输入张量 B 的每个维度必须与目标张量 C 的相应维度匹配，或者必须等于 1。在后一种情况下，输入张量 B 各维度的值将与张量 C 中对应的值混合。
- 输入张量 A 和 B 以及目标张量 C 的数据类型必须满足下表

支持的数据类型

opTensorDesc 中的 opTensorComp Type	A	B	C (destination)
FLOAT	FLOAT	FLOAT	FLOAT
FLOAT	INT8	INT8	FLOAT
FLOAT	HALF	HALF	FLOAT
FLOAT	BFLOAT16	BFLOAT16	FLOAT
DOUBLE	DOUBLE	DOUBLE	DOUBLE
FLOAT	FLOAT	FLOAT	HALF
FLOAT	HALF	HALF	HALF
FLOAT	INT8	INT8	INT8
FLOAT	FLOAT	FLOAT	INT8
FLOAT	FLOAT	FLOAT	BFLOAT16
FLOAT	BFLOAT16	BFLOAT16	BFLOAT16

注解: 不支持 MCDNN_TENSOR_NCHW_VECT_C 作为输入张量格式。支持维度大于 5 的所有张量。此函数不支持超出这些维度的张量格式。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

opTensorDesc

输入。已初始化的 op 张量描述符的句柄。

alpha1, alpha2, beta

输入。指向缩放系数（主机内存中）的指针，用于将源数值与目标张量中的先验值混合，如下所示：

```
dstValue = alpha[0]*resultValue + beta[0]*priorDstValue
```

aDesc, bDesc, cDesc

输入。已初始化的张量描述符的句柄。

A, B

输入。指针，分别指向由 aDesc 和 bDesc 描述符描述的张量数据。

C

输入/输出。指向 cDesc 描述符描述的张量数据的指针。

返回值

MCDNN_STATUS_SUCCESS

此函数执行成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。不受支持的配置示例如下：

- bias 张量和输出张量的维度大于 5。

- opTensorCompType 未按上述方式设置。

MCDNN_STATUS_BAD_PARAM

无法识别目标张量 C 的数据类型，或者不符合上述对输入和目标张量的限制。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.70 mcdnnPoolingForward()

此函数计算输入值池化（即多个相邻值的最大值或平均值），以生成高度和/或宽度较小的输出。

```
mcdnnStatus_t mcdnnPoolingForward(
    mcdnnHandle_t                    handle,
    const mcdnnPoolingDescriptor_t   poolingDesc,
    const void                      *alpha,
    const mcdnnTensorDescriptor_t    xDesc,
    const void                      *x,
    const void                      *beta,
    const mcdnnTensorDescriptor_t    yDesc,
    void                            *y)
```

支持所有张量格式，使用 HW-packed 张量时性能最佳。只支持 2 个和 3 个空间维度。仅支持具有 2 个空间维度的向量化张量。输出张量 yDesc 的维度可以小于或大于 mcdnnGetPooling2dForwardOutputDim() 或 mcdnnGetPoolingNdForwardOutputDim() 函数建议的维度。对于平均池化，即使是整数输入和输出数据类型，计算类型也是浮点型。如果超出范围，则输出向最近偶数舍入，并保持在负数最小值和正数最大值之间。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

poolingDesc

输入。已初始化的池化描述符的句柄。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将计算结果与输出层中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

xDesc

输入。已初始化的输入张量描述符的句柄。类型必须为 FLOAT，DOUBLE，HALF，INT8，INT8x4，INT8x32 或 BFLOAT16。更多信息，参见 mcdnnDataType_t。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存。

yDesc

输入。已初始化的输出张量描述符的句柄。类型必须为 FLOAT，DOUBLE，HALF，INT8，INT8x4，INT8x32 或 BFLOAT16。更多信息，参见 mcdnnDataType_t。

y

输出。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 输入张量和输出张量的 n，c 维度不同。
- 输入张量和输出张量的数据类型不同。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.71 mcdnnQueryRuntimeError()

mcDNN 库函数在启动 GPU 内核之前执行广泛的输入参数检查。最后一步是验证 GPU 内核是否实际启动。当内核启动失败时，相应的 API 调用将返回 MCDNN_STATUS_EXECTION_FAILED。通常，在 GPU 内核启动后，内核本身不会执行运行时检查-数值结果仅会写入输出缓冲区。

```
mcdnnStatus_t mcdnnQueryRuntimeError(
    mcdnnHandle_t handle,
    mcdnnStatus_t *rstatus,
    mcdnnErrQueryMode_t mode,
    mcdnnRuntimeTag_t *tag)
```

当在 mcdnnBatchNormalizationForwardTraining() 或 mcdnnBatchNormalizationBackward() 中选择 MCDNN_BATCHNORM_SPATICATE_PERSISTENT 模式时，算法可能会遇到数值溢出，其中 MCDNN_BATCHNORM_SPATICAL 执行得很好，尽管速度较慢。用户可以调用 mcdnnQueryRuntimeError()，以确保内核执行期间不会发生数值溢出。这些问题由执行计算的内核报告。mcdnnQueryRuntimeError() 可以用于轮询和阻塞软件控制流。有两种轮询模式（MCDNN_ERRQUERY_RAWCODE 和 MCDNN_ERRQUERY_NONBLOCKED）和一种阻塞模式 MCDNN_ERRQUERY_BLOCKED。无论内核完成状态如何，MCDNN_ERRQUERY_RAWCODE 都读取错误存储位置。内核甚至可能无法启动，并且错误存储（按 mcDNN 句柄分配）可能由先前的调用使用。

MCDNN_ERRQUERY_NONBLOCKED 检查用户流中的所有任务是否已完成。如果用户流中的某些任务处于待处理状态，mcdnnQueryRuntimeError() 函数将立即返回并在 rstatus 中报告 MCDNN_STATUS_RUNTIME_IN_PROGRESS。否则，函数会将远程内核错误代码复制到 rstatus。在阻塞模式（MCDNN_ERRQUERY_BLOCKEG）中，该函数等待用户流中的所有任务结束，再报告远程内核错误代码。可以通过调用带有 macaDeviceScheduleSpin，macaDeviceScheduleYield 或 macaDeviceScheduleBlockingSync 标记的 macaSetDeviceFlags 来进一步调整阻塞样式。

在 mcDNN 句柄中更改用户流时，不应使用 MCDNN_ERRQUERY_NONBLOCKING 和 MCDNN_ERRQUERY_BLOCKING 模式。即，在报告运行时内核错误的函数和 mcdnnQueryRuntimeError() 函数之间调用 mcdnnSetStream()。

rstatus 中报告的远程错误状态可以设置为：MCDNN_STATUS_SUCCESS，MCDNN_STATUS_RUNTIME_IN_PROGRESS 或 MCDNN_STATUS_RUNTIME_FP_OVERFLOW。mcdnnQueryRuntimeError() 会自动清除远程内核错误。

注解： 当 mcdnnBatchNormalMode_t 参数为 MCDNN_BATCHNORM_SPATIATION_PERSISTENT 时，mcdnnQueryRuntimeError() 函数应与 mcdnnBatchNormalizationForwardTraining() 和 mcdnnBatchNormalizationBackward() 一起使用。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rstatus

输出。指向用户错误代码存储的指针。

mode

输入。远程错误查询模式。

tag

输入/输出。当前，此参数应为 NULL。

返回值

MCDNN_STATUS_SUCCESS

未检测到错误（rstatus 保留一个有效值）。

MCDNN_STATUS_BAD_PARAM

输入参数无效。

MCDNN_STATUS_INTERNAL_ERROR

阻塞同步流或非阻塞流查询失败。

MCDNN_STATUS_MAPPING_ERROR

设备无法访问零拷贝内存来报告内核错误。

2.2.1.72 mcdnnReduceTensor()

该函数通过实现公式 C = alpha * reduce op (A)+ beta * C 来归约张量 A，给定张量 A 和 C 以及缩放系数 alpha 和 beta。要使用的归约 op 由 reduceTensorDesc 描述符表示。当前支持的 op 由 mcdnnReduceTensorOp_t 枚举列出。

```
mcdnnStatus_t mcdnnReduceTensor(
    mcdnnHandle_t handle,
    const mcdnnReduceTensorDescriptor_t reduceTensorDesc,
    void *indices,
    size_t indicesSizeInBytes,
    void *workspace,
    size_t workspaceSizeInBytes,
    const void *alpha,
    const mcdnnTensorDescriptor_t aDesc,
    const void *A,
    const void *beta,
    const mcdnnTensorDescriptor_t cDesc,
    void *C)
```

输出张量 C 的每个维度必须与输入张量 A 的相应维度匹配，或者必须等于 1。等于 1 的维度表示张量 A 要归约的维度。

此实现仅为最小和最大 op 生成索引，如 reduceTensorDesc 的 mcdnnReduceTensorIndices_t 枚举所示。请求其他归约 op 的索引会导致错误。索引的数据类型由 mcdnnIndicesType_t 枚举表示；当前仅支持 32 位（无符号 int）类型。

返回的索引不是绝对索引，而是相对于归约维度的索引。索引也是平展的，即不是坐标元组。

如果类型为 double，则张量 A 和 C 的数据类型必须匹配。在这种情况下，alpha 和 beta 以及 reduceTensorDesc 的计算枚举都假定为 double 类型。

HALF 和 INT8 数据类型可以与 FLOAT 数据类型混合。在这些情况下，reduceTensorDesc 的计算枚举必须为 FLOAT 数据类型。

注解： 维度不超过 8 时，支持所有张量格式。如果超过这些维度，则不支持此函数。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

reduceTensorDesc

输入。已初始化的归约张量描述符的句柄。

indices

输出。已分配用于写入索引的空间的句柄。

indicesSizeInBytes

输入。上述已分配空间的大小。

workspace

输入。已分配用于实现归约的空间的句柄。

workspaceSizeInBytes

输入。上述已分配空间的大小。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将源数值与目标张量中的先验值混合，如下所示：

```
dstValue = alpha[0]*resultValue + beta[0]*priorDstValue
```

aDesc, cDesc

输入。已初始化的张量描述符的句柄。

A

输入。指针，指向 aDesc 描述符描述的张量数据。

C

输入/输出。指针，指向 cDesc 描述符描述的张量数据。

返回值

MCDNN_STATUS_SUCCESS

此函数执行成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。不受支持的配置示例如下：

- 输入张量和输出张量的维度大于 8。
- reduceTensorCompType 未按上述方式设置。

MCDNN_STATUS_BAD_PARAM

输入和输出张量的相应维度均匹配，或者未满足上述中的条件。

MCDNN_INVALID_VALUE

索引或工作空间的分配不足。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.73 mcdnnRestoreAlgorithm()

此函数从 algoSpace 中用户提供的主机内存空间读取算法元数据，允许用户使用从之前的 mcDNN session 中查找的 RNN 结果。

```
mcdnnStatus_t mcdnnRestoreAlgorithm(
    mcdnnHandle_t handle,
    void* algoSpace,
    size_t algoSpaceSizeInBytes,
    mcdnnAlgorithmDescriptor_t algoDesc)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

algoDesc

输入。已创建的算法描述符。

algoSpace

输入。指向要读取的主机内存的指针。

algoSpaceSizeInBytes

输入。作为工作空间所需的主机内存量，以便能够从指定的 algoDesc 中保存元数据。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

元数据来自不同的 mcDNN 版本。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 任一参数为 NULL。
- 元数据已损坏。

2.2.1.74 mcdnnRestoreDropoutDescriptor()

```
mcdnnStatus_t mcdnnRestoreDropoutDescriptor(
    mcdnnDropoutDescriptor_t dropoutDesc,
    mcdnnHandle_t handle,
    float dropout,
    void *states,
    size_t stateSizeInBytes,
    unsigned long long seed)
```

此函数将丢弃描述符还原到之前的已保存关闭状态。

参数

dropoutDesc

输入/输出。已创建的丢弃描述符。

handle

输入。已创建的 mcDNN 上下文的句柄。

dropout

输入。执行丢弃时，输入张量值设置为 0 的概率。

states

输入。指向 GPU 内存的指针，该内存保存由先前的 mcdnnSetDropoutDescriptor() 调用初始化的随机数生成器状态。

stateSizeInBytes

输入。保存随机数生成器状态的缓冲区大小（以字节为单位）。

seed

输入。在先前的 mcdnnSetDropoutDescriptor() 调用中，用于初始化状态缓冲区的种子。使用与此不同的种子不会产生任何影响。通过调用 mcdnnSetDropoutDescriptor()，可以更改种子并随后更新随机数生成器状态。

返回值

MCDNN_STATUS_SUCCESS

调用成功。

MCDNN_STATUS_INVALID_VALUE

状态缓冲区（如 stateSizeInBytes 中所示）太小。

2.2.1.75 mcdnnSaveAlgorithm()

此函数从 algoSpace 中用户提供的主机内存空间写入算法元数据，允许用户在退出 mcDNN 后保留查找的 RNN 结果。

```
mcdnnStatus_t mcdnnSaveAlgorithm(
    mcdnnHandle_t handle,
    mcdnnAlgorithmDescriptor_t algoDesc,
    void* algoSpace
    size_t algoSpaceSizeInBytes)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

algoDesc

输入。已创建的算法描述符。

algoSpace

输入。指向要写入的主机内存的指针。

algoSpaceSizeInBytes

输入。作为工作空间所需的主机内存量，以便能够从指定的 algoDesc 中保存元数据。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 任一参数为 NULL。
- algoSpaceSizeInBytes 太小。

2.2.1.76 mcdnnScaleTensor()

```
mcdnnStatus_t mcdnnScaleTensor(
mcdnnHandle_t handle,
const mcdnnTensorDescriptor_t yDesc,
void *y,
const void *alpha)
```

该函数按给定系数缩放张量的所有元素。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

yDesc

输入。已初始化的张量描述符的句柄。

y

输入/输出。指针，指向 yDesc 描述符描述的张量数据。

alpha

输入。主机内存中指向单个值的指针，张量的所有元素都将按该值进行缩放。。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

任一已提供的指针为 nil。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.77 mcdnnSetActivationDescriptor()

```
mcdnnStatus_t mcdnnSetActivationDescriptor(
    mcdnnActivationDescriptor_t         activationDesc,
    mcdnnActivationMode_t               mode,
    mcdnnNanPropagation_t               reluNanOpt,
    double                              coef)
```

此函数用于初始化已创建的通用激活描述符对象。

参数

activationDesc

输入/输出。已创建的激活描述符的句柄。

mode

输入。指定激活模式的枚举。

reluNanOpt

输入。指定 NaN 传播模式的枚举。

coef

输入。浮点数。当激活模式（请参见 mcdnnActivationMode_t）设置为 MCDNN_ACTIVATION_CHIPPED_RELU 时，此输入指定裁剪阈值；当激活模式设置为 MCDNN_ACTIVATION_RELU 时，此输入指定上限。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

mode 或 reluNanOpt 具有无效的枚举值。

2.2.1.78 mcdnnSetActivationDescriptorSwishBeta()

```
mcdnnStatus_t mcdnnSetActivationDescriptorSwishBeta(
    mcdnnActivationDescriptor_t activationDesc,
    double swish_beta)
```

此函数用于将 Swish 激活函数的 beta 参数设置为 swish_beta。

参数

activationDesc

输入/输出。已创建的激活描述符的句柄。

swish_beta

输入。设置为 Swish 激活函数 beta 参数的值。

返回值

MCDNN_STATUS_SUCCESS

值设置成功。

MCDNN_STATUS_BAD_PARAM

此激活描述符是一个 NULL 指针。

2.2.1.79 mcdnnSetAlgorithmDescriptor()

此函数用于初始化已创建的通用算法描述符对象。

```
mcdnnStatus_t mcdnnSetAlgorithmDescriptor(
    mcdnnAlgorithmDescriptor_t algorithmDesc,
    mcdnnAlgorithm_t algorithm)
```

参数

algorithmDesc

输入/输出。已创建的算法描述符的句柄。

algorithm

输入。指定算法的结构。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

2.2.1.80 mcdnnSetAlgorithmPerformance()

此函数用于初始化已创建的通用算法性能对象。

```
mcdnnStatus_t mcdnnSetAlgorithmPerformance(
    mcdnnAlgorithmPerformance_t algoPerf,
    mcdnnAlgorithmDescriptor_t algoDesc,
    mcdnnStatus_t status,
    float time,
    size_t memory)
```

参数

algoPerf

输入/输出。已创建的算法性能对象的句柄。

algoDesc

输入。性能结果描述的算法描述符。

status

输入。运行 algoDesc 算法返回的 mcDNN 状态。

time

输入。GPU 运行 algoDesc 算法消耗的时间。

memory

输入。运行 algoDesc 算法所需的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

mode 或 reluNanOpt 具有无效的枚举值。

2.2.1.81 mcdnnSetCallback()

此函数用于设置 mcDNN 错误报告功能的内部状态。

```
mcdnnStatus_t mcdnnSetCallback(
    unsigned mask,
    void *udata,
    mcdnnCallback_t fptr)
```

参数

mask

输入。无符号整数。此无符号整数的四个 LSB 用于打开和关闭不同级别的错误报告消息。这适用于默认回调和自定义回调。该比特位与 mcdnnSeverity_t 枚举对应。用户可以使用预定义的 MCDNN_SEV_ERROR_EN，MCDNN_SEV_WARNING_EN 和 MCDNN_SEV_INFO_EN 宏来组成位掩码。当一个 bit 设置为 1 时，将启用相应的消息通道。

例如，当 bit 3 设置为 1 时，将启用 API 日志记录。目前，仅支持 MCDNN_SEV_INFO 级别的日志输出；其他输出尚未实现。当使用默认回调打开和关闭日志记录时，用户可以将 NULL 传入 udata 和 fptr。此外，必须设置环境变量 MCDNN_LOGDEST_DBG。。

- MCDNN_SEV_INFO_EN= 0b1000（功能已实现）。

- MCDNN_SEV_ERROR_EN= 0b0010（功能暂未实现）。
- MCDNN_SEV_WARNING_EN= 0b0100（功能暂未实现）。

MCDNN_SEV_FATAL 的输出始终处于启用状态且不能禁用。

udata

输入。用户提供的指针。此指针将传入用户的自定义日志记录回调函数。它所指向的数据不会被读取，mcDNN 也不会更改。此指针可以有多种用途，例如在互斥锁（mutex）或通信套接字（communication socket）中用于用户的日志记录回调函数。如果用户正在使用默认的回调函数，或者不想在自定义的回调函数中使用此输入，可以传入 NULL。

fptr

输入。指向用户提供的回调函数的指针。将 NULL 传入该指针时，mcDNN 切换回内置的缺省回调函数。用户提供的回调函数原型必须类似于以下内容（也在头文件中定义）：

```
void customizedLoggingCallback (mcdnnSeverity_t sev, void *udata, const
mcdnnDebug_t *dbg, const char *msg);
```

- 在头文件中定义了 mcdnnDebug_t 结构。它提供了用户可以选择打印或存储在自定义回调中的元数据，如时间，自启动以来的时间，流 ID，进程和线程 ID。
- 变量 msg 是 mcDNN 生成的日志消息。此消息的每一行都由 0 终止，消息的结尾由 00 终止。用户可以选择需要在日志中显示的内容，也可以重新格式化字符串。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

2.2.1.82 mcdnnSetDropoutDescriptor()

```
mcdnnStatus_t mcdnnSetDropoutDescriptor(mcdnnDropoutDescriptor_t
↪dropoutDesc,
    mcdnnHandle_t handle,
    float dropout,
    void *states,
    size_t stateSizeInBytes,
    unsigned long long seed)
```

此函数用于初始化已创建的丢弃描述符对象。如果 states 参数等于 NULL，则不会初始化随机数生成器状态，并且只会设置 dropout 值。在计算期间，用户不应更改 states 所指向的内存。

当 states 参数不为 NULL 时，mcdnnSetDropoutDescriptor() 调用一个随机初始化内核。此内核需要大量 GPU 内存用于堆栈。内核完成时释放内存。当没有足够的内存可用于 GPU 堆栈时，将返回 MCDNN_STATUS_ALLOC_FAILED 状态。

参数

dropoutDesc

输入/输出。已创建的丢弃描述符对象。

handle

输入。已创建的 mcDNN 上下文的句柄。

dropout

输入。在丢弃层输入值设置为 0 的概率。

states

输出。指向用户分配的 GPU 内存的指针，该内存将保存随机数生成器状态。

stateSizeInBytes

输入。指定为 states 所提供内存的大小（以字节为单位）。

seed

输入。用于初始化随机数生成器状态的种子（Seed）。

返回值

MCDNN_STATUS_SUCCESS

调用成功。

MCDNN_STATUS_INVALID_VALUE

sizeInBytes 参数小于 mcdnnDropoutGetStatesSize() 返回的值。

MCDNN_STATUS_ALLOC_FAILED

此函数临时扩展 GPU 堆栈失败。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_INTERNAL_ERROR

内部使用的 MXMACA 函数返回错误状态。

2.2.1.83 mcdnnSetFilter4dDescriptor()

```
mcdnnStatus_t mcdnnSetFilter4dDescriptor(
mcdnnFilterDescriptor_t filterDesc,
mcdnnDataType_t dataType,
mcdnnTensorFormat_t format,
int k,
int c,
int h,
int w)
```

此函数将已创建的卷积核描述符对象初始化为 4D 卷积核。卷积核的布局在内存中必须是连续的。张量格式 MCDNN_TENSOR_NHWC 在 mcdnnConvolutionForward()，mcdnnConvolutionBackwardData() 和 mcdnnConvolutionBackwardFilter() 中的支持有限。

参数

filterDesc

输入/输出。已创建的卷积核描述符的句柄。

datatype

输入。数据类型。

format

输入。卷积核布局格式的类型。如果将此输入设置为 MCDNN_TENSOR_NCHW（mcdnnTensorFormat_t 描述符支持的枚举值之一），则卷积核的布局采用 KCRS 的形式，其中：

- K 是输出特征图的数量
- C 是输入特征图的数量
- R 是每个卷积核的行数
- S 是每个卷积核的列数

如果此输入设置为 MCDNN_TENSOR_NHWC，则卷积核的布局采用 KRSC 的形式。更多信息，参见 mcdnnTensorFormat_t。

k

输入。输出特征图的数量。

c

输入。输入特征图的数量。

h

输入。每个卷积核的高度。

w

输入。每个卷积核的宽度。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

k，c，h，w 参数中至少有一个为负值，或者 dataType 或 format 具有无效的枚举值。

2.2.1.84 mcdnnSetFilterNdDescriptor()

```
mcdnnStatus_t mcdnnSetFilterNdDescriptor(
mcdnnFilterDescriptor_t filterDesc,
mcdnnDataType_t dataType,
mcdnnTensorFormat_t format,
int nbDims,
const int filterDimA[])
```

此函数用于初始化已创建的卷积核描述符对象。卷积核的布局在内存中必须是连续的。张量格式 MCDNN_TENSOR_NHWC 在 mcdnnConvolutionForward()，mcdnnConvolutionBackwardData() 和 mcdnnConvolutionBackwardFilter() 中的支持有限。

参数

filterDesc

输入/输出。已创建的卷积核描述符的句柄。

datatype

输入。数据类型。

format

输入。卷积核布局格式的类型。如果将此输入设置为 MCDNN_TENSOR_NCHW（mcdnnTensorFormat_t 描述符支持的枚举值之一），则卷积核的布局如下：

- N=4 时，即 4D 卷积核描述符，卷积核布局采用 KCRS 的形式：
  - K 是输出特征图的数量
  - C 是输入特征图的数量
  - R 是每个卷积核的行数
  - S 是每个卷积核的列数
- N=3 时，即 3D 卷积核描述符，将省略数字 S（每个卷积核的列数）。
- N=5 及更大时，更高维度的布局立即遵循 RS。

另外，如果此输入设置为 MCDNN_TENSOR_NHWC，则卷积核的布局如下：

- N=4 时，即 4D 卷积核描述符，卷积核布局采用 KRSC 的形式。

- N=3 时，即 3D 卷积核描述符，将省略数字 S（每个卷积核的列数），且 C 的布局立即遵循 R。
- N=5 及更大时，更高维度的布局插入在 S 和 C 之间。

更多信息，参见 mcdnnTensorFormat_t。

nbDims

输入。卷积核的维度。

filterDimA

输入。nbDims 维度的数组，包含每个维度的卷积核大小。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

filterDimA 数组中至少有一个元素为负值，或者 dataType 或 format 具有无效的枚举值。

MCDNN_STATUS_NOT_SUPPORTED

nbDims 参数大于 MCDNN_DIM_MAX。

2.2.1.85 mcdnnSetLRNDescriptor()

此函数用于初始化已创建的 LRN 描述符对象。

```
mcdnnStatus_t mcdnnSetLRNDescriptor(
    mcdnnLRNDescriptor_t normDesc,
    unsigned lrnN,
    double lrnAlpha,
    double lrnBeta,
    double lrnK)
```

注解:

- mcdnn.h 中定义的 MCDNN_LRN_MIN_N，MCDNN_LRN_MAX_N，MCDNN_LRN_MIN_K，MCDNN_LRN_MIN_BETA 宏指定参数的有效范围。
- 在计算过程中，双精度参数的值将向下转换（cast down）为张量数据类型。

参数

normDesc

输出。已创建的 LRN 描述符的句柄。

lrnN

输入。元素中的归一化窗口宽度。LRN 层使用一个 [centerlookBehind, center+Lookahead] 窗口，其中 lookBehind = floor((lrnN-1)/2)，lookAhead = lrnN-lookBehind-1。因此，对于 n=10，窗口为 [k-4…k..k+5]，共有 10 个样本。对于 DivisiveNormalization 层，窗口在所有空间维度 (dimA[2]，dimA[3]，dimA[4]) 中具有与上述相同的范围。默认情况下,lrnN 在 mcdnnCreateLRNDescriptor() 中设置为 5。

lrnAlpha

输入。归一化公式中 alpha 方差缩放参数的值。在库代码中，此值在 LRN 层除以窗宽，在 DivisiveNormalization 层中除以 (窗宽)^#spatialDimensions。默认情况下，此值在 mcdnnCreateLRNDescriptor() 中设置为 1e-4。

lrnBeta

输入。归一化公式中 beta power 参数的值。默认情况下，此值在 mcdnnCreateLRNDescriptor() 中设置为 0.75。

lrnK

输入。归一化公式中 k 参数的值。默认情况下，此值设置为 2.0。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

其中一个输入参数超出上述有效范围。

2.2.1.86 mcdnnSetOpTensorDescriptor()

```
mcdnnStatus_t mcdnnSetOpTensorDescriptor(
 mcdnnOpTensorDescriptor_t opTensorDesc,
 mcdnnOpTensorOp_t opTensorOp,
 mcdnnDataType_t opTensorCompType,
 mcdnnNanPropagation_t opTensorNanOpt)
```

此函数用于初始化张量逐点数学运算描述符。

参数

opTensorDesc

输出。指针，指向包含张量逐点数学运算描述符的结构。

opTensorOp

输入。此张量逐点数学运算描述符的张量逐点数学运算。

opTensorCompType

输入。此张量逐点数学运算描述符的计算数据类型。

opTensorNanOpt

输入。NaN 传播策略。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

MCDNN_STATUS_BAD_PARAM

传入的输入参数至少有一个是无效的。

2.2.1.87 mcdnnSetPooling2dDescriptor()

```
mcdnnStatus_t mcdnnSetPooling2dDescriptor(
    mcdnnPoolingDescriptor_t    poolingDesc,
    mcdnnPoolingMode_t          mode,
    mcdnnNanPropagation_t       maxpoolingNanOpt,
    int                         windowHeight,
    int                         windowWidth,
```

(下页继续)

(续上页)

```
    int                             verticalPadding,
    int                             horizontalPadding,
    int                             verticalStride,
    int                             horizontalStride)
```

此函数将已创建的通用池化描述符对象初始化为 2D 描述符。

参数

poolingDesc

输入/输出。已创建的池化描述符的句柄。

mode

输入。指定池化模式的枚举。

maxpoolingNanOpt

输入。指定 NaN 传播模式的枚举。

windowHeight

输入。池化窗口的高度。

windowWidth

输入。池化窗口的宽度。

verticalPadding

输入。垂直填充的大小。

horizontalPadding

输入。水平填充的大小。

verticalStride

输入。池化垂直步幅。

horizontalStride

输入。池化水平步幅。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

windowHeight、windowWidth、verticalStride、horizontalStride 参数中至少有一个为负值，或者 mode 或 maxpoolingNanOpt 具有无效的枚举值。

2.2.1.88 mcdnnSetPoolingNdDescriptor()

```
mcdnnStatus_t mcdnnSetPoolingNdDescriptor(
    mcdnnPoolingDescriptor_t     poolingDesc,
    const mcdnnPoolingMode_t     mode,
    const mcdnnNanPropagation_t  maxpoolingNanOpt,
    int                          nbDims,
    const int                    windowDimA[],
    const int                    paddingA[],
    const int                    strideA[])
```

此函数用于初始化已创建的通用池化描述符对象。

参数

poolingDesc

输入/输出。已创建的池化描述符的句柄。

mode

输入。指定池化模式的枚举。

maxpoolingNanOpt

输入。指定 NaN 传播模式的枚举。

nbDims

输入。池化操作的维度。必须大于零。

windowDimA

输入。nbDims 维度的数组，包含每个维度的窗口大小。数组元素的值必须大于零。

paddingA

输入。nbDims 维度的数组，包含每个维度的填充大小。允许负填充。

strideA

输入。nbDims 维度的数组，包含每个维度的步幅大小。数组元素的值必须大于零（即不允许负的步幅）。

返回值

MCDNN_STATUS_SUCCESS

对象初始化成功。

MCDNN_STATUS_NOT_SUPPORTED

如果 (nbDims > MCDNN_DIM_MAX-2)。

MCDNN_STATUS_BAD_PARAM

nbDims，或者 windowDimA 或 strideA 数组中至少有一个元素为负值。或者 mode 或 maxpoolingNanOpt 具有无效的枚举值。

2.2.1.89 mcdnnSetReduceTensorDescriptor()

```
mcdnnStatus_t mcdnnSetReduceTensorDescriptor(
 mcdnnReduceTensorDescriptor_t reduceTensorDesc,
 mcdnnReduceTensorOp_t reduceTensorOp,
 mcdnnDataType_t reduceTensorCompType,
 mcdnnNanPropagation_t reduceTensorNanOpt,
 mcdnnReduceTensorIndices_t reduceTensorIndices,
 mcdnnIndicesType_t reduceTensorIndicesType)
```

此函数用于初始化已创建的归约张量描述符对象。

参数

reduceTensorDesc

输入/输出。已创建的归约张量描述符的句柄。

reduceTensorOp

输入。指定归约张量操作的枚举。

reduceTensorCompType

输入。指定归约计算数据类型的枚举。

reduceTensorNanOpt

输入。指定 NaN 传播模式的枚举。

reduceTensorIndices

输入。指定归约张量索引的枚举。

reduceTensorIndicesType

输入。指定归约张量索引类型的枚举。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

reduceTensorDesc 为 NULL（reduceTensorOp，reduceTensorCompType，reduceTensorNanOpt，reduceTensorIndices 或 reduceTensorIndicesType 具有无效的枚举值）。

2.2.1.90 mcdnnSetSpatialTransformerNdDescriptor()

此函数用于初始化已创建的通用空间转换描述符对象。

```
mcdnnStatus_t mcdnnSetSpatialTransformerNdDescriptor(
    mcdnnSpatialTransformerDescriptor_t stDesc,
    mcdnnSamplerType_t samplerType,
    mcdnnDataType_t dataType,
    const int nbDims,
    const int dimA[])
```

参数

stDesc

输入/输出。已创建的空间转换描述符对象。

samplerType

输入。指定采样器类型的枚举。

dataType

输入。数据类型。

nbDims

输入。转换张量的维度。

dimA

输入。nbDims 维度的数组，包含每个维度的转换张量大小。

返回值

MCDNN_STATUS_SUCCESS

调用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- stDesc 或 dimA 为 NULL。
- dataType 或 samplerType 具有无效的枚举值。

2.2.1.91 mcdnnSetStream()

此函数在 mcDNN 句柄中设置用户 MXMACA 流。新的流将用于启动 mcDNN GPU 内核或在内部流中启动 mcDNN 内核时同步到此流。如果未设置 mcDNN 库流，则所有内核都使用默认的（NULL）流。在 mcDNN 句柄中设置用户流可确保在同一流中启动的 mcDNN 调用和其他 GPU 计算核的执行顺序。

```
mcdnnStatus_t mcdnnSetStream(
    mcdnnHandle_t handle,
    mcStream_t streamId)
```

内部流与上次调用此函数时设置的流具有相同的优先级。

参数

handle

输入。指向 mcDNN 句柄的指针。

streamID

输入。要写入 mcDNN 句柄的新 MXMACA 流。

返回值

MCDNN_STATUS_BAD_PARAM

无效（NULL）句柄。

MCDNN_STATUS_MAPPING_ERROR

用户流和 mcDNN 句柄上下文之间不匹配。

MCDNN_STATUS_SUCCESS

新流设置成功。

2.2.1.92 mcdnnSetTensor()

```
mcdnnStatus_t mcdnnSetTensor(
 mcdnnHandle_t handle,
 const mcdnnTensorDescriptor_t yDesc,
 void *y,
 const void *valuePtr)
```

该函数将张量所有元素设置为一个给定值。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

yDesc

输入。已初始化的张量描述符的句柄。

y

输入/输出。指针，指向 yDesc 描述符描述的张量数据。

valuePtr

输入。主机内存中指向单个值的指针。y 张量的所有元素都将设置为 value[0]。value[0] 中元素的数据类型必须与 y 张量的数据类型匹配。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

任一已提供的指针为 nil。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.93 mcdnnSetTensor4dDescriptor()

```
mcdnnStatus_t mcdnnSetTensor4dDescriptor(
                mcdnnTensorDescriptor_t tensorDesc,
                mcdnnTensorFormat_t format,
                mcdnnDataType_t dataType,
                int n,
                int c,
                int h,
                int w)
```

此函数将已创建的通用张量描述符对象初始化为 4D 张量。从 format 参数推断出四个维度的步幅，并以这样的方式设置：数据在内存中是连续的，在维度之间没有填充。

参数

tensorDesc

输入/输出。已创建的张量描述符的句柄。

format

输入。格式类型。

datatype

输入。数据类型。

n

输入。图像数量。

c

输入。每个图像的特征图数量。

h

输入。每个特征图的高度。

w

输入。每个特征图的宽度。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

n，c，h，w 参数中至少有一个为负值，或者 format 或 dataType 具有无效的枚举值。

MCDNN_STATUS_NOT_SUPPORTED

张量描述符的总大小超过最大限制 2 Giga-elements。

2.2.1.94 mcdnnSetTensor4dDescriptorEx()

该函数将已创建的通用张量描述符对象初始化为 4D 张量，类似于 mcdnnSetTensor4dDescriptor()，但将步幅作为参数显式传递。这可用于以任何顺序排列 4D 张量，或仅定义维度之间的间隙。

```
mcdnnStatus_t mcdnnSetTensor4dDescriptorEx(
mcdnnTensorDescriptor_t tensorDesc,
mcdnnDataType_t dataType,
int n,
int c,
int h,
int w,
int nStride,
int cStride,
int hStride,
int wStride)
```

目前，某些 mcDNN 函数对步幅的支持有限。如果使用了步幅不受支持的 4D 张量对象，则这些函数将返回 MCDNN_STATUS_NOT_SUPPORTED。mcdnnTransformTensor() 可用于将数据转换为支持的布局。一个张量的总大小（包括维度之间的潜在填充）限制为 2 Giga-elements 的 dataType。

参数

tensorDesc

输入/输出。已创建的张量描述符的句柄。

datatype

输入。数据类型。

n

输入。图像数量。

c

输入。每个图像的特征图数量。

h

输入。每个特征图的高度。

w

输入。每个特征图的宽度。

nStride

输入。两个连续图像之间的步幅。

cStride

输入。两个连续特征图之间的步幅。

hStride

输入。两个连续行之间的步幅。

wStride

输入。两个连续列之间的步幅。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

n，c，h，w 参数中或 nStride，cStride，hStride，wStride 中至少有一个为负值，或者 dataType 具有无效的枚举值。

MCDNN_STATUS_NOT_SUPPORTED

张量描述符的总大小超过最大限制 2 Giga-elements。

2.2.1.95 mcdnnSetTensorNdDescriptor()

此函数用于初始化已创建的通用张量描述符对象。

```
mcdnnStatus_t mcdnnSetTensorNdDescriptor(
mcdnnTensorDescriptor_t tensorDesc,
mcdnnDataType_t dataType,
int nbDims,
const int dimA[],
const int strideA[])
```

一个张量的总大小（包括维度之间的潜在填充）限制为该数据类型的元素个数问 2G。张量必须至少有 4 个维度，至多有 MCDNN_DIM_MAX 个维度（在 mcdnn.h 中定义）。使用低维数据时，建议用户创建 4D 张量，并将未使用维度的大小设置为 1。

参数

tensorDesc

输入/输出。已创建的张量描述符的句柄。

datatype

输入。数据类型。

nbDims

输入。张量的维度。

注解: 请勿使用 2 维。更多信息，参见 mcdnnGetRNNLinLayerBiasParams()。

---

dimA

输入。nbDims 维度的数组，包含每个维度的张量大小。未使用维度的大小应设置为 1。根据惯例，数组中的维度顺序遵循 [N, C, D, H, W] 格式，其中 W 占数组中最小的索引。

strideA

输入。nbDims 维度的数组，包含每个维度的张量步幅。根据惯例，数组中的步幅顺序遵循 [Nstride, Cstride, Dstride, Hstride, Wstride] 格式，其中 Wstride 占数组中最小的索引。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

dimA 数组中至少有一个元素为负值或 0，或者 dataType 具有无效的枚举值。

MCDNN_STATUS_NOT_SUPPORTED

参数 nbDims 超出 [4, MCDNN_DIM_MAX] 范围，或张量描述符的总大小超过最大限制 2 Giga-elements。

2.2.1.96 mcdnnSetTensorNdDescriptorEx()

```
mcdnnStatus_t mcdnnSetTensorNdDescriptorEx(
mcdnnTensorDescriptor_t tensorDesc,
mcdnnTensorFormat_t format,
mcdnnDataType_t dataType,
int nbDims,
const int dimA[])
```

此函数用于初始化 n-D 张量描述符。

参数

tensorDesc

输出。指针，指向要初始化的张量描述符结构。

format

输入。张量格式。

dataType

输入。张量数据类型。

nbDims

输入。张量的维度。

注解: 请勿使用 2 维。更多信息，参见 mcdnnGetRNNLinLayerBiasParams()。

dimA

输入。包含每个维度大小的数组。

返回值

MCDNN_STATUS_SUCCESS

此函数执行成功。

MCDNN_STATUS_BAD_PARAM

张量描述符分配不正确；或输入参数设置不正确。

MCDNN_STATUS_NOT_SUPPORTED

请求的维度尺寸大于支持的最大尺寸。

2.2.1.97 mcdnnSetTensorTransformDescriptor()

```
mcdnnStatus_t mcdnnSetTensorTransformDescriptor(
 mcdnnTensorTransformDescriptor_t transformDesc,
 const uint32_t nbDims,
 const mcdnnTensorFormat_t destFormat,
 const int32_t padBeforeA[],
 const int32_t padAfterA[],
 const uint32_t foldA[],
 const mcdnnFoldingDirection_t direction);
```

此函数用于初始化张量转换描述符，此描述符是已使用 mcdnnCreateTensorTransformDescriptor() 函数创建的。

参数

transformDesc

输出。要初始化的张量转换描述符。

nbDims

输入。转换操作对象的维度。必须大于 2。更多信息，参见张量描述符。

destFormat

输入。需要的目标格式。

padBeforeA[]

输入。包含每个维度前应添加填充量的数组。设置为 NULL，不进行填充。

padAfterA[]

输入。包含每个维度后应添加填充量的数组。设置为 NULL，不进行填充。

foldA[]

输入。包含每个空间维度（2 维及以上）folding 参数的数组。设置为 NULL，不进行折叠。

direction

输入。选择折叠或展开。当 folding 参数全部 <=1 时，此输入无效。更多信息，参见 mcdnnFoldingDirection_t。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

若 direction 无效或 nbDims<=2，transformDesc 参数为 NULL。

MCDNN_STATUS_NOT_SUPPORTED

如果请求的维度尺寸大于支持的最大尺寸（即其中一个 nbDims 大于 MCDNN_DIM_MAX），或者 destFrom 不是 NCHW 或 NHWC。

2.2.1.98 mcdnnSoftmaxForward()

此函数计算 softmax 函数。

```
mcdnnStatus_t mcdnnSoftmaxForward(
    mcdnnHandle_t handle,
    mcdnnSoftmaxAlgorithm_t algo,
    mcdnnSoftmaxMode_t mode,
    const void *alpha,
    const mcdnnTensorDescriptor_t xDesc,
    const void *x,
    const void *beta,
    const mcdnnTensorDescriptor_t yDesc,
    void *y)
```

所有张量格式都支持 4D 和 5D 张量的所有模式和算法。NCHW 完全压缩格式（fully packed）张量的性能最佳。超过 5 个维度的张量，必须在其空间维度中进行压缩。

数据类型

此函数支持以下数据类型：

- MCDNN_DATA_FLOAT
- MCDNN_DATA_DOUBLE

- MCDNN_DATA_HALF
- MCDNN_DATA_BFLOAT16
- MCDNN_DATA_INT8

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

algorithm

输入。指定 softmax 算法的枚举。

mode

输入。指定 softmax 模式的枚举。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将计算结果与输出层中的先验值混合，如下所示：`dstValue = alpha[0]*result + beta[0]*priorDstValue`

xDesc

输入。已初始化的输入张量描述符的句柄。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存。

yDesc

输入。已初始化的输出张量描述符的句柄。

y

输出。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 输入张量和输出张量的 n，c，h，w 维度不同。
- 输入张量和输出张量的数据类型不同。
- algorithm 参数或 mode 参数具有无效的枚举值。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.99 mcdnnSpatialTfGridGeneratorForward()

该函数在输入张量中生成一个坐标网格，对应于输出张量中的每个像素。

```
mcdnnStatus_t mcdnnSpatialTfGridGeneratorForward(
    mcdnnHandle_t handle,
    const mcdnnSpatialTransformerDescriptor_t stDesc,
    const void *theta,
    void *grid)
```

仅支持 2D 转换。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

stDesc

输入。已创建的空间转换描述符对象。

theta

输入。仿射变换矩阵。对于 2D 转换，其大小应为 n*2*3，其中 n 是 stDesc 中指定的图像数。

grid

输出。坐标网格。对于 2D 转换，其大小应为 n*h*w*2，其中 n，h，w 是 stDesc 中指定的。在第四维中，第一个坐标为 x，第二个坐标为 y。

返回值

MCDNN_STATUS_SUCCESS

调用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- handle 为 NULL。
- grid 参数或 theta 参数任一为 NULL。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。不受支持的配置示例如下：

- stDesc 中指定的转换张量的维度 > 4。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.100 mcdnnSpatialTfSamplerForward()

此函数执行采样器操作，并使用网格生成器提供的网格生成输出张量。

```
mcdnnStatus_t mcdnnSpatialTfSamplerForward(
mcdnnHandle_t handle,
const mcdnnSpatialTransformerDescriptor_t stDesc,
const void *alpha,
const mcdnnTensorDescriptor_t xDesc,
const void *x,
const void *grid,
const void *beta,
mcdnnTensorDescriptor_t yDesc,
void *y)
```

仅支持 2D 转换。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

stDesc

输入。已创建的空间转换描述符对象。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将源数值与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*srcValue + beta[0]*priorDstValue

xDesc

输入。已初始化的输入张量描述符的句柄。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存。

grid

输入。由 mcdnnSpatialTfGridGeneratorForward() 生成的坐标网格。

yDesc

输入。已初始化的输出张量描述符的句柄。

y

输出。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

调用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- handle 为 NULL。
- x、y、grid 参数任一为 NULL。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。不受支持的配置示例如下：

- 转换张量的维度 > 4。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.101 mcdnnTransformFilter()

此函数根据描述在不同格式、数据类型或维度之间转换卷积核。它可用于将布局格式不受支持的卷积核转换为布局格式受支持的卷积核。

```
mcdnnStatus_t mcdnnTransformFilter(
    mcdnnHandle_t handle,
    const mcdnnTensorTransformDescriptor_t transDesc,
    const void *alpha,
    const mcdnnFilterDescriptor_t srcDesc,
```

(下页继续)

(续上页)

```
    const void *srcData,
    const void *beta,
    const mcdnnFilterDescriptor_t destDesc,
    void *destData);
```

此函数将缩放数据从 srcDesc 输入卷积核复制到具有不同布局的 destDesc 输出张量。如果 srcDesc 和 destDesc 卷积核描述符具有不同的维度，它们的折叠和填充量以及 transDesc 中指定的顺序必须一致。

srcDesc 和 destDesc 张量不能以任何方式重叠（即，张量不能就地转换）。

注解： 执行折叠转换或零填充转换时，缩放系数 (alpha,beta) 应设置为 (1,0)。但是，非折叠转换支持任何 (alpha,beta) 值。此函数是线程安全的。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

transDesc

输入。包含请求的卷积核转换详细信息的描述符。更多信息，参见 mcdnnTensorTransformDescriptor_t。

alpha, beta

输入。指针，指向用于缩放 srcDesc 输入张量数据的缩放系数，位于主机内存中。beta 用于缩放目标张量，alpha 用于缩放源张量。在折叠和零填充情况下，不接受 beta 缩放值。非折叠支持任何 (alpha，beta) 值。

srcDesc, destDesc

输入。已启用的卷积核描述符的句柄。srcDesc 和 destDesc 不能重叠。更多信息，参见 mcdnnTensorDescriptor_t。

srcData

输入。指针，指向由 srcDesc 描述的张量数据，位于主机内存中。

destData

输出。指针，指向由 destDesc 描述的张量数据，位于主机内存中。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

参数未初始化或初始化不正确，或者 srcDesc 和 destDesc 的维数不同。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。此外，在折叠和填充路径中，除 A=1 和 B=0 之外的任何值都将导致 MCDNN_STATUS_NOT_SUPPORTED。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.102 mcdnnTransformTensor()

```
mcdnnStatus_t mcdnnTransformTensor(
 mcdnnHandle_t handle,
 const void *alpha,
 const mcdnnTensorDescriptor_t xDesc,
 const void *x,
 const void *beta,
 const mcdnnTensorDescriptor_t yDesc,
 void *y)
```

此函数将缩放数据从一个张量复制到具有不同布局的另一个张量。这些描述符需要具有相同的维度，但可以有不同的步幅。输入和输出张量不能以任何方式重叠（即，张量不能就地转换）。该函数可用于将格式不受支持的张量转换为格式受支持的张量。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将源数值与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*srcValue + beta[0]*priorDstValue。

xDesc

输入。已初始化的张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

x

输入。指针，指向 xDesc 描述符描述的张量数据。

yDesc

输入。已初始化的张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

y

输出。指针，指向 yDesc 描述符描述的张量数据。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

两个张量描述符的 n、c、h、w 维度或 dataType 不同。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

2.2.1.103 mcdnnTransformTensorEx()

```
mcdnnStatus_t mcdnnTransformTensorEx(
 mcdnnHandle_t handle,
 const mcdnnTensorTransformDescriptor_t transDesc,
 const void *alpha,
```

(下页继续)

(续上页)

```
const mcdnnTensorDescriptor_t srcDesc,
const void *srcData,
const void *beta,
const mcdnnTensorDescriptor_t destDesc,
void *destData);
```

此函数在不同格式之间转换张量布局。它可用于将布局格式不受支持的张量转换为布局格式受支持的张量。此函数将缩放数据从 srcDesc 输入张量复制到具有不同布局的 destDesc 输出张量。srcDesc 和 destDesc 的张量描述符需要具有相同的维度，但可以有不同的步幅。srcDesc 和 destDesc 张量不能以任何方式重叠（即，张量不能就地转换）。

注解： 执行折叠转换或零填充转换时，缩放系数 (alpha,beta) 应设置为 (1,0)。但是，非折叠转换支持任何 (alpha,beta) 值。此函数是线程安全的。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

transDesc

输入。包含请求的张量转换详细信息的描述符。更多信息，参见 mcdnnTensorTransformDescriptor_t。

alpha, beta

输入。指针，指向用于缩放 srcDesc 输入张量数据的缩放系数，位于主机内存中。beta 用于缩放目标张量，alpha 用于缩放源张量。在折叠和零填充情况下，不接受 beta 缩放值。非折叠支持任何 (alpha，beta) 值。

srcDesc, destDesc

输入。已启用的张量描述符的句柄。srcDesc 和 destDesc 不能重叠。更多信息，参见 mcdnnTensorDescriptor_t。

srcData

输入。指针，指向由 srcDesc 描述的张量数据，位于主机内存中。

destData

输出。指针，指向由 destDesc 描述的张量数据，位于主机内存中。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

参数未初始化或初始化不正确，或者 srcDesc 和 destDesc 的维数不同。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。此外，在折叠和填充路径中，除 A=1 和 B=0 之外的任何值都将导致 MCDNN_STATUS_NOT_SUPPORTED。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

3 mcdnn_ops_train

此实体包含常用的训练函数算法，例如批量归一化，softmax，丢弃等。mcdnn_ops_train 库依赖于 mcdnn_ops_infer。

3.1 API 参考

3.1.1 API 函数

3.1.1.1 mcdnnActivationBackward()

此函数计算神经元激活函数的梯度。

```
mcdnnStatus_t mcdnnActivationBackward(
mcdnnHandle_t                    handle,
mcdnnActivationDescriptor_t      activationDesc,
const void                       *alpha,
const mcdnnTensorDescriptor_t    yDesc,
const void                       *y,
const mcdnnTensorDescriptor_t    dyDesc,
const void                       *dy,
const mcdnnTensorDescriptor_t    xDesc,
const void                       *x,
const void                       *beta,
const mcdnnTensorDescriptor_t    dxDesc,
void                             *dx)
```

该函数支持就地操作，这意味着 dy 和 dx 指针可以相等。但是，这要求相应的张量描述符相同（特别是，输入和输出的步幅必须匹配才能支持就地操作）。所有张量格式都支持 4 维和 5 维。但是，当 xDesc 和 yDesc 的步幅相等且都为 HW-packed 时，可以获得最佳性能。对于超过 5 维的张量，必须压缩其空间维度。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

activationDesc

输入。激活描述符。更多信息，参见 mcdnnActivationDescriptor_t。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将计算结果与输出层中的先验值混合，如下所示：dstValue = alpha[0]*result + beta[0]*priorDstValue。

yDesc

输入。已初始化的输入张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

y

输入。数据指针，指向与张量描述符 yDesc 关联的 GPU 内存。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

dy

输入。数据指针，指向与张量描述符 dyDesc 关联的 GPU 内存。

xDesc

输入。已初始化的输出张量描述符的句柄。

x

输入。数据指针，指向与输出张量描述符 xDesc 关联的 GPU 内存。

dxDesc

输入。已初始化的输出差分张量描述符的句柄。

dx

输出。数据指针，指向与输出张量描述符 dxDesc 关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 输入差分张量和输出差分张量的 nStride，cStride，hStride，wStride 步幅不同，并使用就地操作。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。不受支持的配置示例如下：

- 输入张量和输出张量的 n，c，h，w 维度不同。
- 输入张量和输出张量的数据类型不同。
- 输入张量和输入差分张量的 nStride、cStride、hStride 和 wStride 步幅不同。
- 输出张量和输出差分张量的 nStride、cStride、hStride 和 wStride 步幅不同。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

3.1.1.2 mcdnnBatchNormalizationBackward()

该函数执行反向 BN 层计算。对于 BN 层，可参见 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift。

```
mcdnnStatus_t mcdnnBatchNormalizationBackward(
mcdnnHandle_t                   handle,
mcdnnBatchNormMode_t            mode,
const void                      *alphaDataDiff,
const void                      *betaDataDiff,
const void                      *alphaParamDiff,
```

(下页继续)

(续上页)

```
const void                          *betaParamDiff,
const mcdnnTensorDescriptor_t       xDesc,
const void                          *x,
const mcdnnTensorDescriptor_t       dyDesc,
const void                          *dy,
const mcdnnTensorDescriptor_t       dxDesc,
void                                *dx,
const mcdnnTensorDescriptor_t       bnScaleBiasDiffDesc,
const void                          *bnScale,
void                                *resultBnScaleDiff,
void                                *resultBnBiasDiff,
double                              epsilon,
const void                          *savedMean,
const void                          *savedInvVariance)
```

Only 4D and 5D tensors are supported. 在训练，反向传播和推理过程中的 epsilon 值必须相同。当所有 x、dy 和 dx 都使用 HW-packed 张量时，可以获得更高的性能。有关此函数中使用的参数的辅助张量描述符生成信息，请参见 mcdnnDeriveBNTensorDescriptor()。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式(per-channel 或 per-activation)。更多信息，参见 mcdnnBatchNormMode_t。

alphaDataDiff, betaDataDiff

输入。指向缩放系数（主机内存中）的指针，用于将 dx 梯度输出与目标张量中的先验值混合，如下所示：dstValue = alphaDataDiff[0]*resultValue + betaDataDiff[0]*priorDstValue。

alphaParamDiff, betaParamDiff

输入。指向缩放系数（主机内存中）的指针，用于将 resultBnScaleDiff 和 resultBnBiasDiff 梯度输出与目标张量中的先验值混合，如下所示：dstValue = alphaParamDiff[0]*resultValue + betaParamDiff[0]*priorDstValue。

xDesc, dxDesc, dyDesc

输入。已初始化的张量描述符的句柄。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存，用于层的 x 数据。

dy

输入。数据指针，指向与张量描述符 dyDesc 关联的 GPU 内存，用于反向传播差分 dy 输入。

dx

输入/输出。数据指针，指向与张量描述符 dxDesc 关联的 GPU 内存，用于生成与 x 相关的差分输出。

bnScaleBiasDiffDesc

输入。以下五个张量的共享张量描述符：bnScale，resultBnScaleDiff，resultBnBiasDiff，savedMean 和 savedInvVariance。此张量描述符的维度取决于归一化模式。更多信息，参见 mcdnnDeriveBNTensorDescriptor()。

注解： 对于 FP16 和 FP32 输入张量，此张量描述符的数据类型必须为 float；对于 FP64 输入张量，必须为 double。

---

bnScale

输入。设备内存中指向 BN scale 参数的指针。

注解： 该层的计算不需要 bnBias 参数。

resultBnScaleDiff, resultBnBiasDiff

输出。指针，指向此函数计算得到的 scale 差分和 bias 差分，位于设备内存中。请注意，这些 scale 梯度和 bias 梯度是此 BN 操作特定的权重梯度，根据定义不会反向传播。

epsilon

输入。BN 公式中使用的 Epsilon 值。其值应等于或大于 mcdnn.h 中为 MCDNN_BN_MIN_EPSILON 定义的值。在正向和反向函数中应使用相同的 Epsilon 值。

savedMean, savedInvVariance

输入。可选的 cache 参数，包含在正向传递过程中计算并保存的中间结果。为保证工作正常，在调用此反向函数之前，层的 x 和 bnScale 数据必须保持不变。

注解： 这两个参数可以为 NULL，但只能同时为 NULL。建议使用此 cache，因为内存开销相对较小。

支持的配置

此函数支持多种描述符的以下数据类型组合。

数据类型配置	xDesc	bnScale BiasMean VarDesc	alphaData Diff, beta DataDiff ,alphaParamDiff, betaParam Diff	dyDesc	dxDesc
PSEUDO_HALF_CONFIG	MCDNN_DATA_HALF	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_HALF	MCDNN_DATA_HALF
FLOAT_CONFIG	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT
DOUBLE_CONFIG	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- alpha、beta、x、dy、dx、bnScale、resultBnScaleDiff 和 resultBnBiasDiff 中任意指针为 NULL。
- xDesc、yDesc 或 dxDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。

- spatial 模式下，4D 张量中 bnScaleBiasDiffDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- savedMean 和 savedInvVariance 指针中只有一个为 NULL。
- epsilon 值小于 MCDNN_BN_MIN_EPSILON。
- 任意一对 xDesc、dyDesc 和 dxDesc 的维度或数据类型不匹配。

3.1.1.3 mcdnnBatchNormalizationBackwardEx()

此函数是 mcdnnBatchNormalizationBackward() 的扩展，用于使用快速 NHWC 半持久内核执行反向 BN 层计算。

```
mcdnnStatus_t mcdnnBatchNormalizationBackwardEx (
mcdnnHandle_t                       handle,
mcdnnBatchNormMode_t                mode,
mcdnnBatchNormOps_t                 bnOps,
const void                          *alphaDataDiff,
const void                          *betaDataDiff,
const void                          *alphaParamDiff,
const void                          *betaParamDiff,
const mcdnnTensorDescriptor_t       xDesc,
const void                          *xData,
const mcdnnTensorDescriptor_t       yDesc,
const void                          *yData,
const mcdnnTensorDescriptor_t       dyDesc,
const void                          *dyData,
const mcdnnTensorDescriptor_t       dzDesc,
void                                *dzData,
const mcdnnTensorDescriptor_t       dxDesc,
void                                *dxData,
const mcdnnTensorDescriptor_t       dBnScaleBiasDesc,
const void                          *bnScaleData,
const void                          *bnBiasData,
void                                *dBnScaleData,
void                                *dBnBiasData,
double                               epsilon,
const void                          *savedMean,
const void                          *savedInvVariance,
const mcdnnActivationDescriptor_t   activationDesc,
void                                *workspace,
size_t                              workSpaceSizeInBytes
void                                *reserveSpace
size_t                              reserveSpaceSizeInBytes);
```

如果 workspace 为 NULL 并且传入的 workSpaceSizeInBytes 为零，则该 API 的功能将与非扩展函数 mcdnnBatchNormalizationBackward 完全相同。此 workspace 不需要清理。此外，workspace 不必在正向和反向传递之间保持不变，因为它不用于传递任何信息。

此扩展函数可以接受指向 GPU workspace 的 workspace 指针，以及来自用户的 workSpaceSizeInBytes（workspace 大小）。

bnOps 输入可用于将此函数设置为仅执行 BN，或先执行 BN 再执行激活，或先执行 BN 再执行元素级加法，然后执行激活。

仅支持 4D 和 5D 张量。在训练，反向传播和推理过程中的 epsilon 值必须相同。张量布局为 NCHW 时，若 x、dy 和 dx 都使用 HW-packed 张量时，可以获得更高的性能。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式（spatial 或 per-activation）。更多信息，参见 mcdnnBatchNormMode_t。

bnOps

输入。操作模式。目前，仅在 NHWC 布局中支持 MCDNN_BATCHNORM_OPS_BN_ACTIVATION 和 MCDNN_BATCHNORM_OPS_BN_ADD_ACTIVATION。更多信息，参见 mcdnnBatchNormOps_t。此输入可用于将此函数设置为仅执行 BN，或先执行 BN 再执行激活，或先执行 BN 再执行元素级加法，然后执行激活。

alphaDataDiff, betaDataDiff

输入。指向缩放系数（主机内存中）的指针，用于将 dx 梯度输出与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

alphaParamDiff, betaParamDiff

输入。指向缩放系数（主机内存中）的指针，用于将 dBnScaleData 和 dBnBiasData 梯度输出与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

xDesc, x, yDesc, yData, dyDesc, dyData

输入。用于层的 x 数据，反向传播梯度输入 dy，原始正向输出 y 数据的张量描述符和指针，位于设备内存中。如果 bnOps 设置为 MCDNN_BATCHNORM_OPS_BN，则不需要 yDesc 和 yData，用户可以传入 NULL。更多信息，参见 mcdnnTensorDescriptor_t。

dzDesc, dxDesc

输入。用于计算梯度输出 dz 和 dx 的张量描述符和指针，位于设备内存中。当 bnOps 为 MCDNN_BATCHNORM_OPS_BN 或 MCDNN_BATCHNORM_OPS_BN_ACTIVATION 时，不需要 dzDesc，用户可以传入 NULL。更多信息，参见 mcdnnTensorDescriptor_t。

dzData, dxData

输出。用于计算梯度输出 dz 和 dx 的张量描述符和指针，位于设备内存中。当 bnOps 为 MCDNN_BATCHNORM_OPS_BN 或 MCDNN_BATCHNORM_OPS_BN_ACTIVATION 时，不需要 dzData，用户可以传入 NULL。更多信息，参见 mcdnnTensorDescriptor_t。

dBnScaleBiasDesc

输入。以下 6 个张量的共享张量描述符：bnScaleData、bnBiasData、dBnScaleData、dBnBiasData、savedMean 和 savedInvVariance。更多信息，参见 mcdnnDeriveBNTensorDescriptor()。此张量描述符的维度取决于归一化模式。

注解： 对于 FP16 和 FP32 输入张量，此张量描述符的数据类型必须为 float；对于 FP64 输入张量，必须为 double。

---

bnScaleData

输入。设备内存中指向 BN scale 参数的指针。(在 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift 中，数量 scale 称为 gamma)。

bnBiasData

输入。设备内存中指向 BN bias 参数的指针。(在 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift 中，bias 称为 beta)。此参数仅在执行激活时使用。

dBnScaleData, dBnBiasData

输出。设备内存中分别指向 bnScaleData 和 bnBiasData 梯度的指针。

epsilon

输入。BN 公式中使用的 Epsilon 值。其值应等于或大于 mcdnn.h 中为 MCDNN_BN_MIN_EPSILON 定义的值。在正向和反向函数中应使用相同的 Epsilon 值。

savedMean, savedInvVariance

输入。可选的 cache 参数，包含在正向传递过程中计算并保存的中间结果。为保证工作正常，在调用此反向函数之前，层的 x 和 bnScaleData、bnBiasData 数据必须保持不变。请注意这两个参数可以为 NULL，但只能同时为 NULL。建议使用此 cache，因为内存开销相对较小。

activationDesc

输入。激活操作的描述符。

当 bnOps 输入设置为 MCDNN_BATCHNORM_OPS_BN_ACTIVATION 或 MCDNN_BATCHNORM_OPS_BN_ADD_ACTIVATION 时，将使用此激活，否则用户可以传入 NULL。

workspace

输入。指向 GPU 工作空间的指针。如果 workspace 为 NULL 并且传入的 workSpaceSizeInBytes 为零，则该 API 的功能将与非扩展函数 mcdnnBatchNormalizationBackward() 完全相同。

workSpaceSizeInBytes

输入。workspace 的大小。它必须足够大，才能通过此函数触发快速 NHWC 半持久内核。

reserveSpace

输入。指针，指向用作 reserveSpace 的 GPU 工作空间。

reserveSpaceSizeInBytes

输入。reserveSpace 的大小。它必须等于或大于 mcdnnGetBatchNormalizationTrainingExReserveSpaceSize() 所需的量。

mcdnnBatchNormalizationBackwardEx() 支持的配置

数据类型配置	xDesc, yDesc	dBnScaleBiasDesc	alphaDataDiff, betaDataDiff,alphaParamDiff, betaParamDiff	dyDesc, dzDesc, dxDesc
PSEUDO_HALF_CONFIG	MCDNN_DATA_HALF	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_HALF
FLOAT_CONFIG	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT
DOUBLE_CONFIG	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- alphaDataDiff、betaDataDiff、alphaParamDiff、betaParamDiff、x、dy、dx、bnScale、resultBnScaleDiff 和 resultBnBiasDiff 中任意指针为 NULL。
- xDesc、yDesc 或 dxDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。

- spatial 模式下，4D 张量中 dBnScaleBiasDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- savedMean 和 savedInvVariance 指针中只有一个为 NULL。
- epsilon 值小于 MCDNN_BN_MIN_EPSILON。
- 任意一对 xDesc、dyDesc 或 dxDesc 的维度或数据类型不匹配。

3.1.1.4 mcdnnBatchNormalizationForwardTraining

此函数在训练阶段执行正向 BN 层计算。对于 BN 层，可参见 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift。

```
mcdnnStatus_t mcdnnBatchNormalizationForwardTraining(
mcdnnHandle_t                    handle,
mcdnnBatchNormMode_t             mode,
const void                       *alpha,
const void                       *beta,
const mcdnnTensorDescriptor_t    xDesc,
const void                       *x,
const mcdnnTensorDescriptor_t    yDesc,
void                             *y,
const mcdnnTensorDescriptor_t    bnScaleBiasMeanVarDesc,
const void                       *bnScale,
const void                       *bnBias,
double                           exponentialAverageFactor,
void                             *resultRunningMean,
void                             *resultRunningVariance,
double                           epsilon,
void                             *resultSaveMean,
void                             *resultSaveInvVariance)
```

仅支持 4D 和 5D 张量。

在训练，反向传播和推理过程中的 epsilon 值必须相同。对于推理阶段，使用 mcdnnBatchNormalizationForwardInference。

当 x 和 y 都使用 HW-packed 张量时，可以获得更高的性能。有关此函数中使用的参数的辅助张量描述符生成信息，请参见 mcdnnDeriveBNTensorDescriptor()。

参数

handle

已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

操作模式（spatial 或 per-activation）。更多信息，参见 mcdnnBatchNormMode_t。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将层输出值与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

xDesc, yDesc

设备内存中用于层的 x 和 y 数据的张量描述符和指针。更多信息，参见 mcdnnTensorDescriptor_t。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存，用于层的 x 输入数据。

y

输入。数据指针，指向与张量描述符 yDesc 关联的 GPU 内存，用于 BN 层的 y 输出。

bnScaleBiasMeanVarDesc

共享张量描述符，用于通过 mcdnnDeriveBNTensorDescriptor() 获取的辅助张量。此张量描述符的维度取决于归一化模式。

bnScale, bnBias

输入。设备内存中用于 BN scale 和 bias 参数的指针。请注意，bnBias 参数可以替换上一层的 bias 参数以提高效率。

exponentialAverageFactor

输入。移动平均（Moving Average，MA）计算中使用的系数如下：

runningMean = runningMean*(1-factor) + newMean*factor

在对函数的第 N 次调用中使用 factor=1/(1+n) 可进行累积移动平均（Cumulative Moving Average，CMA）计算，例如：

CMA[n] = (x[1]+…+x[n])/n

例如：

CMA[n+1] = (n*CMA[n]+x[n+1])/(n+1)

= ((n+1)*CMA[n]-CMA[n])/(n+1) + x[n+1]/(n+1)

= CMA[n]*(1-1/(n+1))+x[n+1]*1/(n+1)

= CMA[n]*(1-factor) + x(n+1)*factor

resultRunningMean, resultRunningVariance

输入/输出。移动均值（Running mean）和方差（Running variance）张量（与 scale 和 bias 张量具有相同的描述符）。这两个指针可以为 NULL，但只能同时为 NULL。存储在 resultRunningVariance 中（或作为推理模式中的输入被传入）的值是样本方差，是方差 [x] 的移动平均值，其中方差是根据模式在 batch 或 spatial+batch 维度上进行计算的。如果这些指针不为 NULL，则张量应初始化为一些合理的值或 0。

epsilon

输入。BN 公式中使用的 Epsilon 值。其值应等于或大于 mcdnn.h 中为 MCDNN_BN_MIN_EPSILON 定义的值。在正向和反向函数中应使用相同的 Epsilon 值。

resultSaveMean, resultSaveInvVariance

输出。可选的 cache 参数，用于保存在正向传递过程中计算得到的中间结果。当用于 mcdnnBatchNormalizationBackward() 函数时，这些缓冲区可用来加速反向传递。用户不能直接使用存储在 resultSaveMean 和 resultSaveInvVariance 缓冲区的中间结果。根据 BN 模式，存储在 resultSaveInvVariance 中的结果可能会有所不同。为保证 cache 正常工作，在调用反向函数之前，输入层数据必须保持不变。请注意这两个参数可以为 NULL，但只能同时为 NULL。在这种情况下，将不会保存中间统计信息，并且 mcdnnBatchNormalizationBackward() 必须进行重新计算。建议使用此 cache，由于这些张量的维数乘积远小于数据张量的维数乘积，因此内存开销相对较小。

支持的 mcdnnBatchNormalizationForwardTraining() 配置

数据类型配置	xDesc	bnScaleBias MeanVarDesc	alpha, beta	yDesc
PSEUDO_HALF_CONFIG	MCDNN_DATA_HALF	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_HALF
FLOAT_CONFIG	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT
DOUBLE_CONFIG	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE
PSEUDO_BFLOAT16_CONFIG	MCDNN_DATA_BFLOAT16	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_BFLOAT16

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- alpha、beta、x、y、bnScale 和 bnBias 中任一指针为 NULL。
- xDesc 或 yDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。
- spatial 模式下，4D 张量中 bnScaleBiasMeanVarDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- resultSaveMean 和 resultSaveInvVariance 指针中只有一个为 NULL。
- resultRunningMean 和 resultRunningInvVariance 指针中只有一个为 NULL。
- epsilon 值小于 MCDNN_BN_MIN_EPSILON。
- xDesc 或 yDesc 的维度或数据类型不匹配。

3.1.1.5 mcdnnBatchNormalizationForwardTrainingEx()

该函数是 mcdnnBatchNormalizationForwardTraining() 的扩展，用于执行正向 BN 层计算。

```
mcdnnStatus_t mcdnnBatchNormalizationForwardTrainingEx(
mcdnnHandle_t                       handle,
mcdnnBatchNormMode_t                mode,
mcdnnBatchNormOps_t                 bnOps,
const void                          *alpha,
const void                          *beta,
const mcdnnTensorDescriptor_t       xDesc,
const void                          *xData,
const mcdnnTensorDescriptor_t       zDesc,
const void                          *zData,
const mcdnnTensorDescriptor_t       yDesc,
void                                *yData,
const mcdnnTensorDescriptor_t       bnScaleBiasMeanVarDesc,
const void                          *bnScaleData,
const void                          *bnBiasData,
double                              exponentialAverageFactor,
void                                *resultRunningMeanData,
```

(下页继续)

(续上页)

```
void                                *resultRunningVarianceData,
double                              epsilon,
void                                *saveMean,
void                                *saveInvVariance,
const mcdnnActivationDescriptor_t   activationDesc,
void                                *workspace,
size_t                              workSpaceSizeInBytes
void                                *reserveSpace
size_t                              reserveSpaceSizeInBytes);
```

当满足以下条件时，此 API 将触发新的半持久 NHWC 内核：

- 所有张量，即 x、y、dz、dy 和 dx，必须是 NHWC 完全压缩，并且类型必须是 MCDNN_DATA_HALF。
- 输入参数模式必须设置为 MCDNN_BATCHNORM_SPATIAL_PERSISTENT。
- workspace 不是 NULL。
- 之前，C 张量维度应始终是 4 的倍数。只有当 bnOps 为 MCDNN_BATCHNORM_OPS_BN_ADD_ACTIVATION 时，C 张量维度应是 4 的倍数。
- WorkSpaceSizeInBytes 等于或大于 mcdnnGetBatchNormalizationForwardTrainingExWorkspace-Size() 所需的量。
- ReserveSpaceSizeInBytes 等于或大于 mcdnnGetBatchNormalizationTrainingExReserveSpace-Size() 所需的量。
- 必须保留使用 mcdnnBatchNormalizationForwardTrainingEx() 存储在 reserveSpace 中的内容。

如果 workspace 为 NULL 并且传入的 workSpaceSizeInBytes 为零，则该 API 的功能将与非扩展函数 mcdnnBatchNormalizationForwardTraining() 完全相同。此 workspace 不需要清理。此外，workspace 不必在正向和反向传递之间保持不变，因为它不用于传递任何信息。

此扩展函数可以接受指向 GPU workspace 的 workspace 指针，以及来自用户的 workSpaceSizeInBytes（workspace 大小）。

bnOps 输入可用于将此函数设置为仅执行 BN，或先执行 BN 再执行激活，或先执行 BN 再执行元素级加法，然后执行激活。

仅支持 4D 和 5D 张量。在训练，反向传播和推理过程中的 epsilon 值必须相同。

张量布局为 NCHW 时，若 x、dy 和 dx 都使用 HW-packed 张量时，可以获得更高的性能。

参数

handle

已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

操作模式（spatial 或 per-activation）。更多信息，参见 mcdnnBatchNormMode_t。

bnOps

输入。快速 NHWC 内核的操作模式。更多信息，参见 mcdnnBatchNormOps_t。此输入可用于将此函数设置为仅执行 BN，或先执行 BN 再执行激活，或先执行 BN 再执行元素级加法，然后执行激活。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将层输出值与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

xDesc, xData, zDesc, zData, yDesc, yData

设备内存中的张量描述符和指针，用于层的输入 x 和输出 y，以及用于在激活之前对 BN 操作的结果进行残差相加的可选 z 张量输入。可选的 zDes 和 zData 描述符仅在 bnOps 为

MCDNN_BATCHNORM_OPS_BN_ADD_ACTIVATION 时使用，否则用户可以传入 NULL。使用时，z 的维度应与 x 和最终输出 y 的维度完全相同。更多信息，参见 mcdnnTensorDescriptor_t。

bnScaleBiasMeanVarDesc

共享张量描述符，用于通过 mcdnnDeriveBNTensorDescriptor() 获取的辅助张量。此张量描述符的维度取决于归一化模式。

bnScaleData, bnBiasData

输入。设备内存中用于 BN scale 和 bias 参数的指针。请注意，bnBiasData 参数可以替换上一层的 bias 参数以提高效率。

exponentialAverageFactor

输入。移动平均（Moving Average，MA）计算中使用的系数如下：

runningMean = runningMean*(1-factor) + newMean*factor

在对函数的第 N 次调用中使用 factor=1/(1+n) 可进行累积移动平均（Cumulative Moving Average，CMA）计算，例如：

CMA[n] = (x[1]+…+x[n])/n

例如：

CMA[n+1] = (n*CMA[n]+x[n+1])/(n+1)

= ((n+1)*CMA[n]-CMA[n])/(n+1) + x[n+1]/(n+1)

= CMA[n]*(1-1/(n+1))+x[n+1]*1/(n+1)

= CMA[n]*(1-factor) + x(n+1)*factor

resultRunningMeanData, resultRunningVarianceData

输入/输出。指向移动均值和移动方差数据的指针。这两个指针可以为 NULL，但只能同时为 NULL。存储在 resultRunningVarianceData 中（或作为推理模式中的输入被传入）的值是样本方差，是方差 [x] 的移动平均值，其中方差是根据模式在 batch 或 spatial+batch 维度上进行计算的。如果这些指针不为 NULL，则张量应初始化为一些合理的值或 0。

epsilon

输入。BN 公式中使用的 Epsilon 值。其值应等于或大于 mcdnn.h 中为 MCDNN_BN_MIN_EPSILON 定义的值。在正向和反向函数中应使用相同的 Epsilon 值。

saveMean, saveInvVariance

输出。可选的 cache 参数，包含在正向传递过程中计算并保存的中间结果。为保证工作正常，在调用此反向函数之前，层的 x 和 bnScaleData、bnBiasData 数据必须保持不变。请注意这两个参数可以为 NULL，但只能同时为 NULL。建议使用此 cache，因为内存开销相对较小。

activationDesc

输入。激活操作的张量描述符。

当 bnOps 输入设置为 MCDNN_BATCHNORM_OPS_BN_ACTIVATION 或 MCDNN_BATCHNORM_OPS_BN_ADD_ACTIVATION 时，将使用此激活，否则用户可以传入 NULL。

workspace, workSpaceSizeInBytes

输入。workspace 是指向 GPU 工作空间的指针，workSpaceSizeInBytes 是 workspace 的大小。当 workspace 不为 NULL 且 workSpaceSizeInBytes 足够大，张量布局为 NHWC 且数据类型配置受支持时，此函数将触发一个新的半持久 NHWC 内核以进行 BN 操作。此 workspace 不需要清理。此外，workspace 不需要在正向和反向传递之间保持不变。

reserveSpace

输入。指针，指向用作 reserveSpace 的 GPU 工作空间。

reserveSpaceSizeInBytes

输入。reserveSpace 的大小。必须等于或大于 mcdnnGetBatchNormalizationTrainingExReserveSpaceSize() 所需的量。

支持的 mcdnnBatchNormalizationForwardTrainingEx() 配置

数据类型配置	xDesc	bnScale BiasMean VarDesc	alpha, beta	zDesc	yDesc
PSEUDO_HALF_CONFIG	MCDNN_DATA_HALF	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_HALF	MCDNN_DATA_HALF
FLOAT_CONFIG	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	不支持	MCDNN_DATA_FLOAT
DOUBLE_CONFIG	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE	不支持	MCDNN_DATA_DOUBLE

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- alpha、beta、x、y、bnScaleData 和 bnBiasData 中任一指针为 NULL。
- xDesc 或 yDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。
- spatial 模式下，4D 张量中 bnScaleBiasMeanVarDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- saveMean 和 saveInvVariance 指针中只有一个为 NULL。
- resultRunningMeanData 和 resultRunningInvVarianceData 指针中只有一个为 NULL。
- epsilon 值小于 MCDNN_BN_MIN_EPSILON。
- xDesc 和 yDesc 的维度或数据类型不匹配。

3.1.1.6 mcdnnDivisiveNormalizationBackward()

该函数执行反向 DivisiveNormalization 层计算。

```
mcdnnStatus_t mcdnnDivisiveNormalizationBackward(
mcdnnHandle_t                   handle,
mcdnnLRNDescriptor_t            normDesc,
mcdnnDivNormMode_t              mode,
const void                      *alpha,
const mcdnnTensorDescriptor_t   xDesc,
const void                      *x,
const void                      *means,
const void                      *dy,
void                            *temp,
```

(下页继续)

(续上页)

```
void                                *temp2,
const void                          *beta,
const mcdnnTensorDescriptor_t       dxDesc,
void                                *dx,
void                                *dMeans)
```

支持的张量格式为 NCHW（适用于 4D）和 NCDHW（适用于 5D），具有任意非重叠非负值步幅。仅支持 4D 和 5D 张量。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

normDesc

输入。已初始化的 LRN 参数描述符的句柄（此描述符用于 LRN 和 DivisiveNormalization 层）。

mode

输入。DivisiveNormalization 层操作模式。目前仅支持实现 MCDNN_DIVNORM_PRECOMPUTED_MEANS。归一化是使用用户要预计算的均值输入张量来执行的。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将层输出值与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

xDesc, x, means

输入。设备内存中用于层的 x 和 means 数据的张量描述符和指针。请注意，means 张量应由用户进行预计算。它还可以包含任意有效值（不需要是实际 means 值，例如可以是高斯内核卷积的结果）。

dy

输入。设备内存中的张量指针，用于层的 dy 累积损失差分数据（cumulative loss differential data）（错误反向传播）。

temp, temp2

Workspace. 设备内存中的临时张量。这些张量用于在反向传递过程中计算中间值。但在从正向到反向传递的过程中可以不保留。正向和反向传递都使用 xDesc 作为其描述符。

dxDesc

输入。dx 和 dMeans 的张量描述符。

dx, dMeans

输出。设备内存中的张量指针，用于输出累积梯度 dx 和 dMeans（dLoss/dx 和 dLoss/dMeans）的层。两者共享相同的描述符。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- x，dx，temp 和 temp2 中任一张量指针为 NULL。
- 任意输入张量或输出张量的维数不在 [4,5] 范围内。
- alpha 或 beta 指针为 NULL。

- xDesc 和 dxDesc 之间的维度不匹配。
- LRN 描述符参数超出其有效范围。
- 任意张量步幅均为负的。

MCDNN_STATUS_UNSUPPORTED

该函数不支持已提供的配置，例如，不支持输入和输出张量之间的步幅不匹配（同一维度下）。

3.1.1.7 mcdnnDropoutBackward()

此函数以 dy 执行反向丢弃操作，以 dx 返回结果。如果在正向丢弃操作期间，x 上的值被传播到 y，那么在反向丢弃操作期间，dy 的值会传播到 dx，否则，dx 值将设置为 0。

```
mcdnnStatus_t mcdnnDropoutBackward(
mcdnnHandle_t                    handle,
const mcdnnDropoutDescriptor_t   dropoutDesc,
const mcdnnTensorDescriptor_t    dydesc,
const void                       *dy,
const mcdnnTensorDescriptor_t    dxdesc,
void                             *dx,
void                             *reserveSpace,
size_t                           reserveSpaceSizeInBytes)
```

完全压缩的张量可获得更好的性能。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

dropoutDesc

输入。已创建的丢弃描述符对象。

dyDesc

输入。已初始化的张量描述符的句柄。

dy

输入。指针，指向 dyDesc 描述符描述的张量数据。

dxDesc

输入。已初始化的张量描述符的句柄。

dx

输出。指针，指向 dxDesc 描述符描述的张量数据。

reserveSpace

输入。指针，指向此函数使用的用户分配的 GPU 内存。在调用 mcdnnDropoutForward 时填充的 reserveSpace，且未更改。

reserveSpaceSizeInBytes

输入。指定为预留空间提供的内存大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

调用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 输入张量和输出张量的元素数不同。
- 输入张量和输出张量的数据类型不同。
- 输入张量和输出张量的步幅不同，并使用就地操作（即 x 和 y 指针相等）。
- 提供的 reserveSpaceSizeInBytes 小于 mcdnnDropoutGetReserveSpaceSize() 返回的值。
- 没有在带有非 NULL 状态参数的 dropoutDesc 上调用 mcdnnSetDropoutDescriptor()。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

3.1.1.8 mcdnnGetBatchNormalizationBackwardExWorkspaceSize()

此函数返回用户应分配的工作空间 GPU 内存量，以便能够为指定的 bnOps 输入设置来调用 mcdnnGetBatchNormalizationBackwardExWorkspaceSize() 函数。然后，分配的 workspace 将传入 mcdnnGetBatchNormalizationBackwardExWorkspaceSize() 函数。

```
mcdnnStatus_t mcdnnGetBatchNormalizationBackwardExWorkspaceSize(
mcdnnHandle_t                       handle,
mcdnnBatchNormMode_t                mode,
mcdnnBatchNormOps_t                 bnOps,
const mcdnnTensorDescriptor_t       xDesc,
const mcdnnTensorDescriptor_t       yDesc,
const mcdnnTensorDescriptor_t       dyDesc,
const mcdnnTensorDescriptor_t       dzDesc,
const mcdnnTensorDescriptor_t       dxDesc,
const mcdnnTensorDescriptor_t       dBnScaleBiasDesc,
const mcdnnActivationDescriptor_t   activationDesc,
size_t                              *sizeInBytes)
```

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式（spatial 或 per-activation）。更多信息，参见 mcdnnBatchNormMode_t。

bnOps

输入。快速 NHWC 内核的操作模式。更多信息，参见 mcdnnBatchNormOps_t。此输入可用于将此函数设置为仅执行 BN，或先执行 BN 再执行激活，或先执行 BN 再执行元素级加法，然后执行激活。

xDesc, yDesc, dyDesc, dzDesc, dxDesc

设备内存中的张量描述符和指针，用于层的 x 数据，反向传播差分 dy（输入），可选 y 输入数据，可选 dz 输出和 dx 输出（关于 x 的结果差分）。更多信息，参见 mcdnnTensorDescriptor_t。

dBnScaleBiasDesc

输入。以下 6 个张量的共享张量描述符：bnScaleData、bnBiasData、dBnScaleData、dBnBiasData、savedMean 和 savedInvVariance。共享张量描述符，用于通过 mcdnnDeriveBNTensorDescriptor() 获取的辅助张量。此张量描述符的维度取决于归一化模式。请注意，对于 FP16 和 FP32 输入张量，此张量描述符的数据类型必须为 float；对于 FP64 输入张量，此张量描述符的数据类型必须为 double。

activationDesc

输入。激活操作的描述符。

当 bnOps 输入设置为 MCDNN_BATCHNORM_OPS_BN_ACTIVATION 或 MCDNN_BATCHNORM_OPS_BN_ADD_ACTIVATION 时，将使用此激活，否则用户可以传入 NULL。

sizeInBytes

输出。工作空间所需的 GPU 内存量，由该函数确定，以便能够使用指定的 bnOps 输入设置来执行 mcdnnGetBatchNormalizationForwardTrainingExWorkspaceSize() 函数。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- xDesc、yDesc 或 dxDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。
- spatial 模式下，4D 张量中 dBnScaleBiasDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- 任意一对 xDesc、dyDesc 或 dxDesc 的维度或数据类型不匹配。

3.1.1.9 mcdnnGetBatchNormalizationForwardTrainingExWorkspaceSize()

此函数返回用户应分配的工作空间 GPU 内存量，以便能够为指定的 bnOps 输入设置来调用 mcdnnGetBatchNormalizationForwardTrainingExWorkspaceSize() 函数。然后，分配的 workspace 应由用户传入该函数。

```
mcdnnGetBatchNormalizationForwardTrainingExWorkspaceSize().
mcdnnStatus_t mcdnnGetBatchNormalizationForwardTrainingExWorkspaceSize(
mcdnnHandle_t                       handle,
mcdnnBatchNormMode_t                mode,
mcdnnBatchNormOps_t                 bnOps,
const mcdnnTensorDescriptor_t       xDesc,
const mcdnnTensorDescriptor_t       zDesc,
const mcdnnTensorDescriptor_t       yDesc,
const mcdnnTensorDescriptor_t       bnScaleBiasMeanVarDesc,
const mcdnnActivationDescriptor_t   activationDesc,
size_t                              *sizeInBytes);
```

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式（spatial 或 per-activation）。更多信息，参见 mcdnnBatchNormMode_t。

bnOps

输入。快速 NHWC 内核的操作模式。更多信息，参见 mcdnnBatchNormOps_t。此输入可用于将此函数设置为仅执行 BN，或先执行 BN 再执行激活，或先执行 BN 再执行元素级加法，然后执行激活。

xDesc, zDesc, yDesc

设备内存中的张量描述符和指针，用于层的 x 数据，可选 z 输入数据和 y 输出。仅当 bnOps 为 MCDNN_BATCHNORM_OPS_BN_ADD_ACTIVATION 时才需要 zDesc，否则用户可以传入 NULL。更多信息，参见 mcdnnTensorDescriptor_t。

bnScaleBiasMeanVarDesc

输入。以下 6 个张量的共享张量描述符：bnScaleData、bnBiasData、dBnScaleData、dBnBiasData、savedMean 和 savedInvVariance。共享张量描述符，用于通过 mcdnnDeriveBNTensorDescriptor() 获取的辅助张量。此张量描述符的维度取决于归一化模式。请注意，对于 FP16 和 FP32 输入张量，此张量描述符的数据类型必须为 float；对于 FP64 输入张量，此张量描述符的数据类型必须为 double。

activationDesc

输入。激活操作的描述符。

当 bnOps 输入设置为 MCDNN_BATCHNORM_OPS_BN_ACTIVATION 或 MCDNN_BATCHNORM_OPS_BN_ADD_ACTIVATION 时，将使用此激活，否则用户可以传入 NULL。

sizeInBytes

输出。工作空间所需的 GPU 内存量，由该函数确定，以便能够使用指定的 bnOps 输入设置来执行 mcdnnGetBatchNormalizationForwardTrainingExWorkspaceSize() 函数。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- xDesc、yDesc 或 dxDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。
- spatial 模式下，4D 张量中 dBnScaleBiasDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- xDesc 或 yDesc 的维度或数据类型不匹配。

3.1.1.10 mcdnnGetBatchNormalizationTrainingExReserveSpaceSize()

此函数返回用户应分配的预留工作空间 GPU 内存量，以用于为指定的 bnOps 输入设置执行 BN 操作。与工作空间不同，预留空间应保留在正向和反向调用之间，并且不应更改数据。

```
mcdnnStatus_t mcdnnGetBatchNormalizationTrainingExReserveSpaceSize(
mcdnnHandle_t                       handle,
mcdnnBatchNormMode_t                mode,
mcdnnBatchNormOps_t                 bnOps,
const mcdnnActivationDescriptor_t   activationDesc,
const mcdnnTensorDescriptor_t       xDesc,
size_t                              *sizeInBytes);
```

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式（spatial 或 per-activation）。更多信息，参见 mcdnnBatchNormMode_t。

bnOps

输入。快速 NHWC 内核的操作模式。更多信息，参见 mcdnnBatchNormOps_t。此输入可用于将此函数设置为仅执行 BN，或先执行 BN 再执行激活，或先执行 BN 再执行元素级加法，然后执行激活。

xDesc

用于层的 x 数据的张量描述符。更多信息，参见 mcdnnTensorDescriptor_t。

activationDesc

输入。激活操作的描述符。

当 bnOps 输入设置为 MCDNN_BATCHNORM_OPS_BN_ACTIVATION 或 MCDNN_BATCHNORM_OPS_BN_ADD_ACTIVATION 时，将使用此激活，否则用户可以传入 NULL。

sizeInBytes

输出。预留的 GPU 内存量。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- xDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。

3.1.1.11 mcdnnGetNormalizationBackwardWorkspaceSize()

此函数返回用户应分配的工作空间 GPU 内存量，以便能够为指定的 normOps 和 algo 输入设置来调用 mcdnnNormalizationBackward() 函数。然后，分配的 workspace 将传入 mcdnnNormalizationBackward() 函数。

```
mcdnnStatus_t
mcdnnGetNormalizationBackwardWorkspaceSize(mcdnnHandle_t handle,
mcdnnNormMode_t                 mode,
mcdnnNormOps_t                  normOps,
mcdnnNormAlgo_t                 algo,
const mcdnnTensorDescriptor_t   xDesc,
const mcdnnTensorDescriptor_t   yDesc,
const mcdnnTensorDescriptor_t   dyDesc,
const mcdnnTensorDescriptor_t   dzDesc,
const mcdnnTensorDescriptor_t   dxDesc,
const mcdnnTensorDescriptor_t
                                dNormScaleBiasDesc,
const mcdnnActivationDescriptor_t
                                activationDesc,
const mcdnnTensorDescriptor_t
                                normMeanVarDesc,
size_t                          *sizeInBytes,
int                             groupCnt);
```

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式（per-channel 或 per-activation）。更多信息，参见 mcdnnNormMode_t。

normOps

输入。post-operative 模式。目前，仅在 NHWC 布局中支持 MCDNN_NORM_OPS_NORM_ACTIVATION 和 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION。更多信息，参见 mcdnnNormOps_t。此输入可用于将此函数设置为仅执行归一化，或先执行归一化再执行激活，或先执行归一化再执行元素级加法，然后执行激活。

algo

输入。要执行的算法。更多信息，参见 mcdnnNormAlgo_t。

xDesc, yDesc, dyDesc, dzDesc, dxDesc

输入。设备内存中的张量描述符和指针，用于层的 x 数据，反向传播差分 dy（输入），可选 y 输入数据，可选 dz 输出和 dx 输出（关于 x 的结果差分）。更多信息，参见 mcdnnTensorDescriptor_t。

dNormScaleBiasDesc

输入。以下 4 个张量的共享张量描述符：normScaleData、normBiasData、dNormScaleData、dNormBiasData。此张量描述符的维度取决于归一化模式。请注意，对于 FP16 和 FP32 输入张量，此张量描述符的数据类型必须为 float；对于 FP64 输入张量，此张量描述符的数据类型必须为 double。

activationDesc

输入。激活操作的描述符。

当 normOps 输入设置为 MCDNN_NORM_OPS_NORM_ACTIVATION 或 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION 时，将使用此激活，否则用户可以传入 NULL。

normMeanVarDesc

输入。以下张量的共享张量描述符：savedMean 和 savedInvVariance。此张量描述符的维度取决于归一化模式。请注意，对于 FP16 和 FP32 输入张量，此张量描述符的数据类型必须为 float；对于 FP64 输入张量，此张量描述符的数据类型必须为 double。

sizeInBytes

输出。工作空间所需的 GPU 内存量，由该函数确定，以便能够使用指定的 normOps 输入设置来执行 mcdnnGetNormalizationForwardTrainingWorkspaceSize() 函数。

groupCnt

输入。分组卷积数。当前仅支持 1。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- xDesc、yDesc 或 dxDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。

- per-channel 模式下，4D 张量中 dNormScaleBiasDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- 任意一对 xDesc、dyDesc 或 dxDesc 的维度或数据类型不匹配。

3.1.1.12 mcdnnGetNormalizationForwardTrainingWorkspaceSize()

此函数返回用户应分配的工作空间 GPU 内存量，以便能够为指定的 normOps 和 algo 输入设置来调用 mcdnnNormalizationForwardTraining() 函数。The workspace allocated should t

```
mcdnnStatus_t
mcdnnGetNormalizationForwardTrainingWorkspaceSize(mcdnnHandle_t handle,
mcdnnNormMode_t                 mode,
mcdnnNormOps_t                  normOps,
mcdnnNormAlgo_t                 algo,
const mcdnnTensorDescriptor_t   xDesc,
const mcdnnTensorDescriptor_t
                                zDesc,
const mcdnnTensorDescriptor_t
                                yDesc,
const mcdnnTensorDescriptor_t
                                normScaleBiasDesc,
const mcdnnActivationDescriptor_t
                                activationDesc,
const mcdnnTensorDescriptor_t
                                normMeanVarDesc,
size_t                          *sizeInBytes,
int                             groupCnt);
```

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式（per-channel 或 per-activation）。更多信息，参见 mcdnnNormMode_t。

normOps

输入。post-operative 模式。目前，仅在 NHWC 布局中支持 MCDNN_NORM_OPS_NORM_ACTIVATION 和 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION。更多信息，参见 mcdnnNormOps_t。此输入可用于将此函数设置为仅执行归一化，或先执行归一化再执行激活，或先执行归一化再执行元素级加法，然后执行激活。

algo

输入。要执行的算法。更多信息，参见 mcdnnNormAlgo_t。

xDesc, zDesc, yDesc

设备内存中的张量描述符和指针，用于层的 x 数据，可选 z 输入数据和 y 输出。仅当 normOps 为 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION 时才需要 zDesc，否则用户可以传入 NULL。更多信息，参见 mcdnnTensorDescriptor_t。

normScaleBiasDesc

输入。以下张量的共享张量描述符：normScaleData 和 normBiasData。此张量描述符的维度取决于归一化模式。请注意，对于 FP16 和 FP32 输入张量，此张量描述符的数据类型必须为 float；对于 FP64 输入张量，此张量描述符的数据类型必须为 double。

activationDesc

输入。激活操作的描述符。

当 normOps 输入设置为 MCDNN_NORM_OPS_NORM_ACTIVATION 或 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION 时，将使用此激活，否则用户可以传入 NULL。

normMeanVarDesc

输入。以下张量的共享张量描述符：savedMean 和 savedInvVariance。此张量描述符的维度取决于归一化模式。请注意，对于 FP16 和 FP32 输入张量，此张量描述符的数据类型必须为 float；对于 FP64 输入张量，此张量描述符的数据类型必须为 double。

sizeInBytes

输出。工作空间所需的 GPU 内存量，由该函数确定，以便能够使用指定的 normOps 输入设置来执行 mcdnnGetNormalizationForwardTrainingWorkspaceSize() 函数。

groupCnt

输入。分组卷积数。当前仅支持 1。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- xDesc、yDesc 或 zDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。
- per-channel 模式下，4D 张量中 normScaleBiasDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- xDesc 或 yDesc 的维度或数据类型不匹配。

3.1.1.13 mcdnnGetNormalizationTrainingReserveSpaceSize()

此函数返回用户应分配的预留工作空间 GPU 内存量，以用于为指定的 normOps 输入设置执行归一化操作。与工作空间不同，预留空间应保留在正向和反向调用之间，并且不应更改数据。

```
mcdnnStatus_t
mcdnnGetNormalizationTrainingReserveSpaceSize(mcdnnHandle_t handle,
mcdnnNormMode_t                 mode,
mcdnnNormOps_t                  normOps,
mcdnnNormAlgo_t                 algo,
const mcdnnActivationDescriptor_t
                                activationDesc,
const mcdnnTensorDescriptor_t   xDesc,
size_t                          *sizeInBytes,
int                             groupCnt);
```

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式（per-channel 或 per-activation）。更多信息，参见 mcdnnNormMode_t。

normOps

输入。post-operative 模式。目前，仅在 NHWC 布局中支持 MCDNN_NORM_OPS_NORM_ACTIVATION 和 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION。更多信息，参见 mcdnnNormOps_t。此输入可用于将此函数设置为仅执行归一化，或先执行归一化再执行激活，或先执行归一化再执行元素级加法，然后执行激活。

algo

输入。要执行的算法。更多信息，参见 mcdnnNormAlgo_t。

xDesc

用于层的 x 数据的张量描述符。更多信息，参见 mcdnnTensorDescriptor_t。

activationDesc

输入。激活操作的描述符。

当 normOps 输入设置为 MCDNN_NORM_OPS_NORM_ACTIVATION 或 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION 时，将使用此激活，否则用户可以传入 NULL。

sizeInBytes

输出。预留的 GPU 内存量。

groupCnt

输入。分组卷积数。当前仅支持 1。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- xDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。

3.1.1.14 mcdnnLRNCrossChannelBackward()

该函数执行反向 LRN 层计算。

```
mcdnnStatus_t mcdnnLRNCrossChannelBackward(
mcdnnHandle_t                   handle,
mcdnnLRNDescriptor_t            normDesc,
mcdnnLRNMode_t                  lrnMode,
const void                      *alpha,
const mcdnnTensorDescriptor_t   yDesc,
const void                      *y,
const mcdnnTensorDescriptor_t   dyDesc,
const void                      *dy,
const mcdnnTensorDescriptor_t   xDesc,
const void                      *x,
const void                      *beta,
const mcdnnTensorDescriptor_t   dxDesc,
void                            *dx)
```

支持的格式包括：positive-strided，NCHW 和 NHWC（适用于 4D x 和 y），以及 5D 中仅支持 NCDHW DHW-packed（适用于 x 和 y）。仅支持非重叠 4D 和 5D 张量。使用 NCHW 布局，性能更优。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

normDesc

输入。已初始化的 LRN 参数描述符的句柄。

lrnMode

输入。LRN 层操作模式。目前仅支持实现 MCDNN_LRN_CROSS_CHANNEL_DIM1。归一化在张量的 dimA[1] 维中执行。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将层输出值与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

yDesc, y

输入。设备内存中用于层的 y 数据的张量描述符和指针。

dyDesc, dy

输入。设备内存中的张量描述符和指针，用于层的 dy 输入累积损失差分数据（包括错误反向传播）。

xDesc, x

输入。设备内存中用于层的 x 数据的张量描述符和指针。请注意，在反向传播过程中不会修改这些值。

dxDesc, dx

输出。设备内存中的张量描述符和指针，用于层的 dx 累积损失差分数据结果（包括错误反向传播）。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- x 和 y 中任一张量指针为 NULL。
- 输入张量维数小于或等于 2。
- LRN 描述符参数超出其有效范围。
- 任一张量参数为 5D，但不是 NCDHW DHW-packed 格式。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。不受支持的配置示例如下：

- 任意输入张量数据类型都与输出张量数据类型不同。
- x，y，dx 或 dy 的任何成对（pairwise）张量维度不匹配。
- 任意张量参数步幅均为负的。

3.1.1.15 mcdnnNormalizationBackward()

此函数执行模式指定的反向归一化层计算。对于 per-channel 归一化层，可参见 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift。

```
mcdnnStatus_t
mcdnnNormalizationBackward(mcdnnHandle_t handle,
mcdnnNormMode_t                 mode,
mcdnnNormOps_t                  normOps,
mcdnnNormAlgo_t                 algo,
const void                      *alphaDataDiff,
const void                      *betaDataDiff,
const void                      *alphaParamDiff,
const void                      *betaParamDiff,
const mcdnnTensorDescriptor_t   xDesc,
const void                      *xData,
const mcdnnTensorDescriptor_t   yDesc,
const void                      *yData,
const mcdnnTensorDescriptor_t   dyDesc,
const void                      *dyData,
const mcdnnTensorDescriptor_t   dzDesc,
void                            *dzData,
const mcdnnTensorDescriptor_t   dxDesc,
void                            *dxData,
const mcdnnTensorDescriptor_t   dNormScaleBiasDesc,
const void                      *normScaleData,
const void                      *normBiasData,
void                            *dNormScaleData,
void                            *dNormBiasData,
double                          epsilon,
const mcdnnTensorDescriptor_t   normMeanVarDesc,
const void                      *savedMean,
const void                      *savedInvVariance,
mcdnnActivationDescriptor_t     activationDesc,
void                            *workSpace,
size_t                          workSpaceSizeInBytes,
void                            *reserveSpace,
size_t                          reserveSpaceSizeInBytes,
int                             groupCnt)
```

仅支持 4D 和 5D 张量。

在训练，反向传播和推理过程中的 epsilon 值必须相同。此 workspace 不需要清理。此外，workspace 不必在正向和反向传递之间保持不变，因为它不用于传递任何信息。

此函数可以接受指向 GPU workspace 的 workspace 指针，以及来自用户的 workSpaceSizeInBytes (workspace 大小)。

normOps 输入可用于将此函数设置为仅执行归一化，或先执行归一化再执行激活，或先执行归一化再执行元素级加法，然后执行激活。

张量布局为 NCHW 时，若 x、dy 和 dx 都使用 HW-packed 张量时，可以获得更高的性能。

当满足以下条件时，可以获得 MCDNN_NORM_PER_CHANNEL 模式的更高性能：

- 所有张量，即 x、y、dz、dy 和 dx，必须是 NHWC 完全压缩，并且类型必须是 MCDNN_DATA_HALF。
- C 张量维度应为 4 的倍数。
- 输入参数模式必须设置为 MCDNN_NORM_PER_CHANNEL。
- 输入参数算法必须设置为 MCDNN_NORM_ALGO_PERSIST。

- workspace 不是 NULL。
- WorkSpaceSizeInBytes 等于或大于 mcdnnGetNormalizationBackwardWorkspaceSize() 所需的量。
- ReserveSpaceSizeInBytes 等于或大于 mcdnnGetNormalizationTrainingReserveSpaceSize() 所需的量。
- 必须保留使用 mcdnnNormalizationForwardTraining() 存储在 reserveSpace 中的内容。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式（per-channel 或 per-activation）。更多信息，参见 mcdnnNormMode_t。

normOps

输入。post-operative 模式。目前，仅在 NHWC 布局中支持 MCDNN_NORM_OPS_NORM_ACTIVATION 和 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION。更多信息，参见 mcdnnNormOps_t。此输入可用于将此函数设置为仅执行归一化，或先执行归一化再执行激活，或先执行归一化再执行元素级加法，然后执行激活。

algo

输入。要执行的算法。更多信息，参见 mcdnnNormAlgo_t。

alphaDataDiff, betaDataDiff

输入。指向缩放系数（主机内存中）的指针，用于将 dx 梯度输出与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

alphaParamDiff, betaParamDiff

输入。指向缩放系数（主机内存中）的指针，用于将 dNormScaleData 和 dNormBiasData 梯度输出与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

xDesc, xData, yDesc, yData, dyDesc, dyData

输入。用于层的 x 数据，反向传播梯度输入 dy，原始正向输出 y 数据的张量描述符和指针，位于设备内存中。如果 normOps 设置为 MCDNN_NORM_OPS_NORM，则不需要 yDesc 和 yData，用户可以传入 NULL。更多信息，参见 mcdnnTensorDescriptor_t。

dzDesc, dxDesc

输入。用于计算梯度输出 dz 和 dx 的张量描述符和指针，位于设备内存中。当 normOps 为 MCDNN_NORM_OPS_NORM 或 MCDNN_NORM_OPS_NORM_ACTIVATION 时，不需要 dzDesc，用户可以传入 NULL。更多信息，参见 mcdnnTensorDescriptor_t。

dzData, dxData

输出。用于计算梯度输出 dz 和 dx 的张量描述符和指针，位于设备内存中。当 normOps 为 MCDNN_NORM_OPS_NORM 或 MCDNN_NORM_OPS_NORM_ACTIVATION 时，不需要 dzData，用户可以传入 NULL。更多信息，参见 mcdnnTensorDescriptor_t。

dNormScaleBiasDesc

输入。以下 4 个张量的共享张量描述符：normScaleData、normBiasData、dNormScaleData、dNormBiasData。此张量描述符的维度取决于归一化模式。

注解:

- 对于 FP16 和 FP32 输入张量，此张量描述符的数据类型必须为 float。
- 对于 FP64 输入张量，必须为 double。

normScaleData

输入。设备内存中指向归一化 scale 参数的指针。(在 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift 中，数量 scale 称为 gamma)。

normBiasData

输入。设备内存中指向归一化 bias 参数的指针。(在 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift 中，bias 称为 beta)。此参数仅在执行激活时使用。

dNormScaleData, dNormBiasData

输出。设备内存中分别指向 normScaleData 和 normBiasData 梯度的指针。

epsilon

输入。归一化公式中使用的 Epsilon 值。其值应等于或大于零。在正向和反向函数中应使用相同的 Epsilon 值。

normMeanVarDesc

输入。以下张量的共享张量描述符：savedMean 和 savedInvVariance。此张量描述符的维度取决于归一化模式。

注解： 对于 FP16 和 FP32 输入张量，此张量描述符的数据类型必须为 float；对于 FP64 输入张量，必须为 double。

savedMean, savedInvVariance

输入。可选的 cache 参数，包含在正向传递过程中计算并保存的中间结果。为保证工作正常，在调用此反向函数之前，层的 x 和 normScaleData、normBiasData 数据必须保持不变。请注意这两个参数可以为 NULL，但只能同时为 NULL。建议使用此 cache，因为内存开销相对较小。

activationDesc

输入。激活操作的描述符。

当 normOps 输入设置为 MCDNN_NORM_OPS_NORM_ACTIVATION 或 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION 时，将使用此激活，否则用户可以传入 NULL。

workspace

输入。指向 GPU 工作空间的指针。

workSpaceSizeInBytes

输入。workspace 的大小。它必须足够大，才能通过此函数触发快速 NHWC 半持久内核。

reserveSpace

输入。指针，指向用作 reserveSpace 的 GPU 工作空间。

reserveSpaceSizeInBytes

输入。reserveSpace 的大小。它必须等于或大于 mcdnnGetNormalizationTrainingReserveSpaceSize() 所需的量。

groupCnt

输入。分组卷积的数量。当前仅支持 1。

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- alphaDataDiff、betaDataDiff、alphaParamDiff、betaParamDiff、xData、dyData、dxData、normScaleData、dNormScaleData 和 dNormBiasData 中任意指针为 NULL。
- xDesc、yDesc 或 dxDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。
- per-channel 模式下，4D 张量中 dNormScaleBiasDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- savedMean 和 savedInvVariance 指针中只有一个为 NULL。
- epsilon 的值小于 0。
- 任意一对 xDesc、dyDesc、dxDesc、dNormScaleBiasDesc 或 normMeanVarDesc 的维度或数据类型不匹配。

3.1.1.16 mcdnnNormalizationForwardTraining()

此函数在训练阶段执行正向归一化层计算。根据模式，将执行不同的归一化操作。对于 Per-channel 层，可参见 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift。

```
mcdnnStatus_t
mcdnnNormalizationForwardTraining(mcdnnHandle_t handle,
mcdnnNormMode_t                 mode,
mcdnnNormOps_t                  normOps,
mcdnnNormAlgo_t                 algo,
const void                      *alpha,
const void                      *beta,
const mcdnnTensorDescriptor_t   xDesc,
const void                      *xData,
const mcdnnTensorDescriptor_t   normScaleBiasDesc,
const void                      *normScale,
const void                      *normBias,
double                          exponentialAverageFactor,
const mcdnnTensorDescriptor_t   normMeanVarDesc,
void                            *resultRunningMean,
void                            *resultRunningVariance,
double                          epsilon,
void                            *resultSaveMean,
void                            *resultSaveInvVariance,
mcdnnActivationDescriptor_t     activationDesc,
const mcdnnTensorDescriptor_t   zDesc,
const void                      *zData,
const mcdnnTensorDescriptor_t   yDesc,
void                            *yData,
void                            *workspace,
size_t                          workSpaceSizeInBytes,
void                            *reserveSpace,
size_t                          reserveSpaceSizeInBytes,
int                             groupCnt);
```

仅支持 4D 和 5D 张量。

在训练，反向传播和推理过程中的 epsilon 值必须相同。

有关推理阶段的信息，请参见 mcdnnNormalizationForwardInference()。

当 x 和 y 都使用 HW-packed 张量时，可以获得更高的性能。

当满足以下条件时，此 API 将触发新的半持久 NHWC 内核：

- 所有张量，即 xData 和 yData，必须是 NHWC 完全压缩，并且类型必须是 MCDNN_DATA_HALF。
- C 张量维度应为 4 的倍数。
- 输入参数模式必须设置为 MCDNN_NORM_PER_CHANNEL。
- 输入参数算法必须设置为 MCDNN_NORM_ALGO_PERSIST。
- workspace 不是 NULL。
- WorkSpaceSizeInBytes 等于或大于 mcdnnGetNormalizationForwardTrainingWorkspaceSize() 所需的量。
- ReserveSpaceSizeInBytes 等于或大于 mcdnnGetNormalizationTrainingReserveSpaceSize() 所需的量。
- 必须保留使用 mcdnnNormalizationForwardTraining() 存储在 reserveSpace 中的内容。

此 workspace 不需要清理。此外，workspace 不必在正向和反向传递之间保持不变，因为它不用于传递任何信息。此扩展函数可以接受指向 GPU workspace 的 workspace 指针，以及来自用户的 workSpaceSizeInBytes（workspace 大小）。

normOps 输入可用于将此函数设置为仅执行归一化，或先执行归一化再执行激活，或先执行归一化再执行元素级加法，然后执行激活。

仅支持 4D 和 5D 张量。在训练，反向传播和推理过程中的 epsilon 值必须相同。

张量布局为 NCHW 时，若 xData 和 yData 都使用 HW-packed 张量时，可以获得更高的性能。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。更多信息，参见 mcdnnHandle_t。

mode

输入。操作模式（per-channel 或 per-activation）。更多信息，参见 mcdnnNormMode_t。

normOps

输入。post-operative 模式。目前，仅在 NHWC 布局中支持 MCDNN_NORM_OPS_NORM_ACTIVATION 和 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION。更多信息，参见 mcdnnNormOps_t。此输入可用于将此函数设置为仅执行归一化，或先执行归一化再执行激活，或先执行归一化再执行元素级加法，然后执行激活。

algo

输入。要执行的算法。更多信息，参见 mcdnnNormAlgo_t。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将层输出值与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

xDesc, yDesc

输入。已初始化的张量描述符的句柄。

xData

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存，用于层的 x 输入数据。

yData

输出。数据指针，指向与张量描述符 yDesc 关联的 GPU 内存，用于归一化层的 y 输出。

zDesc, zData

输入。在激活之前，设备内存中用于归一化运算结果残差相加的张量描述符和指针。zDesc 和 zData 是可选的，仅当 normOps 为 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION 时使用，否则用户可以传入 NULL。使用时，z 的维度应与 xData 和最终输出 y 的维度完全相同。更多信息，参见 mcdnnTensorDescriptor_t。

normScaleBiasDesc, normScale, normBias

输入。设备内存中用于归一化 scale 和 bias 参数的张量描述符和指针。此张量描述符的维度取决于归一化模式。

exponentialAverageFactor

输入。移动平均（Moving Average，MA）计算中使用的系数如下：

runningMean = runningMean*(1-factor) + newMean*factor

在对函数的第 N 次调用中使用 factor=1/(1+n) 可进行累积移动平均（Cumulative Moving Average，CMA）计算，例如：

CMA[n] = (x[1]+⋯+x[n])/n

例如：

CMA[n+1] = (n*CMA[n]+x[n+1])/(n+1)

= ((n+1)*CMA[n]-CMA[n])/(n+1) + x[n+1]/(n+1)

= CMA[n]*(1-1/(n+1))+x[n+1]*1/(n+1)

= CMA[n]*(1-factor) + x(n+1)*factor

normMeanVarDesc

输入。用于以下张量的张量描述符：resultRunningMean, resultRunningVariance, result-SaveMean, resultSaveInvVariance。

resultRunningMean, resultRunningVariance

输入/输出。指向移动均值和移动方差数据的指针。这两个指针可以为 NULL，但只能同时为 NULL。存储在 resultRunningVariance 中（或作为推理模式中的输入被传入）的值是样本方差，是方差 [x] 的移动平均值，其中方差是根据模式在 batch 或 spatial+batch 维度上进行计算的。如果这些指针不为 NULL，则张量应初始化为一些合理的值或 0。

epsilon

输入。归一化公式中使用的 Epsilon 值。其值应等于或大于零。

resultSaveMean, resultSaveInvVariance

输出。可选的 cache 参数，包含在正向传递过程中计算并保存的中间结果。为保证工作正常，在调用此反向函数之前，层的 x 和 normScale、normBias 数据必须保持不变。请注意这两个参数可以为 NULL，但只能同时为 NULL。建议使用此 cache，因为内存开销相对较小。

activationDesc

输入。激活操作的张量描述符。

当 normOps 输入设置为 MCDNN_NORM_OPS_NORM_ACTIVATION 或 MCDNN_NORM_OPS_NORM_ADD_ACTIVATION 时，将使用此激活，否则用户可以传入 NULL。

workspace, workSpaceSizeInBytes

输入。workspace 是指向 GPU 工作空间的指针，workSpaceSizeInBytes 是 workspace 的大小。当 workspace 不为 NULL 且 workSpaceSizeInBytes 足够大，张量布局为 NHWC 且数据类型配置受支持时，此函数将触发一个半持久 NHWC 内核以进行归一化操作。此 workspace 不需要清理。此外，workspace 不需要在正向和反向传递之间保持不变。

reserveSpace

输入。指针，指向用作 reserveSpace 的 GPU 工作空间。

reserveSpaceSizeInBytes

输入。reserveSpace 的大小。必须等于或大于 mcdnnGetNormalizationTrainingReserveSpaceSize() 所需的量。

groupCnt

输入。分组卷积数。当前仅支持 1。

支持的 mcdnnNormalizationForwardTraining() 配置

数据类型配置	wDesc，dyDesc，zDesc 数据类型	normScaleBiasDesc，normMeanVarDesc 数据类型
PSEUDO _HALF_CONFIG	MCDNN_DATA_HALF	MCDNN_DATA_FLOAT
PSEUDO _BFLOAT16_CONFIG	MCDNN_DATA_BFLOAT16	MCDNN_DATA_FLOAT
FLOAT_CONFIG	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT
DOUBLE_CONFIG	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE

返回值

MCDNN_STATUS_SUCCESS

计算已成功执行。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- alpha、beta、xData、yData、normScale 和 normBias 中任一指针为 NULL。
- xDesc 或 yDesc 张量描述符维数不在 [4,5] 范围内（仅支持 4D 和 5D 张量）。
- per-channel 模式下，4D 张量中 normScaleBiasDesc 的维度不是 1xCx1x1，5D 张量中不是 1xCx1x1x1；per-activation 模式下，4D 张量中不是 1xCxHxW，5D 张量中不是 1xCxDxHxW。
- resultSaveMean 和 resultSaveInvVariance 指针中只有一个为 NULL。
- resultRunningMean 和 resultRunningInvVariance 指针中只有一个为 NULL。
- epsilon 的值小于 0。
- xDesc 或 yDesc 的维度或数据类型不匹配。

3.1.1.17 mcdnnOpsTrainVersionCheck()

此函数检查库的 OpsTrain 子集的版本是否与其他子库一致。

```
mcdnnStatus_t mcdnnOpsTrainVersionCheck(void)
```

返回值

MCDNN_STATUS_SUCCESS

版本与其他子库一致。

MCDNN_STATUS_VERSION_MISMATCH

OpsTrain 的版本与其他子库不一致。用户应检查并确保所有子组件安装版本一致。

3.1.1.18 mcdnnPoolingBackward()

该函数计算池化操作的梯度。

```
mcdnnStatus_t mcdnnPoolingBackward(
mcdnnHandle_t                    handle,
const mcdnnPoolingDescriptor_t   poolingDesc,
const void                       *alpha,
const mcdnnTensorDescriptor_t    yDesc,
const void                       *y,
const mcdnnTensorDescriptor_t    dyDesc,
const void                       *dy,
const mcdnnTensorDescriptor_t    xDesc,
const void                       *xData,
const void                       *beta,
const mcdnnTensorDescriptor_t    dxDesc,
void                             *dx)
```

注解: 此函数中的任意张量描述符参数都不支持张量向量化。使用 HW-packed 张量，性能会最佳。只支持 2 个和 3 个空间维度。

mcdnnPoolingBackward() 允许平均池化的 x 和 y 数据指针（以及相关的张量描述符句柄）为 NULL。这可以节省内存空间和带宽。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

poolingDesc

输入。已初始化的池化描述符的句柄。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将计算结果与输出层中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

yDesc

输入。已初始化的输入张量描述符的句柄。平均池化中可以为 NULL。

y

输入。数据指针，指向与张量描述符 yDesc 关联的 GPU 内存。平均池化中可以为 NULL。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。类型必须为 FLOAT、DOUBLE、HALF 或 BFLOAT16。更多信息，参见 mcdnnDataType_t。

dy

输入。数据指针，指向与张量描述符 dyDesc 关联的 GPU 内存。

xDesc

输入。已初始化的输出张量描述符的句柄。平均池化中可以为 NULL。

x

输入。数据指针，指向与输出张量描述符 xDesc 关联的 GPU 内存。平均池化中可以为 NULL。

dxDesc

输入。已初始化的输出差分张量描述符的句柄。类型必须为 FLOAT、DOUBLE、HALF 或 BFLOAT16。更多信息，参见 mcdnnDataType_t。

dx

输出。数据指针，指向与输出张量描述符 dxDesc 关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- yDesc 和 dyDesc 张量的 n，c，h，w 维度不同。
- yDesc 和 dyDesc 张量的 nStride、cStride、hStride 和 wStride 步幅不同。
- xDesc 和 dxDesc 张量的 n，c，h，w 维度不同。
- xDesc 和 dxDesc 张量的 nStride、cStride、hStride 和 wStride 步幅不同。
- 4 个张量的 datatype 不一样。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。不受支持的配置示例如下：

- 输入张量或输出张量的 wStride 不是 1。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

3.1.1.19 mcdnnSoftmaxBackward()

此函数计算 softmax 函数的梯度。

```
mcdnnStatus_t mcdnnSoftmaxBackward(
mcdnnHandle_t                   handle,
mcdnnSoftmaxAlgorithm_t         algorithm,
mcdnnSoftmaxMode_t              mode,
const void                      *alpha,
const mcdnnTensorDescriptor_t   yDesc,
const void                      *yData,
const mcdnnTensorDescriptor_t   dyDesc,
const void                      *dy,
const void                      *beta,
const mcdnnTensorDescriptor_t   dxDesc,
void                            *dx)
```

该函数支持就地操作（In-place Operation）；这意味着，dy 和 dx 指针可以一样。但是，这要求 dyDesc 描述符和 dxDesc 描述符必须相同（特别是输入和输出的步幅必须匹配，才能支持就地操作）。所有张量格式都支持 4D 和 5D 张量的所有模式和算法。NCHW 完全压缩格式（fully packed）张量的性能最佳。超过 5 个维度的张量，必须在其空间维度中进行压缩。

数据类型

此函数支持以下数据类型：

- MCDNN_DATA_FLOAT
- MCDNN_DATA_DOUBLE
- MCDNN_DATA_HALF

- MCDNN_DATA_BFLOAT16

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

algorithm

输入。指定 softmax 算法的枚举。

mode

输入。指定 softmax 模式的枚举。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将计算结果与输出层中的先验值混合，如下所示：dstValue = alpha[0]*result + beta[0]*priorDstValue。

yDesc

输入。已初始化的输入张量描述符的句柄。

y

输入。数据指针，指向与张量描述符 yDesc 关联的 GPU 内存。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

dy

输入。数据指针，指向与张量描述符 dyDesc 关联的 GPU 内存。

dxDesc

输入。已初始化的输出差分张量描述符的句柄。

dx

输出。数据指针，指向与输出张量描述符 dxDesc 关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- yDesc、dyDesc 和 dxDesc 张量的 n，c，h，w 维度不同。
- yDesc 和 dyDesc 张量的 nStride、cStride、hStride 和 wStride 步幅不同。
- 3 个张量的 datatype 不一样。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

3.1.1.20 mcdnnSpatialTfGridGeneratorBackward()

该函数计算网格生成（Grid Generation）操作的梯度。

```
mcdnnStatus_t mcdnnSpatialTfGridGeneratorBackward(
mcdnnHandle_t                                handle,
const mcdnnSpatialTransformerDescriptor_t    stDesc,
const void                                   *dgrid,
void                                         *dtheta)
```

仅支持 2D 转换。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

stDesc

输入。已创建的空间转换描述符对象。

dgrid

输入。数据指针，指向包含输入差分数据的 GPU 内存。

dtheta

输出。数据指针，指向包含输出差分数据的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

调用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- handle 为 NULL。
- dgrid 和 dtheta 中任一参数为 NULL。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。不受支持的配置示例如下：

- stDesc 中指定的转换张量的维度 > 4。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

3.1.1.21 mcdnnSpatialTfSamplerBackward()

该函数计算采样（Sampling）操作的梯度。

```
mcdnnStatus_t mcdnnSpatialTfSamplerBackward(
mcdnnHandle_t                                handle,
const mcdnnSpatialTransformerDescriptor_t    stDesc,
const void                                   *alpha,
const mcdnnTensorDescriptor_t                xDesc,
const void                                   *x,
const void                                   *beta,
const mcdnnTensorDescriptor_t                dxDesc,
void                                         *dx,
```

(下页继续)

(续上页)

```
const void                                  *alphaDgrid,
const mcdnnTensorDescriptor_t               dyDesc,
const void                                  *dy,
const void                                  *grid,
const void                                  *betaDgrid,
void                                        *dgrid)
```

仅支持 2d 转换。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

stDesc

输入。已创建的空间转换描述符对象。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将源数值与目标张量中的先验值混合，如下所示：dstValue = alpha[0]*srcValue + beta[0]*priorDstValue。

xDesc

输入。已初始化的输入张量描述符的句柄。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存。

dxDesc

输入。已初始化的输出差分张量描述符的句柄。

dx

输出。数据指针，指向与输出张量描述符 dxDesc 关联的 GPU 内存。

alphaDgrid, betaDgrid

输入。指向缩放系数（主机内存中）的指针，用于将 dgrid 梯度输出与目标指针中的先验值混合，如下所示：dstValue = alpha[0]*srcValue + beta[0]*priorDstValue。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

dy

输入。数据指针，指向与张量描述符 dyDesc 关联的 GPU 内存。

grid

输入。由 mcdnnSpatialTfGridGeneratorForward() 生成的坐标网格。

dgrid

输出。数据指针, 指向包含输出差分数据的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

调用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- handle 为 NULL。

- x、dx、y、dy、grid、dgrid 中任一参数为 NULL。
- dy 的维度与 stDesc 中指定的维度不同。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。不受支持的配置示例如下：

- 转换张量的维度 > 4。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

4 mcdnn_cnn_infer

此实体包含在推理时需要用到的卷积神经网络相关的所有函数。mcdnn_cnn_infer 库依赖于 mcdnn_ops_infer。有关后端数据和描述符类型，请参见 mcdnn_backend。

4.1 数据类型参考

4.1.1 不透明结构类型的指针

4.1.1.1 mcdnnConvolutionDescriptor_t

mcdnnConvolutionDescriptor_t 是指向不透明结构的指针，该结构包含卷积操作的说明。mcdnnCreateConvolutionDescriptor() 用于创建实例，且必须用 mcdnnSetConvolutionNdDescriptor() 或 mcdnnSetConvolution2dDescriptor() 来初始化此实例。

4.1.2 结构类型

4.1.2.1 mcdnnConvolutionBwdDataAlgoPerf_t

mcdnnConvolutionBwdDataAlgorithm_t 是一个结构，包含由 mcdnnFindConvolutionBackwardDataAlgorithm() 返回的执行结果或由 mcdnnGetConvolutionBackwardDataAlgorithm_v7() 返回的启发式结果（heuristic results）。

数据成员（Data Members）

mcdnnConvolutionBwdDataAlgo_t algo

运行该算法以获取相关的性能指标。

mcdnnStatus_t status

如果在调用 mcdnnConvolutionBackwardData() 或工作空间分配期间发生任何错误，此状态将表示该错误。否则，此状态将是 mcdnnConvolutionBackwardData() 的返回状态。

- **MCDNN_STATUS_ALLOC_FAILED** 如果工作空间分配期间发生任何错误或提供的工作空间不足。
- **MCDNN_STATUS_INTERNAL_ERROR** 如果计算期间或工作空间释放期间出现任何错误。
- 否则，此状态将是 mcdnnConvolutionBackwardData() 的返回状态。

float time

mcdnnConvolutionBackwardData() 的执行时间（以毫秒为单位）。

size_t memory

workspace 大小（以字节为单位）。

mcdnnDeterminism_t determinism

算法的确定性。

mcdnnMathType_t mathType

提供给算法的数学类型。

int reserved[3]

为后续属性预留的空间。

4.1.2.2 mcdnnConvolutionFwdAlgoPerf_t

mcdnnConvolutionFwdAlgoPerf_t 是一个结构，包含由 mcdnnFindConvolutionForwardAlgorithm() 返回的执行结果或由 mcdnnGetConvolutionForwardAlgorithm_v7() 返回的启发式结果。

数据成员（Data Members）

mcdnnConvolutionFwdAlgo_t algo

运行该算法以获取相关的性能指标。

mcdnnStatus_t status

如果在调用 mcdnnConvolutionForward() 或工作空间分配期间发生任何错误，此状态将表示该错误。否则，此状态将是 mcdnnConvolutionForward() 的返回状态。

- **MCDNN_STATUS_ALLOC_FAILED** 如果工作空间分配期间发生任何错误或提供的工作空间不足。
- **MCDNN_STATUS_INTERNAL_ERROR** 如果计算期间或工作空间释放期间出现任何错误。
- 否则，此状态将是 mcdnnConvolutionForward() 的返回状态。

float time

mcdnnConvolutionForward() 的执行时间（以毫秒为单位）。

size_t memory

workspace 大小（以字节为单位）。

mcdnnDeterminism_t determinism

算法的确定性。

mcdnnMathType_t mathType

提供给算法的数学类型。

int reserved[3]

为后续属性预留的空间。

4.1.3 枚举类型

4.1.3.1 mcdnnConvolutionBwdDataAlgo_t

mcdnnConvolutionBwdDataAlgo_t 是一种枚举类型，暴露可用于执行反向数据卷积操作的不同算法。

值

MCDNN_CONVOLUTION_BWD_DATA_ALGO_1

该算法将卷积表示为矩阵乘积，但实际上并未显式形成包含输入张量数据的矩阵。结果具有确定性。

MCDNN_CONVOLUTION_BWD_DATA_ALGO_FFT_TILING

该算法使用快速傅立叶变换（Fast-Fourier Transform，FFT）方法，但将输入拆分为分块（tile）。

4.1.3.2 mcdnnConvolutionBwdFilterAlgo_t

mcdnnConvolutionBwdFilterAlgo_t 是一种枚举类型，暴露可用于执行反向卷积核卷积操作的不同算法。

值

MCDNN_CONVOLUTION_BWD_FILTER_ALGO_1

该算法将卷积表示为矩阵乘积，但实际上并未显式形成包含输入张量数据的矩阵。结果具有确定性。

MCDNN_CONVOLUTION_BWD_FILTER_ALGO_FFT_TILING

该算法使用快速傅立叶变换方法计算卷积，但将输入张量拆分为分块（tile）。可能需要一个足够大的 workspace 来存储中间结果。结果具有确定性。

4.1.3.3 mcdnnConvolutionFwdAlgo_t

mcdnnConvolutionFwdAlgo_t 是一种枚举类型，暴露可用于执行正向卷积操作的不同算法。

值

MCDNN_CONVOLUTION_FWD_ALGO_IMPLICIT_PRECOMP_GEMM

该算法将卷积表示为矩阵乘积，实际上没有显式形成包含输入张量数据的矩阵，但仍需要一些内存工作空间来预计算某些索引，以促成包含输入张量数据的隐式矩阵构造。

MCDNN_CONVOLUTION_FWD_ALGO_FFT_TILING

该算法使用快速傅立叶变换（Fast-Fourier Transform，FFT）方法，但将输入拆分为分块（tile）。

4.1.3.4 mcdnnConvolutionMode_t

mcdnnConvolutionMode_t 是一种枚举类型，用于 mcdnnSetConvolution2dDescriptor() 配置卷积描述符。用于卷积的卷积核可以用两种不同的方式应用，在数学上对应于卷积或交叉相关（cross-correlation）。（交叉相关相当于卷积核旋转 180 度的卷积。）

值

MCDNN_CONVOLUTION

在此模式下，将卷积核应用于图像时，将执行卷积操作。

MCDNN_CROSS_CORRELATION

在此模式下，将卷积核应用于图像时，将执行交叉相关操作。

4.1.3.5 mcdnnReorderType_t

mcdnnReorderType_t 是一种枚举类型，用于设置卷积重排序类型。重排序类型可以由 mcdnnSetConvolutionReorderType() 设置，其状态可以由 mcdnnGetConvolutionReorderType() 读取。

```
typedef enum {
    MCDNN_DEFAULT_REORDER = 0,
    MCDNN_NO_REORDER = 1,
} mcdnnReorderType_t;
```

4.2 API References

4.2.1 API Functions

4.2.1.1 mcdnnCnnInferVersionCheck()

此函数检查库的 CnnInfer 子集的版本是否与其他子库一致。

```
mcdnnStatus_t mcdnnCnnInferVersionCheck(void)
```

返回值

MCDNN_STATUS_SUCCESS

版本与其他子库一致。

MCDNN_STATUS_VERSION_MISMATCH

CnnInfer 的版本与其他子库不一致。用户应检查并确保所有子组件安装版本一致。

4.2.1.2 mcdnnConvolutionBackwardData()

该函数计算张量 dy 的卷积数据梯度，其中 y 是 mcdnnConvolutionForward() 中正向卷积的输出。它使用指定的算法，并在输出张量 dx 中返回结果。缩放系数 alpha 和 beta 可用于缩放计算结果或 dx 张量。

```
mcdnnStatus_t mcdnnConvolutionBackwardData(
    mcdnnHandle_t                       handle,
    const void                          *alpha,
    const mcdnnFilterDescriptor_t       wDesc,
    const void                          *w,
    const mcdnnTensorDescriptor_t       dyDesc,
    const void                          *dy,
    const mcdnnConvolutionDescriptor_t  convDesc,
    mcdnnConvolutionBwdDataAlgo_t       algo,
    void                                *workSpace,
    size_t                              workSpaceSizeInByte,
    const void                          *bet,
    const mcdnnTensorDescriptor_t       dxDesc,
    void                                *dx)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将计算结果与输出层中的先验值混合，如下所示：

```
dstValue = alpha[0] * result + beta[0] * priorDstValue
```

wDesc

输入。已初始化的卷积核描述符的句柄。更多信息，参见 mcdnnFilterDescriptor_t。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

dy

输入。数据指针，指向与输入差分张量描述符 dyDesc 关联的 GPU 内存。

convDesc

输入。已初始化的卷积描述符。更多信息，参见 mcdnnConvolutionDescriptor_t。

algo

输入。指定应使用哪个反向数据卷积算法来计算结果的枚举。更多信息，参见 mcdnnConvolutionBwdDataAlgo_t。

workSpace

输入。数据指针，指向执行指定算法所需的工作空间 GPU 内存。如果特定算法不需要工作空间，则该指针可以为 nil。

workSpaceSizeInBytes

输入。指定已提供的 workSpace 大小（以字节为单位）。

dxDesc

输入。已初始化的输出张量描述符的句柄。

dx

输入/输出。数据指针，指向与 dxDesc 输出张量描述符（携带结果）关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

操作成功启动。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 以下至少有一个为 NULL：handle、dyDesc、wDesc、convDesc、dxDesc、dy、w、dx、alpha、beta
- wDesc 和 dyDesc 的维数不匹配
- wDesc 和 dxDesc 的维数不匹配
- wDesc 的维数小于 3
- wDesc、dxDesc 和 dyDesc 的数据类型不匹配。
- wDesc 和 dxDesc 每个图像（或分组卷积情况下的组）的输入特征图数不匹配。
- dyDesc 空间大小与 mcdnnGetConvolutionNdForwardOutputDim 中设定的大小不匹配

MCDNN_STATUS_NOT_SUPPORTED

需至少满足以下任一条件：

- dyDesc 或 dxDesc 具有负张量步幅

- dyDesc，wDesc 或 dxDesc 的维数不是 4 或 5
- 所选 algo 不支持提供的参数；有关每个 algo 支持的参数，参见以下详尽列表
- dyDesc 或 wDesc 表示输出通道计数，其不是组计数的倍数（如果在 convDesc 中设置了组计数）。

MCDNN_STATUS_MAPPING_ERROR

在纹理对象（texture object）创建的纹理绑定过程中发生错误，该纹理对象与卷积核数据或输入差分张量数据相关联。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

支持的 mcdnnConvolutionBackwardData() 配置

数据类型配置	wDesc，dyDesc，dxDesc 数据类型	convDesc 数据类型
TRUE_HALF_CONFIG（仅支持 true FP16 架构）	MCDNN_DATA_HALF	MCDNN_DATA_HALF
PSEUDO_ HALF_CONFIG	MCDNN_DATA_HALF	MCDNN_DATA_FLOAT
PSEUDO_ BFLOAT16_CONFIG	MCDNN_DATA_BFLOAT16	MCDNN_DATA_FLOAT
FLOAT_CONFIG	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT
DOUBLE_CONFIG	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE

支持的算法

- MCDNN_CONVOLUTION_BWD_DATA_ALGO_1 (_ALGO_1)
- MCDNN_CONVOLUTION_BWD_DATA_ALGO_FFT_TILING (_FFT_TILING)
- MCDNN_TENSOR_NCHW (_NCHW)
- MCDNN_TENSOR_NHWC (_NHWC)
- MCDNN_TENSOR_NCHW_VECT_C (_NCHW_VECT_C)

mcdnnConvolutionBackwardData() 2D 卷积支持的算法：wDesc:_NHWC

算法名称	确定性 (Yes or No)	dyDesc 支持的张量格式	dxDesc 支持的张量格式	支持的数据类型配置	重要
_ALGO_1		NHWC HWC-packed	NHWC HWC-packed	TRUE_ HALF _CONFIG PSEUDO _HALF _CONFIG PSEUDO _BFLOAT16 _CONFIG FLOAT _CONFIG	

mcdnnConvolutionBackwardData() 2D 卷积支持的算法：wDesc:_NCHW

算法名称	确定性 (Yes or No)	dyDesc 支持的张量格式	dxDesc 支持的张量格式	支持的数据类型配置	重要
_ALGO_1	Yes	NCHW CHW-packed	除 _NCHW_VECT_C 之外的所有其他格式	TRUE _HALF _CONFIG PSEUDO _HALF _CONFIG PSEUDO _BFLOAT16 _CONFIG FLOAT _CONFIG DOUBLE _CONFIG	所有维度大于 0 convDesc 组计数支持：大于 0
_FFT _TILING	Yes	NCHW CHW-packed	NCHW HW-packed	PSEUDO _HALF _CONFIG FLOAT _CONFIG 当任务可以由 1D FFT 处理时，也支持 DOUBLE _CONFIG， 即卷积核宽度或高度为 1	1 适用于所有维度 convDesc 组计数支持：大于 0 当 wDesc 卷积核维度都不为 1 时，卷积核宽度和高度不得大于 32 当 wDesc 卷积核任一维度为 1 时，卷积核最大维度不应超过 256 当卷积核宽度或高度为 1 时，convDesc 垂直与水平卷积核步幅必须为 1；否则，步幅可以为 1 或 2 wDesc 卷积核高度必须大于 convDesc 零填充高度 wDesc 卷积核宽度必须大于 convDesc 零填充宽度

mcdnnConvolutionBackwardData() 3D 卷积支持的算法：wDesc:_NCHW

算法名称	确定性（Yes or No）	dyDesc 支持的张量格式	dxDesc 支持的张量格式	支持的数据类型配置	重要
_ALGO_1	Yes	NCDHW CDHW-packed	NCDHW CDHW-packed	TRUE _HALF _CONFIG PSEUDO _BFLOAT16 _CONFIG PSEUDO _HALF _CONFIG FLOAT_CO NFIGDOUBLE _CONFIG	1 适用于所有维度 convDesc 组计数支持：大于 0

下页继续

表 4.2 – 续上页

算法名称	确定性（Yes or No）	dyDesc 支持的张量格式	dxDesc 支持的张量格式	支持的数据类型配置	重要
_FFT _TILING	Yes	NCDHW CDHW-packed	NCDHW DHW-packed	PSEUDO _HALF _CONFIG FLOAT _CONFIG DOUBLE _CONFIG	1 适用于所有维度 convDesc 组计数支持：大于 0 wDesc 卷积核高度必须小于等于 16 wDesc 卷积核宽度必须小于等于 16 wDesc 卷积核深度必须小于等于 16 convDesc 所有卷积核步幅为 1 wDesc 卷积核高度必须大于 convDesc 零填充高度 wDesc 卷积核宽度必须大于 convDesc 零填充宽度 wDesc 卷积核深度必须大于 convDesc 零填充深度

mcdnnConvolutionBackwardData() 3D 卷积支持的算法：wDesc:_NHWC

算法名称	确定性（Yes or No）	dyDesc 支持的张量格式	dxDesc 支持的张量格式	支持的数据类型配置	重要
_ALGO_1	Yes	NDHWC DHWC-packed	NDHWC DHWC-packed	TRUE _HALF _CONFIG PSEUDO _HALF _CONFIG PESUDO _BFLOAT16 _CONFIG FLOAT _CONFIG	所有维度大于 0 convDesc 组计数支持：大于 0

4.2.1.3 mcdnnConvolutionBiasActivationForward()

该函数对 mcdnnConvolutionForward() 的卷积或交叉相关操作，应用 bias 然后再应用激活，以 y 返回结果。完整计算遵循公式：y = act (alpha1 * conv(x) + alpha2 * z + bias)。

```
mcdnnStatus_t mcdnnConvolutionBiasActivationForward(
    mcdnnHandle_t                       handle,
    const void                          *alpha1,
    const mcdnnTensorDescriptor_t       xDesc,
    const void                          *x,
    const mcdnnFilterDescriptor_t       wDesc,
    const void                          *w,
    const mcdnnConvolutionDescriptor_t  convDesc,
    mcdnnConvolutionFwdAlgo_t           algo,
    void                                *workSpace,
    size_t                              workSpaceSizeInBytes,
    const void                          *alpha2,
    const mcdnnTensorDescriptor_t       zDesc,
```

(下页继续)

(续上页)

```
    const void                          *z,
    const mcdnnTensorDescriptor_t       biasDesc,
    const void                          *bias,
    const mcdnnActivationDescriptor_t   activationDesc,
    const mcdnnTensorDescriptor_t       yDesc,
    void                                *y)
```

mcdnnGetConvolution2dForwardOutputDim() 或 mcdnnGetConvolutionNdForwardOutputDim() 函数可用于确定输出张量描述符 yDesc 的正确维度，yDesc 是关于 xDesc、convDesc 和 wDesc 的输出张量描述符。仅 MCDNN_CONVOLUTION_FWD_ALGO_IMPLICIT_PRECOMP_GEMM 算法可以与 MCDNN_ACTIVATION_IDENTITY 一起启用。换言之，在 activationDesc 输入的 mcdnnActivationDescriptor_t 结构中，如果 mcdnnActivationMode_t 字段的模式设置为枚举值 MCDNN_ACTIVATION_IDENTITY，则 mcdnnConvolutionBiasActivationForward() 函数的 mcdnnConvolutionFwdAlgo_t 输入必须设置为枚举值 MCDNN_CONVOLUTION_FWD_ALGO_IMPLICIT_PRECOMP_GEMM。更多信息，参见 mcdnnSetActivationDescriptor()。设备指针 z 和 y 可能指向同一缓冲区，但是 x 不能与 z 或 y 指向同一缓冲区。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

alpha1, alpha2

输入。指向缩放系数（主机内存中）的指针，用于将卷积计算结果与 z 和 bias 混合，如下所示：

```
y = act ( alpha1 * conv(x) + alpha2 * z + bias )
```

xDesc

输入。已初始化的张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存。

wDesc

输入。已初始化的卷积核描述符的句柄。更多信息，参见 mcdnnFilterDescriptor_t。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

convDesc

输入。已初始化的卷积描述符。更多信息，参见 mcdnnConvolutionDescriptor_t。

algo

输入。指定应使用哪个卷积算法来计算结果的枚举。更多信息，参见 mcdnnConvolutionFwdAlgo_t。

workSpace

输入。数据指针，指向执行指定算法所需的工作空间 GPU 内存。如果特定算法不需要工作空间，则该指针可以 NIL。

workSpaceSizeInBytes

输入。指定已提供的 workSpace 大小（以字节为单位）。

zDesc

输入。已初始化的张量描述符的句柄。

z

输入。数据指针，指向与张量描述符 zDesc 关联的 GPU 内存。

biasDesc

输入。已初始化的张量描述符的句柄。

bias

输入。数据指针，指向与张量描述符 biasDesc 关联的 GPU 内存。

activationDesc

输入。已初始化的激活描述符的句柄。更多信息，参见 mcdnnActivationDescriptor_t。

yDesc

输入。已初始化的张量描述符的句柄。

y

输入/输出。数据指针，指向与 yDesc 张量描述符（携带卷积结果）关联的 GPU 内存。

返回值

除了 mcdnnConvolutionForward() 中列出的错误值之外，下面列出了此函数可能返回的错误值及其含义。

MCDNN_STATUS_SUCCESS

操作成功启动。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：* 以下至少有一个为 NULL：handle、xDesc、wDesc、convDesc、yDesc、zDesc、biasDesc、activationDesc、xData、wData、yData、zData、bias、alpha1、alpha2。

- xDesc、wDesc、yDesc 和 zDesc 的维数不等于 convDesc 数组长度 + 2。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。不受支持的配置示例如下：

- activationDesc 的模式不是 MCDNN_ACTIVATION_RELU 或 MCDNN_ACTIVATION_IDENTITY。
- activationDesc 的 reluNanOpt 不是 MCDNN_NOT_PROPAGATE_NAN。
- biasDesc 的第二个步幅不等于 1。
- biasDesc 的第一维不等于 1。
- biasDesc 的第二维和 filterDesc 的第一维不相等。
- biasDesc 的数据类型和 yDesc 的数据类型，没有与以下数据类型表列出的组合匹配。
- zDesc 和 destDesc 不匹配。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

mcdnnConvolutionBiasActivationForward() 支持的数据类型组合 (X = MCDNN_DATA)

x	w	conv Desc	y 和 z	bias	alpha1/ alpha2
X_ DOUBLE	X_ DOUBLE	X_ DOUBLE	X_ DOUBLE	X_ DOUBLE	X_ DOUBLE
X _FLOAT	X _FLOAT	X _FLOAT	X _FLOAT	X _FLOAT	X _FLOAT
X_ HALF	X_ HALF	X _FLOAT	X_ HALF	X_ HALF	X _FLOAT
X_BF LOAT16	X_BF LOAT16	X _FLOAT	X_BF LOAT16	X_BF LOAT16	X _FLOAT
X_ INT8	X_ INT8	X _INT32	X_ INT8	X _FLOAT	X _FLOAT
X_ INT8	X_ INT8	X _INT32	X _FLOAT	X _FLOAT	X _FLOAT
X_ INT8x4	X_ INT8x4	X _INT32	X_ INT8x4	X _FLOAT	X _FLOAT
X_ INT8x4	X_ INT8x4	X _INT32	X _FLOAT	X _FLOAT	X _FLOAT
X _UINT8	X_ INT8	X _INT32	X_ INT8	X _FLOAT	X _FLOAT
X _UINT8	X_ INT8	X _INT32	X _FLOAT	X _FLOAT	X _FLOAT
X_U INT8x4	X_ INT8x4	X _INT32	X_ INT8x4	X _FLOAT	X _FLOAT
X_U INT8x4	X_ INT8x4	X _INT32	X _FLOAT	X _FLOAT	X _FLOAT
X_I NT8x32	X_I NT8x32	X _INT32	X_I NT8x32	X _FLOAT	X _FLOAT

4.2.1.4 mcdnnConvolutionForward()

```
mcdnnStatus_t mcdnnConvolutionForward(
    mcdnnHandle_t                       handle,
    const void                          *alpha,
    const mcdnnTensorDescriptor_t       xDesc,
    const void                          *x,
    const mcdnnFilterDescriptor_t       wDesc,
    const void                          *w,
    const mcdnnConvolutionDescriptor_t convDesc,
    mcdnnConvolutionFwdAlgo_t           algo,
    void                                *workSpace,
    size_t                              workSpaceSizeInBytes,
    const void                          *beta,
    const mcdnnTensorDescriptor_t       yDesc,
    void                                *y)
```

该函数使用 w 指定的卷积核以 x 执行卷积或交叉相关操作，以 y 返回结果。缩放系数 alpha 和 beta 可分别用于缩放输入和输出张量。

注解： mcdnnGetConvolution2dForwardOutputDim() 或 mcdnnGetConvolutionNdForwardOutput-Dim() 函数可用于确定输出张量描述符 yDesc 的正确维度，yDesc 是关于 xDesc、convDesc 和 wDesc 的输出张量描述符。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将计算结果与输出层中的先验值混合，如下所示：

```
dstValue = alpha[0]*result + beta[0]*priorDstValue
```

xDesc

输入。已初始化的张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存。

wDesc

输入。已初始化的卷积核描述符的句柄。更多信息，参见 mcdnnFilterDescriptor_t。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

convDesc

输入。已初始化的卷积描述符。更多信息，参见 mcdnnConvolutionDescriptor_t。

algo

输入。指定应使用哪个卷积算法来计算结果的枚举。更多信息，参见 mcdnnConvolution-FwdAlgo_t。

workSpace

输入。数据指针，指向执行指定算法所需的工作空间 GPU 内存。如果特定算法不需要工作空间，则该指针可以为 nil。

workSpaceSizeInBytes

输入。指定已提供的 workSpace 大小（以字节为单位）。

yDesc

输入。已初始化的张量描述符的句柄。

y

输入/输出。数据指针，指向与 yDesc 张量描述符（携带卷积结果）关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

操作成功启动。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 以下至少有一个为 NULL：handle、xDesc、wDesc、convDesc、yDesc、xData、w、yData、alpha、beta
- xDesc 和 yDesc 的维数不匹配
- xDesc 和 wDesc 的维数不匹配
- xDesc 的维数小于 3
- xDesc 的维数不等于 convDesc 数组长度 + 2
- xDesc 和 wDesc 每个图像（或分组卷积情况下的组）的输入特征图数不匹配
- yDesc 或 wDesc 表示输出通道计数，其不是组计数的倍数（如果在 convDesc 中设置了组计数）。
- xDesc、wDesc 和 yDesc 的数据类型不匹配
- 对于某些空间维度，wDesc 的空间大于输入空间（包括零填充大小）

MCDNN_STATUS_NOT_SUPPORTED

需至少满足以下任一条件：

- xDesc 或 yDesc 具有负张量步幅
- xDesc，wDesc 或 yDesc 的维数不是 4 或 5

- yDesc 空间大小与 mcdnnGetConvolutionNdForwardOutputDim() 中设定的大小不匹配
- 所选 algo 不支持提供的参数；有关每个 algo 支持的参数，参见以下详尽列表

MCDNN_STATUS_MAPPING_ERROR

创建与卷积核数据关联的纹理对象时发生错误。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

mcdnnConvolutionForward() 支持的配置

数据类型配置	xDesc 和 wDesc	convDesc	yDesc
TRUE_HALF_CONFIG （仅支持 true FP16 架构）	MCDNN_ DATA_HALF	MCDNN_ DATA_HALF	MCDNN_ DATA_HALF
PSEUDO_HALF_CONFIG	MCDNN_ DATA_HALF	MCDNN_ DATA_FLOAT	MCDNN_ DATA_HALF
PSEUDO_BFLOAT16_CONFIG （仅支持 bfloat16 架构	MCDNN_ DATA_BFLOAT16	MCDNN_ DATA_FLOAT	MCDNN_ DATA_BFLOAT16
FLOAT_CONFIG	MCDNN_ DATA_FLOAT	MCDNN_ DATA_FLOAT	MCDNN_ DATA_FLOAT
DOUBLE_CONFIG	MCDNN_ DATA_DOUBLE	MCDNN_ DATA_DOUBLE	MCDNN_ DATA_DOUBLE
INT8_CONFIG （仅支持 DP4A 架构）	MCDNN_ DATA_INT8	MCDNN_ DATA_INT32	MCDNN_ DATA_INT8
INT8_ EXT_CONFIG （仅支持 DP4A 架构）	MCDNN_ DATA_INT8	MCDNN_ DATA_INT32	MCDNN_ DATA_FLOAT
INT8x4_ CONFIG（仅支持 DP4A 架构）	MCDNN_ DATA_INT8x4	MCDNN_ DATA_INT32	MCDNN_ DATA_INT8x4
INT8x4_EXT_CONFIG （仅支持 DP4A 架构）	MCDNN_ DATA_INT8x4	MCDNN_ DATA_INT32	MCDNN_ DATA_FLOAT
UINT8_CONFIG（仅支持 DP4A 架构）	xDesc: MCDNN_DATA_ UINT8 wDesc: MCDNN_DATA_ INT8	MCDNN_ DATA_INT32	MCDNN_ DATA_INT8
UINT8x4_ CONFIG （仅支持 DP4A 架构）	xDesc: MCDNN_DATA_ UINT8x4 wDesc: MCDNN_DATA_ INT8x4	MCDNN_ DATA_INT32	MCDNN_ DATA_INT8x4
UINT8_EXT_CONFIG （仅支持 DP4A 架构）	xDesc: MCDNN_DATA_ UINT8 wDesc: MCDNN_DATA_ INT8	MCDNN_ DATA_INT32	MCDNN_ DATA_FLOA
UINT8x4_ EXT_CONFIG （仅支持 DP4A 架构）	xDesc: MCDNN_ DATA_UINT8x4 wDesc: MCDNN_DATA _INT8x4	MCDNN_DATA _INT32	MCDNN_DATA _FLOAT
INT8x32_ CONFIG （仅支持 DP4A 架构）	MCDNN_ DATA_INT8x32	MCDNN_ DATA_INT32	MCDNN_ DATA_INT8x32

支持的算法

- MCDNN_CONVOLUTION_FWD_ALGO_IMPLICIT_PRECOMP_GEMM (_IMPLICIT_PRECOMP_GEMM)
- MCDNN_CONVOLUTION_FWD_ALGO_FFT_TILING (_FFT_TILING)
- MCDNN_TENSOR_NCHW (_NCHW)
- MCDNN_TENSOR_NHWC (_NHWC)
- MCDNN_TENSOR_NCHW_VECT_C (_NCHW_VECT_C)

mcdnnConvolutionForward() 2D 卷积支持的算法：wDesc: _NCHW

算法名称	xDesc 支持的张量格式	yDesc 支持的张量格式	支持的数据类型配置	重要
IMPLICIT PRECOMP_ GEMM	除 _NCHW_ VECT_C 之外的其他格式	除 _NCHW_ VECT_C 之外的其他格式	TRUE_HALF_ CONFIG PSEUDO_HALF _CONFIG PSEUDO_ BFLOAT16_ CONFIG FLOAT_ CONFIG DOUBLE_ CONFIG	1 适用于所有维度
FFT TILING			PSEUDO_HALF _CONFIG FLOAT _CONFIG 当任务可以由 1D FFT 处理时，也支持 DOUBLE_ CONFIG，即卷积核宽度或高度为 1	1 适用于所有维度 当 wDesc 卷积核维度都不为 1 时，卷积核宽度和高度不得大于 32 当 wDesc 卷积核任一维度为 1 时，卷积核最大维度不应超过 256 当卷积核宽度或高度为 1 时，convDesc 垂直与水平卷积核步幅必须为 1；否则，步幅可以为 1 或 2 wDesc 卷积核高度必须大于 convDesc 零填充高度 wDesc 卷积核宽度必须大于 convDesc 零填充宽度

mcdnnConvolutionForward() 2D 卷积支持的算法：wDesc: _NCHWC

算法名称	xDesc	yDesc	支持的数据类型配置	重要
IMPLICIT PRECOMP _GEMM	_NCHW _VECT_C	_NCHW_ VECT_C	INT8 x4_CONFIG UINT8 x4_CONFIG	1 适用于所有维度
IMPLICIT PRECOMP_ GEMM	_NCHW_ VECT_C	_NCHW_ VECT_C	INT8x 32_CONFIG	1 适用于所有维度

mcdnnConvolutionForward() 2D 卷积支持的算法：wDesc: _NHWC

算法名称	xDesc	yDesc	支持的数据类型配置	重要
_IMPLICIT _PRECOMP_ GEMM	NHWC fully-packed	NHWC fully-packed	INT8 _CONFIG INT8_EXT _CONFIG UINT8 _CONFIG UINT8_EXT _CONFIG	1 适用于所有维度 输入和输出特征图必须为 4 的倍数 在 INT8_EXT _CONFIG 或 UINT8_EXT _CONFIG 的情况下，输出特征图可以为非倍数
_IMPLICIT _PRECOMP _GEMM	NHWC HWC-packed	NHWC HWC-packed NCHW CHW-packed	TRUE_HALF_CONFIG PSEUDO_HALF_CONFIG PSEUDO _BFLOAT16_CONFIG FLOAT _CONFIG DOUBLE _CONFIG	

mcdnnConvolutionForward() 3D 卷积支持的算法：wDesc: _NCHW

算法名称	xDesc	yDesc	支持的数据类型配置	重要
_IMPLICIT _PRECOMP_ GEMM				所有维度大于 0
_FFT _TILING	NCDHW DHW-packed	NCDHW DHW-packed		1 适用于所有维度 wDesc 卷积核高度必须小于等于 16 wDesc 卷积核宽度必须小于等于 16 wDesc 卷积核深度必须小于等于 16 convDesc 所有卷积核步幅为 1 wDesc 卷积核高度必须大于 convDesc 零填充高度 wDesc 卷积核宽度必须大于 convDesc 零填充宽度 wDesc 卷积核深度必须大于 convDesc 零填充深度

mcdnnConvolutionForward() 3D 卷积支持的算法：wDesc: _NHWC

算法名称	xDesc	yDesc	支持的数据类型配置	重要
_IMPLICIT _PRECOMP _GEMM	NDHWC DHWC-packed	NDHWC DHWC-packed	PSEUDO_HALF_CONFIG PSEUDO _BFLOAT16_CONFIG FLOAT _CONFIG	所有维度大于 0

4.2.1.5 mcdnnCreateConvolutionDescriptor()

```
mcdnnStatus_t mcdnnCreateConvolutionDescriptor(
                  mcdnnConvolutionDescriptor_t *convDesc)
```

此函数通过分配保存卷积描述符不透明结构所需内存的方式，创建卷积描述符对象。更多信息，参见 mcdnnConvolutionDescriptor_t。

参数

convDesc

输出。卷积层描述符。

返回值

MCDNN_STATUS_SUCCESS

对象创建成功。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

4.2.1.6 mcdnnDestroyConvolutionDescriptor()

```
mcdnnStatus_t mcdnnDestroyConvolutionDescriptor(
                  mcdnnConvolutionDescriptor_t convDesc)
```

此函数用于销毁已创建的卷积描述符对象。

返回值

MCDNN_STATUS_SUCCESS

描述符销毁成功。

4.2.1.7 mcdnnFindConvolutionBackwardDataAlgorithm()

```
mcdnnStatus_t mcdnnFindConvolutionBackwardDataAlgorithm(
                  mcdnnHandle_t handle,
                  const mcdnnFilterDescriptor_t wDesc,
                  const mcdnnTensorDescriptor_t dyDesc,
                  const mcdnnConvolutionDescriptor_t convDesc,
                  const mcdnnTensorDescriptor_t dxDesc,
                  int requestedAlgoCount,
                  int *returnedAlgoCount,
                  mcdnnConvolutionBwdDataAlgoPerf_t *perfResults)
```

此函数尝试所有可用于 mcdnnConvolutionBackwardData() 的算法。它将尝试提供的 convDesc MathType 和 MCDNN_FMA_MATH（假设两者不同）。

注解:

- 只能使用 MCDNN_FMA_MATH 尝试没有 MCDNN_TENSOR_OP_MATH 可用性的算法，并以同样方式返回。
- 对于 MCDNN_DATA_FLOAT，MCDNN_FMA_MATH 将使用 float 类型计算，其它 MATH_TYPE 将使用 tf32 类型计算。

通过 mcMalloc() 分配内存。在用户分配的 mcdnnConvolutionBwdDataAlgoPerf_t 数组中返回性能指标。这些指标以一种有序的方式写入，其中第一个元素的计算时间最短。可用算法的总数可以通过 mcdnnGetConvolutionBackwardDataAlgorithmMaxCount() API 查询。

注解:

- 此函数是主机阻塞。
- 建议在分配层数据之前运行此函数；否则可能会由于资源使用问题而不必要地禁止某些算法选项。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

wDesc

输入。已初始化的卷积核描述符的句柄。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

convDesc

输入。已初始化的卷积描述符。

dxDesc

输入。已初始化的输出张量描述符的句柄。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 未正确分配 handle。
- 未正确分配 wDesc、dyDesc 或 dxDesc。
- wDesc，dyDesc 或 dxDesc 的维度小于 1。
- returnedCount 或 perfResults 为 nil。
- requestedCount 小于 1。

MCDNN_STATUS_ALLOC_FAILED

此函数无法分配内存来存储样本输入，卷积核和输出。

MCDNN_STATUS_INTERNAL_ERROR

遇到一些内部错误。

4.2.1.8 mcdnnFindConvolutionBackwardDataAlgorithmEx()

```
mcdnnStatus_t mcdnnFindConvolutionBackwardDataAlgorithmEx(
    mcdnnHandle_t                       handle,
    const mcdnnFilterDescriptor_t       wDesc,
    const void                          *w,
    const mcdnnTensorDescriptor_t       dyDesc,
    const void                          *dy,
    const mcdnnConvolutionDescriptor_t  convDesc,
    const mcdnnTensorDescriptor_t       dxDesc,
    void                                *dx,
    int                                 requestedAlgoCount,
    int                                 *returnedAlgoCount,
    mcdnnConvolutionBwdDataAlgoPerf_t   *perfResults,
    void                                *workSpace,
    size_t                              workSpaceSizeInBytes)
```

此函数尝试所有可用于 mcdnnConvolutionBackwardData() 的算法。它将尝试提供的 convDesc MathType 和 MCDNN_TENSOR_OP_MATH（假设两者不同）。

注解： 只能使用 MCDNN_FMA_MATH 尝试没有 MCDNN_TENSOR_OP_MATH 可用性的算法，并以同样方式返回。对于 MCDNN_DATA_FLOAT，MCDNN_FMA_MATH 将使用 float 类型计算，其它 MATH_TYPE 将使用 tf32 类型计算。

通过 mcMalloc() 分配内存。在用户分配的 mcdnnConvolutionBwdDataAlgoPerf_t 数组中返回性能指标。这些指标以一种有序的方式写入，其中第一个元素的计算时间最短。可用算法的总数可以通过 mcdnnGetConvolutionBackwardDataAlgorithmMaxCount() API 查询。

注解： 此函数是主机阻塞。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

wDesc

输入。已初始化的卷积核描述符的句柄。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

dy

输入。数据指针，指向与卷积核描述符 dyDesc 关联的 GPU 内存。

convDesc

输入。已初始化的卷积描述符。

dxDesc

输入。已初始化的输出张量描述符的句柄。

dxDesc

输入/输出。数据指针，指向与张量描述符 dxDesc 关联的 GPU 内存。此张量的内容会被任意值覆盖。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

workSpace

输入。指向 GPU 内存的数据指针，此内存是某些算法所必需的工作空间。此工作空间的大小将决定算法的可用性。nil 指针被视为 0 字节的 workSpace。

workSpaceSizeInBytes

输入。指定已提供的 workSpace 大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 未正确分配 handle。
- 未正确分配 wDesc、dyDesc 或 dxDesc。
- wDesc，dyDesc 或 dxDesc 的维度小于 1。
- w，dy 或 dx 为 nil。
- returnedCount 或 perfResults 为 nil。
- requestedCount 小于 1。

MCDNN_STATUS_INTERNAL_ERROR

需至少满足以下任一条件：

- 此函数不能用来分配必要的计时对象。
- 此函数不能用来释放必要的计时对象。
- 此函数不能用来释放样本输入，卷积核和输出。

4.2.1.9 mcdnnFindConvolutionForwardAlgorithm()

```
mcdnnStatus_t mcdnnFindConvolutionForwardAlgorithm(
                  mcdnnHandle_t                          handle,
                  const mcdnnTensorDescriptor_t          xDesc,
                  const mcdnnFilterDescriptor_t          wDesc,
                  const mcdnnConvolutionDescriptor_t     convDesc,
                  const mcdnnTensorDescriptor_t          yDesc,
                  int                                    requestedAlgoCount,
                  int                                    *returnedAlgoCount,
                  mcdnnConvolutionFwdAlgoPerf_t          *perfResults)
```

此函数尝试所有可用于 mcdnnConvolutionForward() 的算法。它将尝试提供的 convDesc MathType 和 MCDNN_TENSOR_OP_MATH（假设两者不同）。

注解: 只能使用 MCDNN_FMA_MATH 尝试没有 MCDNN_TENSOR_OP_MATH 可用性的算法，并以同样方式返回。对于 MCDNN_DATA_FLOAT，MCDNN_FMA_MATH 将使用 float 类型计算，其它 MATH_TYPE 将使用 tf32 类型计算。

通过 mcMalloc() 分配内存。在用户分配的 mcdnnConvolutionFwdAlgoPerf_t 数组中返回性能指标。这些指标以一种有序的方式写入，其中第一个元素的计算时间最短。可用算法的总数可以通过 mcdnnGetConvolutionForwardAlgorithmMaxCount() API 查询。

注解:

- 此函数是主机阻塞。
- 建议在分配层数据之前运行此函数；否则可能会由于资源使用问题而不必要地禁止某些算法选项。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

xDesc

输入。已初始化的输入张量描述符的句柄。

wDesc

输入。已初始化的卷积核描述符的句柄。

convDesc

输入。已初始化的卷积描述符。

yDesc

输入。已初始化的输出张量描述符的句柄。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 未正确分配 handle。
- 未正确分配 xDesc、wDesc 或 yDesc。
- xDesc、wDesc 或 yDesc 的维度小于 1。
- returnedCount 或 perfResults 为 nil。
- requestedCount 小于 1。

MCDNN_STATUS_ALLOC_FAILED

此函数无法分配内存来存储样本输入，卷积核和输出。

MCDNN_STATUS_INTERNAL_ERROR

需至少满足以下任一条件：

- 此函数不能用来分配必要的计时对象。
- 此函数不能用来释放必要的计时对象。
- 此函数不能用来释放样本输入，卷积核和输出。

4.2.1.10 mcdnnFindConvolutionForwardAlgorithmEx()

```
mcdnnStatus_t mcdnnFindConvolutionForwardAlgorithmEx(
      mcdnnHandle_t                       handle,
      const mcdnnTensorDescriptor_t       xDesc,
      const void                          *x,
      const mcdnnFilterDescriptor_t       wDesc,
      const void                          *w,
      const mcdnnConvolutionDescriptor_t convDesc,
      const mcdnnTensorDescriptor_t       yDesc,
      void                                *y,
      int                                 requestedAlgoCount,
      int                                 *returnedAlgoCount,
      mcdnnConvolutionFwdAlgoPerf_t       *perfResults,
      void                                *workSpace,
      size_t                              workSpaceSizeInBytes)
```

此函数尝试所有可用于 mcdnnConvolutionForward() 的算法。它将尝试提供的 convDesc MathType 和 MCDNN_TENSOR_OP_MATH（假设两者不同）。

注解： 只能使用 MCDNN_FMA_MATH 尝试没有 MCDNN_TENSOR_OP_MATH 可用性的算法，并以同样方式返回。对于 MCDNN_DATA_FLOAT，MCDNN_FMA_MATH 将使用 float 类型计算，其它 MATH_TYPE 将使用 tf32 类型计算。

通过 mcMalloc() 分配内存。在用户分配的 mcdnnConvolutionFwdAlgoPerf_t 数组中返回性能指标。这些指标以一种有序的方式写入，其中第一个元素的计算时间最短。可用算法的总数可以通过 mcdnnGetConvolutionForwardAlgorithmMaxCount() API 查询。

注解： 此函数是主机阻塞。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

xDesc

输入。已初始化的输入张量描述符的句柄。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存。

wDesc

输入。已初始化的卷积核描述符的句柄。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

convDesc

输入。已初始化的卷积描述符。

yDesc

输入。已初始化的输出张量描述符的句柄。

y

输入/输出。数据指针，指向与张量描述符 yDesc 关联的 GPU 内存。此张量的内容会被任意值覆盖。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

workSpace

输入。指向 GPU 内存的数据指针，此内存是某些算法所必需的工作空间。此工作空间的大小将决定算法的可用性。nil 指针被视为 0 字节的 workSpace。

workSpaceSizeInBytes

输入。指定已提供的 workSpace 大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 未正确分配 handle。
- 未正确分配 xDesc、wDesc 或 yDesc。
- xDesc、wDesc 或 yDesc 的维度小于 1。
- x、w 或 y 为 nil。
- returnedCount 或 perfResults 为 nil。
- requestedCount 小于 1。

MCDNN_STATUS_INTERNAL_ERROR

需至少满足以下任一条件：

- 此函数不能用来分配必要的计时对象。
- 此函数不能用来释放必要的计时对象。
- 此函数不能用来释放样本输入，卷积核和输出。

4.2.1.11 mcdnnGetConvolution2dDescriptor()

```
mcdnnStatus_t mcdnnGetConvolution2dDescriptor(
      const mcdnnConvolutionDescriptor_t convDesc,
      int                                *pad_h,
      int                                *pad_w,
      int                                *u,
      int                                *v,
      int                                *dilation_h,
      int                                *dilation_w,
      mcdnnConvolutionMode_t             *mode,
      mcdnnDataType_t                    *computeType)
```

此函数用于查询已初始化的 2D 卷积描述符对象。

参数

convDesc

输入/输出。已创建的卷积描述符的句柄。

pad_h

输出。零填充高度：隐式连接到输入图像顶部和底部的零的行数。

pad_w

输出。零填充宽度：隐式连接到输入图像的左侧和右侧的零的列数。

u

输出。垂直卷积核步幅。

v

输出。水平卷积核步幅。

dilation_h

输出。卷积核高度扩张。

dilation_w

输出。卷积核宽度扩张。

mode

输出。卷积模式。

computeType

输出。计算精度。

返回值

MCDNN_STATUS_SUCCESS

操作成功。

MCDNN_STATUS_BAD_PARAM

convDesc 参数为 nil。

4.2.1.12 mcdnnGetConvolution2dForwardOutputDim()

```
mcdnnStatus_t mcdnnGetConvolution2dForwardOutputDim(
      const mcdnnConvolutionDescriptor_t convDesc,
      const mcdnnTensorDescriptor_t      inputTensorDesc,
      const mcdnnFilterDescriptor_t      filterDesc,
```

(下页继续)

(续上页)

```
    int                                 *n,
    int                                 *c,
    int                                 *h,
    int                                 *w)
```

给定卷积描述符，输入张量描述符和卷积核描述符，此函数返回 2D 卷积的 4D 张量的维度。此函数可帮助设置输出张量，并在启动实际卷积之前分配适当的内存量。

输出图像的各维度 h 和 w 计算如下：

```
outputDim = 1 + (inputDim + 2 * pad - (((filterDim - 1) * dilation) + 1) )
↪/ convolutionStride;
```

注解： 调用 mcdnnConvolutionForward() 或 mcdnnConvolutionBackwardBias() 时，必须严格遵守此函数提供的维度。卷积函数不支持提供更小或更大的输出张量。

参数

convDesc

输入。已创建的卷积描述符的句柄。

inputTensorDesc

输入。已初始化的张量描述符的句柄。

filterDesc

输入。已初始化的卷积核描述符的句柄。

n

输出。输出图像数量。

c

输出。每个图像的输出特征图数量。

h

输出。每个输出特征图的高度。

w

输出。每个输出特征图的宽度。

返回值

MCDNN_STATUS_BAD_PARAM

一个或多个描述符未正确创建，或者 inputTensorDesc 和 filterDesc 的特征图不匹配。

MCDNN_STATUS_SUCCESS

对象设置成功。

4.2.1.13 mcdnnGetConvolutionBackwardDataAlgorithmMaxCount()

```
mcdnnStatus_t mcdnnGetConvolutionBackwardDataAlgorithmMaxCount(
                    mcdnnHandle_t handle, int *count)
```

该函数返回可从 mcdnnFindConvolutionBackwardDataAlgorithm() 和 mcdnnGetConvolutionForwardAlgorithmer_v7() 返回的算法的最大数量。这是所有算法总和加上当前设备支持的具有 Tensor Core 操作的算法总和。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

count

输出。得到的算法最大数量。

返回值

MCDNN_STATUS_SUCCESS

此函数执行成功。

MCDNN_STATUS_BAD_PARAM

未正确分配提供的 handle。

4.2.1.14 mcdnnGetConvolutionBackwardDataAlgorithm_v7()

该函数用作启发式函数，用于为给定卷积参数获取 mcdnnConvolutionBackwardData() 的最适合算法。此函数将返回按期望的（基于内部启发式）相对性能排序的所有算法（包括 MCDNN_TENSOR_OP_MATH 和 MCDNN_DEFAULT_MATH 版本的算法，其中 MCDNN_TENSOR_OP_MATH 可能可用），最快的是 perfResults 的索引 0。要全面搜索最快的算法，请使用 mcdnnFindConvolutionBackwardDataAlgorithm()。可用算法的总数可以通过 returnedAlgoCount 变量查询。

```
mcdnnStatus_t mcdnnGetConvolutionBackwardDataAlgorithm_v7(
    mcdnnHandle_t                       handle,
    const mcdnnFilterDescriptor_t       wDesc,
    const mcdnnTensorDescriptor_t       dyDesc,
    const mcdnnConvolutionDescriptor_t  convDesc,
    const mcdnnTensorDescriptor_t       dxDesc,
    const int                           requestedAlgoCount,
    int                                 *returnedAlgoCount,
    mcdnnConvolutionBwdDataAlgoPerf_t   *perfResults)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

wDesc

输入。已初始化的卷积核描述符的句柄。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

convDesc

输入。已初始化的卷积描述符。

dxDesc

输入。已初始化的输出张量描述符的句柄。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 以下任一参数为 NULL：handle、wDesc、dyDesc、convDesc、dxDesc、perfResults、returnedAlgoCount。
- 输入张量和输出张量的特征图数量不同。
- 两个张量描述符或卷积核的 dataType 不同。

requestedAlgoCount 小于或等于 0。

4.2.1.15 mcdnnGetConvolutionBackwardDataWorkspaceSize()

```
mcdnnStatus_t mcdnnGetConvolutionBackwardDataWorkspaceSize(
    mcdnnHandle_t                       handle,
    const mcdnnFilterDescriptor_t       wDesc,
    const mcdnnTensorDescriptor_t       dyDesc
    const mcdnnConvolutionDescriptor_t convDesc,
    const mcdnnTensorDescriptor_t       dxDesc,
    mcdnnConvolutionBwdDataAlgo_t       algo,
    size_t                              *sizeInBytes)
```

此函数返回用户应分配的 GPU 内存工作空间量，以便能够使用指定算法来调用 mcdnnConvolutionBackwardData() 函数。然后，分配的 workspace 将传入 mcdnnConvolutionBackwardData() 函数。指定的算法可以是调用 mcdnnGetConvolutionBackwardDataAlgorithm_v7() 的结果，也可以由用户任意选择。请注意，并非每个算法都可用于输入张量的每个配置和/或卷积描述符的每个配置。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

wDesc

输入。已初始化的卷积核描述符的句柄。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

convDesc

输入。已初始化的卷积描述符。

dxDesc

输入。已初始化的输出张量描述符的句柄。

algo

输入。指定所选卷积算法的枚举。

sizeInBytes

输出。作为工作空间所需的 GPU 内存量，以便能够使用指定的算法执行正向卷积。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 输入张量和输出张量的特征图数量不同。
- 两个张量描述符或卷积核的 dataType 不同。

MCDNN_STATUS_NOT_SUPPORTED

指定的算法不支持张量描述符，卷积核描述符和卷积描述符的组合。

4.2.1.16 mcdnnGetConvolutionForwardAlgorithmMaxCount()

```
mcdnnStatus_t mcdnnGetConvolutionForwardAlgorithmMaxCount(
                  mcdnnHandle_t handle, int *count)
```

该函数返回可从 mcdnnFindConvolutionForwardAlgorithm() 和 mcdnnGetConvolutionForwardAlgorithmer_v7() 返回的算法的最大数量。这是所有算法总和加上当前设备支持的具有 Tensor Core 操作的算法总和。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

count

输出。得到的算法最大数量。

返回值

MCDNN_STATUS_SUCCESS

此函数执行成功。

MCDNN_STATUS_BAD_PARAM

未正确分配提供的 handle。

4.2.1.17 mcdnnGetConvolutionForwardAlgorithm_v7

该函数用作启发式函数，用于为给定卷积参数获取 mcdnnConvolutionForward() 的最适合算法。此函数将返回按期望的（基于内部启发式）相对性能排序的所有算法（包括 MCDNN_TENSOR_OP_MATH 和 MCDNN_DEFAULT_MATH 版本的算法，其中 MCDNN_TENSOR_OP_MATH 可能可用），最快的是 perfResults 的索引 0。要全面搜索最快的算法，请使用 mcdnnFindConvolutionForwardAlgorithm()。可用算法的总数可以通过 returnedAlgoCount 变量查询。

```
mcdnnStatus_t mcdnnGetConvolutionForwardAlgorithm_v7(
    mcdnnHandle_t                       handle,
    const mcdnnTensorDescriptor_t       xDesc,
    const mcdnnFilterDescriptor_t       wDesc,
    const mcdnnConvolutionDescriptor_t  convDesc,
    const mcdnnTensorDescriptor_t       yDesc,
    const int                           requestedAlgoCount,
    int                                 *returnedAlgoCount,
    mcdnnConvolutionFwdAlgoPerf_t       *perfResults)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

xDesc

输入。已初始化的输入张量描述符的句柄。

wDesc

输入。已初始化的卷积卷积核描述符的句柄。

convDesc

输入。已初始化的卷积描述符。

yDesc

输入。已初始化的输出张量描述符的句柄。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 以下任一参数为 NULL：handle、xDesc、wDesc、convDesc、yDesc、perfResults、returnedAlgoCount。
- yDesc 或 wDesc 的维度与 xDesc 不同。
- xDesc，yDesc 或 wDesc 张量的数据类型不全相同。
- xDesc 和 wDesc 中的特征图数量不同。
- xDesc 张量的维度小于 3。
- requestedAlgoCount 小于或等于 0。

4.2.1.18 mcdnnGetConvolutionForwardWorkspaceSize()

```
mcdnnStatus_t mcdnnGetConvolutionForwardWorkspaceSize(
    mcdnnHandle_t                       handle,
    const mcdnnTensorDescriptor_t       xDesc,
    const mcdnnFilterDescriptor_t       wDesc,
    const mcdnnConvolutionDescriptor_t convDesc,
    const mcdnnTensorDescriptor_t       yDesc,
    mcdnnConvolutionFwdAlgo_t           algo,
    size_t                              *sizeInBytes)
```

此函数返回用户应分配的 GPU 内存工作空间量，以便能够使用指定算法来调用 mcdnnConvolutionForward() 函数。然后，分配的工作空间将传入 mcdnnConvolutionForward() 函数。指定的算法可以是调用 mcdnnGetConvolutionForwardAlgorithm_v7() 的结果，也可以由用户任意选择。请注意，并非每个算法都可用于输入张量的每个配置和/或卷积描述符的每个配置。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

xDesc

输入。已初始化的 x 张量描述符的句柄。

wDesc

输入。已初始化的卷积核描述符的句柄。

convDesc

输入。已初始化的卷积描述符。

yDesc

输入。已初始化的 y 张量描述符的句柄。

algo

输入。指定所选卷积算法的枚举。

sizeInBytes

输出。作为工作空间所需的 GPU 内存量，以便能够使用指定的算法执行正向卷积。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 以下任一参数为 NULL：handle、xDesc、wDesc、convDesc、yDesc。
- yDesc 或 wDesc 张量与 xDesc 的维度不同。
- xDesc，yDesc 或 wDesc 张量的数据不全相同类型。
- xDesc 和 wDesc 中的特征图数量不同。
- xDesc 张量的维度小于 3。

MCDNN_STATUS_NOT_SUPPORTED 指定的算法不支持张量描述符，卷积核描述符和卷积描述符的组合。

4.2.1.19 mcdnnGetConvolutionGroupCount()

```
mcdnnStatus_t mcdnnGetConvolutionGroupCount(
      mcdnnConvolutionDescriptor_t convDesc,
      int                          *groupCount);
```

此函数返回给定卷积描述符中指定的组计数。

参数

convDesc

输入。卷积层描述符。

groupCount

Output。组数。

返回值

MCDNN_STATUS_SUCCESS

组计数返回成功。

MCDNN_STATUS_BAD_PARAM

提供的卷积描述符无效。

4.2.1.20 mcdnnGetConvolutionMathType()

```
mcdnnStatus_t mcdnnGetConvolutionMathType(
mcdnnConvolutionDescriptor_t    convDesc,
mcdnnMathType_t                 *mathType)
```

返回值

MCDNN_STATUS_SUCCESS

数学类型返回成功。

MCDNN_STATUS_BAD_PARAM

提供的卷积描述符无效。

4.2.1.21 mcdnnGetConvolutionNdDescriptor()

```
mcdnnStatus_t mcdnnGetConvolutionNdDescriptor(
const mcdnnConvolutionDescriptor_t  convDesc,
int                                 arrayLengthRequested,
int                                 *arrayLength,
int                                 padA[],
int                                 filterStrideA[],
int                                 dilationA[],
mcdnnConvolutionMode_t              *mode,
mcdnnDataType_t                     *dataType)
```

参数

convDesc

输入/输出。已创建的卷积描述符的句柄。

arrayLengthRequested

输入。期望的卷积描述符的维度。也是 padA、filterStrideA、dilationA 数组能够保存结果的的最小空间。

arrayLength

输出。卷积描述符的实际维度。

padA

输出。至少包含 arrayLengthRequested 维度的数组，由提供的卷积描述符中的 padding 参数填充。

filterStrideA

输出。至少包含 arrayLengthRequested 维度的数组，由提供的卷积描述符中的 filter 步幅填充。

dilationA

输出。至少包含 arrayLengthRequested 维度的数组，由提供的卷积描述符中的 dilation 参数填充。

mode

输出。提供的描述符的卷积模式。

datatype

输出。提供的描述符的数据类型。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- convDesc 描述符为 nil。
- arrayLengthRequest 为负值。

MCDNN_STATUS_NOT_SUPPORTED

arrayLengthRequested 大于 MCDNN_DIM_MAX - 2。

4.2.1.22 mcdnnGetConvolutionNdForwardOutputDim()

给定卷积描述符，输入张量描述符和卷积核描述符，此函数返回 nbDims-2-D 卷积的 Nd 张量的维度。此函数可帮助设置输出张量，并在启动实际卷积之前分配适当的内存量。

```
mcdnnStatus_t mcdnnGetConvolutionNdForwardOutputDim(
const mcdnnConvolutionDescriptor_t  convDesc,
const mcdnnTensorDescriptor_t       inputTensorDesc,
const mcdnnFilterDescriptor_t       filterDesc,
int                                 nbDims,
int                                 tensorOuputDimA[])
```

输出张量的 (nbDims-2)-D 图像的每个维度计算如下：

```
outputDim = 1 + ( inputDim + 2*pad - (((filterDim-1)*dilation)+1) )/
↪convolutionStride;
```

注解： 调用 mcdnnConvolutionForward() 或 mcdnnConvolutionBackwardBias() 时，必须严格遵守此函数提供的维度。卷积函数不支持提供更小或更大的输出张量。

参数

convDesc

输入。已创建的卷积描述符的句柄。

inputTensorDesc

输入。已初始化的张量描述符的句柄。

filterDesc

输入。已初始化的卷积核描述符的句柄。

nbDims

输入。输出张量的维度。

tensorOuputDimA

输出。nbDims 维度数组，该数组包含此函数退出时输出张量的大小。

返回值

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 以下任一参数为 nil：convDesc、inputTensorDesc、filterDesc。
- filterDesc 卷积核描述符的维度与 inputTensorDesc 输入张量描述符的维度不同。
- 卷积描述符的维度与 inputTensorDesc-2 输入张量描述符的维度不同。
- filterDesc 卷积核描述符的特征图与 inputTensorDesc 输入张量描述符的特征图不同。
- 扩张卷积核 filterDesc 的尺寸大于输入张量的填充尺寸。
- 输出数组的 nbDims 维为负值或大于输入张量描述符 inputTensorDesc 的维度。

MCDNN_STATUS_SUCCESS

函数退出成功。

4.2.1.23 mcdnnGetConvolutionReorderType()

```
mcdnnStatus_t mcdnnGetConvolutionReorderType(
      mcdnnConvolutionDescriptor_t convDesc,
      mcdnnReorderType_t           *reorderType)
```

此函数从给定的卷积描述符检索卷积重排序类型。

参数

convDesc

输入。要从中检索重排序类型的卷积描述符。

reorderType

输出。检索到的重排序类型。更多信息，参见 mcdnnReorderType_t。

返回值

MCDNN_STATUS_SUCCESS

已成功检索重排序类型。

MCDNN_STATUS_BAD_PARAM

此函数的任一输入无效。

4.2.1.24 mcdnnGetFoldedConvBackwardDataDescriptors()

此函数计算折叠描述符的反向数据梯度。将数据描述符和卷积描述符作为输入，并计算折叠数据描述符和折叠转换（folding transform）描述符。然后可以使用它们来执行实际的折叠转换。

```
mcdnnStatus_t
mcdnnGetFoldedConvBackwardDataDescriptors(const mcdnnHandle_t handle,
const mcdnnFilterDescriptor_t       filterDesc,
const mcdnnTensorDescriptor_t       diffDesc,
const mcdnnConvolutionDescriptor_t  convDesc,
const mcdnnTensorDescriptor_t       gradDesc,
const mcdnnTensorFormat_t           transformFormat,
mcdnnFilterDescriptor_t             foldedFilterDesc,
mcdnnTensorDescriptor_t             paddedDiffDesc,
mcdnnConvolutionDescriptor_t        foldedConvDesc,
```

(下页继续)

(续上页)

```
mcdnnTensorDescriptor_t             foldedGradDesc,
mcdnnTensorTransformDescriptor_t    filterFoldTransDesc,
mcdnnTensorTransformDescriptor_t    diffPadTransDesc,
mcdnnTensorTransformDescriptor_t    gradFoldTransDesc,
mcdnnTensorTransformDescriptor_t    gradUnfoldTransDesc);
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

filterDesc

输入。折叠前的卷积核描述符。

diffDesc

输入。折叠前的 diff 描述符。

convDesc

输入。折叠前的卷积描述符。

gradDesc

输入。折叠前的梯度描述符。

transformFormat

输入。用于折叠的转换格式。

foldedFilterDesc

输出。已折叠的卷积核描述符。

paddedDiffDesc

输出。已填充的 diff 描述符。

foldedConvDesc

输出。已折叠的卷积描述符。

foldedGradDesc

输出。已折叠的梯度描述符。

filterFoldTransDesc

输出。卷积核的折叠转换描述符。

diffPadTransDesc

输出。Desc 的折叠转换描述符。

gradFoldTransDesc

输出。梯度的折叠转换描述符。

gradUnfoldTransDesc

输出。已折叠梯度的展开转换描述符。

返回值

MCDNN_STATUS_SUCCESS

已成功计算折叠描述符。

MCDNN_STATUS_BAD_PARAM

如果任意输入参数为 NULL 或如果输入张量超过 4 个维度。

MCDNN_STATUS_EXECUTION_FAILED

计算折叠描述符失败。

4.2.1.25 mcdnnIm2Col()

该函数构造执行 GEMM 卷积正向传递所需的 A 矩阵。

```
mcdnnStatus_t mcdnnIm2Col(
mcdnnHandle_t                   handle,
mcdnnTensorDescriptor_t         srcDesc,
const void                      *srcData,
mcdnnFilterDescriptor_t         filterDesc,
mcdnnConvolutionDescriptor_t    convDesc,
void                            *colBuffer)
```

A 矩阵的高度为 batch_size*y_height*y_width，宽度为 input_channels*filter_height*filter_width，其中：

- batch_size 是 srcDesc 第一维
- y_height 和 y_width 是由 mcdnnGetConvolutionNdForwardOutputDim() 计算得出
- input_channels 是 srcDesc 第二维（在 NCHW 布局中）
- filter_height 和 filter_width 是 wDesc 第三和第四维

A 矩阵以 HW 完全压缩格式存储在 GPU 内存中。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

srcDesc

输入。已初始化的张量描述符的句柄。

srcData

输入。数据指针，指向与输入张量描述符关联的 GPU 内存。

filterDesc

输入。已初始化的卷积核描述符的句柄。

convDesc

输入。已初始化的卷积描述符的句柄。

colBuffer

输出。数据指针，指向存储输出矩阵的 GPU 内存。

返回值

MCDNN_STATUS_BAD_PARAM

srcData 或 colBuffer 为 NULL。

MCDNN_STATUS_NOT_SUPPORTED

srcDesc、filterDesc、convDesc 的 dataType 为 MCDNN_DATA_INT8，MCDNN_DATA_INT8x4，MCDNN_DATA_INT8 或 MCDNN_DATA_INT8x4。convDesc 的 groupCount 大于 1。

MCDNN_STATUS_EXECUTION_FAILED

MXMACA 内核执行失败。

MCDNN_STATUS_SUCCESS

输出数据组生成成功。

4.2.1.26 mcdnnReorderFilterAndBias

mcdnnReorderFilterAndBias() 函数对数据类型为 MCDNN_DATA_INT8x32 且张量格式为 MCDNN_TENSOR_NCHW_VECT_C 的张量，进行卷积核和 bias 值的重排序。它可用于通过分离重排序操作与卷积来加速推理时间。当前仅支持 2D 卷积核（filter）。

```
mcdnnStatus_t mcdnnReorderFilterAndBias(
mcdnnHandle_t                   handle,
const mcdnnFilterDescriptor_t   filterDesc,
mcdnnReorderType_t              reorderType,
const void                      *filterData,
void                            *reorderedFilterData,
int                             reorderBias,
const void                      *biasData,
void                            *reorderedBiasData);
```

数据类型为 MCDNN_DATA_INT8x32 的卷积核和 bias 张量（也表示 MCDNN_TENSOR_NCHW_VECT_C 张量格式）需要进行输出通道轴置换，以便利用 Tensor Core IMMA 指令。当卷积描述符的 reorder 类型属性设置为 MCDNN_DEFAULT_REORDER 时，在每次 mcdnnConvolutionForward() 和 mcdnnConvolutionBiasActivationForward() 调用中完成此操作。用户可以避免重复的重排序内核调用，方法是首先使用此调用对卷积核和 bias 张量进行重排序，然后调用正向卷积 API，重排序类型设置为 MCDNN_NO_REORDER。例如，多层神经网络中的卷积可能需要对每个层的内核进行重排序，这可能占用总推理时间的很大部分。使用此函数，可以对卷积核和 bias 数据进行一次重排序。随后在多层执行卷积操作，从而加速推理时间。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

filterDesc

输入。内核数据集的描述符。

reorderType

输入。设置是否执行重排序。更多信息，参见 mcdnnReorderType_t。

filterData

输入。指针，指向设备内存中卷积核（内核）数据位置。

reorderedFilterData

输出。指针，指向通过此函数写入已重排序卷积核数据的位置（设备内存中）。此张量与 filterData 具有相同的维度。

reorderBias

输入。如果 > 0，则也对 bias 数据执行重排序。如果 <= 0，则不对 bias 数据执行重排序。

biasData

输入。指向设备内存中 bias 数据位置的指针。

reorderedBiasData

输出。指针，指向通过此函数写入已重排序 bias 数据的位置（设备内存中）。此张量与 biasData 具有相同的维度。

返回值

MCDNN_STATUS_SUCCESS

重排序成功。

MCDNN_STATUS_EXECUTION_FAILED

卷积核和 bias 数据的重新排序失败。

MCDNN_STATUS_BAD_PARAM

句柄，卷积核描述符，卷积核数据或已重排序数据为 NULL。或者，如果请求 bias 重排序（reorderBias > 0），bias 数据或已重排序的 bias 数据为 NULL。如果卷积核维度大小不是 4，也会返回此状态。

MCDNN_STATUS_NOT_SUPPORTED

卷积核描述符数据类型不是 MCDNN_DATA_INT8x32；卷积核描述符张量不是矢量化布局（MCDNN_TENSOR_NCHW_VECT_C）。

4.2.1.27 mcdnnSetConvolution2dDescriptor()

```
mcdnnStatus_t mcdnnSetConvolution2dDescriptor(
      mcdnnConvolutionDescriptor_t convDesc,
      int                          pad_h,
      int                          pad_w,
      int                          u,
      int                          v,
      int                          dilation_h,
      int                          dilation_w,
      mcdnnConvolutionMode_t       mode,
      mcdnnDataType_t              computeType)
```

此函数将已创建的卷积描述符对象初始化为二维相关（2D correlation）。此函数假定张量和卷积核描述符与正向卷积路径相对应，并检查其设置是否有效。同一卷积描述符可以在对应同一层的反向路径中重用。

参数

convDesc

输入/输出。已创建的卷积描述符的句柄。

pad_h

输入。零填充高度：隐式连接到输入图像顶部和底部的零的行数。

pad_w

输入。零填充宽度：隐式连接到输入图像的左侧和右侧的零的列数。

u

输入。垂直卷积核步幅。

v

输入。水平卷积核步幅。

dilation_h

输入。卷积核高度扩张。

dilation_w

输入。卷积核宽度扩张。

mode

输入。在 MCDNN_CONVOLUTION 和 MCDNN_CROSS_CORRELATION 之间选择。

computeType

输入。计算精度。

返回值

MCDNN_STATUS_SUCCESS

目标设置成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- convDesc 描述符为 nil。
- pad_h 或 pad_w 参数为小于 0 的负数。
- u 或 v 参数为负数或 0。
- dilation_h 或 dilation_w 参数为负数或 0。
- mode 参数具有无效的枚举值。

4.2.1.28 mcdnnSetConvolutionGroupCount()

```
mcdnnStatus_t mcdnnSetConvolutionGroupCount(
      mcdnnConvolutionDescriptor_t convDesc,
      int                          groupCount)
```

此函数允许用户指定要在关联卷积中使用的组数。

参数

groupCount

输入。组数。

convDesc

输出。卷积层描述符。

返回值

MCDNN_STATUS_SUCCESS

组数设置成功。

MCDNN_STATUS_BAD_PARAM

提供的卷积描述符无效。

4.2.1.29 mcdnnSetConvolutionMathType

此函数允许用户指定是否允许在与给定卷积描述符关联的库函数中使用 op 张量。

```
mcdnnStatus_t mcdnnSetConvolutionMathType(
mcdnnConvolutionDescriptor_t    convDesc,
mcdnnMathType_t                 mathType)
```

返回值

MCDNN_STATUS_SUCCESS

数学类型设置成功。

MCDNN_STATUS_BAD_PARAM

提供了无效的卷积描述符或指定了无效的数学类型。

4.2.1.30 mcdnnSetConvolutionNdDescriptor()

此函数将已创建的通用卷积描述符对象初始化为 Nd 相关。同一卷积描述符可以在对应同一层的反向路径中重用。卷积计算将以指定的 dataType 完成，这可能与输入/输出张量不同。

```
mcdnnStatus_t mcdnnSetConvolutionNdDescriptor(
mcdnnConvolutionDescriptor_t    convDesc,
int                             arrayLength,
const                           int padA[],
const                           int filterStrideA[],
const                           int dilationA[],
mcdnnConvolutionMode_t          mode,
mcdnnDataType_t                 dataType)
```

参数

convDesc

输入/输出。已创建的卷积描述符的句柄。

arrayLength

输入。卷积维度。

padA

输入。arrayLength 维度的数组，包含每个维度的零填充大小。对于每个维度，在该维度每个元素的起始和结尾处，隐式连接的零的数量即为填充。

filterStrideA

输入。arrayLength 维度的数组，包含每个维度的卷积核步幅。对于每个维度，到达下一个点的过滤窗口的下一个起始处，要滑动的元素数量即为卷积核步幅。

dilationA

输入。arrayLength 维度的数组，包含每个维度的扩张系数。

mode

输入。在 MCDNN_CONVOLUTION 和 MCDNN_CROSS_CORRELATION 之间选择。

datatype

输入。选择要进行计算的数据类型。

注 解: 不建议在 mcdnnSetConvolutionNdDescriptor() 中使用带有 HALF_CONVOLUTION_BWD_FILTER 的 MCDNN_DATA_HALF，因为它对任何用于训练的实际用例都无用，并且在将来的 mcDNN 版本中将被视为被阻止。建议在 mcdnnSetTensorNdDescriptor() 中将 MCDNN_DATA_HALF 用于输入张量，在 mcdnnSetConvolutionNdDescriptor() 中使用带有 HALF_CONVOLUTION_BWD_FILTER 的 MCDNN_DATA_FLOAT，并在许多通用深度学习框架中与自动混合精度（Automatic Mixed Precision，AMP）训练一起使用。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- convDesc 描述符为 nil。
- arrayLengthRequest 为负值。
- mode 具有无效的枚举值。
- datatype 具有无效的枚举值。
- padA 中任一元素为小于 0 的负数。
- strideA 中任一元素为负数或 0。
- dilationA 中任一元素为负数或 0。

MCDNN_STATUS_NOT_SUPPORTED

需至少满足以下任一条件：arrayLengthRequest 大于 MCDNN_DIM_MAX。

4.2.1.31 mcdnnSetConvolutionReorderType()

```
mcdnnStatus_t mcdnnSetConvolutionReorderType(
      mcdnnConvolutionDescriptor_t convDesc,
      mcdnnReorderType_t           reorderType)
```

此函数对给定的卷积描述符设置卷积重排序类型。

参数

convDesc

输入。要为其设置重排序类型的卷积描述符。

reorderType

输入。将重排序类型设置为此值。更多信息，参见 mcdnnReorderType_t。

返回值

MCDNN_STATUS_SUCCESS

已成功设置重排序类型。

MCDNN_STATUS_BAD_PARAM

不支持提供的重排序类型。

5 mcdnn_cnn_train

此实体包含在训练时需要用到的卷积神经网络相关的所有函数。mcdnn_cnn_train 库依赖于 mcdnn_ops_infer，mcdnn_ops_train 和 mcdnn_adv_infer。有关后端数据和描述符类型，请参见 mcdnn_backend。

5.1 数据类型参考

5.1.1 不透明结构类型的指针

5.1.1.1 mcdnnFusedOpsConstParamPack_t

mcdnnFusedOpsConstParamPack_t 是指向不透明结构的指针，该结构包含 mcdnnFusedOps 常量参数描述。使用 mcdnnCreateFusedOpsConstParamPack() 函数创建一个此结构的实例，使用 mcdnnDestroyFusedOpsConstParamPack() 函数销毁已创建的描述符。

5.1.1.2 mcdnnFusedOpsPlan_t

mcdnnFusedOpsPlan_t 是指向不透明结构的指针，该结构包含 mcdnnFusedOpsPlan 描述。此描述符包含计划信息，包括问题类型和大小，应运行哪些内核以及内部工作空间分区。使用 mcdnnCreateFusedOptionsPlan() 函数创建一个此结构的实例，使用 mcdnnDestroyFusedOptionsPlan() 函数销毁已创建的描述符。

5.1.1.3 mcdnnFusedOpsVariantParamPack_t

mcdnnFusedOpsVariantParamPack_t 是指向不透明结构的指针，该结构包含 mcdnnFusedOps 变量参数描述。使用 mcdnnCreateFusedOpsVariantParamPack() 函数创建一个此结构的实例，使用函数 mcdnnDestroyFusedOpsVariantParamPack() 销毁已创建的描述符。

5.1.2 结构类型

5.1.2.1 mcdnnConvolutionBwdFilterAlgoPerf_t

mcdnnConvolutionBwdFilterAlgoPerf_t 是一个结构，包含由 mcdnnFindConvolutionBackwardFilterAlgorithm() 返回的执行结果或由 mcdnnGetConvolutionBackwardFilterAlgorithm_v7() 返回的启发式结果。

数据成员（Data Members）

mcdnnConvolutionBwdFilterAlgo_t algo

运行该算法以获取相关的性能指标。

mcdnnStatus_t status

如果在调用 mcdnnConvolutionBackwardFilter() 或工作空间分配期间发生任何错误，此状态将表示该错误。否则，此状态将是 mcdnnConvolutionBackwardFilter() 的返回状态。

- MCDNN_STATUS_ALLOC_FAILED 如果工作空间分配期间发生任何错误或提供的工作空间不足。
- MCDNN_STATUS_INTERNAL_ERROR 如果计算期间或工作空间释放期间出现任何错误。
- 否则，此状态将是 mcdnnConvolutionBackwardFilter() 的返回状态。

float time

mcdnnConvolutionBackwardFilter() 的执行时间（以毫秒为单位）。

size_t memory

workspace 大小（以字节为单位）。

mcdnnDeterminism_t determinism

算法的确定性。

mcdnnMathType_t mathType

提供给算法的数学类型。

int reserved[3]

为后续属性预留的空间。

5.1.3 枚举类型

5.1.3.1 mcdnnFusedOps_t

mcdnnFusedOps_t 是一种枚举类型，用于选择要在融合运算中执行的特定计算序列。

成员与描述

MCDNN_FUSED_SCALE_BIAS_ACTIVATION_CONV_BNSTATS = 0

在 per-channel 基础上，它按以下顺序执行这些操作：缩放，添加 bias，激活，卷积和生成 batchnorm 统计信息。

MCDNN_FUSED_SCALE_BIAS_ACTIVATION_WGRAD = 1

在 per-channel 基础上，它按以下顺序执行这些操作：缩放，添加 bias，激活，卷积反向权重和生成 batchnorm 统计信息。

MCDNN_FUSED_BN_FINALIZE_STATISTICS_TRAINING = 2

从 ySum，ySqSum 和已学习的 scale 及 bias 计算等效 scale 和 bias。（可选）更新运行统计信息并生成已保存的统计信息。

MCDNN_FUSED_BN_FINALIZE_STATISTICS_INFERENCE = 3

从已学习的运行统计信息、scale、bias 计算等效 scale 和 bias。

MCDNN_FUSED_CONV_SCALE_BIAS_ADD_ACTIVATION = 4

在 per-channel 基础上，它按以下顺序执行这些操作：卷积，缩放，添加 bias，使用另一个张量进行元素级相加，激活。

MCDNN_FUSED_SCALE_BIAS_ADD_ACTIVATION_GEN_BITMASK = 5

在 per-channel 基础上，它按以下顺序执行这些操作：在一个张量上进行缩放和 bias，在第二个张量上进行缩放和 bias，对这两个张量进行元素级相加，在得到的张量上执行激活并生成激活位掩码。

MCDNN_FUSED_DACTIVATION_FORK_DBATCHNORM = 6

在 per-channel 基础上，它按以下顺序执行这些操作：反向激活，fork （即，写出残差分支的梯度），反向 BN。

5.1.3.2 mcdnnFusedOpsConstParamLabel_t

mcdnnFusedOpsConstParamLabel_t 是一种枚举类型，用于选择 mcdnnFusedOps 描述符的类型。更多信息，参见 mcdnnSetFusedOpsConstParamPackAttribute()。

```
typedef enum {
MCDNN_PARAM_XDESC                              = 0,
MCDNN_PARAM_XDATA_PLACEHOLDER                  = 1,
MCDNN_PARAM_BN_MODE                            = 2,
MCDNN_PARAM_BN_EQSCALEBIAS_DESC                = 3,
MCDNN_PARAM_BN_EQSCALE_PLACEHOLDER             = 4,
MCDNN_PARAM_BN_EQBIAS_PLACEHOLDER              = 5,
MCDNN_PARAM_ACTIVATION_DESC                    = 6,
MCDNN_PARAM_CONV_DESC                          = 7,
MCDNN_PARAM_WDESC                              = 8,
MCDNN_PARAM_WDATA_PLACEHOLDER                  = 9,
MCDNN_PARAM_DWDESC                             = 10,
MCDNN_PARAM_DWDATA_PLACEHOLDER                 = 11,
MCDNN_PARAM_YDESC                              = 12,
MCDNN_PARAM_YDATA_PLACEHOLDER                  = 13,
MCDNN_PARAM_DYDESC                             = 14,
MCDNN_PARAM_DYDATA_PLACEHOLDER                 = 15,
MCDNN_PARAM_YSTATS_DESC                        = 16,
MCDNN_PARAM_YSUM_PLACEHOLDER                   = 17,
MCDNN_PARAM_YSQSUM_PLACEHOLDER                 = 18,
MCDNN_PARAM_BN_SCALEBIAS_MEANVAR_DESC          = 19,
MCDNN_PARAM_BN_SCALE_PLACEHOLDER               = 20,
MCDNN_PARAM_BN_BIAS_PLACEHOLDER                = 21,
MCDNN_PARAM_BN_SAVED_MEAN_PLACEHOLDER          = 22,
MCDNN_PARAM_BN_SAVED_INVSTD_PLACEHOLDER        = 23,
MCDNN_PARAM_BN_RUNNING_MEAN_PLACEHOLDER        = 24,
MCDNN_PARAM_BN_RUNNING_VAR_PLACEHOLDER         = 25,
MCDNN_PARAM_ZDESC                              = 26,
MCDNN_PARAM_ZDATA_PLACEHOLDER                  = 27,
MCDNN_PARAM_BN_Z_EQSCALEBIAS_DESC              = 28,
MCDNN_PARAM_BN_Z_EQSCALE_PLACEHOLDER           = 29,
MCDNN_PARAM_BN_Z_EQBIAS_PLACEHOLDER            = 30,
MCDNN_PARAM_ACTIVATION_BITMASK_DESC            = 31,
MCDNN_PARAM_ACTIVATION_BITMASK_PLACEHOLDER     = 32,
MCDNN_PARAM_DXDESC                             = 33,
MCDNN_PARAM_DXDATA_PLACEHOLDER                 = 34,
MCDNN_PARAM_DZDESC                             = 35,
MCDNN_PARAM_DZDATA_PLACEHOLDER                 = 36,
MCDNN_PARAM_BN_DSCALE_PLACEHOLDER              = 37,
MCDNN_PARAM_BN_DBIAS_PLACEHOLDER               = 38,
} mcdnnFusedOpsConstParamLabel_t;
```

mcdnnFusedOpsConstParamLabel_t 表格说明

使用的简称	含义
Setter	mcdnnSetFusedOpsConstParamPackAttribute()
Getter	mcdnnGetFusedOpsConstParamPackAttribute()
X_PointerPlaceHolder_t	mcdnnFusedOpsPointerPlaceHolder_t
属性列中的 X_ 前缀	枚举数名称中的 MCDNN_PARAM_

mcdnnFusedOpsConstParamLabel_t 中的 MCDNN_FUSED_SCALE_BIAS_ACTIVATION_CONV_BNSTATS

属性	Setter 中期望传入的描述符类型	说明	默认值
X_XDESC	Setter 中，*param 应为 xDesc，一个指向已初始化的 mcdnnTensorDescriptor_t 的指针	描述 x（输入）张量大小，布局，数据类型的张量描述符	NULL
X_XDATA_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 xData 指针是否为 NULL，如果不是，则为用户允许的指针对齐	MCDNN_PTR_NULL
X_BN_MODE	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnBatchNormMode_t	描述 scale，bias，统计数据的操作模式 mcDNN 只支持 MCDNN_BATCHNORM_SPATIAL 和 MCDNN_BATCHNORM_SPATIAL_PERSISTENT，即 scale，bias，统计数据均为 per-channel 模式	MCDNN_BATCHNORM_PER_ACTIVATION
X_BN_EQSCALEBIAS_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensor Descriptor_t*	张量描述符，描述 batchNorm 等效 scale 和 bias 张量的大小，布局，数据类型 形状必须与 MCDNN_PARAM_BN_MODE 中指定的模式匹配 如果设置为 NULL，scale 和 bias 操作均会变为 NoP	NULL
X_BN_EQSCALE_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 batchNorm 等效 scale 指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则 scale 操作会变为 NoP	MCDNN_PTR_NULL
X_BN_EQBIAS_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParam Pack 中 batchNorm 等效 bias 指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则 bias 操作会变为 NoP	MCDNN_PTR_NULL

下页继续

表 5.1 – 续上页

属性	Setter 中期望传入的描述符类型	说明	默认值
X_ACTIVATION_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnActivation Descriptor_t	描述激活操作 mcDNN 只支持 MCDNN_ACTIVATION _RELU 和 MCDNN_ACTIVATION _IDENTITY 激活模式 若设为 NULL 或激活模式设为 MCDNN_ACTIVATION _IDENTITY，则 op 序列中的激活操作会变为 NoP	NULL
X_CONV_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnn Convolution Descriptor_t	描述卷积操作	NULL
X_WDESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnFilter Descriptor_t	卷积核描述符，描述 w （卷积核）张量的大小，布局，数据类型	NULL
X_WDATA_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 w (卷积核) 张量指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_NULL
X_YDESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensor Descriptor_t	张量描述符，描述 y （输出）张量大小，布局，数据类型	NULL
X_YDATA_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 y （输出）张量指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_NULL
X_YSTATS_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensor Descriptor_t	张量描述符，描述 y 的和与 y 平方和的张量大小，布局，数据类型 形状必须与 MCDNN_PARAM_BN _MODE 中指定的模式匹配 若设置为 NULL，y 统计数据生成操作会变为 NoP	NULL
X_YSUM_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 y 的和的指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN _PTR_NULL，y 统计数据生成操作会变为 NoP	MCDNN_PTR_NULL
X_YSQSUM_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParam Pack 中 y 平方和的指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN _PTR_NULL，y 统计数据生成操作会变为 NoP	MCDNN_PTR _NULL

注解:

- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_NULL，则 VariantParamPack 中的设备指针也需要为 NULL。
- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_ELEM_ALIGNED 或 MCDNN_PTR_16B_ALIGNED，则 VariantParamPack 中的设备指针可能不是 NULL，分别至少需要元素对齐或 16 字节对齐。

mcdnnFusedOpsConstParamLabel_t 完全融合快速路径（正向）的条件

参数	条件
MCDNN_PARAM_XDESC MCDNN_PARAM_XDATA_PLACEHOLDER	张量为 4 维 数据类型为 MCDNN_DATA_HALF 布局为 NHWC 完全压缩格式 对齐为 MCDNN_PTR_16B_ALIGNED 张量的 C 维为 8 的倍数
MCDNN_PARAM_BN_EQSCALEBIAS_DESC MCDNN_PARAM_BN_EQSCALE_PLACEHOLDER MCDNN_PARAM_BN_EQBIAS_PLACEHOLDER	scale 或 bias 操作不是 NoP：张量为 4 维，形状为 1xCx1x1 数据类型为 MCDNN_DATA_HALF 布局为完全压缩格式 对齐为 MCDNN_PTR_16B_ALIGNED
MCDNN_PARAM_CONV_DESC MCDNN_PARAM_WDESC MCDNN_PARAM_WDATA_PLACEHOLDER	卷积描述符的 mode 需要设为 MCDNN_CROSS_CORRELATION 卷积描述符的 dataType 需要设为 MCDNN_DATA_FLOAT 卷积描述符的 dilationA 为 (1,1) 卷积描述符的组计数需要为 1 卷积描述符的 mathType 需要设为 MCDNN_TENSOR_OP_MATH 或 MCDNN_TENSOR_OP_MATH_ALLOW_CONVERSION 卷积核布局为 NHWC 卷积核数据类型为 MCDNN_DATA_HALF 卷积核的 K 维为 32 的倍数 卷积核的 RxS 尺寸为 1x1 或 3x3 若卷积核的 RxS 尺寸为 1x1，卷积描述符的 padA 需为 (0,0)，filterStrideA 需为 (1,1) 卷积核的对齐为 MCDNN_PTR_16B_ALIGNED
MCDNN_PARAM_YDESC MCDNN_PARAM_YDATA_PLACEHOLDER	张量为 4 维 数据类型为 MCDNN_DATA_HALF 布局为 NHWC 完全压缩格式 对齐为 MCDNN_PTR_16B_ALIGNED
MCDNN_PARAM_YSTATS_DESC MCDNN_PARAM_YSUM_PLACEHOLDER MCDNN_PARAM_YSQSUM_PLACEHOLDER	统计数据生成操作不是 NoP：张量为 4 维，形状为 1xKx1x1 数据类型为 MCDNN_DATA_FLOAT 布局为完全压缩 对齐为 MCDNN_PTR_16B_ALIGNED

mcdnnFusedOpsConstParamLabel_t 中的

MCDNN_FUSED_SCALE_BIAS_ACTIVATION_WGRAD

属性	Setter 中期望传入的描述符类型	说明	默认值
X_XDESC	Setter 中，*param 应为 xDesc，一个指针，指向已初始化的 mcdnnTensorDescriptor_t	张量描述符，描述 x（输入）张量大小，布局，数据类型	NULL
X_XDATA_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 xData 指针是否为 NULL，如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL
X_BN_MODE	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnBatchNorm Mode_t	描述 scale，bias，统计数据的操作模式 mcDNN 只支持 MCDNN_BATCHNORM_SPATIAL 和 MCDNN_BATCHNORM_SPATIAL _PERSISTENT，即 scale，bias，统计数据均为 per-channel 模式	MCDNN_BATCHNORM_PER_ACTIVATION
X_BN_EQSCALEBIAS_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensorDescriptor_t	张量描述符，描述 batchNorm 等效 scale 和 bias 张量的大小，布局，数据类型 形状必须与 MCDNN_PARAM_BN _MODE 中指定的模式匹配 如果设置为 NULL，scale 和 bias 操作均会变为 NoP	NULL
X_BN_EQSCALE_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 等效 scale 指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则 scale 操作会变为 NoP	MCDNN_PTR_ NULL
X_BN_EQBIAS_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 等效 bias 指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则 bias 操作会变为 NoP	MCDNN_PTR_ NULL
X_ACTIVATION_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnActivation Descriptor_t	描述激活操作 mcDNN 只支持 MCDNN_ACTIVATION _RELU 和 MCDNN_ACTIVATION _IDENTITY 激活模式 若设为 NULL 或激活模式设为 MCDNN_ACTIVATION _IDENTITY，则 op 序列中的激活操作会变为 NoP	NULL
X_CONV_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnConvolution Descriptor_t	描述卷积操作	NULL

下页继续

表 5.3 – 续上页

属性	Setter 中期望传入的描述符类型	说明	默认值
X_DWDESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnFilterDescriptor_t	卷积核描述符，描述 dw（卷积核梯度输出）张量的大小，布局，数据类型	NULL
X_DWDATA_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 dw（卷积核梯度输出）张量指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL
X_DYDESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensorDescriptor_t	张量描述符，描述 dy（梯度输入）张量大小，布局，数据类型	NULL
X_YDATA_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 dy（梯度输入）张量指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL

注解:

- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_NULL，则 VariantParamPack 中的设备指针也需要为 NULL。
- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_ELEM_ALIGNED 或 MCDNN_PTR_16B_ALIGNED，则 VariantParamPack 中的设备指针可能不是 NULL，分别至少需要元素对齐或 16 字节对齐。

mcdnnFusedOpsConstParamLabel_t 完全融合快速路径 (反向) 的条件

参数	条件
MCDNN_PARAM_XDESC MCDNN_PARAM_XDATA_PLACEHOLDER	张量为 4 维 数据类型为 MCDNN_DATA_HALF 布局为 NHWC 完全压缩格式 对齐为 MCDNN_PTR_16B_ALIGNED 张量的 C 维为 8 的倍数
MCDNN_PARAM_BN_EQSCALEBIAS_DESC MCDNN_PARAM_BN_EQSCALE_PLACEHOLDER MCDNN_PARAM_BN_EQBIAS_PLACEHOLDER	scale 或 bias 操作不是 NoP：张量为 4 维，形状为 1xCx1x1 数据类型为 MCDNN_DATA_HALF 布局为完全压缩 对齐为 MCDNN_PTR_16B_ALIGNED

下页继续

表 5.4 – 续上页

参数	条件
MCDNN_PARAM_CONV_DESC MCDNN_PARAM_DWDESC MCDNN_PARAM_DWDATA_PLACEHOLDER	卷积描述符的 mode 需要设为 MCDNN_CROSS_CORRELATION 卷积描述符的 dataType 需要设为 MCDNN_DATA_FLOAT 卷积描述符的 dilationA 为 (1,1) 卷积描述符的组计数需要为 1 卷积描述符的 mathType 需要设为 MCDNN_TENSOR_OP_MATH 或 MCDNN_TENSOR_OP_MATH_ALLOW _CONVERSION 卷积核布局为 NHWC 卷积核数据类型为 MCDNN_DATA_HALF 卷积核的 K 维为 32 的倍数 卷积核的 RxS 尺寸为 1x1 或 3x3 若卷积核的 RxS 尺寸为 1x1，卷积描述符的 padA 需为 (0,0)，filterStrideA 需为 (1,1) 卷积核的对齐为 MCDNN_PTR_16B_ALIGNED
MCDNN_PARAM_DYDESC MCDNN_PARAM_DYDATA_PLACEHOLDER	张量为 4 维 数据类型为 MCDNN_DATA_HALF 布局为 NHWC 完全压缩格式 对齐为 MCDNN_PTR_16B_ALIGNED

mcdnnFusedOpsConstParamLabel_t 中的

MCDNN_FUSED_BN_FINALIZE_STATISTICS_TRAINING

适用于 mcdnnFusedOp_t 中 MCDNN_FUSED_BN_FINALIZE_STATISTICS_TRAINING 属性			
属性	Setter 中期望传入的描述符类型	说明	默认值
X_BN_MODE	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnBatchNorm Mode_t	描述 scale，bias，统计数据的操作模式 mcDNN 只支持 MCDNN_BATCHNORM_ SPATIAL 和 MCDNN_BATCHNORM_SPATIAL _PERSISTENT，即 scale，bias，统计数据均为 per-channel 模式	MCDNN _BATCHNORM_PER _ACTIVATION
X_YSTATS _DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensor Descriptor_t	张量描述符，描述 y 的和与 y 平方和的张量大小，布局，数据类型 形状必须与 MCDNN_PARAM_BN _MODE 中指定的模式匹配	NULL
X_YSUM_ PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 y 的和的指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL
X_YSQSUM_ PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 y 的和的指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL

下页继续

表 5.5 – 续上页

适用于 mcdnnFusedOp_t 中 MCDNN_FUSED_BN_FINALIZE_STATISTICS_TRAINING 属性			
X_BN_SCALEBIAS_MEANVAR_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensorDescriptor_t	通用张量描述符，描述 batchNorm 训练的 scale，bias 和统计数据张量的大小，布局，数据类型 形状必须与 MCDNN_PARAM_BN_MODE 中指定的模式匹配（与 mcdnnBatchNormalization* API 中的 bnScale BiasMeanVarDesc 字段类似）	NULL
X_BN_SCALE_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 训练的 scale 指针是否为 NULL 如果不是，则为用户允许的指针对齐 若不需要 BN_EQSCALE 的输出，则此属性值可以为 NULL	MCDNN_PTR_ NULL
X_BN_BIAS_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 训练的 bias 指针是否为 NULL 如果不是，则为用户允许的指针对齐 若不需要 BN_EQSCALE 和 BN_EQBIAS 的输出，则此属性值可以为 NULL	MCDNN_PTR_ NULL
X_BN_SAVED_MEAN_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 保存的均值指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则此输出计算变为 NoP	MCDNN_PTR_ NULL
X_BN_SAVED_INVSTD_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 保存的逆标准差指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则此输出计算变为 NoP	MCDNN_PTR_ NULL
X_BN_RUNNING_MEAN_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 移动均值指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则此输出计算变为 NoP	MCDNN_PTR_ NULL

下页继续

表 5.5 – 续上页

适用于 mcdnnFusedOp_t 中 MCDNN_FUSED_BN_FINALIZE_STATISTICS_TRAINING 属性			
X_BN_RUNNING_VAR_ PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 batchNorm 移动方差指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则此输出计算变为 NoP	MCDNN_PTR_ NULL
X_BN_EQSCALEBIAS_ _DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensor Descriptor_t	张量描述符，描述 batch-Norm 等效 scale 和 bias 张量的大小，布局，数据类型 形状必须与 MCDNN_PARAM_ BN_MODE 中指定的模式匹配 若不需要 BN_EQSCALE 和 BN_EQBIAS 的输出，则此参数可以为 NULL	NULL
X_BN_EQSCALE_ PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 batchNorm 等效 scale 指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则 scale 操作会变为 NoP	MCDNN_PTR_ NULL
X_BN_EQBIAS_ PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 batchNorm 等效 bias 指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则 bias 操作会变为 NoP	MCDNN_PTR_ NULL

mcdnnFusedOpsConstParamLabel_t 中的

MCDNN_FUSED_BN_FINALIZE_STATISTICS_INFERENCE

适用于 mcdnnFusedOp_t 中 MCDNN_FUSED_BN_FINALIZE_STATISTICS_INFERENCE 属性			
属性	Setter 中期望传入的描述符类型	说明	默认值
X_BN_MODE	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnBatch-Norm Mode_t	描述 scale，bias，统计数据的操作模式 mcDNN 只支持 MCDNN_BATCHNORM _SPATIAL 和 MCDNN_BATCHNORM _SPATIAL _PERSIS-TENT，即 scale，bias，统计数据均为 per-channel 模式	MCDNN _BATCHNORM_PER _ACTIVATION

下页继续

表 5.6 – 续上页

适用于 mcdnnFusedOp_t 中 MCDNN_FUSED_BN_FINALIZE_STATISTICS_INFERENCE 属性			
X_BN_SCALEBIAS_MEANVAR_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensorDescriptor_t	通用张量描述符，描述 batchNorm 训练的 scale，bias 和统计数据张量的大小，布局，数据类型 形状必须与 MCDNN_PARAM_BN_MODE 中指定的模式匹配（与 mcdnnBatchNormalization* API 中的 bnScaleBiasMeanVarDesc 字段类似）	NULL
X_BN_SCALE_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 训练的 scale 指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL
X_BN_BIAS_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 训练的 bias 指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL
X_BN_RUNNING_MEAN_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 移动均值指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL
X_BN_RUNNING_VAR_ PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 移动方差指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL
X_BN_EQSCALEBIAS__DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensorDescriptor_t	张量描述符，描述 batchNorm 等效 scale 和 bias 张量的大小，布局，数据类型 形状必须与 MCDNN_PARAM_BN_MODE 中指定的模式匹配	NULL
X_BN_EQSCALE_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm 等效 scale 指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则此输出计算会变为 NoP	MCDNN_PTR_ NULL

下页继续

表 5.6 – 续上页

适用于 mcdnnFusedOp_t 中 MCDNN_FUSED_BN_FINALIZE_STATISTICS_INFERENCE 属性			
X_BN_EQBIAS_ PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 batchNorm 等效 bias 指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则输出计算会变为 NoP	MCDNN_PTR_ NULL

mcdnnFusedOpsConstParamLabel_t 中的

MCDNN_FUSED_CONVOLUTION_SCALE_BIAS_ADD_RELU

属性	Setter 中期望传入的描述符类型	说明	默认值
X_XDESC	Setter 中，*param 应为 xDesc，一个指向已初始化的 mcdnnTensor Descriptor_t 的指针	描述 x（输入）张量大小，布局，数据类型的张量描述符	NULL
X_XDATA PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 xData 指针是否为 NULL，如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL
X_CONV_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnConvolution Descriptor_t	描述卷积操作	NULL
X_WDESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnFilter Descriptor_t	卷积核描述符，描述 w（卷积核）张量的大小，布局，数据类型	NULL
X_WDATA_ PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 w（卷积核）张量指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL
X_BN _EQSCALEBIAS _DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensor Descriptor_t	张量描述符，描述 alpha1 scale 和 bias 张量的大小，布局，数据类型 张量形状为 (1,K,1,1)，其中 K 是输出特征图的数量	NULL
X_BN_EQSCALE_ PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_Pointer PlaceHolder_t	描述 VariantParamPack 中 batchNorm 等效 scale 或 alpha1 张量指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则 alpha1 缩放操作变为 NoP	MCDNN_PTR_ NULL

下页继续

表 5.7 – 续上页

属性	Setter 中期望传入的描述符类型	说明	默认值
X_ZDESC	Setter 中，*param 应为 xDesc，一个指针，指向已初始化的 mcdnnTensorDescriptor_t	张量描述符，描述 z 张量的大小，布局，数据类型的张量描述符 如果不设置，则 z 缩放加法（scale-add）运算会变为 NoP	NULL
MCDNN_PARAM_ZDATA_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 z 张量指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则 z 缩放加法运算会变为 NoP	MCDNN_PTR_ NULL
MCDNN_PARAM_BN_Z_EQSCALEBIAS_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensorDescriptor_t	张量描述符，描述 alpha2 张量的大小，布局，数据类型 如果设置为 NULL，则输入 z 变为 NoP	NULLPTR
MCDNN_PARAM_BN_Z_EQSCALE PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 batchNorm z 等效缩放指针是否为 NULL 如果不是，则为用户允许的指针对齐 如果设置为 MCDNN_PTR_NULL，则输入 z 缩放会变为 NoP	MCDNN_PTR_ NULL
X_ACTIVATION_DESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnActivation Descriptor_t	描述激活操作 mcDNN 只支持 MCDNN_ACTIVATION_RELU 和 MCDNN_ACTIVATION_IDENTITY 激活模式 若设为 NULL 或激活模式设为 MCDNN_ACTIVATION_IDENTITY，则 op 序列中的激活操作会变为 NoP	NULL
X_YDESC	Setter 中，*param 应为一个指针，指向已初始化的 mcdnnTensorDescriptor_t	张量描述符，描述 y（输出）张量大小，布局，数据类型	NULL
X_YDATA_PLACEHOLDER	Setter 中，*param 应为一个指针，指向已初始化的 X_PointerPlaceHolder_t	描述 VariantParamPack 中 y（输出）张量指针是否为 NULL 如果不是，则为用户允许的指针对齐	MCDNN_PTR_ NULL

5.1.3.3 mcdnnFusedOpsPointerPlaceHolder_t

mcdnnFusedOpsPointerPlaceHolder_t 是一种枚举类型，用于选择 mcdnnFusedOps 描述符指针的对齐类型。

项	说明
MCDNN_PTR_NULL = 0	表示指向 variantPack 中张量的指针为 NULL
MCDNN_PTR_ELEM_ALIGNED = 1	表示指向 variantPack 中张量的指针不为 NULL，且会元素对齐
MCDNN_PTR_16B_ALIGNED = 2	表示指向 variantPack 中张量的指针不为 NULL，且会 16 字节对齐

5.1.3.4 mcdnnFusedOpsVariantParamLabel_t

mcdnnFusedOpsVariantParamLabel_t 是一种枚举类型，用于设置缓冲区指针。这些缓冲区指针可以在每次迭代中更改。

```
typedef enum {
    MCDNN_PTR_XDATA                            = 0,
    MCDNN_PTR_BN_EQSCALE                       = 1,
    MCDNN_PTR_BN_EQBIAS                        = 2,
    MCDNN_PTR_WDATA                            = 3,
    MCDNN_PTR_DWDATA                           = 4,
    MCDNN_PTR_YDATA                            = 5,
    MCDNN_PTR_DYDATA                           = 6,
    MCDNN_PTR_YSUM                             = 7,
    MCDNN_PTR_YSQSUM                           = 8,
    MCDNN_PTR_WORKSPACE                        = 9,
    MCDNN_PTR_BN_SCALE                         = 10,
    MCDNN_PTR_BN_BIAS                          = 11,
    MCDNN_PTR_BN_SAVED_MEAN                    = 12,
    MCDNN_PTR_BN_SAVED_INVSTD                  = 13,
    MCDNN_PTR_BN_RUNNING_MEAN                  = 14,
    MCDNN_PTR_BN_RUNNING_VAR                   = 15,
    MCDNN_PTR_ZDATA                            = 16,
    MCDNN_PTR_BN_Z_EQSCALE                     = 17,
    MCDNN_PTR_BN_Z_EQBIAS                      = 18,
    MCDNN_PTR_ACTIVATION_BITMASK               = 19,
    MCDNN_PTR_DXDATA                           = 20,
    MCDNN_PTR_DZDATA                           = 21,
    MCDNN_PTR_BN_DSCALE                        = 22,
    MCDNN_PTR_BN_DBIAS                         = 23,
    MCDNN_SCALAR_SIZE_T_WORKSPACE_SIZE_IN_BYTES  = 100,
    MCDNN_SCALAR_INT64_T_BN_ACCUMULATION_COUNT   = 101,
    MCDNN_SCALAR_DOUBLE_BN_EXP_AVG_FACTOR        = 102,
    MCDNN_SCALAR_DOUBLE_BN_EPSILON               = 103,
    } mcdnnFusedOpsVariantParamLabel_t;
```

mcdnnFusedOpsVariantParamLabel_t 表格说明

使用的简称	含义
Setter	mcdnnSetFusedOpsVariantParamPack Attribute()
Getter	mcdnnGetFusedOpsVariantParamPack Attribute()
属性键列中的 X_ 前缀	枚举数名称中的 MCDNN_PTR_ 或 MCDNN_SCALAR_

mcdnnFusedOpsVariantParamLabel_t 中的

MCDNN_FUSED_SCALE_BIAS_ACTIVATION_CONV_BNSTATS

属性键	Setter 中期望传入的描述符类型	I/O 类型	说明	默认值
X_XDATA	void *	input	设备上指向 x（输入）张量的指针 需与已设置的 MCDNN_PARAM_XDATA_ PLACEHOLDER 属性一致	NULL
X_BN_ EQSCALE	void *	input	设备上指向 batchNorm 等效 scale 张量的指针 需与已设置的 MCDNN_PARAM_BN_EQSCALE_ PLACEHOLDER 属性一致	NULL
X_ BN_EQBIAS	void *	input	设备上指向 batchNorm 等效 bias 张量的指针 需要与已设置的 MCDNN_PARAM_BN_ EQBIAS_PLACE HOLDER 属性一致	NULL
X_WDATA	void *	input	设备上指向 w (卷积核) 张量的指针 需要与已设置的 MCDNN_PARAM_WDATA_ PLACEHOLDER 属性一致	NULL
X_YDATA	void *	output	设备上指向 y (输出) 张量的指针 需要与已设置的 MCDNN_PARAM_YDATA_ PLACEHOLDER 属性一致	NULL
X_YSUM	void *	output	设备上指向 y 的和张量的指针 需要与已设置的 MCDNN_PARAM_YSUM_P LACEHOLDER 属性一致	NULL
X_YSQSUM	void *	output	设备上指向 y 的平方和张量的指针 需要与已设置的 MCDNN_PARAM_YSQSUM _PLACEHOLDER 属性一致	NULL
X_ WORKSPACE	void *	input	设备上指向用户已分配的工作空间的指针 若需求的工作空间大小为 0，可以为 NULL	NULL
X_SIZE_T _WORKSPACE_ SIZE_IN_ BYTES	size_t*	input	主机内存中指向一个 size_t 值的指针 描述用户已分配的工作空间大小(以字节为单位) 需要大于或等于 mcdnn MakeFusedOps Plan 的值	0

注解:

- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_NULL，则 VariantParamPack 中的设备指针也需要为 NULL。
- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_ELEM_ALIGNED 或 MCDNN_PTR_16B_ALIGNED，则 VariantParamPack 中的设备指针可能不是 NULL，分别至少需要元素对齐或 16 字节对齐。

mcdnnFusedOpsVariantParamLabel_t 中的 MCDNN_FUSED_SCALE_BIAS_ACTIVATION_WGRAD

属性键	Setter 中期望传入的描述符类型	I/O 类型	说明	默认值
X_XDATA	void *	input	设备上指向 x（输入）张量的指针 需要与已设置的 MCDNN_PARAM_XDATA _PLACEHOLDER 属性一致	NULL
X_BN_ EQSCALE	void *	input	设备上指向 batchNorm 等效 scale 张量的指针 需要与已设置的 MCDNN_PARAM_BN_ EQSCALE_PLACEHOLDER 属性一致	NULL
X_ BN_EQBIAS	void *	input	设备上指向 batchNorm 等效 bias 张量的指针 需要与已设置的 MCDNN_PARAM_BN_EQBIAS_ PLACEHOLDER 属性一致	NULL
X_DWDATA	void *	output	设备上指向 dw (卷积核梯度输出) 张量的指针 需要与已设置的 MCDNN_PARAM_WDATA_ PLACEHOLDER 属性一致	NULL
X_DYDATA	void *	input	设备上指向 dy (梯度输入) 张量的指针 需要与已设置的 MCDNN_PARAM_YDATA_ PLACEHOLDER 属性一致	NULL
X_ WORKSPACE	void *	input	设备上指向用户已分配的工作空间的指针 若需求的工作空间大小为 0，可以为 NULL	NULL
X_SIZE_T _WORKSPACE_ SIZE_IN_ BYTES	size_t*	input	主机内存中指向一个 size_t 值的指针 描述用户已分配的工作空间大小(以字节为单位) 需要大于或等于 mcdnn MakeFusedOps Plan 的值	0

注解:

- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_NULL，则 VariantParamPack 中的设备指针也需要为 NULL。
- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_ELEM_ALIGNED 或 MCDNN_PTR_16B_ALIGNED，则 VariantParamPack 中的设备指针可能不是 NULL，分别至少需要元素对齐或 16 字节对齐。

mcdnnFusedOpsVariantParamLabel_t 中的 MCDNN_FUSED_BN_FINALIZE_STATISTICS_TRAINING

属性键	Setter 中期望传入的描述符类型	I/O 类型	说明	默认值
X_YSUM	void *	input	设备上指向 y 的和张量的指针 需与已设置的 MCDNN_PARAM_YSUM_ PLACEHOLDER 属性一致	NULL

下页继续

表 5.10 – 续上页

属性键	Setter 中期望传入的描述符类型	I/O 类型	说明	默认值
X_YSQSUM	void *	input	设备上指向 y 的平方和张量的指针 需与已设置的 MCDNN_PARAM_YSQSUM_ PLACEHOLDER 属性一致	NULL
X _BN_SCALE	void *	input	设备上指向 batchNorm scale 张量的指针 需与已设置的 MCDNN_PARAM_BN_SCALE_ PLACEHOLDER 属性一致	NULL
X_ BN_BIAS	void *	input	设备上指向 batchNorm bias 张量的指针 需与已设置的 MCDNN_PARAM_BN_BIAS_ PLACEHOLDER 属性一致	NULL
X_BN_SAVED _MEAN	void *	output	设备上指向 batchNorm 保存的均值张量的指针 需与已设置的 MCDNN_PARAM_BN_SAVED _MEAN_ PLACEHOLDER 属性一致	NULL
X_BN_SAVED _INVSTD	void *	output	设备上指向 batchNorm 保存的逆标准差张量的指针 需与已设置的 MCDNN_PARAM_BN_SAVED _INVSTD_ PLACEHOLDER 属性一致	NULL
X_BN_ RUNNING_ MEAN	void *	input/ output	设备上指向 batchNorm 移动均值张量的指针 需与已设置的 MCDNN_PARAM_BN_RUNNING _MEAN_ PLACEHOLDER 属性一致	NULL
X_BN_ RUNNING_ VAR	void *	input/ output	设备上指向 batchNorm 移动方差张量的指针 需与已设置的 MCDNN_PARAM_BN_RUNNING _VAR_ PLACEHOLDER 属性一致	NULL
X_BN_ EQSCALE	void *	output	设备上指向 batchNorm 等效 scale 张量的指针 需与已设置的 MCDNN_PARAM_BN_EQSCALE_ PLACEHOLDER 属性一致	NULL
X_BN_ EQBIAS	void *	output	设备上指向 batchNorm 等效 bias 张量的指针 需与已设置的 MCDNN_PARAM_BN_EQBIAS_ PLACEHOLDER 属性一致	NULL

下页继续

表 5.10 – 续上页

属性键	Setter 中期望传入的描述符类型	I/O 类型	说明	默认值
X_INT64_T_BN_ACCUMULATION_COUNT	int64_t*	input	主机内存中指向 int64_t 中一个标量值的指针 该标量值描述 y 的和与平方和张量累加的张量元素数 例如，在单个 GPU 用例中，若 mode 为 MCDNN _BATCHNORM_SPATIAL 或 MCDNN _BATCHNORM _SPATIAL_ PERSISTENT，该值应等于张量的 N*H*W，该张量为从中计算统计数据的张量 在多 GPU 用例中，若已在 y 的和与平方和张量上执行 all-reduce，该值应为每个 GPU 上单 GPU 累积计数的总和	0
X_DOUBLE_BN_EXP_AVG_FACTOR	double*	input	主机内存中指向一个双精度标量值的指针 在移动平均计算中使用的系数 参见 mcdnnBatch Normaliza tion* API 中的 exponen tialAverage Factor	0.0
X_DOUBLE_BN_EPSILON	double*	input	主机内存中指向一个双精度标量值的指针 批量归一化公式中使用的条件常量 其值应大于或等于 mcdnn.h 中定义的 MCDNN_BN_MIN _EPSILON 的值 参见 mcdnnBatch Normaliza tion* API 中的 exponen tialAverage Factor	0.0
X_ WORKSPACE	void *	input	设备上指向用户已分配的工作空间的指针 若需求的工作空间大小为 0，可以为 NULL	NULL
X_SIZE_T_WORKSPACE_SIZE_IN_BYTES	size_t*	input	主机内存中指向一个 size_t 值的指针 描述用户已分配的工作空间大小（以字节为单位） 需要大于或等于 mcdnn MakeFusedOps Plan 的值	0

注解:

- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_NULL，则 VariantParamPack 中的设备指针也需要为 NULL。
- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_ELEM_ALIGNED 或 MCDNN_PTR_16B_ALIGNED，则 VariantParamPack 中的设备指针可能不是 NULL，分别至少需要元素对齐或 16 字节对齐。

mcdnnFusedOpsVariantParamLabel_t 中的

MCDNN_FUSED_BN_FINALIZE_STATISTICS_INFERENCE

属性键	Setter 中期望传入的描述符类型	I/O 类型	说明	默认值
X _BN_SCALE	void *	input	设备上指向 batchNorm scale 张量的指针 需与已设置的 MCDNN_PARAM _BN_SCALE_ PLACEHOLDER 属性一致	NULL
X_ BN_BIAS	void *	input	设备上指向 batchNorm bias 张量的指针 需与已设置的 MCDNN_PARAM _BN_BIAS_ PLACEHOLDER 属性一致	NULL
X_BN_ RUNNING_ MEAN	void *	input/ output	设备上指向 batchNorm 移动均值张量的指针 需与已设置的 MCDNN_PARAM _BN_RUNNING _MEAN_ PLACE-HOLDER 属性一致	NULL
X_BN_ RUNNING_ VAR	void *	input/ output	设备上指向 batchNorm 移动方差张量的指针 需与已设置的 MCDNN_PARAM _BN_RUNNING _VAR_ PLACE-HOLDER 属性一致	NULL
X_BN_ EQSCALE	void *	output	设备上指向 batchNorm 等效 scale 张量的指针 需与已设置的 MCDNN_PARAM _BN_EQSCALE_ PLACEHOLDER 属性一致	NULL
X_BN_ EQBIAS	void *	output	设备上指向 batchNorm 等效 bias 张量的指针 需与已设置的 MCDNN_PARAM _BN_EQBIAS_ PLACEHOLDER 属性一致	NULL
X_DOUBLE_BN _EPSILON	double*	input	主机内存中指向一个双精度标量值的指针 批量归一化公式中使用的条件常量 其值应大于或等于 mcdnn.h 中定义的 MCDNN_BN _MIN_ EPSILON 的值 参见 mcdnnBatch Normaliza tion* API 中的 exponen tialAverage Fac-tor	0.0
X_ WORKSPACE	void *	input	设备上指向用户已分配的工作空间的指针 若需求的工作空间大小为 0，可以为 NULL	NULL
X_SIZE_T _WORKSPACE_ SIZE_IN_ BYTES	size_t*	inpu	主机内存中指向一个 size_t 值的指针 描述用户已分配的工作空间大小（以字节为单位） 需要大于或等于 mcdnn Make-FusedOps Plan 的值	0
X_XDATA	void *	input	设备上指向 x（图像）张量的指针 需与已设置的 MCDNN_PARAM _XDATA PLACEHOLDER 属性一致	NULL

下页继续

表 5.11 – 续上页

属性键	Setter 中期望传入的描述符类型	I/O 类型	说明	默认值
X_WDATA	void *	input	设备上指向 w （卷积核）张量的指针 需与已设置的 MCDNN_PARAM_WDATA_ PLACEHOLDER 属性一致	NULL
X_BN_ EQSCALE	void *	output	设备上指向 alpha1 或 batchNorm 等效 scale 张量的指针 需与已设置的 MCDNN_PARAM_BN _EQSCALE_ PLACEHOLDER 属性一致	NULL
X_ZDATA	void *	input	设备上指向 z 张量的指针 需已设置的 MCDNN_PARAM_YDATA_ PLACEHOLDER 属性一致	NULL
X_BN_ Z_EQSCALE	void *	input	设备上指向 z 的 alpha2 或等效 scale 张量的指针 需与已设置的 MCDNN_PARAM_BN_Z _EQSCALE_ PLACEHOLDER 属性一致	NULL
X_BN_ Z_EQBIAS	void *	input	设备上指向 z 的 batchNorm 等效 bias 张量的指针 需与已设置的 MCDNN_PARAM_BN_Z _EQBIAS_ PLACEHOLDER 属性一致	NULL
X_YDATA	void *	output	设备上指向 y （输出）张量的指针 需与已设置的 MCDNN_PARAM_YDATA_ PLACEHOLDER 属性一致	NULL
X_ WORKSPACE	void *	input	设备上指向用户已分配的工作空间的指针 若需求的工作空间大小为 0，可以为 NULL	NULL
X_SIZE_T _WORKSPACE_ SIZE_IN_ BYTES	size_t*	input	主机内存中指向一个 size_t 值的指针 描述用户已分配的工作空间大小（以字节为单位） 需要大于或等于 mcdnn MakeFusedOps Plan 的值	0

注解:

- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_NULL，则 VariantParamPack 中的设备指针也需要为 NULL。
- 如果 ConstParamPack 中对应的指针占位符设置为 MCDNN_PTR_ELEM_ALIGNED 或 MCDNN_PTR_16B_ALIGNED，则 VariantParamPack 中的设备指针可能不是 NULL，分别至少需要元素对齐或 16 字节对齐。

5.2 API 参考

5.2.1 API 函数

5.2.1.1 mcdnnCnnTrainVersionCheck()

此函数检查库的 CnnTrain 子集的版本是否与其他子库一致。

```
mcdnnStatus_t mcdnnCnnTrainVersionCheck(void)
```

返回值

MCDNN_STATUS_SUCCESS

版本与其他子库一致。

MCDNN_STATUS_VERSION_MISMATCH

CnnTrain 的版本与其他子库不一致。用户应检查并确保所有子组件安装版本一致。

5.2.1.2 mcdnnConvolutionBackwardBias()

该函数计算与 bias 相关的卷积函数梯度，其是输入张量所有图像上属于同一特征图的元素的总和。因此，得到的元素数等于输入张量的特征图数。

```
mcdnnStatus_t mcdnnConvolutionBackwardBias(
mcdnnHandle_t                       handle,
const void                          *alpha,
const mcdnnTensorDescriptor_t       dyDesc,
const void                          *dy,
const void                          *beta,
const mcdnnTensorDescriptor_t       dbDesc,
void                                *db)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将计算结果与输出层中的先验值混合，如下所示：dstValue = alpha[0]*resultValue + beta[0]*priorDstValue。

dyDesc

输入。已初始化的输入张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

dy

输入。数据指针，指向与张量描述符 dyDesc 关联的 GPU 内存。

dbDesc

输入。已初始化的输出张量描述符的句柄。

db

输出。数据指针，指向与输出张量描述符 dbDesc 关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

操作成功启动。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- One of the parameters n, height, or width of the output tensor is not 1.
- 输入张量和输出张量的特征图数量不同。
- 两个张量描述符的 dataType 不同。

5.2.1.3 mcdnnConvolutionBackwardFilter()

```
mcdnnStatus_t mcdnnConvolutionBackwardFilter(
    mcdnnHandle_t                       handle,
    const void                          *alpha,
    const mcdnnTensorDescriptor_t       xDesc,
    const void                          *x,
    const mcdnnTensorDescriptor_t       dyDesc,
    const void                          *dy,
    const mcdnnConvolutionDescriptor_t  convDesc,
    mcdnnConvolutionBwdFilterAlgo_t     algo,
    void                                *workSpace,
    size_t                              workSpaceSizeInBytes,
    const void                          *beta,
    const mcdnnFilterDescriptor_t       dwDesc,
    void                                *dw)
```

该函数计算张量 dy 的卷积权重梯度，其中 y 是 mcdnnConvolutionForward() 中正向卷积的输出。它使用指定的算法，并在输出张量 dw 中返回结果。缩放系数 alpha 和 beta 可用于缩放计算结果或与当前 dw 累加。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

alpha, beta

输入。指向缩放系数（主机内存中）的指针，用于将计算结果与输出层中的先验值混合，如下所示：

```
dstValue = alpha[0] * result + beta[0] * priorDstValue
```

xDesc

输入。已初始化的张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

x

输入。数据指针，指向与张量描述符 xDesc 关联的 GPU 内存。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

dy

输入。数据指针，指向与反向传播梯度张量描述符 dyDesc 关联的 GPU 内存。

convDesc

输入。已初始化的卷积描述符。更多信息，参见 mcdnnConvolutionDescriptor_t。

algo

输入。指定应使用哪个卷积算法来计算结果的枚举。更多信息，参见 mcdnnConvolutionBwdFilterAlgo_t。

workSpace

输入。数据指针，指向执行指定算法所需的工作空间 GPU 内存。如果特定算法不需要工作空间，则该指针可以为 nil。

workSpaceSizeInBytes

输入。指定已提供的 workSpace 大小（以字节为单位）。

dwDesc

输入。已初始化的卷积核梯度描述符的句柄。更多信息，参见 mcdnnFilterDescriptor_t。

dw

输入/输出。数据指针，指向与卷积核梯度描述符 dwDesc（携带结果）关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

操作成功启动。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 以下至少有一个为 NULL：handle、xDesc、dyDesc、convDesc、dwDesc、xData、dyData、dwData、alpha、beta
- xDesc 和 dyDesc 的维数不匹配
- xDesc 和 dwDesc 的维数不匹配
- xDesc 的维数小于 3
- xDesc、dyDesc 和 dwDesc 的数据类型不匹配。
- xDesc 和 dwDesc 每个图像（或分组卷积情况下的组）的输入特征图数不匹配。
- yDesc 或 dwDesc 表示输出通道计数，其不是组计数的倍数（如果在 convDesc 中设置了组计数)。

MCDNN_STATUS_NOT_SUPPORTED

需至少满足以下任一条件：

- xDesc 或 dyDesc 具有负张量步幅
- xDesc，dwDesc 或 dyDesc 的维数不是 4 或 5
- 所选 algo 不支持提供的参数；有关每个 algo 支持的参数，参见以下详尽列表

MCDNN_STATUS_MAPPING_ERROR

创建与卷积核数据关联的纹理对象时发生错误。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

支持的 mcdnnConvolutionBackwardFilter() 配置

数据类型配置	xDesc，dyDesc，dwDesc 数据类型	convDesc 数据类型
TRUE_HALF_CONFIG （仅支持 true FP16 架构）	MCDNN_DATA_HALF	MCDNN_DATA_HALF
PSEUDO_HALF_ CONFIG	MCDNN_DATA_HALF	MCDNN_DATA_FLOAT
PSEUDO_BFLOAT16_ CONFIG	MCDNN_DATA_BFLOAT16	MCDNN_DATA_FLOAT
FLOAT_CONFIG	MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT
DOUBLE_CONFIG	MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE

注解： 指定单独的算法会导致性能、支持和计算决定因素发生变化。有关算法选项及其各自支持的参数和确定性的详尽列表，请参见下表。

支持的算法

- MCDNN_CONVOLUTION_BWD_FILTER_ALGO_1 (_ALGO_1)
- MCDNN_CONVOLUTION_BWD_FILTER_ALGO_FFT_TILING (_FFT_TILING)
- MCDNN_TENSOR_NCHW (_NCHW)
- MCDNN_TENSOR_NHWC (_NHWC)
- MCDNN_TENSOR_NCHW_VECT_C (_NCHW_VECT_C)

mcdnnConvolutionBackwardFilter() 2D 卷积支持的算法：dwDesc: _NHWC

算法名称	确定性 (Yes or No)	dyDesc 支持的张量格式	dxDesc 支持的张量格式	支持的数据类型配置	重要
_ALGO_1		NHWC HWC-packed	NHWC HWC-packed	PSEUDO _HALF_ CONFIG PSEUDO _BFLOAT16 _CONFIG FLOAT _CONFIG	

mcdnnConvolutionBackwardFilter() 2D 卷积支持的算法：dwDesc: _NCHW

算法名称	确定性 (Yes or No)	dyDesc 支持的张量格式	dxDesc 支持的张量格式	支持的数据类型配置	重要
_ALGO_1	Yes	除 _NCHW_ VECT_C 之外的所有其他格式	NCHW CHW-packed	PSEUDO _HALF _CONFIG TRUE_HALF _CONFIG PSEUDO _BFLOAT16 _CONFIG FLOAT _CONFIG DOUBLE _CONFIG	所有维度大于 0 convDesc 组计数支持：大于 0

下页继续

表 5.12 – 续上页

算法名称	确定性 (Yes or No)	dyDesc 支持的张量格式	dxDesc 支持的张量格式	支持的数据类型配置	重要
_FFT _TILING	Yes	NCHW CHW-packed	NCHW CHW-packed	PSEUDO _HALF _CONFIG FLOAT _CONFIG DOUBLE _CONFIG	1 适用于所有维度 convDesc 组计数支持：大于 0 dyDesc 宽度或高度必须等于 1 (与 xDesc 相同的维度) 其他维度必须小于或等于 256，即目前支持的最大 1D 分块尺寸 convDesc 垂直和水平卷积核步幅必须等于 1 dwDesc 卷积核高度必须大于 convDesc 零填充高度 dwDesc 卷积核宽度必须大于 convDesc 零填充宽度

mcdnnConvolutionBackwardFilter() 3D 卷积支持的算法：dwDesc: _NCHW

算法名称	确定性 (Yes or No)	dyDesc 支持的张量格式	dxDesc 支持的张量格式	支持的数据类型配置	重要
_ALGO_1	No	除 _NCDHW _VECT_C 之外的所有其他格式	NCDHW CDHW -packed NCDHW W-packed NDHWC	PSEUDO _HALF _CONFIG PSEUDO _BFLOAT16 _CONFIG FLOAT _CONFIG DOUBLE _CONFIG	所有维度大于 0 convDesc 组计数支持：大于 0

mcdnnConvolutionBackwardFilter() 3D 卷积支持的算法：dwDesc: _NHWC

算法名称	确定性 (Yes or No)	dyDesc 支持的张量格式	dxDesc 支持的张量格式	支持的数据类型配置	重要
_ALGO_1	Yes	NDHWC HWC-packed	NDHWC HWC-packed	PSEUDO _HALF _CONFIG PSEUDO _BFLOT16 _CONFIG FLOAT _CONFIG TRUE_HALF _CONFIG	所有维度大于 0 convDesc 组计数支持：大于 0

5.2.1.4 mcdnnCreateFusedOpsConstParamPack()

此函数创建一个不透明结构来存储所选 mcdnnFusedOps 计算序列的各种 problem size 信息，例如形状，布局和张量类型，以及卷积和激活描述符。

```
mcdnnStatus_t mcdnnCreateFusedOpsConstParamPack(
mcdnnFusedOpsConstParamPack_t       *constPack,
mcdnnFusedOps_t                     ops);
```

参数

constPack

输入。由该函数创建的不透明结构。更多信息，参见 mcdnnFusedOpsConstParamPack_t。

ops

输入。要在 mcdnnFusedOps 计算中执行的特定计算序列，定义在枚举类型 mcdnnFusedOps_t 中。

返回值

MCDNN_STATUS_BAD_PARAM

constPack 或 ops 为 NULL。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

MCDNN_STATUS_SUCCESS

描述符创建成功。

5.2.1.5 mcdnnCreateFusedOpsPlan()

此函数创建 mcdnnFusedOps 计算的 plan 描述符。此描述符包含计划信息，包括问题类型和大小，应运行哪些内核以及内部工作空间分区。

```
mcdnnStatus_t mcdnnCreateFusedOpsPlan(
mcdnnFusedOpsPlan_t         *plan,
mcdnnFusedOps_t             ops);
```

参数

plan

输入。指针，指向此函数创建的描述符实例。

ops

输入。此 plan 描述符对应的融合操作计算的特定序列。更多信息，参见 mcdnnFusedOps_t。

返回值

MCDNN_STATUS_BAD_PARAM

输入 plan 为 NULL 或 ops 输入不是有效的 mcdnnFusedOp 枚举。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

MCDNN_STATUS_SUCCESS

plan 描述符创建成功。

5.2.1.6 mcdnnCreateFusedOpsVariantParamPack()

此函数创建 mcdnnFusedOps 计算的 variant pack 描述符。

```
mcdnnStatus_t mcdnnCreateFusedOpsVariantParamPack(
mcdnnFusedOpsVariantParamPack_t *varPack,
mcdnnFusedOps_t                 ops);
```

参数

varPack

输入。指针，指向此函数创建的描述符。更多信息，参见 mcdnnFusedOpsVariantParam-Pack_t。

ops

输入。此描述符对应的融合操作计算的特定序列。

返回值

MCDNN_STATUS_SUCCESS

描述符创建成功。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

MCDNN_STATUS_BAD_PARAM

任何输入都无效。

5.2.1.7 mcdnnDestroyFusedOpsConstParamPack()

此函数用于销毁已创建的 mcdnnFusedOpsConstParamPack_t 结构。

```
mcdnnStatus_t mcdnnDestroyFusedOpsConstParamPack(
mcdnnFusedOpsConstParamPack_t constPack);
```

参数

constPack

输入。要销毁的 mcdnnFusedOpsConstParamPack_t 结构。

返回值

MCDNN_STATUS_SUCCESS

描述符已销毁成功。

MCDNN_STATUS_INTERNAL_ERROR

ops 枚举值不受支持或无效。

5.2.1.8 mcdnnDestroyFusedOpsPlan()

此函数用于销毁已提供的 plan 描述符。

```
mcdnnStatus_t mcdnnDestroyFusedOpsPlan(
mcdnnFusedOpsPlan_t plan);
```

参数

plan

输入。需要通过此函数销毁的描述符。

返回值

MCDNN_STATUS_SUCCESS

plan 描述符为 NULL 或描述符已成功销毁。

5.2.1.9 mcdnnDestroyFusedOpsVariantParamPack()

此函数用于销毁已为 mcdnnFusedOps 常量参数创建的描述符。

```
mcdnnStatus_t mcdnnDestroyFusedOpsVariantParamPack(
mcdnnFusedOpsVariantParamPack_t varPack);
```

参数

varPack

输入。要销毁的描述符。

返回值

MCDNN_STATUS_SUCCESS

描述符销毁成功。

5.2.1.10 mcdnnFindConvolutionBackwardFilterAlgorithm()

```
mcdnnStatus_t mcdnnFindConvolutionBackwardFilterAlgorithm(
    mcdnnHandle_t                       handle,
    const mcdnnTensorDescriptor_t       xDesc,
    const mcdnnTensorDescriptor_t       dyDesc,
    const mcdnnConvolutionDescriptor_t  convDesc,
    const mcdnnFilterDescriptor_t       dwDesc,
    int                                 requestedAlgoCount,
    int                                 *returnedAlgoCount,
    mcdnnConvolutionBwdFilterAlgoPerf_t *perfResults)
```

此函数尝试所有可用于 mcdnnConvolutionBackwardFilter() 的算法。它将尝试提供的 convDesc MathType 和 MCDNN_TENSOR_OP_MATH（假设两者不同）。

注解: 只能使用 MCDNN_FMA_MATH 尝试没有 MCDNN_TENSOR_OP_MATH 可用性的算法，并以同样方式返回。对于 MCDNN_DATA_FLOAT，MCDNN_FMA_MATH 将使用 float 类型计算，其它 MATH_TYPE 将使用 tf32 类型计算。

通过 mcMalloc() 分配内存。在用户分配的 mcdnnConvolutionBwdFilterAlgoPerf_t 数组中返回性能指标。这些指标以一种有序的方式写入，其中第一个元素的计算时间最短。可用算法的总数可以通过 mcdnnGetConvolutionBackwardFilterAlgorithmMaxCount() API 查询。

注解:

- 此函数是主机阻塞。
- 建议在分配层数据之前运行此函数；否则可能会由于资源使用问题而不必要地禁止某些算法选项。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

xDesc

输入。已初始化的输入张量描述符的句柄。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

convDesc

输入。已初始化的卷积描述符。

dwDesc

输入。已初始化的卷积核描述符的句柄。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 未正确分配 handle。
- 未正确分配 xDes、dyDesc 或 dwDesc。
- xDesc，dyDesc 或 dwDesc 的维度小于 1。
- returnedCount 或 perfResults 为 nil。
- requestedCount 小于 1。

MCDNN_STATUS_ALLOC_FAILED

此函数无法分配内存来存储样本输入，卷积核和输出。

MCDNN_STATUS_INTERNAL_ERROR

需至少满足以下任一条件：

- 此函数不能用来分配必要的计时对象。
- 此函数不能用来释放必要的计时对象。
- 此函数不能用来释放样本输入，卷积核和输出。

5.2.1.11 mcdnnFindConvolutionBackwardFilterAlgorithmEx()

```
mcdnnStatus_t mcdnnFindConvolutionBackwardFilterAlgorithmEx(
    mcdnnHandle_t                       handle,
    const mcdnnTensorDescriptor_t       xDesc,
    const void                          *x,
    const mcdnnTensorDescriptor_t       dyDesc,
    const void                           *dy,
    const mcdnnConvolutionDescriptor_t  convDesc,
    const mcdnnFilterDescriptor_t       dwDesc,
    void                                *dw,
    int                                 requestedAlgoCount,
    int                                 *returnedAlgoCount,
    mcdnnConvolutionBwdFilterAlgoPerf_t *perfResults,
    void                                *workSpace,
    size_t                              workSpaceSizeInBytes)
```

此函数尝试所有可用于 mcdnnConvolutionBackwardFilter() 的算法。它将尝试提供的 convDesc MathType 和 MCDNN_TENSOR_OP_MATH（假设两者不同）。

注解： 只能使用 MCDNN_TENSOR_OP_MATH 尝试没有 MCDNN_FMA_MATH 可用性的算法，并以同样方式返回。对于 MCDNN_DATA_FLOAT，MCDNN_FMA_MATH 将使用 float 类型计算，其它 MATH_TYPE 将使用 tf32 类型计算。

通过 mcMalloc() 分配内存。在用户分配的 mcdnnConvolutionBwdFilterAlgoPerf_t 数组中返回性能指标。这些指标以一种有序的方式写入，其中第一个元素的计算时间最短。可用算法的总数可以通过 mcdnnGetConvolutionBackwardFilterAlgorithmMaxCount() API 查询。

注解： 此函数是主机阻塞。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

xDesc

输入。已初始化的输入张量描述符的句柄。

x

输入。数据指针，指向与卷积核描述符 xDesc 关联的 GPU 内存。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

dy

输入。数据指针，指向与张量描述符 dyDesc 关联的 GPU 内存。

convDesc

输入。已初始化的卷积描述符。

dwDesc

输入。已初始化的卷积核描述符的句柄。

dw

输入/输出。数据指针，指向与卷积核描述符 dwDesc 关联的 GPU 内存。此张量的内容会被任意值覆盖。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

workSpace

输入。指向 GPU 内存的数据指针，此内存是某些算法所必需的工作空间。此工作空间的大小将决定算法的可用性。nil 指针被视为 0 字节的 workSpace。

workSpaceSizeInBytes

输入。指定已提供的 workSpace 大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 未正确分配 handle。
- 未正确分配 xDes、dyDesc 或 dwDesc。
- xDesc，dyDesc 或 dwDesc 的维度小于 1。
- x，dy 或 dw 为 nil。
- returnedCount 或 perfResults 为 nil。
- requestedCount 小于 1。

MCDNN_STATUS_INTERNAL_ERROR

需至少满足以下任一条件：

- 此函数不能用来分配必要的计时对象。
- 此函数不能用来释放必要的计时对象。
- 此函数不能用来释放样本输入，卷积核和输出。

5.2.1.12 mcdnnFusedOpsExecute()

此函数执行 mcdnnFusedOps 操作的序列。

```
mcdnnStatus_t mcdnnFusedOpsExecute(
mcdnnHandle_t                    handle,
const mcdnnFusedOpsPlan_t        plan,
mcdnnFusedOpsVariantParamPack_t varPack);
```

参数

handle

输入。指向 mcDNN 库上下文的指针。

plan

输入。指向已创建和初始化的 plan 描述符的指针。

varPack

输入。指向变量参数包描述符的指针。

返回值

MCDNN_STATUS_BAD_PARAM

plan 描述符中的 mcdnnFusedOps_t 类型不受支持。

5.2.1.13 mcdnnGetConvolutionBackwardFilterAlgorithmMaxCount()

```
mcdnnStatus_t mcdnnGetConvolutionBackwardFilterAlgorithmMaxCount(
      mcdnnHandle_t handle,
      int           *count)
```

该函数返回可从 mcdnnFindConvolutionBackwardFilterAlgorithm() 和 mcdnnGetConvolutionForwardAlgorithmer_v7() 返回的算法的最大数量。这是所有算法总和加上当前设备支持的具有 Tensor Core 操作的算法总和。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

count

输出。得到的算法最大计数。

返回值

MCDNN_STATUS_SUCCESS

此函数执行成功。

MCDNN_STATUS_BAD_PARAM

未正确分配提供的 handle。

5.2.1.14 mcdnnGetConvolutionBackwardFilterAlgorithm_v7()

该函数用作启发式函数，用于为给定卷积参数获取 mcdnnConvolutionBackwardFilter() 的最适合算法。此函数将返回按期望的 (基于内部启发式) 相对性能排序的所有算法 (包括 MCDNN_TENSOR_OP_MATH 和 MCDNN_DEFAULT_MATH 版本的算法，其中 MCDNN_TENSOR_OP_MATH 可能可用)，最快的是 perfResults 的索引 0。要全面搜索最快的算法，请使用 mcdnnFindConvolutionBackwardFilterAlgorithm()。可用算法的总数可以通过 returnedAlgoCount 变量查询。

```
mcdnnStatus_t mcdnnGetConvolutionBackwardFilterAlgorithm_v7(
mcdnnHandle_t                       handle,
const mcdnnTensorDescriptor_t       xDesc,
const mcdnnTensorDescriptor_t       dyDesc,
const mcdnnConvolutionDescriptor_t  convDesc,
const mcdnnFilterDescriptor_t       dwDesc,
const int                           requestedAlgoCount,
int                                 *returnedAlgoCount,
mcdnnConvolutionBwdFilterAlgoPerf_t *perfResults)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

xDesc

输入。已初始化的输入张量描述符的句柄。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

convDesc

输入。已初始化的卷积描述符。

dwDesc

输入。已初始化的卷积核描述符的句柄。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 以下任一参数为 NULL：handle、xDesc、dyDesc、convDesc、dwDesc、perfResults、returnedAlgoCount。
- 输入张量和输出张量的特征图数量不同。
- 两个张量描述符或卷积核的 dataType 不同。
- requestedAlgoCount 小于或等于 0。

5.2.1.15 mcdnnGetConvolutionBackwardFilterWorkspaceSize()

```
mcdnnStatus_t mcdnnGetConvolutionBackwardFilterWorkspaceSize(
    mcdnnHandle_t                       handle,
    const mcdnnTensorDescriptor_t       xDesc,
    const mcdnnTensorDescriptor_t       dyDesc,
    const mcdnnConvolutionDescriptor_t convDesc,
    const mcdnnFilterDescriptor_t       gradDesc,
    mcdnnConvolutionBwdFilterAlgo_t     algo,
    size_t                              *sizeInBytes)
```

此函数返回用户应分配的 GPU 内存工作空间量，以便能够使用指定算法来调用 mcdnnConvolutionBackwardFilter() 函数。然后，分配的工作空间将传入 mcdnnConvolutionBackwardFilter() 函数。指定的算法可以是调用 mcdnnGetConvolutionBackwardFilterAlgorithm_v7() 的结果，也可以由用户任意选择。请注意，并非每个算法都可用于输入张量的每个配置和/或卷积描述符的每个配置。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

xDesc

输入。已初始化的输入张量描述符的句柄。

dyDesc

输入。已初始化的输入差分张量描述符的句柄。

convDesc

输入。已初始化的卷积描述符。

dwDesc

输入。已初始化的卷积核描述符的句柄。

algo

输入。指定所选卷积算法的枚举。

sizeInBytes

输出。作为工作空间所需的 GPU 内存量，以便能够使用指定的算法执行正向卷积。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM 需至少满足以下任一条件：

- 输入张量和输出张量的特征图数量不同。
- 两个张量描述符或卷积核的 dataType 不同。

MCDNN_STATUS_NOT_SUPPORTED

指定的算法不支持张量描述符，卷积核描述符和卷积描述符的组合。

5.2.1.16 mcdnnGetFusedOpsConstParamPackAttribute()

此函数检索参数指针输入所指向的描述符的值。描述符的类型由 paramLabel 输入的枚举值指定。

```
mcdnnStatus_t mcdnnGetFusedOpsConstParamPackAttribute(
const mcdnnFusedOpsConstParamPack_t constPack,
mcdnnFusedOpsConstParamLabel_t      paramLabel,
```

(下页继续)

(续上页)

```
void                                    *param,
int                                     *isNULL);
```

参数

constPack

输入。mcdnnFusedOpsConstParamPack_t 不透明结构，用于存储所选 mcdnnFusedOps 计算序列的各种 problem size 信息，例如形状，布局和张量类型，以及卷积和激活描述符。

paramLabel

输入。此 getter 函数可以检索多种类型的描述符。参数输入指向描述符本身，此输入表明参数输入所指向的描述符类型。mcdnnFusedOpsConstParamLabel_t 枚举类型可用于选择描述符的类型。参见以下参数描述。

param

输入。数据指针，指向与要检索的描述符关联的主机内存。此描述符的类型取决于 paramLabel 的值。对于给定的 paramLabel，如果 constPack 中的关联值设置为 NULL 或默认为 NULL，则 mcDNN 将把该值或 constPack 中的不透明结构复制到 param 指向的主机内存缓冲区。有关更多信息，请参见 mcdnnFusedOpsConstParamLabel_t 中的表格。

isNULL

输入/输出。用户必须在此字段中传入主机内存中指向整数的指针。如果与给定 paramLabel 关联的 constPack 中的值默认为 NULL，或用户已设置为 NULL，则 mcDNN 将一个非零值写入 isNULL 所指的位置。

返回值

MCDNN_STATUS_SUCCESS

已成功检索描述符值。

MCDNN_STATUS_BAD_PARAM

constPack，param 或 isNULL 为 NULL，或者 paramLabel 无效。

5.2.1.17 mcdnnGetFusedOpsVariantParamPackAttribute()

此函数用于检索变量参数包描述符的设置。

```
mcdnnStatus_t mcdnnGetFusedOpsVariantParamPackAttribute(
const mcdnnFusedOpsVariantParamPack_t   varPack,
mcdnnFusedOpsVariantParamLabel_t        paramLabel,
void                                    *ptr);
```

参数

varPack

输入。指向 mcdnnFusedOps 变量参数包（varPack）描述符的指针。

paramLabel

输入。缓冲区指针参数的类型（在 varPack 描述符中）。更多信息，参见 mcdnnFusedOpsConstParamLabel_t。检索到的描述符值因类型而异。

ptr

输出。指针，指向此函数写入检索值的主机或设备内存。指针的数据类型和主机/设备内存位置取决于 paramLabel 的输入选择。更多信息，参见 mcdnnFusedOpsVariantParamLabel_t。

返回值

MCDNN_STATUS_SUCCESS

已成功检索描述符值。

MCDNN_STATUS_BAD_PARAM

varPack 或 ptr 为 NULL，或者 paramLabel 的设置值无效。

5.2.1.18 mcdnnMakeFusedOpsPlan()

在 mcdnnFusedOpsExecut() 实际执行融合操作之前，此函数确定要执行的最佳内核以及用户应分配的工作空间大小。

```
mcdnnStatus_t mcdnnMakeFusedOpsPlan(
mcdnnHandle_t                          handle,
mcdnnFusedOpsPlan_t                    plan,
const mcdnnFusedOpsConstParamPack_t constPack,
size_t                                 *workspaceSizeInBytes);
```

参数

handle

输入。指向 mcDNN 库上下文的指针。

plan

输入。指向已创建和初始化的 plan 描述符的指针。

constPack

输入。指向常量参数包描述符的指针。

workspaceSizeInBytes

输出。用户为执行此计划应分配的工作空间大小。

返回值

MCDNN_STATUS_BAD_PARAM

任意输入为 NULL，或者不支持 constPack 描述符中的 mcdnnFusedOps_t 类型。

MCDNN_STATUS_SUCCESS

此函数执行成功。

5.2.1.19 mcdnnSetFusedOpsConstParamPackAttribute()

此函数用于设置参数指针输入所指向的描述符。描述符的类型由 paramLabel 输入的枚举值指定。

```
mcdnnStatus_t mcdnnSetFusedOpsConstParamPackAttribute(
mcdnnFusedOpsConstParamPack_t   constPack,
mcdnnFusedOpsConstParamLabel_t  paramLabel,
const void                      *param);
```

参数

constPack

输入。mcdnnFusedOpsConstParamPack_t 不透明结构，用于存储卷积和激活等操作的各种 problem size 信息，例如形状，布局和张量类型，卷积和激活描述符，以及设置。

paramLabel

输入。此 setter 函数可以设置多种类型的描述符。参数输入指向描述符本身，此输入表明参数输入所指向的描述符类型。mcdnnFusedOpsConstParamLabel_t 枚举类型可用于选择描述符的类型。

param

输入。数据指针，指向与特定描述符关联的主机内存。此描述符的类型取决于 paramLabel 的值。有关更多信息，请参见 mcdnnFusedOpsConstParamLabel_t 中的表格。如果该指针设置为 NULL，则 mcDNN 库将记录空值。如果不是，则此指针指向的值（即，值或底部的不透明结构）将在 mcdnnSetFusedOpsConstParamPackAttribute() 运行期间复制到 constPack 中。

返回值

MCDNN_STATUS_SUCCESS

描述符设置成功。

MCDNN_STATUS_BAD_PARAM

constPack 为 NULL，或者 paramLabel 或 constPack 的 ops 设置无效。

5.2.1.20 mcdnnSetFusedOpsVariantParamPackAttribute()

此函数用于设置变量参数包描述符。

```
mcdnnStatus_t mcdnnSetFusedOpsVariantParamPackAttribute(
mcdnnFusedOpsVariantParamPack_t     varPack,
mcdnnFusedOpsVariantParamLabel_t    paramLabel,
void                                *ptr);
```

参数

varPack

输入。指向 mcdnnFusedOps 变量参数包（varPack）描述符的指针。

paramLabel

输入。通过此函数设置的缓冲区指针参数的类型（在 varPack 描述符中）。更多信息，参见 mcdnnFusedOpsConstParamLabel_t。

ptr

输入。主机或设备内存中的指针，指向设置的描述符参数值。指针的数据类型和主机/设备内存位置取决于 paramLabel 的输入选择。更多信息，参见 mcdnnFusedOpsVariantParamLabel_t。

返回值

MCDNN_STATUS_BAD_PARAM

varPack 为 NULL，或 paramLabel 的设置值无效。

MCDNN_STATUS_SUCCESS

描述符设置成功。

6 mcdnn_adv_infer

此实体包含所有其他功能与算法。包括循环神经网络（Rerrent Neural Network，RNN），CTC 损失（CTC Loss）和多头注意力机制（Multi-Head Attention）。mcdnn_adv_infer 库依赖于 mcdnn_ops_infer。

6.1 数据类型参考

以下为 mcdnn_adv_infer.h 中的数据类型参考。

6.1.1 不透明结构类型的指针

这些是指向 mcdnn_adv_infer.h 中不透明结构类型的指针。

6.1.1.1 mcdnnAttnDescriptor_t

mcdnnAttnDescriptor_t 是一个指针，指向包含多头注意力层的参数的不透明结构，例如：

- 权重和 bias 张量形状（线性投影前后的向量长度）
- 在调用函数来求正向响应和梯度的值时可以提前设置且不会更改的参数（注意力头数，softmax 平滑/锐化系数）
- 计算临时缓冲区大小所需的其他设置。

使用 mcdnnCreateAttnDescriptor() 函数创建注意力描述符对象的实例，使用 mcdnnDestroyAttnDescriptor() 删除已创建的描述符。使用 mcdnnSetAttnDescriptor() 函数配置描述符。

6.1.1.2 mcdnnPersistentRNNPlan_t

mcdnnPersistentRNNPlan_t 是指向不透明结构的指针，该结构包含执行动态持久 RNN 的 plan。mcdnnCreatePersistentRNNPlan() 用于创建和初始化一个实例。

6.1.1.3 mcdnnRNNDataDescriptor_t

mcdnnRNNDataDescriptor_t 是指向不透明结构的指针，该结构包含 RNN 数据集描述。mcdnnCreateRNNDataDescriptor() 函数用于创建一个实例，且必须使用 mcdnnSetRNNDataDescriptor() 初始化该实例。

6.1.1.4 mcdnnRNNDescriptor_t

mcdnnRNNDescriptor_t 是指向不透明结构的指针，该结构包含 RNN 操作的说明。mcdnnCreateRNNDescriptor() 用于创建实例。

6.1.1.5 mcdnnSeqDataDescriptor_t

mcdnnSeqDataDescriptor_t 是一个指针，指向包含序列数据容器或缓冲区参数的不透明结构。序列数据容器用于存储由 VECT 维定义的固定大小的向量。向量排列于另外三个维度：TIME，BATCH 和 BEAM。TIME 维用于将向量捆绑到向量序列中。实际序列可以短于 TIME 维，因此需要有关每个序列长度以及如何保存未使用（填充）向量的附加信息。

假定序列数据容器已完全压缩。当向量按地址的升序排布时，TIME、BATCH、BEAM 维可以按任何顺序排列。可以使用六种数据布局（TIME、BATCH 和 BEAM 的排列）。

mcdnnSeqDataDescriptor_t 对象包含以下参数：

- 向量使用的数据类型
- TIME、BATCH、BEAM 和 VECT 维度
- 数据布局
- 沿 TIME 维度的每个序列的长度
- 要复制到输出填充向量的可选值

使用 mcdnnCreateSeqDataDescriptor() 函数创建序列数据描述符对象的实例，使用 mcdnnDestroySeqDataDescriptor() 删除已创建的描述符。使用 mcdnnSetSeqDataDescriptor() 函数配置描述符。

多头注意力 API 函数使用此描述符。

6.1.2 枚举类型

以下为 mcdnn_adv_infer.h 中的枚举类型。

6.1.2.1 mcdnnDirectionMode_t

mcdnnDirectionMode_t 是一种枚举类型，用于指定 mcdnnRNNForwardInference()，mcdnnRNNForwardTrain()，mcdnnRNNBackwardData() 和 mcdnnRNNBackwardWeights() 函数中的循环模式。

值

MCDNN_UNIDIRECTIONAL

网络从第一个输入循环迭代到最后一个输入。

MCDNN_BIDIRECTIONAL

网络的每一层都分别从第一个输入循环迭代到最后一个输入，并从最后一个输入迭代到第一个输入。两个输出在每次迭代时连接，提供层的输出。

6.1.2.2 mcdnnForwardMode_t

mcdnnForwardMode_t 是一种枚举类型，指定 RNN API 中的推理或训练模式。此参数允许 mcDNN 库更精确地调整工作空间缓冲区的大小，该缓冲区的推理和训练方案可能不同。

值

MCDNN_FWD_MODE_INFERENCE

选择推理模式。

MCDNN_FWD_MODE_TRAINING

选择训练模式。

6.1.2.3 mcdnnMultiHeadAttnWeightKind_t

mcdnnMultiHeadAttnWeightKind_t 是一种枚举类型，用于指定 mcdnnGetMultiHeadAttnWeight() 函数中的一组权重或 bias。

值

MCDNN_MH_ATTN_Q_WEIGHTS

选择查询的输入投影权重。

MCDNN_MH_ATTN_K_WEIGHTS

选择键的输入投影权重。

MCDNN_MH_ATTN_V_WEIGHTS

选择值的输入投影权重。

MCDNN_MH_ATTN_O_WEIGHTS

选择输出投影权重。

MCDNN_MH_ATTN_Q_BIASES

选择查询的输入投影 bias。

MCDNN_MH_ATTN_K_BIASES

选择键的输入投影 bias。

MCDNN_MH_ATTN_V_BIASES

选择值的输入投影 bias。

MCDNN_MH_ATTN_O_BIASES

选择输出投影权重。

6.1.2.4 mcdnnRNNBiasMode_t

mcdnnRNNBiasMode_t 是一种枚举类型，用于指定 RNN 函数的 bias 向量数。有关基于 bias 模式的每个神经元类型的公式，请参见 mcdnnRNNMode_t 枚举类型的描述。

值

MCDNN_RNN_NO_BIAS

应用不使用 bias 的 RNN 单元（cell）公式。

MCDNN_RNN_SINGLE_INP_BIAS

应用在输入 GEMM 中使用一个输入 bias 向量的 RNN 单元公式。

MCDNN_RNN_DOUBLE_BIAS

应用使用两个 bias 向量的 RNN 单元公式。

MCDNN_RNN_SINGLE_REC_BIAS

应用在循环 GEMM 中使用一个循环 bias 向量的 RNN 单元公式。

6.1.2.5 mcdnnRNNClipMode_t

mcdnnRNNClipMode_t 是用于选择 LSTM 单元裁剪（cell clipping）模式的枚举类型。它与 mcdnnRNNSetClim()、mcdnnRNGetClim() 函数一起在 LSTM 单元内部使用。

值

MCDNN_RNN_CLIP_NONE

禁用 LSTM 单元裁剪。

MCDNN_RNN_CLIP_MINMAX

启用 LSTM 单元裁剪。

6.1.2.6 mcdnnRNNDataLayout_t

mcdnnRNNDataLayout_t 是用于选择 RNN 数据布局的枚举类型。它用于 mcdnnGetRNNDataDescriptor() 和 mcdnnSetRNNDataDescriptor() API 调用。

值

MCDNN_RNN_DATA_LAYOUT_SEQ_MAJOR_UNPACKED

使用从一个时间步（time-step）到下一个时间步的外部步幅，来填充数据布局。

MCDNN_RNN_DATA_LAYOUT_SEQ_MAJOR_PACKED

序列长度按基本 RNN API 中的顺序进行排序和填充。

MCDNN_RNN_DATA_LAYOUT_BATCH_MAJOR_UNPACKED

使用从一个 batch 到下一个 batch 的外部步幅，来填充数据布局。

6.1.2.7 mcdnnRNNInputMode_t

mcdnnRNNInputMode_t 是一种枚举类型，用于指定 mcdnnRNNForwardInference()，mcdnnRNNForwardTrain()，mcdnnRNNBackwardData() 和 mcdnnRNNBackwardWeights() 函数中第一层的行为。

值

MCDNN_LINEAR_INPUT

对第一个循环层的输入执行矩阵乘法加 bias。

MCDNN_SKIP_INPUT

不对第一个循环层的输入执行任何操作。如果使用 MCDNN_SKIP_INPUT，则输入张量的前导维度（leading dimension）必须等于网络的隐藏状态大小。

6.1.2.8 mcdnnRNNMode_t

mcdnnRNNMode_t 是一种枚举类型，用于指定 mcdnnRNNForwardInference()，mcdnnRNNForwardTrain()，mcdnnRNNBackwardData() 和 mcdnnRNNBackwardWeights() 函数中使用的网络类型。

值

MCDNN_RNN_RELU

具有 ReLU 激活函数的单门控循环神经网络。

在正向传递中，给定迭代的输出 h_t 可以从循环输入 h_{t-1} 和上一层输入 x_t 计算得到，给定矩阵 `W`，`R` 和 bias 向量，其中 $ReLU\left(x\right) = \max\left(x,\ 0\right)$。如果 `rnnDesc` 中 `mcdnnRNNBiasMode_t`

biasMode 为 MCDNN_RNN_DOUBLE_BIAS （默认模式），则应用以下带有 b_W 和 b_R bias 的方程式：

$$h_t = ReLU\ (W_i x_t + R_i h_{t-1} + b_{Wi} + b_{Ri})$$

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_SINGLE_INP_BIAS 或 MCDNN_RNN_SINGLE_REC_BIAS，则应用以下带有 bb bias 的方程式：

$$h_t = ReLU\ (W_i x_t + R_i h_{t-1} + b_i)$$

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_NO_BIAS，则应用以下方程式：

$$h_t = ReLU\ (W_i x_t + R_i h_{t-1})$$

MCDNN_RNN_TANH

具有 tanh 激活函数的单门控循环神经网络。

在正向传递中，给定迭代的输出 h_t 可以从循环输入 h_{t-1} 和上一层输入 x_t 计算得到，给定矩阵 W，R 和 bias 向量，其中 tanh 是双曲正切函数。

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_DOUBLE_BIAS （默认模式），则应用以下带有 b_W 和 b_R bias 的方程式：

$$h_t = \tanh\ (W_i x_t + R_i h_{t-1} + b_{Wi} + b_{Ri})$$

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_SINGLE_INP_BIAS 或 MCDNN_RNN_SINGLE_REC_BIAS，则应用以下带有 bb bias 的方程式：

$$h_t = \tanh\ (W_i x_t + R_i h_{t-1} + b_i)$$

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_NO_BIAS，则应用以下方程式：

$$h_t = \tanh\ (W_i x_t + R_i h_{t-1})$$

MCDNN_LSTM

没有窥孔连接（peephole connection）的四门长短时记忆（Long Short-Term Memory，LSTM）网络。

在正向传递中，给定迭代的输出 h_t 和单元输出 c_t，可以从循环输入 h_{t-1}，单元输入 c_{t-1} 和上一层输入 x_t 计算得到，给定矩阵 W，R 和 bias 向量。

此外，还适用以下内容：

- σ 是以下方程式的 sigmoid 运算符：$\sigma(x) = 1\ /\ (1 + e^{-x})$，
- $\circ$ 表示逐点乘法，
- tanh 是双曲正切函数，
- i_t, f_t, o_t, c_t 分别代表输入门，遗忘门，输出门和新门。

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_DOUBLE_BIAS （默认模式），则应用以下带有 b_W 和 b_R bias 的方程式：

$$i_t = \sigma\ (W_i x_t + R_i h_{t-1} + b_{Wi} + b_{Ri})$$
$$f_t = \sigma\ (W_f x_t + R_f h_{t-1} + b_{Wf} + b_{Rf})$$
$$o_t = \sigma\ (W_o x_t + R_o h_{t-1} + b_{Wo} + b_{Ro})$$
$$c_t = \tanh\ (W_c x_t + R_c h_{t-1} + b_{Wc} + b_{Rc})$$
$$c_t = f_t \circ c_{t-1} + i_t \circ c_t$$
$$h_t = o_t \circ \tanh\ (c_t)$$

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_SINGLE_INP_BIAS 或 MCDNN_RNN_SINGLE_REC_BIAS，则应用以下带有 bb bias 的方程式：

$$i_t = \sigma\left(W_i x_t + R_i h_{t-1} + b_i\right)$$
$$f_t = \sigma\left(W_f x_t + R_f h_{t-1} + b_f\right)$$
$$o_t = \sigma\left(W_o x_t + R_o h_{t-1} + b_o\right)$$
$$c_t = \tanh\left(W_c x_t + R_c h_{t-1} + b_c\right)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ c_t$$
$$h_t = o_t \circ \tanh\left(c_t\right)$$

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_NO_BIAS，则应用以下方程式：

$$i_t = \sigma\left(W_i x_t + R_i h_{t-1}\right)$$
$$f_t = \sigma\left(W_f x_t + R_f h_{t-1}\right)$$
$$o_t = \sigma\left(W_o x_t + R_o h_{t-1}\right)$$
$$c_t = \tanh\left(W_c x_t + R_c h_{t-1}\right)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ c_t$$
$$h_t = o_t \circ \tanh\left(c_t\right)$$

MCDNN_GRU

由门控循环单元组成的三门网络。

在正向传递中，给定迭代的输出 h_t 可以从循环输入 h_{t-1} 和上一层输入 x_t 计算得到，给定矩阵 W，R 和 bias 向量。

此外，还适用以下内容：

- σ 是以下方程式的 sigmoid 运算符：$\sigma(x) = 1 / (1 + e^{-x})$，
- $\circ$ 表示逐点乘法，
- tanh 是双曲正切函数，
- i_t, r_t,, h_t 分别代表输入门，重置门和新门。

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_DOUBLE_BIAS（默认模式），则应用以下带有 b_W 和 b_R bias 的方程式：

$$i_t = \sigma\left(W_i x_t + R_i h_{t-1} + b_{Wi} + b_{Ru}\right)$$
$$r_t = \sigma\left(W_r x_t + R_r h_{t-1} + b_{Wr} + b_{Rr}\right)$$
$$h_t = \tanh\left(W_h x_t + r_t \circ \left(R_h h_{t-1} + b_{Rh}\right) + b_{Wh}\right)$$
$$h_t = \left(1 - i_t\right) \circ h_t + i_t \circ h_{t-1}$$

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_SINGLE_INP_BIAS，则应用以下带有 bb bias 的方程式：

$$i_t = \sigma\left(W_i x_t + R_i h_{t-1} + b_i\right)$$
$$r_t = \sigma\left(W_r x_t + R_r h_{t-1} + b_r\right)$$
$$h_t = \tanh\left(W_h x_t + r_t \circ \left(R_h h_{t-1}\right) + b_{Wh}\right)$$
$$h_t = \left(1 - i_t\right) \circ h_t + i_t \circ h_{t-1}$$

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_SINGLE_REC_BIAS，则应用以下带有 bb bias 的方程式：

$$i_t = \sigma\left(W_i x_t + R_i h_{t-1} + b_i\right)$$
$$r_t = \sigma\left(W_r x_t + R_r h_{t-1} + b_r\right)$$
$$h_t = \tanh\left(W_h x_t + r_t \circ \left(R_h h_{t-1} + b_{Rh}\right)\right)$$
$$h_t = \left(1 - i_t\right) \circ h_t + i_t \circ h_{t-1}$$

如果 rnnDesc 中 mcdnnRNNBiasMode_t biasMode 为 MCDNN_RNN_NO_BIAS，则应用以下方程式：

$$i_t = \sigma\left(W_i x_t + R_i h_{t-1}\right)$$
$$r_t = \sigma\left(W_r x_t + R_r h_{t-1}\right)$$
$$h_t = \tanh\left(W_h x_t + r_t \circ \left(R_h h_{t-1}\right)\right)$$
$$h_t = \left(1 - i_t\right) \circ h_t + i_t \circ h_{t-1}$$

6.1.2.9 mcdnnRNNPaddingMode_t

mcdnnRNNPaddingMode_t 是一种枚举类型，用于启用或禁用填充输入/输出。

值

MCDNN_RNN_PADDED_IO_DISABLED

禁用填充输入/输出。

MCDNN_RNN_PADDED_IO_ENABLED

启用填充输入/输出。

6.1.2.10 mcdnnSeqDataAxis_t

mcdnnSeqDataAxis_t 是一种枚举类型，用于索引被传入 mcdnnSetSeqDataDescriptor() 函数的 dimA[] 参数中的活动维度，以配置 mcdnnSeqDataDescriptor_t 类型的序列数据描述符。mcdnnSeqDataAxis_t 常量也用于 mcdnnSetSeqDataDescriptor() 调用的 axis[] 参数中，以定义内存中序列数据缓冲区的布局。

有关如何使用 mcdnnSeqDataAxis_t 枚举类型的详细说明，请参见 mcdnnSetSeqDataDescriptor()。

MCDNN_SEQDATA_DIM_COUNT 宏定义 mcdnnSeqDataAxis_t 枚举类型中的常量数。当前该值设为 4。

值

MCDNN_SEQDATA_TIME_DIM

标识 TIME（序列长度）维度或指定数据布局中的 TIME。

MCDNN_SEQDATA_BATCH_DIM

标识 BATCH 维度或指定数据布局中的 BATCH。

MCDNN_SEQDATA_BEAM_DIM

标识 BEAM 维度或指定数据布局中的 BEAM。

MCDNN_SEQDATA_VECT_DIM

标识 VECT（向量）维度或指定数据布局中的 VECT。

6.2 API 参考

6.2.1 API 函数

以下为 mcdnn_adv_infer.h 中的 API 函数。

6.2.1.1 mcdnnAdvInferVersionCheck()

此函数用于查看库的 AdvInfer 子集的版本是否与其他子库一致。

```
mcdnnStatus_t mcdnnAdvInferVersionCheck(void)
```

返回值

MCDNN_STATUS_SUCCESS

版本与其他子库一致。

MCDNN_STATUS_VERSION_MISMATCH

AdvInfer 的版本与其他子库不一致。用户应检查并确保所有子组件安装版本一致。

6.2.1.2 mcdnnBuildRNNDynamic()

当选择 MCDNN_RNN_ALGO_PERST_DYNAMIC algo 时，此函数使用 MXMACA 运行时编译库编译 RNN 持久性代码。代码根据当前 GPU 和特定超参数（miniBatch）进行定制。此函数调用的运行时消耗大，因此不能频繁调用。请注意，MCDNN_RNN_ALGO_PRESERT_DYNAMIC algo 不支持 batch 中可变长度序列。

```
mcdnnStatus_t mcdnnBuildRNNDynamic(
    mcdnnHandle_t handle,
    mcdnnRNNDescriptor_t rnnDesc,
    int32_t miniBatch);
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

miniBatch

输入。一个 batch 中的确切序列数。

返回值

MCDNN_STATUS_SUCCESS

代码已成功构建并链接。

MCDNN_STATUS_MAPPING_ERROR

GPU/MXMACA 资源（如纹理对象，共享内存或零拷贝内存）没有所需的空间，或者用户资源与 mcDNN 内部资源不匹配。例如，在调用 mcdnnSetStream() 时，可能会出现资源不匹配。调用 mcdnnCreate() 时，用户提供的 MXMACA 流与 mcDNN 句柄中实例化的内部 MXMACA 事件之间可能不匹配。此错误状态与纹理尺寸，共享内存大小或零拷贝内存可用性相关时，可能无法纠正。如果 mcdnnSetStream() 返回 MCDNN_STATUS_MAPPING_ERROR，则通常是可纠正的，但是，这意味着 mcDNN 句柄是在一个 GPU 上创建的，且传入此函数的用户流与另一个 GPU 相关联。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

MCDNN_STATUS_RUNTIME_PREREQUISITE_MISSING

找不到必备的运行时库。

MCDNN_STATUS_NOT_SUPPORTED

当前超参数无效。

6.2.1.3 mcdnnCreateAttnDescriptor()

此函数通过为不透明的注意力描述符对象分配主机内存，并初始化所有描述符字段，来创建该对象的一个实例。当无法分配注意力描述符对象时，函数将 NULL 写入 attnDesc。

```
mcdnnStatus_t mcdnnCreateAttnDescriptor(mcdnnAttnDescriptor_t *attnDesc);
```

使用 mcdnnSetAttnDescriptor() 函数配置注意力描述符，并使用 mcdnnDestroyAttnDescriptor() 销毁该描述符并释放分配的内存。

参数

attnDesc

输出。指针，指向要写入新创建的注意力描述符的地址。

返回值

MCDNN_STATUS_SUCCESS

描述符对象创建成功。

MCDNN_STATUS_BAD_PARAM

遇到无效的输入参数（attnDes=NULL）。

MCDNN_STATUS_ALLOC_FAILED

内存分配失败。

6.2.1.4 mcdnnCreatePersistentRNNPlan()

此函数保留 API，不再单独实现。使用 mcdnnBuildRNNDynamic() 代替 mcdnnCreatePersistentRNNPlan()。

```
mcdnnStatus_t mcdnnCreatePersistentRNNPlan(
    mcdnnRNNDescriptor_t         rnnDesc,
    const int                    minibatch,
    const mcdnnDataType_t        dataType,
    mcdnnPersistentRNNPlan_t    *plan)
```

使用 MCDNN_RNN_ALGO_PERSIST_DYNAMIC algo 时，此函数创建执行持久 RNN 的计划。此计划是根据当前的 GPU 和 RNN 模型超参数定制的。此函数调用的运行时消耗大，因此不能频繁调用。但是，每次在 minibatch 中更改输入向量的数量时，用户都必须调用 mcdnnCreatePersistentRNNPlan()。更多信息，参见 mcdnnRNNDescriptor_t，mcdnnDataType_t，mcdnnPersistentRNNPlan_t。

参数

rnnDesc

输入。已初始化的 RNN 描述符。

minibatch

输入。一个 batch 中的确切向量数。

dataType

输入。为 RNN 权重/偏差以及输入和输出数据指定数据类型。

plan

输出。指针，指向要写入新创建的 RNN 持久性计划的地址。

返回值

MCDNN_STATUS_SUCCESS

对象创建成功。

MCDNN_STATUS_MAPPING_ERROR

GPU/MXMACA 资源（如纹理对象，共享内存或零拷贝内存）没有所需的空间，或者用户资源与 mcDNN 内部资源不匹配。例如，在调用 mcdnnSetStream() 时，可能会出现资源不匹配。调用 mcdnnCreate() 时，用户提供的 MXMACA 流与 mcDNN 句柄中实例化的内部 MXMACA 事件之间可能不匹配。

此错误状态与纹理尺寸，共享内存大小或零拷贝内存可用性相关时，可能无法纠正。如果 mcdnnSetStream() 返回 MCDNN_STATUS_MAPPING_ERROR，则通常是可纠正的，但是，这意味着 mcDNN 句柄是在一个 GPU 上创建的，且传入此函数的用户流与另一个 GPU 相关联。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

MCDNN_STATUS_RUNTIME_PREREQUISITE_MISSING

找不到必备的运行时库。

MCDNN_STATUS_NOT_SUPPORTED

当前超参数无效。

6.2.1.5 mcdnnCreateRNNDataDescriptor()

此函数通过分配保存 RNN 数据描述符不透明结构所需内存的方式，创建 RNN 数据描述符对象。

```
mcdnnStatus_t mcdnnCreateRNNDataDescriptor(
    mcdnnRNNDataDescriptor_t *RNNDataDesc)
```

参数

RNNDataDesc

输出。指针，指向要写入新创建的 RNN 数据描述符的地址。

返回值

MCDNN_STATUS_SUCCESS

RNN 数据描述符对象已创建成功。

MCDNN_STATUS_BAD_PARAM

RNNDataDesc 参数为 NULL。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

6.2.1.6 mcdnnCreateRNNDescriptor()

此函数通过分配保存通用 RNN 描述符不透明结构所需内存的方式，创建通用 RNN 描述符对象。

```
mcdnnStatus_t mcdnnCreateRNNDescriptor(
    mcdnnRNNDescriptor_t    *rnnDesc)
```

参数

rnnDesc

输出。指针，指向要写入新创建的 RNN 描述符的地址。

返回值

MCDNN_STATUS_SUCCESS

已成功创建对象。

MCDNN_STATUS_BAD_PARAM

rnnDesc 参数为 NULL。

MCDNN_STATUS_ALLOC_FAILED

资源无法分配。

6.2.1.7 mcdnnCreateSeqDataDescriptor()

此函数通过为不透明的序列数据描述符对象分配主机内存，并初始化所有描述符字段，来创建该对象的一个实例。当无法分配序列数据描述符对象时，函数将 NULL 写入 seqDataDesc。

```
mcdnnStatus_t mcdnnCreateSeqDataDescriptor(mcdnnSeqDataDescriptor_t
↪*seqDataDesc);
```

使用 mcdnnSetSeqDataDescriptor() 函数配置序列数据描述符，并使用 mcdnnDestroySeqDataDescriptor() 销毁该描述符并释放分配的内存。

参数

seqDataDesc

输出。指针，指向要写入新创建的序列数据描述符的地址。

返回值

MCDNN_STATUS_SUCCESS

已成功创建描述符对象。

MCDNN_STATUS_BAD_PARAM

遇到无效的输入参数（seqDataDesc=NULL）。

MCDNN_STATUS_ALLOC_FAILED

内存分配失败。

6.2.1.8 mcdnnDestroyAttnDescriptor()

此函数用于销毁注意力描述符对象，并释放其内存。attnDesc 参数可以为 NULL。使用 NULL 参数调用 mcdnnDestroyAttnDescriptor() 是一个空操作（NOP）。

```
mcdnnStatus_t mcdnnDestroyAttnDescriptor(mcdnnAttnDescriptor_t attnDesc);
```

mcdnnDestroyAttnDescriptor() 函数无法检测 attnDesc 参数是否包含有效的地址。如果传入无效的指针，mcdnnCreateAttnDescriptor() 函数未返回，或者在有效地址的 double deletion 场景下，将出现未定义的行为。

参数

attnDesc

输入。指针，指向要销毁的注意力描述符对象。

返回值

MCDNN_STATUS_SUCCESS

描述符已销毁成功。

6.2.1.9 mcdnnDestroyPersistentRNNPlan()

此函数保留 API，不再单独实现。此函数用于销毁已创建的 RNN 持久性计划对象。使用 NULL 参数调用 mcdnnDestroyPersistentRNNPlan() 是一个空操作 (NOP)。

```
mcdnnStatus_t mcdnnDestroyPersistentRNNPlan(
    mcdnnPersistentRNNPlan_t plan)
```

mcdnnDestroyPersistentRNNPlan() 函数无法检测 plan 参数是否包含有效的地址。如果传入无效的指针，mcdnnCreatePersistentRNNPlan() 函数未返回，或者在有效地址的 double deletion 场景下，将出现未定义的行为。

参数

plan

输入。指针，指向要销毁的 RNN 持久性计划对象。

返回值

MCDNN_STATUS_SUCCESS

对象已销毁成功。

6.2.1.10 mcdnnDestroyRNNDataDescriptor()

此函数用于销毁已创建的 RNN 数据描述符对象。使用 NULL 参数调用 mcdnnDestroyRNNDataDescriptor() 是一个空操作（NOP）

```
mcdnnStatus_t mcdnnDestroyRNNDataDescriptor(
    mcdnnRNNDataDescriptor_t RNNDataDesc)
```

mcdnnDestroyRNNDataDescriptor() 函数无法检测 RNNDataDesc 参数是否包含有效的地址。如果传入无效的指针，mcdnnCreateRNNDataDescriptor() 函数未返回，或者在有效地址的 double deletion 场景下，将出现未定义的行为。

参数

RNNDataDesc

输入。指针，指向要销毁的 RNN 数据描述符对象。

返回值

MCDNN_STATUS_SUCCESS

RNN 数据描述符对象销毁成功。

6.2.1.11 mcdnnDestroyRNNDescriptor()

此函数用于销毁已创建的 RNN 描述符对象。使用 NULL 参数调用 mcdnnDestroyRNNDescriptor() 是一个空操作（NOP）

```
mcdnnStatus_t mcdnnDestroyRNNDescriptor(
    mcdnnRNNDescriptor_t rnnDesc)
```

mcdnnDestroyRNNDescriptor() 函数无法检测 rnnDesc 参数是否包含有效的地址。如果传入无效的指针，mcdnnCreateRNNDescriptor() 函数未返回，或者在有效地址的 double deletion 场景下，将出现未定义的行为。

参数

rnnDesc

输入。指针，指向要销毁的 RNN 描述符对象。

返回值

MCDNN_STATUS_SUCCESS

对象已销毁成功。

6.2.1.12 mcdnnDestroySeqDataDescriptor()

此函数用于销毁序列描述符对象，并释放其内存。seqDataDesc 参数可以为 NULL。使用 NULL 参数调用 mcdnnDestroySeqDataDescriptor() 是一个空操作（NOP）。

```
mcdnnStatus_t mcdnnDestroySeqDataDescriptor(mcdnnSeqDataDescriptor_t
↪seqDataDesc);
```

mcdnnDestroySeqDataDescriptor() 函数无法检测 seqDataDesc 参数是否包含有效的地址。如果传入无效的指针，mcdnnCreateSeqDataDescriptor() 函数未返回，或者在有效地址的 double deletion 场景下，将出现未定义的行为。

参数

seqDataDesc

输入。指针，指向要销毁的序列数据描述符对象。

返回值

MCDNN_STATUS_SUCCESS

描述符已销毁成功。

6.2.1.13 mcdnnFindRNNForwardInferenceAlgorithmEx()

此函数保留 API，不再单独实现。此函数使用用户分配的 GPU 内存尝试 mcdnnRNNForwardInference() 的所有可用 mcDNN 算法。它将影响算法性能的参数输出到用户分配的 mcdnnAlgorithmPerformance_t 数组。这些参数指标以一种有序的方式写入，其中第一个元素的计算时间最短。

```
mcdnnStatus_t mcdnnFindRNNForwardInferenceAlgorithmEx(
    mcdnnHandle_t                    handle,
    const mcdnnRNNDescriptor_t       rnnDesc,
    const int                        seqLength,
    const mcdnnTensorDescriptor_t   *xDesc,
    const void                      *x,
    const mcdnnTensorDescriptor_t    hxDesc,
    const void                      *hx,
    const mcdnnTensorDescriptor_t    cxDesc,
    const void                      *cx,
    const mcdnnFilterDescriptor_t    wDesc,
    const void                      *w,
    const mcdnnTensorDescriptor_t    *yDesc,
    void                            *y,
    const mcdnnTensorDescriptor_t    hyDesc,
```

(下页继续)

(续上页)

```
    void                              *hy,
    const mcdnnTensorDescriptor_t      cyDesc,
    void                              *cy,
    const float                       findIntensity,
    const int                         requestedAlgoCount,
    int                               *returnedAlgoCount,
    mcdnnAlgorithmPerformance_t       *perfResults,
    void                              *workspace,
    size_t                             workSpaceSizeInBytes);
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

seqLength

输入。要展开的迭代次数。seqLength 的值不能超过 mcdnnGetRNNWorkspaceSize() 函数中用于查询执行 RNN 所需的工作空间大小。

xDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代的输入（一个迭代一个描述符）。张量的第一维（批大小）可能从元素 n 减少到元素 1，但不会增加。每个张量描述符必须具有相同的第二维（向量长度）。

x

输入。数据指针，指向与 xDesc 数组中张量描述符关联的 GPU 内存。数据压缩会在迭代 n 的最后一个元素结束后直接从迭代 n+1 的第一个元素开始连续进行。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。

张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。

cxDesc

输入。完全压缩的的张量描述符，描述 LSTM 网络的初始单元（cell）状态。

张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

cx

输入。数据指针，指向与张量描述符 cxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态将初始化为零。

wDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

yDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代的输出。

张量的第二维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第二维应与 hiddenSize 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第二维应是 hiddenSize 参数的 2 倍。

张量 n 的第一维必须与 xDesc 中张量 n 的第一维匹配。

y

输出。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。数据压缩会在迭代 n 的最后一个元素结束后直接从迭代 n+1 的第一个元素开始连续进行。

hyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终隐藏状态。

张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hy

输出。数据指针，指向与张量描述符 hyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终隐藏状态。

cyDesc

输入。完全压缩的张量描述符，描述 LSTM 网络的最终单元状态。

张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

cy

输出。数据指针，指向与张量描述符 cyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终单元状态。

findIntensity

输入。此输入之前未使用。通过选择要搜索的空间占大的笛卡尔积（Cartesian product）空间的百分比，它在 mcDNN 中用于控制 RNN 查找算法的总体运行时。

- 在 (0,1.] 范围内设置 findIntensity，则是设置要搜索的空间占整个 RNN 搜索空间的百分比。
  - 当 findIntensity 设置为 1.0 时，将对所有 RNN 参数执行完整搜索。

  - 当 findIntensity 设置为 0.0f 时，将执行快速，最小的搜索。此设置可获得最佳运行时。

  但是，在这种情况下，此函数返回的参数不会获得算法的最佳性能；更大范围的搜索可能会发现更好的性能参数。此选项将最多执行三个已配置的 RNN problem 的实例。运行时会随 RNN problem size 而成比例地变化，在其他情况下也是如此，因此不能保证明确的时间限制。

- 在 [-1.,0) 范围内设置 findIntensity，则是设置要搜索的空间占归约笛卡尔积空间的百分比。为获得良好的性能，已启发式地选择此归约搜索空间。设置为-1.0 表示在此归约搜索空间上进行完整搜索。
- [-1,1] 范围以外的值被截断到 [-1,1] 范围中，然后按照上述说明进行解释。
- 此函数对大参数空间上的单个 RNN 执行进行乘积—一个参数组合一次执行。此函数返回的时间是运行总时间。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- hxDesc、cxDesc、wDesc、hyDesc、cyDesc 描述符中或者 xDesc、yDesc 描述符中至少有一个是无效的。
- xDesc、hxDesc、cxDesc、wDesc、yDesc、hyDesc、cyDesc 中至少有一个描述符的步幅或维度不正确。
- workSpaceSizeInBytes 太小。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

6.2.1.14 mcdnnGetAttnDescriptor()

此函数用于从已创建的注意力描述符中检索设置。当不需要检索到的值时，用户可以将 NULL 分配给除 attnDesc 之外的任意指针。

```
mcdnnStatus_t mcdnnGetAttnDescriptor(
    mcdnnAttnDescriptor_t attnDesc,
    unsigned *attnMode,
    int *nHeads,
    double *smScaler,
    mcdnnDataType_t *dataType,
    mcdnnDataType_t *computePrec,
    mcdnnMathType_t *mathType,
    mcdnnDropoutDescriptor_t *attnDropoutDesc,
    mcdnnDropoutDescriptor_t *postDropoutDesc,
    int *qSize,
    int *kSize,
    int *vSize,
    int *qProjSize,
    int *kProjSize,
    int *vProjSize,
    int *oProjSize,
    int *qoMaxSeqLength,
    int *kvMaxSeqLength,
    int *maxBatchSize,
    int *maxBeamSize);
```

参数

attnDesc 输入。注意力描述符。

attnMode

输出。指向二进制注意力标记存储的指针。

nHeads

输出。指向注意力头数存储的指针。

smScaler

输出。指向 softmax 平滑/锐化系数存储的指针。

dataType

输出。注意力、权重、序列数据输入和输出的数据类型。

computePrec

输出。指向计算精度存储的指针。

mathType

输出。MMA 设置。

attnDropoutDesc

输出。应用于 softmax 输出的丢弃操作的描述符。

postDropoutDesc

输出。应用于多头注意力输出的丢弃操作的描述符。

qSize, kSize, vSize

输出。Q、K、V 嵌入向量长度。

qProjSize, kProjSize, vProjSize

输出。输入投影（projection）后的 Q、K、V 嵌入向量长度。

oProjSize

输出。存储投影后输出向量长度的指针。

qoMaxSeqLength

输出。与 Q、O、dQ、dO 输入和输出相关的序列数据描述符中的最大序列长度。

kvMaxSeqLength

输出。与 K、V、dK、dV 输入和输出相关的序列数据描述符中的最大序列长度。

maxBatchSize

输出。mcdnnSeqDataDescriptor_t 容器中的最大批大小。

maxBeamSize

输出。mcdnnSeqDataDescriptor_t 容器中的最大束宽（beam size）。

返回值

MCDNN_STATUS_SUCCESS

已成功检索到请求的注意力描述符字段。

MCDNN_STATUS_BAD_PARAM

找到一个无效的输入变量。

6.2.1.15 mcdnnGetMultiHeadAttnBuffers()

此函数计算以下函数使用的权重、工作空间和预留空间缓冲区大小：

- mcdnnMultiHeadAttnForward()
- mcdnnMultiHeadAttnBackwardData()
- mcdnnMultiHeadAttnBackwardWeights()

```
mcdnnStatus_t mcdnnGetMultiHeadAttnBuffers(
    mcdnnHandle_t handle,
    const mcdnnAttnDescriptor_t attnDesc,
    size_t *weightSizeInBytes,
    size_t *workSpaceSizeInBytes,
    size_t *reserveSpaceSizeInBytes);
```

将 NULL 分配给 reserveSpaceSizeInBytes 参数，表示用户不会调用多头注意力梯度函数：mcdnnMultiHeadAttnBackwardData() 和 mcdnnMultiHeadAttnBackwardWeights()。此状态出现在推理模式中。

注解： 不能将 NULL 分配给 weightSizeInBytes 和 workSpaceSizeInBytes 指针。

用户必须使用 mcMalloc() 和已报告的缓冲区大小来分配 GPU 内存中的权重、工作空间和预留空间缓冲区大小。缓冲区也可以从已分配内存的较大块中分离出来，但缓冲区地址必须至少对齐 16B。

工作空间缓冲区用于临时存储。其内容可以在相应 API 启动的所有 GPU 内核完成后丢弃或修改。预留空间缓冲区用于将中间结果从 mcdnnMultiHeadAttnForward() 传送到 mcdnnMultiHeadAttnBackwardData()，以及从 mcdnnMultiHeadAttnBackwardData() 传送到 mcdnnMultiHeadAttnBackwardWeights()。在上述三个多头注意力 API 函数启动的所有 GPU 内核完成之前，预留空间缓冲区的内容无法修改。

所有多头注意力权重和 bias 张量存储在单个权重缓冲区中。为优化速度，mcDNN API 可能会根据提供的注意力参数更改张量布局及其在权重缓冲区中的相对位置。使用 mcdnnGetMultiHeadAttnWeights() 函数获取每个权重或 bias 张量的起始地址和形状。

参数

handle

输入。当前的 mcDNN 上下文句柄。

attnDesc

输入。指针，指向已初始化的注意力描述符。

weightSizeInBytes

输出。存储所有多头注意力可训练参数所需的最小缓冲区大小。

workSpaceSizeInBytes

输出。保存正向和梯度多头注意力 API 使用的所有临时表面（temporary surface）所需的最小缓冲区大小。

reserveSpaceSizeInBytes

输出。存储在正向和反向（梯度）多头注意力函数之间交换的所有中间数据所需的最小缓冲区大小。在推理模式中将该参数设置为 NULL，表示不会调用梯度 API。

返回值

MCDNN_STATUS_ARCH_MISMATCH

GPU 设备不支持输入数据类型。

MCDNN_STATUS_SUCCESS

已成功计算请求的缓冲区大小。

MCDNN_STATUS_BAD_PARAM

找到一个无效的输入参数。

6.2.1.16 mcdnnGetMultiHeadAttnWeights()

该函数用于获取权重或 bias 张量的形状。它还用于检索权重缓冲区中张量数据的起始地址。使用 wKind 参数选择一个特定张量。有关更多枚举类型的说明信息，请参见 mcdnnMultiHeadAttnWeightKind_t。

```
mcdnnStatus_t mcdnnGetMultiHeadAttnWeights(
    mcdnnHandle_t handle,
    const mcdnnAttnDescriptor_t attnDesc,
    mcdnnMultiHeadAttnWeightKind_t wKind,
    size_t weightSizeInBytes,
    const void *weights,
    mcdnnTensorDescriptor_t wDesc,
    void **wAddr);
```

在注意力描述符中设置 MCDNN_ATTN_ENABLE_PROJ_BIASES 标记时，在输入和输出投影中使用 bias。有关控制投影 bias 的标记的说明，请参见 mcdnnSetAttnDescriptor()。

当相应的权重或 bias 张量不存在时，函数将 NULL 写入 wAddr 指向的存储位置，并在 wDesc 张量描述符中返回零。在这种情况下，mcdnnGetMultiHeadAttnWeights() 函数的返回状态为 MCDNN_STATUS_SUCCESS。

mcDNN 虽然应该在 GPU 内存中分配带有权重和 bias 的缓冲区，但用户可以将其复制到主机内存，并使用主机权重地址调用 mcdnnGetMultiHeadAttnWeights() 函数以获取主机内存中的张量指针。此方案允许用户直接在 CPU 内存中检查可训练参数。

参数

handle

输入。当前的 mcDNN 上下文句柄。

attnDesc

输入。已配置的注意力描述符。

wKind

输入。指定应检索哪个权重或 bias 张量的枚举类型。

weightSizeInBytes

输入。用于存储所有多头注意力权重和 bias 的缓冲区大小。

weights

输入。指向主机或设备内存中权重缓冲区的指针。

wDesc

输出。指定权重或 bias 张量形状的描述符。对于权重，wDesc.dimA[] 数组有三个元素：[nHeads, projected size, original size]。对于 bias，wDesc.dimA[] 数组也有三个元素：[nHeads, projected size, 1]。wDesc.strideA[] 数组描述了张量元素在内存中的排列方式。

wAddr

输出。指针，指向应写入所请求张量起始地址的位置。当禁用相应的投影时，写入 wAddr 的地址为 NULL。

返回值

MCDNN_STATUS_SUCCESS

已成功检索设备内存中数据的权重张量描述符和地址。

MCDNN_STATUS_BAD_PARAM

遇到一个无效或不兼容的输入参数。例如，wKind 的值无效，或者 weightSizeInBytes 太小。

6.2.1.17 mcdnnGetRNNBackwardWeightsAlgorithmMaxCount()

此函数保留 API，不再单独实现。

6.2.1.18 mcdnnGetRNNBiasMode()

此函数保留 API，不再单独实现。使用 mcdnnGetRNNDescriptor_v8() 代替 mcdnnGetRNNBiasMode()。

```
mcdnnStatus_t mcdnnGetRNNBiasMode(
    mcdnnRNNDescriptor_t   rnnDesc,
    mcdnnRNNBiasMode_t     *biasMode)
```

此函数用于检索已使用 mcdnnSetRNNBiasMode() 配置的 RNN bias 模式。在 mcdnnCreateRNNDescriptor() 之后，rnnDesc 中 biasMode 的默认值为 MCDNN_RNN_DOUBLE_BIAS。

参数

rnnDesc

输入。已创建的 RNN 描述符。

biasMode

输出。指针，指向应保存 RNN bias 模式的位置。

返回值

MCDNN_STATUS_BAD_PARAM

rnnDesc 或 biasMode 为 NULL。

MCDNN_STATUS_SUCCESS

已成功检索 biasMode 参数。

6.2.1.19 mcdnnGetRNNDataDescriptor()

此函数用于检索已创建的 RNN 数据描述符对象。

```
mcdnnStatus_t mcdnnGetRNNDataDescriptor(
    mcdnnRNNDataDescriptor_t        RNNDataDesc,
    mcdnnDataType_t                 *dataType,
    mcdnnRNNDataLayout_t            *layout,
    int                             *maxSeqLength,
    int                             *batchSize,
    int                             *vectorSize,
    int                             arrayLengthRequested,
    int                             seqLengthArray[],
    void                            *paddingFill);
```

参数

RNNDataDesc

输入。已创建和初始化的 RNN 描述符。

dataType

输出。指针，指向主机内存中用于存储 RNN 数据张量数据类型的位置。

layout

输出。指针，指向主机内存中用于存储 RNN 数据张量内存布局的位置。

maxSeqLength

输出。此 RNN 数据张量内的最大序列长度，包括填充向量。

batchSize

输出。在 mini-batch 内的序列数。

vectorSize

输出。输入或输出张量在每个时间步的向量长度（即，嵌入大小）。

arrayLengthRequested

输入。用户请求的 seqLengthArray 元素数。

seqLengthArray

输出。指针，指向主机内存中用于存储描述每个序列长度（即，时间步的数量）的整数数组的位置。如果 arrayLengthRequested 为 0，则允许该指针为 NULL。

paddingFill

输出。指针，指向主机内存中存储用户自定义符号的位置。该符号表示与 RNN 数据张量相同的数据类型。

返回值

MCDNN_STATUS_SUCCESS

已成功获取参数。

MCDNN_STATUS_BAD_PARAM

满足以下任意一种情况：

- RNNDataDesc、dataType、layout、maxSeqLength、batchSize、vectorSize、paddingFill 中任意一个为 NULL。
- arrayLengthRequested 大于 0 时，seqLengthArray 为 NULL。
- arrayLengthRequested 小于 0。

6.2.1.20 mcdnnGetRNNDescriptor_v6()

此函数保留 API，不再单独实现。使用 mcdnnGetRNNDescriptor_v8() 代替 mcdnnGetRNNDescriptor_v6()。

```
mcdnnStatus_t mcdnnGetRNNDescriptor_v6(
    mcdnnHandle_t handle,
    mcdnnRNNDescriptor_t rnnDesc,
    int *hiddenSize,
    int *numLayers,
    mcdnnDropoutDescriptor_t *dropoutDesc,
    mcdnnRNNInputMode_t *inputMode,
    mcdnnDirectionMode_t *direction,
    mcdnnRNNMode_t *cellMode,
    mcdnnRNNAlgo_t *algo,
    mcdnnDataType_t *mathPrec);
```

此函数用于检索由 mcdnnSetRNNDescriptor_v6() 配置的 RNN 网络参数。所有传入该函数的指针都应为 not-NULL 或报告 MCDNN_STATUS_BAD_PARAM。该函数不用于检查已检索参数的有效性。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

rnnDesc

输入。已创建和初始化的 RNN 描述符。

hiddenSize

输出。指针，指向应存储隐藏状态大小（每个 RNN 层中使用相同的值）的位置。

numLayers

输出。指针，指向应存储 RNN 层数的位置。

dropoutDesc

输出。指针，指向应存储已配置的丢弃描述符句柄的位置。

inputMode

输出。指针，指向应存储 RNN 第一层模式的位置。

direction

输出。指针，指向应存储 RNN 单向/双向模式的位置。

mode

输出。指针，指向应存储 RNN 单元类型的位置。

algo

输出。指针，指向应存储 RNN 算法类型的位置。

mathPrec

输出。指针，指向应存储数学精度类型的位置。

返回值

MCDNN_STATUS_SUCCESS

已从 RNN 描述符成功检索 RNN 参数。

MCDNN_STATUS_BAD_PARAM

传入该函数的指针中至少有一个为 NULL。

6.2.1.21 mcdnnGetRNNDescriptor_v8()

此函数用于检索由 mcdnnSetRNNDescriptor_v8() 配置的 RNN 网络参数。当不需要检索到的值时，用户可以将 NULL 分配给除 rnnDesc 之外的任意指针。该函数不用于检查已检索参数的有效性。

```
mcdnnStatus_t mcdnnGetRNNDescriptor_v8(
    mcdnnRNNDescriptor_t rnnDesc,
    mcdnnRNNAlgo_t *algo,
    mcdnnRNNMode_t *cellMode,
    mcdnnRNNBiasMode_t *biasMode,
    mcdnnDirectionMode_t *dirMode,
    mcdnnRNNInputMode_t *inputMode,
    mcdnnDataType_t *dataType,
    mcdnnDataType_t *mathPrec,
    mcdnnMathType_t *mathType,
    int32_t *inputSize,
    int32_t *hiddenSize,
    int32_t *projSize,
    int32_t *numLayers,
    mcdnnDropoutDescriptor_t *dropoutDesc,
    uint32_t *auxFlags);
```

参数

rnnDesc

输入。已创建和初始化的 RNN 描述符。

algo

输出。指针，指向应存储 RNN 算法类型的位置。

cellMode

输出。指针，指向应存储 RNN 单元类型的位置。

biasMode

输出。指针，指向应保存 RNN bias 模式 mcdnnRNNBiasMode_t 的位置。

dirMode

输出。指针，指向应存储 RNN 单向/双向模式的位置。

inputMode

输出。指针，指向应存储 RNN 第一层模式的位置。

dataType

输出。指针，指向应存储 RNN 权重/bias 数据类型的位置。

mathPrec

输出。指针，指向应存储数学精度类型的位置。

mathType

输出。指针，指向保存 Tensor Core 首选项的位置。

inputSize

输出。指针，指向存储 RNN 输入向量大小的位置。

hiddenSize

输出。指针，指向应存储隐藏状态大小（每个 RNN 层中使用相同的值）的位置。

projSize

输出。指针，指向存储循环投影后 LSTM 单元输出大小的位置。

numLayers

输出。指针，指向应存储 RNN 层数的位置。

dropoutDesc

输出。指针，指向应存储已配置的丢弃描述符句柄的位置。

auxFlags

输出。指针，指向不需要传入额外数值进行配置的其他 RNN 选项（标记）。

返回值

MCDNN_STATUS_SUCCESS

已从 RNN 描述符成功检索 RNN 参数。

MCDNN_STATUS_BAD_PARAM

找到一个无效的输入参数（rnnDesc 为 NULL）。

MCDNN_STATUS_NOT_INITIALIZED

RNN 描述符是使用旧的 mcdnnSetRNNDescriptor_v6() 进行配置的。

6.2.1.22 mcdnnGetRNNForwardInferenceAlgorithmMaxCount()

此函数保留 API，不再单独实现。

6.2.1.23 mcdnnGetRNNLinLayerBiasParams()

此函数保留 API，不再单独实现。使用 mcdnnGetRNNWeightParams() 代替 mcdnnGetRNNLinLayerBiasParams()。

```
mcdnnStatus_t mcdnnGetRNNLinLayerBiasParams(
mcdnnHandle_t                handle,
const mcdnnRNNDescriptor_t   rnnDesc,
const int                    pseudoLayer,
const mcdnnTensorDescriptor_t xDesc,
const mcdnnFilterDescriptor_t wDesc,
const void                   *w,
const int                    linLayerID,
mcdnnFilterDescriptor_t      linLayerBiasDesc,
void                         **linLayerBias)
```

该函数用于获取循环网络中每个伪层（pseudo-layer）的每个 RNN bias 列向量的指针和描述符（由 rnnDesc 定义），以及在 xDesc 中指定的其输入宽度。

注解： 在 mcDNN 中更改了 mcdnnGetRNNLinLayerBiasParams() 函数，以匹配 mcdnnGetRNNLinLayerMatrixParams() 的行为。

---

mcdnnGetRNNLinLayerBiasParams() 函数以两个维度返回 RNN bias 向量大小：行和列。

更多信息，参见 mcdnnGetFilterNdDescriptor()。mcDNN 中，格式为：

```
filterDimA[0]=1,
filterDimA[1]=rows,
filterDimA[2]=1 (number of columns)
```

当通过 mcdnnGetFilterNdDescriptor() 检索时，应忽略卷积核描述符的格式字段。mcDNN 中的 RNN 实现，在单元非线性函数之前使用两个 bias 向量。请注意，mcDNN 中的 RNN 实现取决于单元非线性函数之前的 bias 向量数量。有关基于 rnnDesc 中 mcdnnRNNBiasMode_tbiasMode 值的枚举类型，请参见 mcdnnRNNMode_t 说明中的方程式。如果 linLayerID 引用了不存在的 bias，则该函数将 linLayerBiasDesc 设置为一个归零卷积核描述符，其中：

```
filterDimA[0]=0,
filterDimA[1]=0, and
filterDimA[2]=2
```

并将 linLayerBias 设置为 NULL。请参见函数参数 linLayerID 的详细信息，以确定基于 biasMode 的 linLayerID 的相关值。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

pseudoLayer

输入。要查询的伪层。在单向 RNN 中，伪层与物理层相同（pseudoLayer=0 是 RNN 输入层，pseudoLayer=1 是第一个隐藏层）。在双向 RNN 中，伪层的数量是物理层的两倍。

- pseudoLayer=0 是指物理输入层的正向部分
- pseudoLayer=1 是指物理输入层的反向部分
- pseudoLayer=2 是第一个隐藏层的正向部分，依此类推

xDesc

输入。一个完全压缩的张量描述符，描述一个循环迭代的输入（以检索 RNN 输入宽度）。

wDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

linLayerID

输入。Linear ID index of the weight matrix.

若 rnnDesc 中 cellMode 设置为 MCDNN_RNN_RELU 或 MCDNN_RNN_TANH：

- 值 0 表示与来自上一层的输入或 RNN 模型的输入结合使用的权重矩阵。
- 值 1 表示与上一时间步的隐藏状态或初始隐藏状态结合使用的权重矩阵。

若 rnnDesc 中 cellMode 设置为 MCDNN_LSTM：

- 值 0、1、2、3 表示与来自上一层的输入或 RNN 模型的输入结合使用的权重矩阵。
- 值 4、5、6、7 表示与上一时间步的隐藏状态或初始隐藏状态结合使用的权重矩阵。
- 值 8 对应于投影矩阵（如果已启用）。

值及其 LSTM 门（gate）：

- linLayerID0 和 4 对应于输入门。
- linLayerID1 和 5 对应于遗忘门。
- linLayerID2 和 6 对应于双曲正切的新单元状态计算。
- linLayerID3 和 7 对应于输出门。

若 rnnDesc 中 cellMode 设置为 MCDNN_GRU：

- 值 0、1、2 表示与来自上一层的输入或 RNN 模型的输入结合使用的权重矩阵。
- 值 3、4、5 表示与上一时间步的隐藏状态或初始隐藏状态结合使用的权重矩阵。

值及其 GRU 门（gate）：

- linLayerID0 和 4 对应于重置门。
- linLayerID1 和 4 对应于更新门。
- linLayerID2 和 5 对应于双曲正切的新隐藏状态计算。

linLayerBiasDesc

输出。已创建的卷积核描述符的句柄。

linLayerBias

输出。数据指针，指向与卷积核描述符 linLayerBiasDesc 关联的 GPU 内存。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 以下任一参数为 NULL：handle、rnnDesc、xDesc、wDesc、linLayerBiasDesc、linLayerBias。
- 检测到 rnnDesc 和其他描述符之间数据类型不匹配。
- 不满足 w 指针对齐的最低要求。
- pseudoLayer 或 linLayerID 的值超出范围。

MCDNN_STATUS_INVALID_VALUE

linLayerBias 向量的某些元素位于 wDesc 描述符指定的 w 缓冲区边界之外。

6.2.1.24 mcdnnGetRNNLinLayerMatrixParams()

此函数保留 API，不再单独实现。使用 mcdnnGetRNNWeightParams() 代替 mcdnnGetRNNLinLayer-MatrixParams()。

```
mcdnnStatus_t mcdnnGetRNNLinLayerMatrixParams(
    mcdnnHandle_t                    handle,
    const mcdnnRNNDescriptor_t       rnnDesc,
    const int                        pseudoLayer,
    const mcdnnTensorDescriptor_t    xDesc,
    const mcdnnFilterDescriptor_t    wDesc,
    const void                      *w,
    const int                        linLayerID,
    mcdnnFilterDescriptor_t          linLayerMatDesc,
    void                            **linLayerMat)
```

该函数用于获取循环网络中每个伪层的每个 RNN 权重矩阵的指针和描述符（由 rnnDesc 定义），以及在 xDesc 中指定的其输入宽度。

注解： mcdnnGetRNNLinLayerMatrixParams() 函数已在 mcDNN 中得到增强，但未更改其原型。该函数不报告卷积核描述符 linLayerMatDesc 中每个权重矩阵的元素总数，而是以两个维度返回矩阵大小：行和列。此外，当权重矩阵不存在时（例如，MCDNN_SKIP_INPUT 模式），函数在 linLayerMat 中返回 NULL，并且 linLayerMatDesc 的所有字段均为零。

---

mcdnnGetRNNLinLayerMatrixParams() 函数以两个维度返回 RNN 矩阵大小：行和列。这允许用户轻松打印和初始化 RNN 权重矩阵。每个权重矩阵中的元素按行主序（row-major）排列。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

pseudoLayer

输入。要查询的伪层。在单向 RNN 中，伪层与物理层相同（pseudoLayer=0 是 RNN 输入层，pseudoLayer=1 是第一个隐藏层）。在双向 RNN 中，伪层的数量是物理层的两倍。

- pseudoLayer=0 是指物理输入层的正向部分
- pseudoLayer=1 是指物理输入层的反向部分
- pseudoLayer=2 是第一个隐藏层的正向部分，依此类推

xDesc

输入。一个完全压缩的张量描述符，描述一个循环迭代的输入（以检索 RNN 输入宽度）。

wDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

linLayerID

输入。线性层，获取以下信息：

若 rnnDesc 中 mode 设置为 MCDNN_RNN_RELU 或 MCDNN_RNN_TANH：

- 值 0 表示应用于输入的 bias，输入来自上一层（如果 rnnDesc 中的 biasMode 为 MCDNN_RNN_SINGLE_INP_BIAS 或 MCDNN_RNN_DOUBLE_BIAS，则相关）。
- 值 1 表示应用于循环输入的 bias（如果 rnnDesc 中的 BiasMode 为 MCDNN_RNN_DOUBLE_BIAS 或 MCDNN_RNN_SINGLE_REC_BIAS，则相关）。

若 rnnDesc 中 mode 设置为 MCDNN_LSTM：

- 值 0、1、2 表示应用于输入的 bias，输入来自上一层（如果 rnnDesc 中的 biasMode 为 MCDNN_RNN_SINGLE_INP_BIAS 或 MCDNN_RNN_DOUBLE_BIAS，则相关）。
- 值 4、5、6、7 表示应用于循环输入的 bias（如果 rnnDesc 中的 BiasMode 为 MCDNN_RNN_DOUBLE_BIAS 或 MCDNN_RNN_SINGLE_REC_BIAS，则相关）。

值及其相关门：

- 值 0 和 4 表示输入门。
- 值 1 和 5 表示遗忘门。
- 值 2 和 6 表示新的内存门。
- 值 3 和 7 表示输出门。

若 rnnDesc 中 mode 设置为 MCDNN_GRU：

- 值 0、1、2 表示应用于输入的 bias，输入来自上一层（如果 rnnDesc 中的 biasMode 为 MCDNN_RNN_SINGLE_INP_BIAS 或 MCDNN_RNN_DOUBLE_BIAS，则相关）。
- 值 3、4、5 表示应用于循环输入的 bias（如果 rnnDesc 中的 BiasMode 为 MCDNN_RNN_DOUBLE_BIAS 或 MCDNN_RNN_SINGLE_REC_BIAS，则相关）。

值及其相关门：

- 值 0 和 3 表示重置门。
- 值 1 和 4 表示更新门。
- 值 2 和 5 表示新的内存门。

有关模式和 bias 模式的更多信息，请参见 mcdnnRNNMode_t。

linLayerMatDesc

输出。已创建的卷积核描述符的句柄。当权重矩阵不存在时，返回的卷积核描述符的所有字段设置为零。

linLayerMat

输出。数据指针，指向与卷积核描述符 linLayerMatDesc 关联的 GPU 内存。当权重矩阵不存在时，返回的指针为 NULL。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- 以下任一参数为 NULL：handle、rnnDesc、xDesc、wDesc、linLayerMatDesc、linLayerMat。
- 检测到 rnnDesc 和其他描述符之间数据类型不匹配。
- 不满足 w 指针对齐的最低要求。
- pseudoLayer 或 linLayerID 的值超出范围。

MCDNN_STATUS_INVALID_VALUE

linLayerMat 向量的某些元素位于 wDesc 描述符指定的 w 缓冲区边界之外。

6.2.1.25 mcdnnGetRNNMatrixMathType()

此函数保留 API，不再单独实现。使用 mcdnnGetRNNDescriptor_v8() 代替 mcdnnGetRNNMatrixMathType()。

```
mcdnnStatus_t mcdnnGetRNNMatrixMathType(
    mcdnnRNNDescriptor_t rnnDesc,
    mcdnnMathType_t *mType);
```

此函数用于检索 MMA 的首选设置。有关详细信息，请参见 mcdnnMathType_t 说明。

参数

rnnDesc

输入。已创建和初始化的 RNN 描述符。

mType

输出。存储 Tensor Core 首选设置的地址。

返回值

MCDNN_STATUS_SUCCESS

已成功检索到请求的 RNN 描述符字段。

MCDNN_STATUS_BAD_PARAM

找到一个无效的输入变量（rnnDesc 或 mType 为 NULL）。

6.2.1.26 mcdnnGetRNNPaddingMode()

此函数保留 API，不再单独实现。使用 mcdnnGetRNNDescriptor_v8() 代替 mcdnnGetRNNPaddingMode()。

```
mcdnnStatus_t mcdnnGetRNNPaddingMode(
    mcdnnRNNDescriptor_t        rnnDesc,
    mcdnnRNNPaddingMode_t       *paddingMode)
```

此函数从 RNN 描述符检索 RNN 填充模式。

参数

rnnDesc

输入/输出。已创建的 RNN 描述符。

paddingMode

输入。指针，指向保存 RNN 填充模式的主机内存。

返回值

MCDNN_STATUS_SUCCESS

已成功检索 RNN 填充模式参数。

MCDNN_STATUS_BAD_PARAM

rnnDesc 或 *paddingMode 为 NULL。

6.2.1.27 mcdnnGetRNNParamsSize()

此函数保留 API，不再单独实现。使用 mcdnnGetRNNWeightSpaceSize() 代替 mcdnnGetRNNParamsSize()。

```
mcdnnStatus_t mcdnnGetRNNParamsSize(
    mcdnnHandle_t                    handle,
    const mcdnnRNNDescriptor_t       rnnDesc,
    const mcdnnTensorDescriptor_t    xDesc,
    size_t                          *sizeInBytes,
    mcdnnDataType_t                  dataType)
```

此函数用于查询执行 rnnDesc 描述的 RNN 所需的参数空间量，以及 xDesc 定义的输入维度。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

xDesc

输入。一个完全压缩的张量描述符，描述一个循环迭代的输入。

sizeInBytes

输出。使用指定的描述符和输入张量执行 RNN 所需的最小 GPU 内存量，作为参数空间。

dataType

输入。参数的数据类型。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- xDesc 描述符无效。
- xDesc 描述符未完全压缩。
- dataType 和张量描述符数据类型的组合无效。

MCDNN_STATUS_NOT_SUPPORTED

不支持 RNN 描述符和张量描述符的组合。

6.2.1.28 mcdnnGetRNNProjectionLayers()

此函数保留 API，不再单独实现。使用 mcdnnGetRNNDescriptor_v8() 代替 mcdnnGetRNNProjectionLayers()。

```
mcdnnStatus_t mcdnnGetRNNProjectionLayers(
    mcdnnHandle_t           handle,
    mcdnnRNNDescriptor_t    rnnDesc,
    int                     *recProjSize,
    int                     *outProjSize)
```

此函数用于检索当前 RNN 投影参数。在默认情况下，投影功能是禁用的，因此调用此函数将产生与 hiddenSize 和 outProjSize 相等的 recProjSize（设置为 0）。使用 mcdnnSetRNNProjectionLayers() 启用 RNN 投影。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

rnnDesc

输入。已创建和初始化的 RNN 描述符。

recProjSize

输出。指针，指向应存储循环投影大小的位置。

outProjSize

输出。指针，指向应存储输出投影大小的位置。

返回值

MCDNN_STATUS_SUCCESS

已成功检索 RNN 投影参数。

MCDNN_STATUS_BAD_PARAM

将一个 NULL 指针传入了此函数。

6.2.1.29 mcdnnGetRNNTempSpaceSizes()

此函数根据存储在 rnnDesc 中的 RNN 网络几何图形，fMode 参数定义的指定用法（推理或训练）以及从 xDesc 检索的当前 RNN 数据维度（maxSeqLength，batchSize），计算工作空间和预留空间缓冲区大小。当 RNN 数据维度更改时，必须再次调用 mcdnnGetRNNTempSpaceSizes()，因为 RNN 临时缓冲区大小不是单调性的。

```
mcdnnStatus_t mcdnnGetRNNTempSpaceSizes(
    mcdnnHandle_t handle,
    mcdnnRNNDescriptor_t rnnDesc,
    mcdnnForwardMode_t fMode,
    mcdnnRNNDataDescriptor_t xDesc,
    size_t *workSpaceSize,
    size_t *reserveSpaceSize);
```

当不需要检索到的值时，用户可以将 NULL 分配给 workSpaceSize 或 reserveSpaceSize 指针。

参数

handle

输入。当前的 mcDNN 上下文句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

fMode

输入。指定是在推理模式还是训练模式中使用临时缓冲区。推理期间不使用预留空间缓冲区。因此，当 fMode 参数为 MCDNN_FWD_MODE_INFERENCE 时，返回的预留空间缓冲区大小为 0。

xDesc

输入。指定当前 RNN 数据维度的单个 RNN 数据描述符：maxSeqLength 和 batchSize。

workSpaceSize

输出。作为工作空间缓冲区所需的最小 GPU 内存量（以字节为单位）。工作空间缓冲区不用于在 API 之间传递中间结果，而是用作临时读/写缓冲区。

reserveSpaceSize

输出。作为预留空间缓冲区所需的最小 GPU 内存量（以字节为单位）。预留空间缓冲区用于将中间结果从 mcdnnRNNForward() 传递到 RNN BackwardData 和 BackwardWeights 函数，这些函数计算与 RNN 输入或可训练权重和 bias 相关的一阶导数。

返回值

MCDNN_STATUS_SUCCESS

已成功计算 RNN 临时缓冲区大小。

MCDNN_STATUS_BAD_PARAM

检测到一个无效的输入参数。

MCDNN_STATUS_NOT_SUPPORTED

检测到不兼容或不支持的输入参数组合。

6.2.1.30 mcdnnGetRNNTrainingReserveSize()

此函数保留 API，不再单独实现。使用 mcdnnGetRNNTempSpaceSizes() 代替 mcdnnGetRNNTrainingReserveSize()。

```
mcdnnStatus_t mcdnnGetRNNTrainingReserveSize(
    mcdnnHandle_t                    handle,
    const mcdnnRNNDescriptor_t       rnnDesc,
    const int                        seqLength,
    const mcdnnTensorDescriptor_t   *xDesc,
    size_t                          *sizeInBytes)
```

此函数用于查询训练 rnnDesc 描述的 RNN 所需的预留空间量，以及 xDesc 定义的输入维度。必须对 mcdnnRNNForwardTraining()、mcdnnRNNBackwardData() 和 mcdnnRNNBackwardWeights() 传入相同的预留空间缓冲区。每一个调用都会覆盖预留空间的内容。但是，如果需要内存重用，则可以在调用之间安全备份和恢复预留空间。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

seqLength

输入。要展开的迭代次数。seqLength 的值不能超过 mcdnnGetRNNWorkspaceSize() 函数中用于查询执行 RNN 所需的工作空间大小。

xDesc

输入。张量描述符数组，描述每个循环迭代的输入（一个迭代一个描述符）。张量的第一维（批大小）可能从元素 n 减少到元素 n+1，但可能不会增加。每个张量描述符必须具有相同的第二维（向量长度）。

sizeInBytes

输出。使用指定的描述符和输入张量训练 RNN 所需的最小 GPU 内存量，作为预留空间。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- xDesc 中至少有一个描述符是无效的。
- xDesc 中描述符的第二维，步幅或数据类型不一致。
- xDesc 中描述符的第一维有递增。
- xDesc 中描述符未完全压缩。

MCDNN_STATUS_NOT_SUPPORTED

xDesc 描述的张量的数据类型不受支持。

6.2.1.31 mcdnnGetRNNWeightParams()

此函数用于获取循环网络中每个伪层的每个 RNN 权重矩阵和 bias 向量的起始地址和形状。

```
mcdnnStatus_t mcdnnGetRNNWeightParams(
    mcdnnHandle_t handle,
    mcdnnRNNDescriptor_t rnnDesc,
    int32_t pseudoLayer,
    size_t weightSpaceSize,
    const void *weightSpace,
    int32_t linLayerID,
    mcdnnTensorDescriptor_t mDesc,
    void **mAddr,
    mcdnnTensorDescriptor_t bDesc,
    void **bAddr);
```

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

pseudoLayer

输入。要查询的伪层。在单向 RNN 中，伪层与物理层相同（pseudoLayer=0 是 RNN 输入层，pseudoLayer=1 是第一个隐藏层）。在双向 RNN 中，伪层的数量是物理层的两倍：

- pseudoLayer=0 是指物理输入层的正向子层
- pseudoLayer=1 是指物理输入层的反向子层
- pseudoLayer=2 是第一个隐藏层的正向子层，依此类推

weightSpaceSize

输入。权重空间缓冲区的大小（以字节为单位）。

weightSpace

输入。指向权重空间缓冲区的指针。

linLayerID

输入。权重矩阵或 bias 向量线性 ID 索引。

若 rnnDesc 中 cellMode 设置为 MCDNN_RNN_RELU 或 MCDNN_RNN_TANH：

- 值 0 表示与来自上一层的输入或 RNN 模型的输入结合使用的权重矩阵或 bias 向量。
- 值 1 表示与上一时间步的隐藏状态或初始隐藏状态结合使用的权重矩阵或 bias 向量。

若 rnnDesc 中 cellMode 设置为 MCDNN_LSTM：

- 值 0、1、2、3 表示与来自上一层的输入或 RNN 模型的输入结合使用的权重矩阵或 bias 向量。
- 值 4、5、6、7 表示与上一时间步的隐藏状态或初始隐藏状态结合使用的权重矩阵或 bias 向量。
- 值 8 对应于投影矩阵（如果已启用），此操作中没有 bias。

值及其 LSTM 门（gate）：

- linLayerID0 和 4 对应于输入门。
- linLayerID1 和 5 对应于遗忘门。
- linLayerID2 和 6 对应于双曲正切的新单元状态计算。
- linLayerID3 和 7 对应于输出门。

若 rnnDesc 中 cellMode 设置为 MCDNN_GRU：

- 值 0、1、2 表示与来自上一层的输入或 RNN 模型的输入结合使用的权重矩阵或 bias 向量。
- 值 3、4、5 表示与上一时间步的隐藏状态或初始隐藏状态结合使用的权重矩阵或 bias 向量。

值及其 GRU 门（gate）：

- linLayerID0 和 4 对应于重置门。
- linLayerID1 和 4 对应于更新门。
- linLayerID2 和 5 对应于双曲正切的新隐藏状态计算。

有关模式和 bias 模式的更多信息，请参见 mcdnnRNNMode_t。

mDesc

输出。已创建的张量描述符的句柄。相应的权重矩阵的形状将以以下格式在此描述符中返回：dimA[3] = {1, rows, cols}。当权重矩阵不存在时，报告的张量维数为 0。当选择 MCDNN_SKIP_INPUT 时，第一层的输入 GEMM 矩阵会出现这种情况；当禁用该功能时，LSTM 投影矩阵会出现这种情况。

mAddr

输出。指针，指向权重空间缓冲区内权重矩阵的起始点。当权重矩阵不存在时，返回的地址为 NULL。

bDesc

输出。已创建的张量描述符的句柄。相应的 bias 向量的形状将以以下格式在此描述符中返回：dimA[3] = {1, rows, 1}。当 bias 向量不存在时，报告的张量维数为 0。

bAddr

输出。指针，指向权重空间缓冲区内 bias 向量的起始点。当 bias 向量不存在时，返回的地址为 NULL。

返回值

MCDNN_STATUS_SUCCESS

查询已成功完成。

MCDNN_STATUS_BAD_PARAM

遇到无效的输入参数。例如，pseudoLayer 的值超出范围或 linLayerID 为负或大于 8。

MCDNN_STATUS_INVALID_VALUE

某些权重/bias 元素位于权重空间缓冲区边界之外。

MCDNN_STATUS_NOT_INITIALIZED

RNN 描述符是使用旧的 mcdnnSetRNNDescriptor_v6() 进行配置的。

6.2.1.32 mcdnnGetRNNWeightSpaceSize()

此函数用于报告权重空间缓冲区所需的大小（以字节为单位）。权重空间缓冲区包含所有 RNN 权重矩阵和 bias 向量。

```
mcdnnStatus_t mcdnnGetRNNWeightSpaceSize(
    mcdnnHandle_t handle,
    mcdnnRNNDescriptor_t rnnDesc,
    size_t *weightSpaceSize);
```

参数

handle

输入。当前的 mcDNN 上下文句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

weightSpaceSize

输出。所有 RNN 可训练参数所需的最小 GPU 内存（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

遇到无效的输入参数。例如，任意输入参数均为 NULL。

MCDNN_STATUS_NOT_INITIALIZED

RNN 描述符是使用旧的 mcdnnSetRNNDescriptor_v6() 进行配置的。

6.2.1.33 mcdnnGetRNNWorkspaceSize()

此函数保留 API，不再单独实现。使用 mcdnnGetRNNTempSpaceSizes() 代替 mcdnnGetRNNWorkspaceSize()。

```
mcdnnStatus_t mcdnnGetRNNWorkspaceSize(
    mcdnnHandle_t                   handle,
    const mcdnnRNNDescriptor_t      rnnDesc,
    const int                       seqLength,
    const mcdnnTensorDescriptor_t  *xDesc,
    size_t                         *sizeInBytes)
```

此函数用于查询执行 rnnDesc 描述的 RNN 所需的工作空间量，以及 xDesc 定义的输入维度。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

seqLength

输入。要展开的迭代次数。根据此函数提供的大小来分配的工作空间，不能用于比 seqLength 长的序列。

xDesc

输入。张量描述符数组，描述每个循环迭代的输入（一个迭代一个描述符）。张量的第一维（批大小）可能从元素 n 减少到元素 n+1，但可能不会增加。例如，如果 batch 中有多个时间序列，序列长度可以不同。此维度是序列特定迭代的批大小，因此当 batch 中的序列终止时，维度应减小。每个张量描述符必须具有相同的第二维（向量长度）。

sizeInBytes

输出。使用指定的描述符和输入张量执行 RNN 所需的最小 GPU 内存量，作为工作空间。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- xDesc 中至少有一个描述符是无效的。
- xDesc 中描述符的第二维，步幅或数据类型不一致。
- xDesc 中描述符的第一维在增长。
- xDesc 中描述符未完全压缩。

MCDNN_STATUS_NOT_SUPPORTED

xDesc 描述的张量的数据类型不受支持。

6.2.1.34 mcdnnGetSeqDataDescriptor()

此函数用于从已创建的序列数据描述符中检索设置。当不需要检索到的值时，用户可以将 NULL 分配给除 seqDataDesc 之外的任意指针。nbDimsRequested 参数适用于 dimA[] 和 axes[] 数组。当相应的 dimA[]、axes[] 或 seqLengthArray[] 数组为 NULL 时，将忽略 nbDimsRequested 或 seqLengthSizeRequested 的正值。

```
mcdnnStatus_t mcdnnGetSeqDataDescriptor(
    const mcdnnSeqDataDescriptor_t seqDataDesc,
    mcdnnDataType_t *dataType,
    int *nbDims,
    int nbDimsRequested,
    int dimA[],
    mcdnnSeqDataAxis_t axes[],
    size_t *seqLengthArraySize,
    size_t seqLengthSizeRequested,
    int seqLengthArray[],
    void *paddingFill);
```

mcdnnGetSeqDataDescriptor() 函数不用于报告序列数据缓冲区中的实际步幅。这些步幅可用于计算任意序列数据元素的偏移量。用户必须根据 mcdnnGetSeqDataDescriptor() 函数报告的 axes[] 和 dimA[] 数组来预计算步幅。以下为执行此任务的示例代码：

```
// Array holding sequence data strides.
size_t strA[MCDNN_SEQDATA_DIM_COUNT] = {0};

// Compute strides from dimension and order arrays.
size_t stride = 1;
for (int i = nbDims - 1; i >= 0; i--) {
    int j = int(axes[i]);
    if (unsigned(j) < MCDNN_SEQDATA_DIM_COUNT-1 && strA[j] == 0) {
    strA[j] = stride;
    stride *= dimA[j];
    } else {
        fprintf(stderr, "ERROR: invalid axes[%d]=%d\n\n", i, j);
        abort();
    }
}
```

现在，strA[] 数组可用于计算任意序列数据元素的索引，例如：

```
// Using four indices (batch, beam, time, vect) with ranges already
↪checked.
size_t base = strA[MCDNN_SEQDATA_BATCH_DIM] * batch
            + strA[MCDNN_SEQDATA_BEAM_DIM]  * beam
        + strA[MCDNN_SEQDATA_TIME_DIM]  * time;
val = seqDataPtr[base + vect];
```

以上代码假定四个索引（batch、beam、time、vect）都小于 dimA[] 数组中的相应值。示例代码还省略了 strA[MCDNN_SEQDATA_VECT_DIM] 步幅，因为其值始终为 1，即一个向量的元素占用一个连续的内存块。

参数

seqDataDesc

输入。序列数据描述符。

dataType

输出。序列数据缓冲区中使用的数据类型。

nbDims

输出。dimA[] 和 axes[] 数组中的活动维数。

nbDimsRequested

输入。从索引 0 开始，可写入 dimA[] 和 axes[] 数组的连续元素的最大数量。建议设置此参数的值为 MCDNN_SEQDATA_DIM_COUNT。

dimA[]

输出。包含序列数据维度的整数数组。

axes[]

输出。mcdnnSeqDataAxis_t 数组，用于定义内存中序列数据的布局。

seqLengthArraySize

输出。seqLengthArray[] 中保存所有序列长度所需的元素数。

seqLengthSizeRequested

输入。从索引 0 开始，可写入 seqLengthArray[] 数组的连续元素的最大数量。

seqLengthArray[]

输出。包含序列长度的整数数组。

paddingFill

输出。指针，指向具有填充值的 dataType 存储位置，该填充值应写入所有填充向量。未请求输出填充向量的显式初始化时，使用 NULL。

返回值

MCDNN_STATUS_SUCCESS

已成功检索到请求的序列数据描述符字段。

MCDNN_STATUS_BAD_PARAM

找到一个无效的输入参数。

MCDNN_STATUS_INTERNAL_ERROR

内部状态不一致。

6.2.1.35 mcdnnMultiHeadAttnForward()

mcdnnMultiHeadAttnForward() 函数用于计算多头注意力层的正向响应。当 reserveSpaceSizeInBytes=0 且 reserveSpace=NULL 时，在推理模式下，该函数不调用反向（梯度）函数，否则为训练模式。在训练模式中，预留空间用于将中间结果从 mcdnnMultiHeadAttnForward() 传送到 mcdnnMultiHeadAttnBackwardData()，以及从 mcdnnMultiHeadAttnBackwardData() 传送到 mcdnnMultiHeadAttnBackwardWeights()。

```
mcdnnStatus_t mcdnnMultiHeadAttnForward(
    mcdnnHandle_t handle,
    const mcdnnAttnDescriptor_t attnDesc,
    int currIdx,
    const int loWinIdx[],
    const int hiWinIdx[],
    const int devSeqLengthsQO[],
    const int devSeqLengthsKV[],
    const mcdnnSeqDataDescriptor_t qDesc,
    const void *queries,
    const void *residuals,
    const mcdnnSeqDataDescriptor_t kDesc,
    const void *keys,
    const mcdnnSeqDataDescriptor_t vDesc,
    const void *values,
    const mcdnnSeqDataDescriptor_t oDesc,
    void *out,
    size_t weightSizeInBytes,
    const void *weights,
    size_t workSpaceSizeInBytes,
    void *workSpace,
    size_t reserveSpaceSizeInBytes,
    void *reserveSpace);
```

在推理模式中，currIdx 指定要处理的嵌入向量的时间步或序列索引。在此模式下，用户可以对时间步 0（currIdx=0）执行一次迭代，然后更新 Q，K，V 向量和注意力窗口，并执行下一个时间步（currIdx=1）。所有时间步都可以重复迭代过程。

当所有的 Q 时间步都可用时（例如，在训练模式或自注意力机制编码器端的推理模式中），用户可以为 currIdx 分配一个负值，mcdnnMultiHeadAttnForward() API 将自动扫描所有的 Q 时间步。

loWinIdx[] 和 hiWinIdx[] 主机数组指定每个 Q 时间步的注意力窗口大小。在典型自注意力场景下，用户必须包含所有已访问过的嵌入向量，但不包括当前或未来的向量。在这种情况下，用户需要设置：

```
currIdx=0: loWinIdx[0]=0; hiWinIdx[0]=0;  // initial time-step, no
↪attention window
currIdx=1: loWinIdx[1]=0; hiWinIdx[1]=1;  // attention window spans one
↪vector
currIdx=2: loWinIdx[2]=0; hiWinIdx[2]=2;  // attention window spans two
↪vectors
(...)
```

当 mcdnnMultiHeadAttnForward() 中 currIdx 为负数时，必须为所有时间步完全初始化 loWinIdx[] 和 hiWinIdx[] 数组。当用 currIdx=0，currIdx=1，currIdx=2 等调用 mcdnnMultiHeadAttnForward() 时，用户只能在调用正向响应函数之前更新 loWinIdx[currIdx] 和 hiWinIdx[currIdx] 元素。不会访问 loWinIdx[] 和 hiWinIdx[] 数组中的所有其他元素。任意自适应注意力窗口机制都可以通过这种方式实现。

当注意力窗口应为最大尺寸时（例如，交叉注意力），请使用以下设置：

```
currIdx=0: loWinIdx[0]=0; hiWinIdx[0]=maxSeqLenK;
currIdx=1: loWinIdx[1]=0; hiWinIdx[1]=maxSeqLenK;
currIdx=2: loWinIdx[2]=0; hiWinIdx[2]=maxSeqLenK;
(...)
```

上述 maxSeqLenK 值应等于或大于 kDesc 描述符中的 dimA[MCDNN_SEQDATA_TIME_DIM]。更佳选择是使用 limits.h 中的 maxSeqLenK=INT_MAX。

注解： mcdnnSetSeqDataDescriptor() 的 seqLengthArray[] 中定义的任意 K 序列的实际长度可以短于 maxSeqLenK。有效注意力窗口跨度是根据存储在 K 序列描述符中的 seqLengthArray[] 以及包含在 loWinIdx[] 和 hiWinIdx[] 数组中的索引计算的。

---

devSeqLengthsQO[] 和 devSeqLengthsKV[] 是指向具有 Q，O 和 K，V 序列长度的设备（非主机）数组的指针。请注意，相同的信息也会传入主机端相应的 mcdnnSeqDataDescriptor_t 类型描述符中。mcDNN 调用的异步性质以及专用于 GPU 内核参数的有限常量内存，使得需要额外的设备数组。当 mcdnnMultiHeadAttnForward() API 返回时，存储在描述符中的序列长度数组可在下一次迭代中立即修改。但是，由正向调用启动的 GPU 内核此时可能尚未启动。因此，序列数组需要拷贝到设备端，以便 GPU 内核直接访问。如果没有设备内存分配和 MXMACA 流同步，则无法在 mcdnnMultiHeadAttnForward() 函数内为非常大的 K，V 输入创建这些拷贝。

为了减少 mcdnnMultiHeadAttnForward() API 的开销，未验证 devSeqLengthsQO[] 和 devSeqLengthsKV[] 设备数组是否包含与序列数据描述符中 seqLengthArray[] 相同的设置。

kDesc 和 vDesc 描述符中的序列长度应相同。同样，qDesc 和 oDesc 描述符中的序列长度应匹配。用户可以在 qDesc、kDesc、vDesc 和 oDesc 描述符中定义六种不同的数据布局。有关这些布局的信息，请参见 mcdnnSetSeqDataDescriptor() 函数。所有多头注意力 API 调用都需要在所有序列数据描述符中使用相同的布局。

在 Transformer 模型中，多头注意力块与层归一化和残差连接（residual connection）紧密耦合。mcdnnMultiHeadAttnForward() 不包含层归一化，但可用于处理下图所示的残差连接。

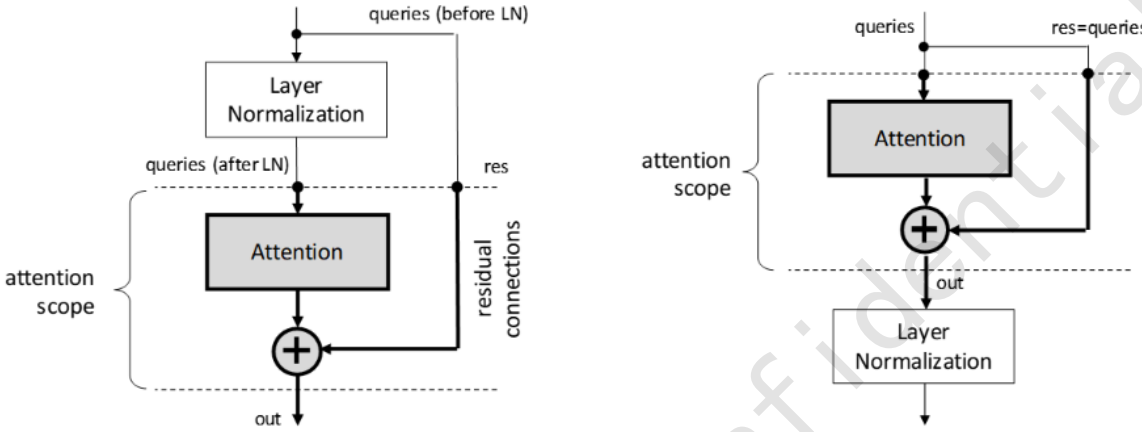

图 1. 多头注意力块与层归一化和残差连接紧密耦合

查询和残差在 mcdnnMultiHeadAttnForward() 中共享相同的 qDesc 描述符。禁用残差连接时，残差指针应为 NULL。启用残差连接时，qDesc 中的向量长度应与 oDesc 描述符中指定的向量长度匹配，因此向量加法是可行的。

queries、keys、values 指针不能为 NULL，即使 K 和 V 是相同的输入或 Q，K，V 是相同的输入。

参数

handle

输入。当前的 mcDNN 上下文句柄。

attnDesc

输入。已初始化的注意力描述符。

currIdx

输入。查询中要处理的时间步。当 currIdx 参数为负值时，将处理所有 Q 时间步。当 currIdx 为零或正时，仅为选定的时间步计算正向响应。后一个输入只能在推理模式中使用，以处理一个时间步，同时更新下一个注意力窗口和 Q，R，K，V 输入之间的调用。

loWinIdx[], hiWinIdx[]

输入。两个主机整数组，指定每个 Q 时间步的注意力窗口的起始和结束索引。包含 K，V 集合中的起始索引，而不包含结束索引。

devSeqLengthsQO[]

输入。设备数组，指定查询、残差和输出序列数据的序列长度。

devSeqLengthsKV[]

输入。设备数组，指定键和值输入数据的序列长度。

qDesc

输入。查询和残差序列数据的描述符。

queries

输入。指向设备内存中查询数据的指针。

residuals

输入。指向设备内存中残差数据的指针。如果不需要残差连接，将此参数设置为 NULL。

kDesc

输入。键序列数据的描述符。

keys

输入。指向设备内存中键数据的指针。

vDesc

输入。值序列数据的描述符。

values

输入。指向设备内存中值数据的指针。

oDesc

输入。多头注意力输出序列数据的描述符。

out

输出。指针，指向应写入输出响应的设备内存。

weightSizeInBytes

输入。权重缓冲区的大小（以字节为单位），存储了所有多头注意力可训练参数。

weights

输入。指向设备内存中权重缓冲区的指针。

workSpaceSizeInBytes

输入。用于临时 API 存储的工作空间缓冲区大小（以字节为单位）。

workSpace

输入/输出。指向设备内存中工作空间缓冲区的指针。

reserveSpaceSizeInBytes

输入。用于在正向和反向（梯度）API 调用之间进行数据交换的预留空间缓冲区大小（以字节为单位）。此参数在推理模式中应为零，在训练模式中应为非零。

reserveSpace

输入/输出。指向设备内存中预留空间缓冲区的指针。此参数在推理模式中应为 NULL，在训练模式中应为非 NULL。

返回值

MCDNN_STATUS_SUCCESS

处理 API 输入参数和启动 GPU 内核时，未检测到错误。

MCDNN_STATUS_BAD_PARAM

遇到一个无效或不兼容的输入参数。例如:

- 所需的输入指针为 NULL
- currIdx 的值超出了范围
- 注意力、查询、键、值、输出的描述符值相互不兼容

MCDNN_STATUS_EXECUTION_FAILED

启动 GPU 内核的过程返回错误，或之前的内核未成功完成。

MCDNN_STATUS_INTERNAL_ERROR

内部状态不一致。

MCDNN_STATUS_NOT_SUPPORTED

请求的选项或输入参数组合不受支持。

MCDNN_STATUS_ALLOC_FAILED

共享内存不足，无法启动 GPU 内核。

6.2.1.36 mcdnnRNNForward()

此函数使用 x、hx、cx 中的输入和 weightSpace 缓冲区中的权重/bias，以计算 rnnDesc 描述的循环神经网络的正向响应。将 RNN 输出写入 y、hy、cy 缓冲区。多层 RNN 模型中的 x、y、hx、cx、hy、cy 信号的位置如下图所示。请注意，时间步之间和层之间的内部 RNN 信号不会向用户公开。

```
mcdnnStatus_t mcdnnRNNForward(
    mcdnnHandle_t handle,
    mcdnnRNNDescriptor_t rnnDesc,
    mcdnnForwardMode_t fwdMode,
    const int32_t devSeqLengths[],
    mcdnnRNNDataDescriptor_t xDesc,
    const void *x,
    mcdnnRNNDataDescriptor_t yDesc,
    void *y,
    mcdnnTensorDescriptor_t hDesc,
    const void *hx,
    void *hy,
    mcdnnTensorDescriptor_t cDesc,
    const void *cx,
    void *cy,
    size_t weightSpaceSize,
    const void *weightSpace,
    size_t workSpaceSize,
    void *workSpace,
    size_t reserveSpaceSize,
    void *reserveSpace);
```

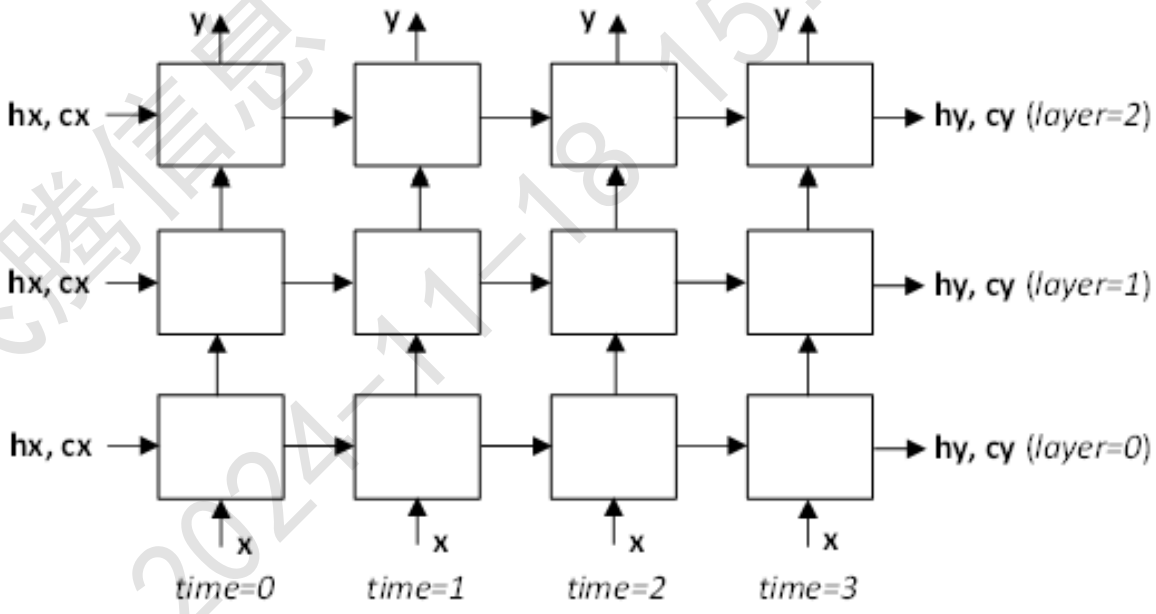

图 2. 多层 RNN 模型中的 x、y、hx、cx、hy、cy 信号的位置

下图描述了 RNN 模型为双向时的数据流。在这种模式下，每个 RNN 物理层由两个连续的伪层组成，每个伪层都有其权重、bias、初始隐藏状态 hx，对于 LSTM，还有初始单元状态 cx。伪层 0、2、4 会从左到右或在正向（F）方向处理输入向量。奇数伪层 1、3、5 会从右向左或在反向（R）方向处理输入向量。两个连续的伪层在相同的输入向量上操作，只是顺序不同。伪层 0 和 1 访问存储在 x 缓冲区中的原始序列。F 和 R 单元的输出是串联的，因此输入到下两个伪层的向量的长度为 2xhiddenSize 或 2xprojSize。后续伪层中的输入 GEMM 将向量长度调整为 1xhiddenSize。

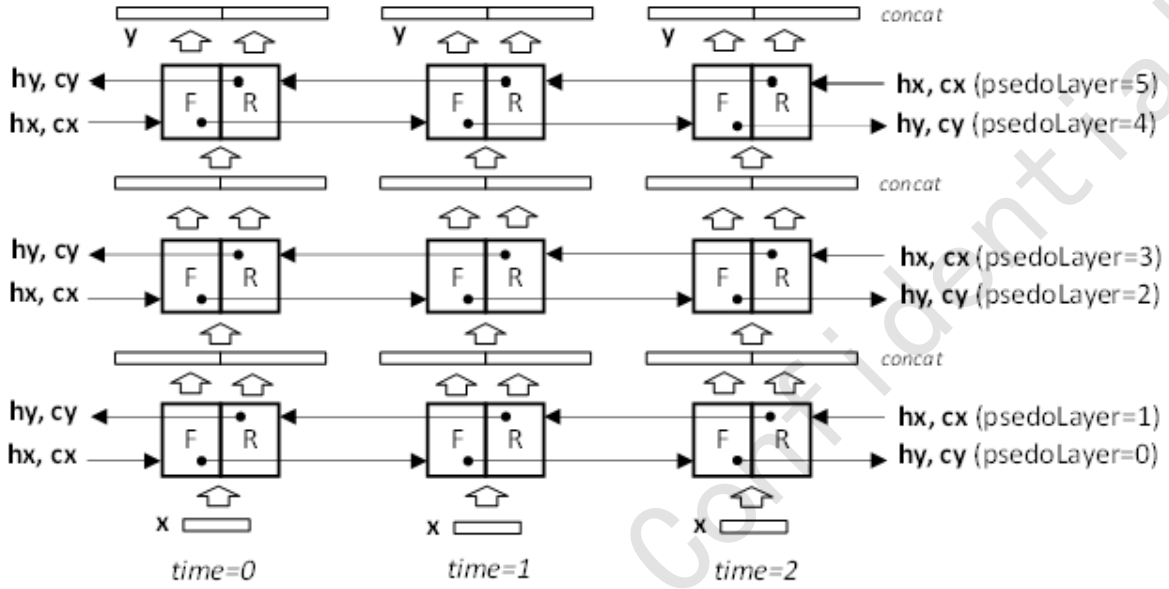

图 3.RNN 模型为双向时的数据流

当 fwdMode 参数设置为 MCDNN_FWD_MODE_TRAINING 时,mcdnnRNNForward () 函数将在预留空间缓冲区中存储计算一阶导数所需的中间数据。工作空间和预留空间缓冲区大小应由 mcdnnGetRNNTempSpaceSizes() 函数计算，该函数的 fwdMode 设置与 mcdnnRNNForward() 调用中使用的设置相同。

必须在 xDesc 和 yDesc 描述符中指定相同的布局类型。必须在 xDesc、yDesc 和设备数组 devSeqLengths 中配置相同的序列长度。mcdnnRNNForward() 函数不用于验证存储在 GPU 内存中 devSeqLengths 的序列长度与 CPU 内存中 xDesc 和 yDesc 描述符中的是否相同。但是，会检查 xDesc 和 yDesc 描述符中的序列长度数组是否一致。

参数

handle

输入。当前的 mcDNN 上下文句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

fwdMode

输入。指定推理或训练模式（MCDNN_FWD_MODE_INFERENCE 和 MCDNN_FWD_MODE_TRAINING）。在训练模式下，其他数据存储在预留空间缓冲区中。此信息用于在反向传递中计算导数。

devSeqLengths

输入。RNN 数据描述符 xDesc 或 yDesc 中 seqLengthArray 的拷贝。devSeqLengths 数组必须存储在 GPU 内存中，因为该数组可能在 mcdnnRNNForward() 函数之后，被 GPU 内核异步访问。此参数不能为 NULL。

xDesc

输入。与 RNN 模型主输入对应的已初始化的描述符。dataType、layout、maxSeqLength、batchSize 和 seqLengthArray 必须与 yDesc 中的匹配。vectorSize 参数必须与传入 mcdnnSetRNNDescriptor_V8 函数的 inputSize 参数匹配。

x

输入。数据指针，指向与 RNN 数据描述符 xDesc 关联的 GPU 内存。向量应根据 xDesc 指定的布局在内存中排列。张量中的元素（包括填充向量）必须紧密压缩。

yDesc

输入。已初始化的 RNN 数据描述符。dataType、layout、maxSeqLength、batchSize 和 seqLengthArray 必须与 xDesc 中相应参数匹配。vectorSize 参数取决于是否启用了 LSTM 投影以及网络是否为双向。具体而言：

- 对于单向模型，vectorSize 参数必须与传入 mcdnnSetRNNDescriptor_v8() 的 hiddenSize 参数匹配。如果启用了 LSTM 投影，则 vectorSize 必须与传入 mcdnnSetRNNDescriptor_v8() 的 projSize 参数相同。
- 对于双向模型，如果 RNN cellMode 为 MCDNN_LSTM 且启用了投影功能，则 vectorSize 参数必须是传入 mcdnnSetRNNDescriptor_v8() 的 projSize 参数的 2 倍。否则，应该为 hiddenSize 的 2 倍。

y

输出。数据指针，指向与 RNN 数据描述符 yDesc 关联的 GPU 内存。向量应根据 yDesc 指定的布局在内存中排布。张量中的元素（包括填充向量中的元素）必须紧密压缩，且不支持步幅。

hDesc

输入。描述 RNN 初始或最终隐藏状态的张量描述符。隐藏状态数据为完全压缩格式。张量的第一维取决于传入 mcdnnSetRNNDescriptor_v8() 函数的 dirMode 参数。

- 如果 dirMode 为 MCDNN_UNIDIRECTIONAL，则第一维应与传入 mcdnnSetRNNDescriptor_v8() 的 numLayers 参数匹配。
- 如果 dirMode 为 MCDNN_BIDIRECTIONAL，则第一维应是传入 mcdnnSetRNNDescriptor_v8() 的 numLayers 参数的 2 倍。

第二维必须与 xDesc 中描述的 batchSize 参数匹配。第三维取决于 RNN 模式是否为 MCDNN_LSTM 以及是否启用了 LSTM 投影。具体而言：如果 RNN 模式为 MCDNN_LSTM 且启用了 LSTM 投影，则第三维必须与传入 mcdnnSetRNNProjectionLayers() 调用的 projSize 参数匹配。否则，第三维必须与传入 mcdnnSetRNNDescriptor_v8() 调用的 hiddenSize 参数匹配，其中 mcdnnSetRNNDescriptor_v8() 是用于初始化 rnnDesc。

hx

输入。指针，指向具有 RNN 初始隐藏状态的 GPU 缓冲区。数据维度由 hDesc 张量描述符描述。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。

hy

输出。指针，指向存储 RNN 最终隐藏状态的 GPU 缓冲区。数据维度由 hDesc 张量描述符描述。如果传入 NULL 指针，将不会保存网络的最终隐藏状态。

cDesc

输入。仅适用于 LSTM 网络。仅描述 LSTM 网络初始或最终单元状态的张量描述符。单元状态数据为完全压缩格式。张量的第一维取决于传入 mcdnnSetRNNDescriptor_v8() 函数的 dirMode 参数。

- 如果 dirMode 为 MCDNN_UNIDIRECTIONAL，则第一维应与传入 mcdnnSetRNNDescriptor_v8() 的 numLayers 参数匹配。
- 如果 dirMode 为 MCDNN_BIDIRECTIONAL，则第一维应是传入 mcdnnSetRNNDescriptor_v8() 的 numLayers 参数的 2 倍。

第二维必须与 xDesc 中描述的 batchSize 参数匹配。否则，第三维必须与传入 mcdnnSetRNNDescriptor_v8() 调用的 hiddenSize 参数匹配。

cx

输入。仅适用于 LSTM 网络。指针，指向存储 LSTM 初始隐藏状态的 GPU 缓冲区。数据维度由 cDesc 张量描述符描述。如果传入 NULL 指针，则网络的初始单元状态将初始化为零。

cy

输出。仅适用于 LSTM 网络。指针，指向存储 LSTM 最终状态数据的 GPU 缓冲区。数据维度由 cDesc 张量描述符描述。如果传入 NULL 指针，将不会保存 LSTM 单元状态。

weightSpaceSize

输入。指定已提供的权重空间缓冲区的大小（以字节为单位）。

weightSpace

输入。GPU 内存中权重空间缓冲区的地址。

workSpaceSize

输入。指定已提供的工作空间缓冲区的大小（以字节为单位）。

workSpace

输入/输出。GPU 内存中用于存储临时数据的工作空间缓冲区地址。

reserveSpaceSize

输入。指定预留空间缓冲区的大小（以字节为单位）。

reserveSpace

输入/输出。GPU 内存中预留空间缓冲区的地址。

返回值

MCDNN_STATUS_SUCCESS

处理 API 输入参数和启动 GPU 内核时，未检测到错误。

MCDNN_STATUS_NOT_SUPPORTED

需至少满足以下任一条件：

- 在指定 MCDNN_RNN_ALGO_PERSIST_STATIC 或 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，传入变量序列长度输入
- 在 pre-Pascal 设备上请求 MCDNN_RNN_ALGO_PERSIST_STATIC 或 MCDNN_RNN_ALGO_PERSIST_DYNAMICC
- “double”浮点类型用于输入/输出和 MCDNN_RNN_ALGO_PERSIST_STATIC 算法

MCDNN_STATUS_BAD_PARAM

遇到一个无效或不兼容的输入参数。例如：

- 一些输入描述符为 NULL
- rnnDesc、xDesc、yDesc、hDesc 或 cDesc 描述符中至少有一个设置无效
- weightSpaceSize、workSpaceSize 或 reserveSpaceSize 太小。

MCDNN_STATUS_EXECUTION_FAILED

启动 GPU 内核的过程返回错误，或之前的内核未成功完成。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配 CPU 内存。

6.2.1.37 mcdnnRNNForwardInference()

此函数保留 API，不再单独实现。使用 mcdnnRNNForward() 代替 mcdnnRNNForwardInference()。

```
mcdnnStatus_t mcdnnRNNForwardInference(
    mcdnnHandle_t                    handle,
    const mcdnnRNNDescriptor_t       rnnDesc,
    const int                        seqLength,
    const mcdnnTensorDescriptor_t   *xDesc,
    const void                      *x,
```

(下页继续)

(续上页)

```
    const mcdnnTensorDescriptor_t   hxDesc,
    const void                     *hx,
    const mcdnnTensorDescriptor_t   cxDesc,
    const void                     *cx,
    const mcdnnFilterDescriptor_t   wDesc,
    const void                     *w,
    const mcdnnTensorDescriptor_t   *yDesc,
    void                           *y,
    const mcdnnTensorDescriptor_t   hyDesc,
    void                           *hy,
    const mcdnnTensorDescriptor_t   cyDesc,
    void                           *cy,
    void                           *workspace,
    size_t                          workSpaceSizeInBytes)
```

此函数使用输入 x、hx、cx，权重 w 和输出 y、hy、cy 来执行 rnnDesc 描述的循环神经网络。需要工作空间用于中间存储。此函数不用于存储训练所需的中间数据；但 mcdnnRNNForwardTraining() 用于此目的。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

seqLength

输入。要展开的迭代次数。seqLength 的值不能超过 mcdnnGetRNNWorkspaceSize() 函数中用于查询执行 RNN 所需的工作空间大小。

xDesc

输入。完全压缩的张量描述符的 seqLength 数组。数组中的每个描述符都应具有三个维度，这些维度将输入数据格式描述为一个循环迭代（一个 RNN 时间步一个描述符）。张量的第一维（批大小）可能从迭代 n 减少到迭代 n+1，但可能不会增加。每个张量描述符必须具有相同的第二维（RNN 输入向量长度 inputSize）。每个张量的第三维需要为 1。输入数据应按列主序（column-major）排列，因此 xDesc 中的步幅应设置如下：

strideA[0]=inputSize, strideA[1]=1, strideA[2]=1

x

输入。数据指针，指向与张量描述符 xDesc 的数组关联的 GPU 内存。输入向量压缩会在迭代 n 的最后一个向量结束后直接从迭代 n+1 的第一个向量开始连续进行。换言之，所有 RNN 时间步的输入向量应填充在 GPU 内存的连续块中，且向量之间没有间隙。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。

cxDesc

输入。完全压缩的的张量描述符，描述 LSTM 网络的初始单元（cell）状态。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 中描述的张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

cx

输入。数据指针，指向与张量描述符 cxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态将初始化为零。

wDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

yDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代输出（一个迭代一个描述符）。张量的第二维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第二维应与 hiddenSize 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第二维应是 hiddenSize 参数的 2 倍。

张量 n 的第一维必须与 xDesc 中张量 n 的第一维匹配。

y

输出。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。数据压缩会在迭代 n 的最后一个元素结束后直接从迭代 n+1 的第一个元素开始连续进行。

hyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终隐藏状态。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 中描述的张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hy

输出。数据指针，指向与张量描述符 hyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终隐藏状态。

cyDesc

输入。完全压缩的张量描述符，描述 LSTM 网络的最终单元状态。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

cy

输出。数据指针，指向与张量描述符 cyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终单元状态。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- hxDesc、cxDesc、wDesc、hyDesc、cyDesc 描述符中或者 xDesc、yDesc 描述符中至少有一个是无效的。
- xDesc、hxDesc、cxDesc、wDesc、yDesc、hyDesc、cyDesc 中任一描述符的步幅或维度不正确。
- workSpaceSizeInBytes 太小。

MCDNN_STATUS_INVALID_VALUE

在 RNN 描述符中选择 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，在当前函数之前未调用 mcdnnSetPersistentRNNPlan()。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

6.2.1.38 mcdnnRNNForwardInferenceEx()

此函数保留 API，不再单独实现。使用 mcdnnRNNForward() 代替 mcdnnRNNForwardInferenceEx()。

```
mcdnnStatus_t mcdnnRNNForwardInferenceEx(
    mcdnnHandle_t                   handle,
    const mcdnnRNNDescriptor_t      rnnDesc,
    const mcdnnRNNDataDescriptor_t  xDesc,
    const void                      *x,
    const mcdnnTensorDescriptor_t   hxDesc,
    const void                      *hx,
    const mcdnnTensorDescriptor_t   cxDesc,
    const void                      *cx,
    const mcdnnFilterDescriptor_t   wDesc,
    const void                      *w,
    const mcdnnRNNDataDescriptor_t  yDesc,
    void                            *y,
    const mcdnnTensorDescriptor_t   hyDesc,
    void                            *hy,
    const mcdnnTensorDescriptor_t   cyDesc,
    void                            *cy,
```

(下页继续)

(续上页)

```
    const mcdnnRNNDataDescriptor_t  kDesc,
    const void                      *keys,
    const mcdnnRNNDataDescriptor_t  cDesc,
    void                            *cAttn,
    const mcdnnRNNDataDescriptor_t  iDesc,
    void                            *iAttn,
    const mcdnnRNNDataDescriptor_t  qDesc,
    void                            *queries,
    void                            *workSpace,
    size_t                          workSpaceSizeInBytes)
```

该函数是 mcdnnRNNForwardInference() 函数的扩展版本。mcdnnRNNForwardTrainingEx() 函数允许用户对输入 x 和输出 y 使用非压缩（填充）布局。在非压缩布局中，mini-batch 中的每个序列都被视为固定长度，由其对应的 RNNDataDescriptor 中的 maxSeqLength 指定。每个固定长度序列（例如，mini-batch 中的第 n 个序列）都由一个有效段（由其对应的 RNNDataDescriptor 中的 seqLengthArray[n] 指定）和一个填充段组成，使组合序列长度等于 maxSeqLength。对于非压缩布局，同时支持序列主要（sequence-major）布局（即，时间主要）和批主要（batch-major）布局。为了向后兼容，支持压缩的序列主要布局。但是，与非扩展函数 mcdnnRNNForwardInference() 类似，mini-batch 中的序列需要根据长度按降序排序。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

xDesc

输入。已初始化的 RNN 数据描述符。dataType、layout、maxSeqLength、batchSize 和 seqLengthArray 必须与 yDesc 中的匹配。

x

输入。数据指针，指向与 RNN 数据描述符 xDesc 关联的 GPU 内存。向量应根据 xDesc 指定的布局在内存中排列。张量中的元素（包括填充向量中的元素）必须紧密压缩，且不支持步幅。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。

张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 中描述的 batchSize 参数匹配。

第三维取决于 RNN 模式是否为 MCDNN_LSTM 以及是否启用了 LSTM 投影。具体而言：

- 如果 RNN 模式为 MCDNN_LSTM 且启用了 LSTM 投影，则第三维必须与传入 mcdnnSetRNNProjectionLayers() 调用的 recProjSize 参数匹配，其中 mcdnnSetRNNProjectionLayers() 是用于设置 rnnDesc 的。
- 否则，第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。

cxDesc

输入。完全压缩的的张量描述符，描述 LSTM 网络的初始单元（cell）状态。

张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 中描述的 batchSize 参数匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。

cx

输入。数据指针，指向与张量描述符 cxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态将初始化为零。

wDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

yDesc

输入。已初始化的 RNN 数据描述符。

dataType、layout、maxSeqLength、batchSize 和 seqLengthArray 必须与 dyDesc 和 dxDesc 中的相应参数匹配。vectorSize 参数取决于 RNN 模式是否设置为 MCDNN_LSTM，是否启用了 LSTM 投影以及网络是否为双向。具体而言：

- 对于单向网络，如果 RNN 模式为 MCDNN_LSTM 且启用了 LSTM 投影，则 vectorSize 参数必须与传入 mcdnnSetRNNProjectionLayers() 调用的 recProjSize 参数匹配，其中 mcdnnSetRNNProjectionLayers() 是用于设置 rnnDesc 的。如果网络是双向的，则将该值乘以 2。
- 否则，对于单向网络，vectorSize 参数必须与初始化 rnnDesc 的 hiddenSize 参数匹配。如果网络是双向的，则将该值乘以 2。

y

输出。数据指针，指向与 RNN 数据描述符 yDesc 关联的 GPU 内存。向量应根据 yDesc 指定的布局在内存中排布。张量中的元素（包括填充向量中的元素）必须紧密压缩，且不支持步幅。

hyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终隐藏状态。此描述符的设置必须与 hxDesc 完全一致。

hy

输出。数据指针，指向与张量描述符 hyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终隐藏状态。

cyDesc

输入。完全压缩的张量描述符，描述 LSTM 网络的最终单元状态。此描述符的设置必须与 cxDesc 完全一致。

cy

输出。数据指针，指向与张量描述符 cyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终单元状态。

kDesc

保留。用户可以传入 NULL。

keys

保留。用户可以传入 NULL。

cDesc

保留。用户可以传入 NULL。

cAttn

保留。用户可以传入 NULL。

iDesc

保留。用户可以传入 NULL。

iAttn

保留。用户可以传入 NULL。

qDesc

保留。用户可以传入 NULL。

queries

保留。用户可以传入 NULL。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

需至少满足以下任一条件：

- 在使用 MCDNN_RNN_ALGO_PERSIST_STATIC 或 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，传入变量序列长度输入。
- 在 pre-Pascal 设备上使用 MCDNN_RNN_ALGO_PERSIST_STATIC 或 MCDNN_RNN_ALGO_PERSIST_DYNAMIC。
- 双输入/输出用于 MCDNN_RNN_ALGO_PERSIST_STATIC。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- xDesc、yDesc、hxDesc、cxDesc、wDesc、hyDesc 或 cyDesc 无效，或者步幅或维度不正确。
- reserveSpaceSizeInBytes 太小。
- workSpaceSizeInBytes 太小。

MCDNN_STATUS_INVALID_VALUE

在 RNN 描述符中选择 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，在当前函数之前未调用 mcdnnSetPersistentRNNPlan()。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

6.2.1.39 mcdnnRNNGetClip()

此函数保留 API，不再单独实现。使用 mcdnnRNNGetClip_v8() 代替 mcdnnRNNGetClip()。

```
mcdnnStatus_t mcdnnRNNGetClip(
    mcdnnHandle_t               handle,
    mcdnnRNNDescriptor_t        rnnDesc,
    mcdnnRNNClipMode_t          *clipMode,
    mcdnnNanPropagation_t       *clipNanOpt,
    double                      *lclip,
    double                      *rclip);
```

检索当前的 LSTM 单元裁剪参数，并将其存储在提供的参数中。

参数

clipMode

输出。指针，指向已检索到的 clipMode 的存储位置。clipMode 可以为 MCDNN_RNN_CLIP_NONE，在这种情况下不执行 LSTM 单元状态裁剪；或者为 MCDNN_RNN_CLIP_MINMAX，在这种情况下，对激活其他单元的单元状态执行裁剪。

lclip, rclip

输出。指针，指向已检索到的 LSTM 单元裁剪范围 [lclip, rclip] 的存储位置。

clipNanOpt

输出。指针，指向已检索到的 clipNanOpt 的存储位置。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

任意已提供的指针参数为 NULL。

6.2.1.40 mcdnnRNNGetClip_v8()

检索当前的 LSTM 单元裁剪参数，并将其存储在提供的参数中。当不需要检索到的值时，用户可以将 NULL 分配给除 rnnDesc 之外的任意指针。该函数不用于检查已检索参数的有效性。

```
mcdnnStatus_t mcdnnRNNGetClip_v8(
    mcdnnRNNDescriptor_t rnnDesc,
    mcdnnRNNClipMode_t *clipMode,
    mcdnnNanPropagation_t *clipNanOpt,
    double *lclip,
    double *rclip);
```

参数

rnnDesc

输入。已初始化的 RNN 描述符。

clipMode

输出。指针，指向已检索到的 mcdnnRNNClipMode_t 值的存储位置。clipMode 可以为 MCDNN_RNN_CLIP_NONE，在这种情况下不执行 LSTM 单元状态裁剪；或者为 MCDNN_RNN_CLIP_MINMAX，在这种情况下，对激活其他单元的单元状态执行裁剪。

clipNanOpt

输出。指针，指向已检索到的 mcdnnNanPropagation_t 值的存储位置。

lclip, rclip

输出。指针，指向已检索到的 LSTM 单元裁剪范围 [lclip, rclip] 的存储位置。

返回值

MCDNN_STATUS_SUCCESS

已从 RNN 描述符成功检索 LSTM 裁剪参数。

MCDNN_STATUS_BAD_PARAM

找到一个无效的输入参数（rnnDesc 为 NULL）。

6.2.1.41 mcdnnRNNSetClip()

此函数保留 API，不再单独实现。使用 mcdnnRNNSetClip_v8() 代替 mcdnnRNNSetClip()。

```
mcdnnStatus_t mcdnnRNNSetClip(
    mcdnnHandle_t               handle,
    mcdnnRNNDescriptor_t        rnnDesc,
    mcdnnRNNClipMode_t          clipMode,
    mcdnnNanPropagation_t       clipNanOpt,
    double                      lclip,
    double                      rclip);
```

设置 LSTM 单元裁剪（cell clipping）模式。默认情况下，LSTM 裁剪是禁用的。启用时，裁剪会应用到所有层。可能会多次调用 mcdnnRNNSetClip() 函数。

参数

clipMode

输入。启用或禁用 LSTM 单元裁剪。当 clipMode 设置为 MCDNN_RNN_CLIP_NONE 时，不会执行 LSTM 单元状态裁剪。当 clipMode 为 MCDNN_RNN_CLIP_MINMAX 时，会对激活其他单元的单元状态执行裁剪。

lclip, rclip

输入。LSTM 单元裁剪应设置的范围 [lclip, rclip]。

clipNanOpt

输入。当设置为 MCDNN_PROPAGATE_NAN 时（请参见 mcdnnNanPropagation_t 的说明），NaN 从 LSTM 单元传播；或者可以设置为裁剪范围边界值之一，以代替传播。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

如果 lclip > rclip，或者如果 lclip 或 rclip 是 NaN，则返回该值。

6.2.1.42 mcdnnRNNSetClip_v8()

设置 LSTM 单元裁剪（cell clipping）模式。默认情况下，LSTM 裁剪是禁用的。启用时，裁剪会应用到所有层。此 mcdnnRNNSetClip() 函数不会影响工作空间、预留空间和权重空间缓冲区大小，且可以多次调用。

```
mcdnnStatus_t mcdnnRNNSetClip_v8(
    mcdnnRNNDescriptor_t rnnDesc,
    mcdnnRNNClipMode_t clipMode,
    mcdnnNanPropagation_t clipNanOpt,
    double lclip,
    double rclip);
```

参数

rnnDesc

输入。已初始化的 RNN 描述符。

clipMode

输入。启用或禁用 LSTM 单元裁剪。当 clipMode 设置为 MCDNN_RNN_CLIP_NONE 时，不会执行 LSTM 单元状态裁剪。当 clipMode 为 MCDNN_RNN_CLIP_MINMAX 时，会对激活其他单元的单元状态执行裁剪。

clipNanOpt

输入。当设置为 MCDNN_PROPAGATE_NAN 时（请参见 mcdnnNanPropagation_t 的说明），NaN 从 LSTM 单元传播；或者可以设置为裁剪范围边界值之一，以代替传播。

lclip, rclip

输入。LSTM 单元裁剪应设置的范围 [lclip, rclip]。

返回值

MCDNN_STATUS_SUCCESS

函数已成功完成。

MCDNN_STATUS_BAD_PARAM

找到一个无效的输入参数，例如：

- rnnDesc 为 NULL
- lclip > rclip
- lclip 或 rclip 为 NaN

6.2.1.43 mcdnnSetAttnDescriptor()

此函数用于配置多头注意力描述符，此描述符是使用 mcdnnCreateAttnDescriptor() 函数创建的。该函数用于设置计算内部缓冲区大小、权重和 bias 张量的维度或选择优化代码路径所需的注意力参数。

```
mcdnnStatus_t mcdnnSetAttnDescriptor(
    mcdnnAttnDescriptor_t attnDesc,
    unsigned attnMode,
    int nHeads,
    double smScaler,
    mcdnnDataType_t dataType,
    mcdnnDataType_t computePrec,
    mcdnnMathType_t mathType,
    mcdnnDropoutDescriptor_t attnDropoutDesc,
    mcdnnDropoutDescriptor_t postDropoutDesc,
    int qSize,
    int kSize,
    int vSize,
    int qProjSize,
    int kProjSize,
```

(下页继续)

(续上页)

```
    int vProjSize,
    int oProjSize,
    int qoMaxSeqLength,
    int kvMaxSeqLength,
    int maxBatchSize,
    int maxBeamSize);
```

mcdnnMultiHeadAttnForward()、mcdnnMultiHeadAttnBackwardData() 和 mcdnnMultiHeadAttnBackwardWeights() 函数中的输入序列数据描述符会与注意力描述符中存储的配置参数进行对照检查。某些参数必须完全匹配，而 max 参数（如 maxBatchSize 或 qoMaxSeqLength）则为相应的维度设置上限。

多头注意力模型可通过以下公式描述：

$$\mathbf{h}_i = (\mathbf{W}_{V,i}\mathbf{V})\, softmax\left(smScaler\left(\mathbf{K}^{\mathbf{T}}\mathbf{W}_{K,i}^{T}\right)(\mathbf{W}_{Q,i}\mathbf{q})\right),\ for\ i = 0\ ...\ nHeads - 1$$

$$MultiHeadAttn\left(\mathbf{q},\mathbf{K},\mathbf{V},\mathbf{W}_Q,\mathbf{W}_K,\mathbf{W}_V,\mathbf{W}_O\right) = \sum_{i=0}^{nHeads-1}\mathbf{W}_{O,i}\mathbf{h}_i$$

其中：

- $nHeads$ 是求值 $\mathbf{h}_i$ 向量的独立注意力头数。
- **q** 是一个主要输入，单个 query 列向量。
- **K**, **V** 是 key 和 value 列向量的两个矩阵。

为了简单起见，上述公式使用单个嵌入向量 **q** 表示，但 API 可以在束搜索（beam search）方案中处理多个 **q** 候选项，可以处理捆绑到一个 batch 中多个序列的 **q** 向量，或者自动迭代序列的所有嵌入向量（时间步）。因此通常情况下，**q**、**K**, **V** 输入是带有附加信息片段的张量，例如每个序列的活动长度或如何保存未使用的填充向量。在一些发布中，$\mathbf{W}_{O,i}$ 矩阵被合并成一个输出投影矩阵，$\mathbf{h}_i$ 向量被显式合并成一个向量。这是等价的。在库中，$\mathbf{W}_{O,i}$ 矩阵在概念上与 $\mathbf{W}_{Q,i}$、$\mathbf{W}_{K,i}$ 或 $\mathbf{W}_{V,i}$ 输入投影权重相同。有关详细信息，请参见 mcdnnGetMultiHeadAttnWeights() 函数的说明。

$\mathbf{W}_{Q,i}$、$\mathbf{W}_{K,i}$、$\mathbf{W}_{V,i}$、$\mathbf{W}_{O,i}$ 权重矩阵发挥类似的作用，调整 **q**、**K**, **V** 输入和多头注意力最终输出中的向量长度。用户可以通过将 $qProjSize$、$kProjSize$、$vProjSize$ 或 $oProjSize$ 参数设置为零，来禁用任意或所有投影。**q**、**K**, **V** 中嵌入向量的大小和投影后的向量长度，需要以上述矩阵乘法可行的方式进行选择。否则 mcdnnSetAttnDescriptor() 函数返回 MCDNN_STATUS_BAD_PARAM。当需要维持 $\mathbf{W}_{KQ,i} = \mathbf{W}_{K,i}^{T}\mathbf{W}_{Q,i}$ 或 $\mathbf{W}_{OV,i} = \mathbf{W}_{O,i}\mathbf{W}_{V,i}$ 矩阵的秩亏缺（rank deficiency），以在每个头的线性转换（linear transformation）中消除一个或多个维度时，这四个权重矩阵全都需要使用。这是特征提取（feature extraction）的一种形式。在这种情况下，投影的大小比原始向量长度小。

对于每个注意力头，权重矩阵大小定义如下：

- $\mathbf{W}_{Q,i}$ - size $[qProjSize\ x\ qSize]$, $i = O\ ..\ nHeads - 1$
- $\mathbf{W}_{K,i}$ - size $[kProjSize\ x\ kSize]$, $i = 0\ ..\ nHeads - 1$, $kProjSize = qProjSize$
- $\mathbf{W}_{V,i}$ - size $[vProjSize\ x\ vSize]$, $i = 0\ ..\ nHeads - 1$
- $\mathbf{W}_{O,i}$ - size $[oProjSize\ x\ (vProjSize\ >\ 0\ ?\ vProjSize\ :\ vSize)]$, $i = 0\ ..\ nHeads - 1$

当禁用输出投影 $(oProjSize = 0)$ 时，输出向量长度为 $nHeads * (vProjSize\ >\ 0\ ?\ vProjSize\ :\ vSize)$，即输出是所有 $\mathbf{h}_i$ 向量的串联。

在另一种解释中，连接矩阵 $\mathbf{W}_O = [\mathbf{W}_{O,0}, \mathbf{W}_{O,1}, \mathbf{W}_{O,2}, ...]$ 构成了单位矩阵 (Identity Matrix)。Softmax 是一个标准化的指数向量函数，它接受和输出相同大小的向量。多头注意力 API 利用 MCDNN_SOFTMAX_ACCURATE 类型的 softmax 来降低浮点溢出的可能性。

smScaler 参数是 softmax 锐化/平滑系数。当 smScaler=1.0 时，softmax 使用自然指数函数 exp(x) 或 2.7183*。当 smScaler<1.0 时，例如 smScaler=0.2，由于 exp(0.2*x)

≈ 1.2214*，softmax 块使用的函数不会增长得那么快。

smScaler 参数可以调整来处理输入到 softmax 的值的更大范围。当范围过大（或 smScaler 对于给定范围不够小）时，softmax 块的输出向量变为类别向量（categorical vector），这意味着一个向量元素接近 1.0，其他输出为 0 或非常接近于 0。发生这种情况时，softmax 块的雅可比（Jacobian）矩阵也接近于 0，因此，除非通过残差连接（如果启用了这些连接），增量（delta）在训练期间不会从输出到输入进行反向传播。用户可以将 smScaler 设置为任何正浮点值，甚至可以设置为零。smScaler 参数不可训练。

qoMaxSeqLength、kvMaxSeqLength、maxBatchSize、maxBeamSize 参数分别声明 mcdnnSeqDataDescriptor_t 容器中的最大序列长度，最大批大小和最大束宽。提供给正向和反向（梯度）API 函数的实际维度不应超过 max 限制。需要谨慎设置 max 参数，因为过大的值将导致因工作空间和预留空间缓冲区过大而使用过多内存。

attnMode 参数被视为二进制掩码，其中设置了各种开/关选项。这些选项可以影响内部缓冲区大小，强制执行某些参数检查，选择优化的代码执行路径或启用不需要额外数值参数的注意力变量。例如，在输入和输出投影中加入 bias。

attnDropoutDesc 和 postDropoutDesc 参数是描述符，用于定义训练模式中的两个活动丢弃层。attnDropoutDesc 定义的第一个丢弃操作直接应用于 softmax 输出。由 postDropoutDesc 指定的第二个丢弃操作，用于在添加残差连接的点之前，更改多头注意力输出。

注解： mcdnnSetAttnDescriptor() 函数执行 attnDropoutDesc 和 postDropoutDesc 的浅拷贝（shallow copy），这意味着两个丢弃描述符的地址都存储在注意力描述符中，而不是存储在整个结构中。因此，用户应在注意力描述符的整个生命周期内保留丢弃描述符。

参数

attnDesc

输出。要配置的注意力描述符。

attnMode

输入。启用不需要额外数值的各种注意力选项。用户应为此参数指定一组首选的 bitwise OR-ed 标记。

nHeads

输入。注意力头数。

smScaler

输入。Softmax 平滑（1.0 >= smScaler >= 0.0）或锐化（smScaler > 1.0）系数。不允许负数值。

dataType

输入。用于表示注意力输入，注意力权重和注意力输出的数据类型。

computePrec

输入。计算精度。

mathType

输入。MMA 设置。

attnDropoutDesc

输入。应用于 softmax 输出的丢弃操作的描述符。

postDropoutDesc

输入。丢弃操作的描述符，此操作应用于在添加残差连接的点之前。

qSize, kSize, vSize

输入。Q、K、V 嵌入向量长度。

qProjSize, kProjSize, vProjSize

输入。输入投影后的 Q、K、V 嵌入向量长度。使用零以禁用相应的投影。

oProjSize

输入。输出投影后的 h_i 向量长度。使用零以禁用此投影。

qoMaxSeqLength

输入。与 Q, O, dQ 和 dO 输入和输出相关的序列数据描述符中的最大序列长度。

kvMaxSeqLength

输入。与 K, V, dK 和 dV 输入和输出相关的序列数据描述符中的最大序列长度。

maxBatchSize

输入。任意 mcdnnSeqDataDescriptor_t 容器中的最大批大小。

maxBeamSize

输入。任意 mcdnnSeqDataDescriptor_t 容器中的最大束宽。

支持的 attnMode 标记

MCDNN_ATTN_QUERYMAP_ALL_TO_ONE

当束宽大于 **Q** 输入中的一个值时，**Q** 和 **K**, **V** 向量之间映射的正向声明。

同一束中的多个 **Q** 向量映射到相同的 **K**, **V** 向量中。这意味着 **K**, **V** 组中的束宽都等于 1。

MCDNN_ATTN_QUERYMAP_ONE_TO_ONE

当束宽大于 **Q** 输入中的一个值时，**Q** 和 **K**, **V** 向量之间映射的正向声明。

同一束中的多个 **Q** 向量映射到不同的 **K**, **V** 向量中。这需要 **K**, **V** 组的束宽与 **Q** 输入中的相同。

MCDNN_ATTN_DISABLE_PROJ_BIASES

不在注意力输入和输出投影中使用 bias。

MCDNN_ATTN_ENABLE_PROJ_BIASES

在注意力输入和输出投影中使用额外的 bias。

在这种情况下，投影 $\overline{\mathbf{K}}$ 向量计算为 $\overline{\mathbf{K}_i} = \mathbf{W}_{K,i}\mathbf{K} + \mathbf{b} * [1, 1, \ldots, 1]_{1\times n}$，其中 n 是 **K** 矩阵中的列数。换言之，在权重矩阵相乘之后，相同的列向量 **b** 添加到 **K** 的所有列中。

支持的 dataType、computePrec 和 mathType 组合

mcdnnSetAttnDescriptor() 支持的组合

dataType	computePrec	mathType
MCDNN_DATA_DOUBLE	MCDNN_DATA_DOUBLE	MCDNN_DEFAULT_MATH
MCDNN_DATA_FLOAT	MCDNN_DATA_FLOAT	MCDNN_DEFAULT_MATH MCDNN_TENSOR_OP_MATH _ALLOW_CONVERSION
MCDNN_DATA_HALF	MCDNN_DATA_HALF MCDNN_DATA_FLOAT	MCDNN_DEFAULT_MATH MCDNN_TENSOR_OP_MATH MCDNN_TENSOR_OP_MATH _ALLOW_CONVERSION

不支持的功能

目前，所有多头注意力函数会忽略 mcdnnSeqDataDescriptor_t 中的 paddingFill 参数。

返回值

MCDNN_STATUS_SUCCESS

注意力描述符配置成功。

MCDNN_STATUS_BAD_PARAM

遇到无效的输入参数。例如:

- post-projection Q 和 K 大小不相等
- dataType、computePrec 或 mathType 无效
- 以下一个或多个参数为负值或零：nHeads、qSize、kSize、vSize、qoMaxSeqLength、kvMaxSeqLength、maxBatchSize、maxBeamSize
- 以下一个或多个参数为负值：qProjSize、kProjSize、vProjSize、smScaler

MCDNN_STATUS_NOT_SUPPORTED

请求的选项或输入参数组合不受支持。

6.2.1.44 mcdnnSetPersistentRNNPlan()

此函数保留 API，不再单独实现。此函数用于设置在使用 rnnDesc 和 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 算法时要执行的持久性 RNN 计划。

```
mcdnnStatus_t mcdnnSetPersistentRNNPlan(
    mcdnnRNNDescriptor_t        rnnDesc,
    mcdnnPersistentRNNPlan_t    plan)
```

返回值

MCDNN_STATUS_SUCCESS

计划设置成功。

MCDNN_STATUS_BAD_PARAM

在 rnnDesc 中选择的算法不是 MCDNN_RNN_ALGO_PERSIST_DYNAMIC。

6.2.1.45 mcdnnSetRNNAlgorithmDescriptor()

此函数保留 API，不再单独实现。

6.2.1.46 mcdnnSetRNNBiasMode()

此函数保留 API，不再单独实现。使用 mcdnnSetRNNDescriptor_v8() 代替 mcdnnSetRNNBiasMode()。

```
mcdnnStatus_t mcdnnSetRNNBiasMode(
    mcdnnRNNDescriptor_t    rnnDesc,
    mcdnnRNNBiasMode_t      biasMode)
```

mcdnnSetRNNBiasMode() 函数用于为已创建和初始化的 RNN 描述符设置 bias 向量的数量。调用此函数以在 RNN 中启用指定的 bias 模式。在 mcdnnCreateRNNDescriptor() 之后，rnnDesc 中 biasMode 的默认值为 MCDNN_RNN_DOUBLE_BIAS。

参数

rnnDesc

输入/输出。已创建的 RNN 描述符。

biasMode

输入。设置 bias 向量的数量。更多信息，参见 mcdnnRNNBiasMode_t。

返回值

MCDNN_STATUS_BAD_PARAM

rnnDesc 为 NULL 或 biasMode 有一个无效的枚举值。

MCDNN_STATUS_SUCCESS

biasMode 设置成功。

MCDNN_STATUS_NOT_SUPPORTED

非默认 bias 模式（除 MCDNN_RNN_DOUBLE_BIAS 之外的枚举类型）应用于 MCDNN_RNN_ALGO_STANDARD 以外的 RNN 算法。

6.2.1.47 mcdnnSetRNNDataDescriptor()

此函数用于初始化已创建的 RNN 数据描述符对象。此数据结构旨在支持扩展的 RNN 推理和训练函数的输入和输出非压缩（填充）布局。还支持压缩（无填充）布局以实现向后兼容。

```
mcdnnStatus_t mcdnnSetRNNDataDescriptor(
    mcdnnRNNDataDescriptor_t       RNNDataDesc,
    mcdnnDataType_t                dataType,
    mcdnnRNNDataLayout_t           layout,
    int                            maxSeqLength,
    int                            batchSize,
    int                            vectorSize,
    const int                      seqLengthArray[],
    void                           *paddingFill);
```

参数

RNNDataDesc

输入/输出。已创建的 RNN 描述符。更多信息，参见 mcdnnRNNDataDescriptor_t。

dataType

输入。RNN 数据张量的数据类型。更多信息，参见 mcdnnDataType_t。

layout

输入。RNN 数据张量的内存布局。

maxSeqLength

输入。此 RNN 数据张量内的最大序列长度。在非压缩（填充）布局中，此输入应包含每个序列中的填充向量。在压缩（未填充）布局中，此输入应等于 seqLengthArray 中的最大元素。

batchSize

输入。在 mini-batch 内的序列数。

vectorSize

输入。输入或输出张量在每个时间步的向量长度（嵌入大小）。

seqLengthArray

输入。整数组，由元素的 batchSize 数组成。描述每个序列的长度（时间步数量）。seqLengthArray 中的每个元素必须大于或等于 0，但小于或等于 maxSeqLength。在压缩布局中，元素应按降序排序，类似于非扩展 RNN 计算函数所需的布局。

paddingFill

输入。用于填充 RNN 输出中填充位置的用户定义符号。仅在描述符描述 RNN 输出且已指定压缩布局时有效。该符号应在主机内存中，表示与 RNN 数据张量相同的数据类型。如果传入 NULL 指针，则不会定义输出中的填充位置。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_NOT_SUPPORTED

dataType 不是 MCDNN_DATA_HALF、MCDNN_DATA_FLOAT 或 MCDNN_DATA_DOUBLE。

MCDNN_STATUS_BAD_PARAM

满足以下任意一种情况：

- RNNDataDesc 为 NULL。
- maxSeqLength、batchSize 或 vectorSize 中的任意一个小于或等于 0。
- seqLengthArray 的元素小于 0 或大于 maxSeqLength。
- layout 不是 MCDNN_RNN_DATA_LAYOUT_SEQ_MAJOR_UNPACKED、MCDNN_RNN_DATA_LAYOUT_SEQ_MAJOR_PACKED 或者 MCDNN_RNN_DATA_LAYOUT_BATCH_MAJOR_UNPACKED。

MCDNN_STATUS_ALLOC_FAILED

内部数组存储分配失败。

6.2.1.48 mcdnnSetRNNDescriptor_v6()

此函数保留 API，不再单独实现。使用 mcdnnSetRNNDescriptor_v8() 代替 mcdnnSetRNNDescriptor_v6()。

```
mcdnnStatus_t mcdnnSetRNNDescriptor_v6(
    mcdnnHandle_t                   handle,
    mcdnnRNNDescriptor_t            rnnDesc,
    const int                       hiddenSize,
    const int                       numLayers,
    mcdnnDropoutDescriptor_t        dropoutDesc,
    mcdnnRNNInputMode_t             inputMode,
    mcdnnDirectionMode_t            direction,
    mcdnnRNNMode_t                  mode,
    mcdnnRNNAlgo_t                  algo,
    mcdnnDataType_t                 mathPrec)
```

此函数用于初始化已创建的 RNN 描述符对象。

注解： 较大的网络（例如，较长的序列或更多的层）预计比较小的网络效率更高。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

rnnDesc

输入/输出。已创建的 RNN 描述符。

hiddenSize

输入。每个层内部隐藏状态的大小。

numLayers

输入。堆叠层数。

dropoutDesc

输入。已创建和初始化的丢弃描述符的句柄。丢弃会在层之间应用，例如，单层网络将不应用任何丢弃。

inputMode

输入。指定第一层输入的行为。

direction

输入。指定循环方式，例如双向。

mode

输入。指定要计算的 RNN 的类型。

algo

输入。指定应使用哪个 RNN 算法来计算结果。

mathPrec

输入。数学精度。此参数用于控制 RNN 中的数学精度。适用以下内容：

- 对于 FP16 的输入/输出，mathPrec 参数可以是 MCDNN_DATA_HALF 或 MCDNN_DATA_FLOAT。
- 对于 FP32 的输入/输出，mathPrec 参数只能是 MCDNN_DATA_FLOAT。
- 对于 FP64 和双精度类型的输入/输出，mathPrec 参数只能是 MCDNN_DATA_DOUBLE。

返回值

MCDNN_STATUS_SUCCESS

对象设置成功。

MCDNN_STATUS_BAD_PARAM

hiddenSize 或 numLayers 参数中至少有一个为 0 或负值；inputMode、direction、mode、algo 或 dataType 中任一具有无效的枚举值；dropoutDesc 是无效的丢弃描述符或 rnnDesc 未正确创建。

6.2.1.49 mcdnnSetRNNDescriptor_v8()

此函数用于初始化已创建的 RNN 描述符对象。通过 mcdnnSetRNNDescriptor_v8() 配置的 RNN 描述符得到了增强，可以存储在 RNN 模型中计算可调整权重/bias 总数所需的所有信息。

```
mcdnnStatus_t mcdnnSetRNNDescriptor_v8(
    mcdnnRNNDescriptor_t rnnDesc,
    mcdnnRNNAlgo_t algo,
    mcdnnRNNMode_t cellMode,
    mcdnnRNNBiasMode_t biasMode,
    mcdnnDirectionMode_t dirMode,
    mcdnnRNNInputMode_t inputMode,
    mcdnnDataType_t dataType,
    mcdnnDataType_t mathPrec,
    mcdnnMathType_t mathType,
    int32_t inputSize,
    int32_t hiddenSize,
    int32_t projSize,
    int32_t numLayers,
    mcdnnDropoutDescriptor_t dropoutDesc,
    uint32_t auxFlags);
```

参数

rnnDesc

输入。已初始化的 RNN 描述符。

algo

输入。RNN 算法（MCDNN_RNN_ALGO_STANDARD，MCDNN_RNN_ALGO_PERSIST_STATIC 或 MCDNN_RNN_ALGO_PERSIST_DYNAMIC）。

cellMode

输入。指定整个模型中的 RNN 单元类型（MCDNN_RNN_RELU、MCDNN_RNN_TANH、MCDNN_RNN_LSTM、MCDNN_RNN_GRU）。

biasMode

输入。设置 bias 向量的数量（MCDNN_RNN_NO_BIAS、MCDNN_RNN_SINGLE_INP_BIAS、MCDNN_RNN_SINGLE_REC_BIAS、MCDNN_RNN_DOUBLE_BIAS）。这两个单独的 bias 设置在功能上与 RELU，TANH 和 LSTM 单元类型的相同。有关 GRU 单元的差异，请参见 mcdnnRNNMode_t 枚举类型中 MCDNN_GRU 的说明。

dirMode

输入。指定循环模式：MCDNN_UNIDIRECTIONAL 或 MCDNN_BIDIRECTIONAL。在双向 RNN 中，在物理层之间传递的隐藏状态是正向和反向隐藏状态的串联。

inputMode

输入。指定第一层如何处理 RNN 模型的输入。当 inputMode 为 MCDNN_LINEAR_INPUT 时，inputSize 的原始输入向量将乘以权重矩阵，以获得 hiddenSize 的向量。当 inputMode 为 MCDNN_SKIP_INPUT 时，第一层的原始输入向量将按原样使用，而不会乘以权重矩阵。

dataType

输入。为 RNN 权重/偏差以及输入和输出数据指定数据类型。

mathPrec

输入。此参数用于控制 RNN 模型中的数学精度。适用以下内容：

- 对于 FP16 的输入/输出，mathPrec 参数可以是 MCDNN_DATA_HALF 或 MCDNN_DATA_FLOAT。
- 对于 FP32 的输入/输出，mathPrec 参数只能是 MCDNN_DATA_FLOAT。
- 对于 FP64 和双精度类型的输入/输出，mathPrec 参数只能是 MCDNN_DATA_DOUBLE。

mathType

输入。设置使用 MMA 加速器的首选项。

- 当 dataType 为 MCDNN_DATA_HALF 时，mathType 参数可以是 MCDNN_DEFAULT_MATH 或 MCDNN_TENSOR_OP_MATH。

对于此数据类型，ALLOW_CONVERSION 设置的处理方式与 MCDNN_TENSOR_OP_MATH 相同。

- 当 dataType 为 MCDNN_DATA_FLOAT 时，mathType 参数可以是 MCDNN_DEFAULT_MATH 或 MCDNN_TENSOR_OP_MATH_ALLOW_CONVERSION。
- 使用后一种设置时，原始权重和中间结果将向下转换为 MCDNN_DATA_HALF，然后再用于另一个递归迭代。
- 当 dataType 为 MCDNN_DATA_DOUBLE 时，mathType 参数可以是 MCDNN_DEFAULT_MATH 。

此选项具有警告含义。例如，由于特定的 GEMM 维度限制，Tensor Core 有时可能无法使用。

inputSize

输入。RNN 模型中输入向量的大小。当 inputMode=MCDNN_SKIP_INPUT，inputSize 的值应与 hiddenSize 的值一致。

hiddenSize

输入。RNN 模型中隐藏状态向量的大小。在所有 RNN 层中使用相同的隐藏大小。

projSize

输入。循环投影后 LSTM 单元输出的大小。该值不应大于 hiddenSize。将 projSize 设置为 hiddenSize 是合法的，但是在这种情况下，将禁用循环投影功能。循环投影是 LSTM 单元中的一个附加矩阵乘法，用于将隐藏状态向量 ht 投影为较小的向量 rt = Wrht，其中 Wr 是一个具有 projSize 行和 hiddenSize 列的矩形矩阵。启用循环投影时，LSTM 单元的输出（到下一层且及时展开）为 rt，而不是 ht。只能为 LSTM 单元和 MCDNN_RNN_ALGO_STANDARD 启用循环投影。

numLayers

输入。深度 RNN 模型中堆叠的物理层数。当 dirMode=MCDNN_BIDIRECTIONAL 时，物理层由与正向和反向对应的两个伪层组成。

dropoutDesc

输入。已创建和初始化的丢弃描述符的句柄。丢弃操作将在物理层之间应用。单层网络不会应用任何丢弃操作。丢弃仅用于训练模式中。

auxFlags

输入。此参数用于传递不需要附加数值来配置相应功能的其他 switch。在将来的 mcDNN 版本中，此参数将用于扩展 RNN 功能，而不添加新的 API 函数（适用的选项应为 bitwise OR-ed）。当前，此参数用于启用或禁用填充输入/输出（MCDNN_RNN_PADDED_IO_DISABLED、MCDNN_RNN_PADDED_IO_ENABLED）。启用填充输入/输出时，RNN 数据描述符中允许 MCDNN_RNN_DATA_LAYOUT_SEQ_MAJOR_UNPACKED 和 MCDNN_RNN_DATA_LAYOUT_BATCH_MAJOR_UNPACKED 布局。

返回值

MCDNN_STATUS_SUCCESS

RNN 描述符配置成功。

MCDNN_STATUS_BAD_PARAM

检测到一个无效的输入变量。

MCDNN_STATUS_NOT_SUPPORTED

bias 张量的维度是指与输出张量维度不兼容的数据量，或者两个张量描述符的 dataType 不同。

MCDNN_STATUS_EXECUTION_FAILED

检测到不兼容或不支持的输入参数组合。

6.2.1.50 mcdnnSetRNNMatrixMathType()

此函数保留 API，不再单独实现。使用 mcdnnSetRNNDescriptor_v8() 代替 mcdnnSetRNNMatrixMathType()。

```
mcdnnStatus_t mcdnnSetRNNMatrixMathType(
    mcdnnRNNDescriptor_t    rnnDesc,
    mcdnnMathType_t         mType)
```

此函数用于设置使用 MMA 加速器的首选项。当 mType 参数为 MCDNN_TENSOR_OP_MATH 时，若权重/bias 类型为 MCDNN_DATA_HALF 或 MCDNN_DATA_FLOAT，RNN 推理和训练 API 将尝试使用 Tensor Core。当 RNN 权重/bias 存储为 MCDNN_DATA_FLOAT 格式时，原始权重和中间结果将向下转换为 MCDNN_DATA_HALF，然后再用于另一个递归迭代。

参数

rnnDesc

输入。已创建和初始化的 RNN 描述符。

mType

输入。执行 RNN GEMM（通用矩阵乘法）时的首选计算选项。此选项具有警告含义。例如，由于特定的 GEMM 维度限制，Tensor Core 有时可能无法使用。

返回值

MCDNN_STATUS_SUCCESS

已成功设置 RNN 网络的首选计算选项。

MCDNN_STATUS_BAD_PARAM

检测到一个无效的输入参数。

6.2.1.51 mcdnnSetRNNPaddingMode()

此函数保留 API，不再单独实现。使用 mcdnnSetRNNDescriptor_v8() 代替 mcdnnSetRNNPaddingMode()。

```
mcdnnStatus_t mcdnnSetRNNPaddingMode(
    mcdnnRNNDescriptor_t        rnnDesc,
    mcdnnRNNPaddingMode_t       paddingMode)
```

此函数启用或禁用已创建和初始化的 RNN 描述符的填充 RNN 输入/输出。在调用 mcdnnGetRNNWorkspaceSize() 和 mcdnnGetRNNTrainingReserveSize() 函数之前，需要此信息以确定是否需要其他工作空间和训练预留空间。默认情况下，不启用填充的 RNN 输入/输出。

参数

rnnDesc

输入/输出。已创建的 RNN 描述符。

paddingMode

输入。启用或禁用填充输入/输出。更多信息，参见 mcdnnRNNPaddingMode_t。

返回值

MCDNN_STATUS_SUCCESS

paddingMode 设置成功。

MCDNN_STATUS_BAD_PARAM

rnnDesc 为 NULL 或 paddingMode 有一个无效的枚举值。

6.2.1.52 mcdnnSetRNNProjectionLayers()

此函数保留 API，不再单独实现。使用 mcdnnSetRNNDescriptor_v8() 代替 mcdnnSetRNNProjectionLayers()。

```
mcdnnStatus_t mcdnnSetRNNProjectionLayers(
    mcdnnHandle_t           handle,
    mcdnnRNNDescriptor_t    rnnDesc,
    int                     recProjSize,
    int                     outProjSize)
```

调用 mcdnnSetRNNProjectionLayers() 函数可以启用递归神经网络（recursive neural network）中的循环和/或输出投影。循环投影是 LSTM 单元中的一个附加矩阵乘法，用于将隐藏状态向量 h_t 投影为较小的向量 $r_t = W_r h_t$，其中 W_r 是一个包含 recProjSize 行和 hiddenSize 列的矩形矩阵。启用循环投影时，LSTM 单元的输出（到下一层且及时展开）为 r_t，而不是 h_t。与非线性函数结合使用的 i_t、f_t、o_t、c_t 向量的维度，与标准 LSTM 单元中保持一致。为了实现这一点，LSTM 公式（请参见 mcdnnRNNMode_t 类型）中的矩阵形状，例如隐藏 RNN 层中的 W_i 或整个网络中的 R_i，在标准 LSTM 模式中变为矩形或正方形。显然，$R_i * W_r$ 的结果是一个方形矩阵，但它是秩亏缺，反映了 LSTM 输出的压缩。对于相同的 hiddenSize 值，当具有投影的 RNN 网络中独立（可调整）权重的数量比标准 LSTM 的少时，通常使用循环投影。

只能为 LSTM 单元和 MCDNN_RNN_ALGO_STANDARD 启用循环投影。recProjSize 参数应小于 hiddenSize 值。设置 recProjSize 等于 hiddenSize 是合法的，但在这种情况下，循环投影功能是禁用的。

输出投影目前尚未实现。

参数

handle

输入。已创建的 mcDNN 库描述符的句柄。

rnnDesc

输入。已创建和初始化的 RNN 描述符。

recProjSize

输入。循环投影后 LSTM 单元输出的大小。该值不应大于 hiddenSize。

outProjSize

输入。此参数应为零。

返回值

MCDNN_STATUS_SUCCESS

已成功设置 RNN 投影参数。

MCDNN_STATUS_BAD_PARAM

检测到无效的输入参数（例如，NULL 句柄，投影参数为负值）。

MCDNN_STATUS_NOT_SUPPORTED

投影应用于 MCDNN_RNN_ALGO_STANDARD 以外的 RNN 算法，MCDNN_LSTM 以外的单元类型，recProjSize 大于 hiddenSize。

6.2.1.53 mcdnnSetSeqDataDescriptor()

此函数用于初始化已创建的序列数据描述符对象。简言之，此描述符定义了四维张量的维度（dimA）和数据布局（axes）。

```
mcdnnStatus_t mcdnnSetSeqDataDescriptor(
    mcdnnSeqDataDescriptor_t seqDataDesc,
    mcdnnDataType_t dataType,
    int nbDims,
    const int dimA[],
    const mcdnnSeqDataAxis_t axes[],
    size_t seqLengthArraySize,
    const int seqLengthArray[],
    void *paddingFill);
```

序列数据描述符的四个维度都具有可用于索引 dimA[] 数组的唯一标识符：

```
MCDNN_SEQDATA_TIME_DIM
MCDNN_SEQDATA_BATCH_DIM
MCDNN_SEQDATA_BEAM_DIM
MCDNN_SEQDATA_VECT_DIM
```

例如，要表示序列数据缓冲区中向量的长度为五个元素，我们需要在 dimA[] 数组中指定 dimA[MCDNN_SEQDATA_VECT_DIM]=5。dimA[] 和 axes[] 数组中的活动维数由 nbDims 参数定义。当前，此参数的值应为 4。dimA[] 和 axes[] 数组的实际大小应使用 MCDNN_SEQDATA_DIM_COUNT 宏进行声明。mcdnnSeqDataDescriptor_t 容器被视为一个向量的集合，这些向量是形成序列的具有固定长度的向量，类似于构造句子的单词（字符向量）。TIME 维度涵盖序列长度。不同的序列会捆绑在一个 batch 中。一个 batch 可以是一组单独的序列或束。一个束是备选序列或候选序列的集群。我们可以把束考虑为从一种语言到另一种语言的翻译任务。在选择最佳译文之前，您可能需要保留并尝试使用原句的几个译文版本。保留的候选项数量是束宽。即使在同一个束内，每个序列都可以有不同的长度，因此朝向序列末尾的向量可以只是填充向量。paddingFill 参数指定如何在输出序列数据缓冲区中写入填充向量。paddingFill 参数指向一个 dataType 的值，该值应复制到填充向量中的所有元素。目前，paddingFill 唯一支持的值是 NULL，即应该忽略此选项。在这种情况下，输出缓冲区中填充向量的元素将具有未定义的值。假定非空序列始终从时间索引零开始。seqLengthArray[] 必须指定容器中的所有序列长度，因此该数组的总大小应为 dimA[MCDNN_SEQDATA_BATCH_DIM] * dimA[MCDNN_SEQDATA_BEAM_DIM]。seqLengthArray[] 数组的每个元素都应有一个非负值，该值应小于或等于 dimA[MCDNN_SEQDATA_TIME_DIM]；最大序列长度。seqLengthArray[] 中的元素始终按相同的批主序（batch-major order）排列，这意味着，考虑束和 batch 的维度时，当我们按地址的升序遍历数组，batch 是外部或较慢变化的索引。使用一个简单的示例，seqLengthArray[] 数组应按以下顺序保存序列长度：

```
{batch_idx=0, beam_idx=0}
{batch_idx=0, beam_idx=1}
{batch_idx=1, beam_idx=0}
{batch_idx=1, beam_idx=1}
{batch_idx=2, beam_idx=0}
{batch_idx=2, beam_idx=1}
```

当 dimA[MCDNN_SEQDATA_BATCH_DIM]=3 和 dimA[MCDNN_SEQDATA_BEAM_DIM]=2 时。存储在 mcdnnSeqDataDescriptor_t 容器中的数据必须符合以下条件：

所有数据都是完全压缩格式。单独的向量元素或连续向量之间没有未使用的空间或间隙。容器的最内维度是向量。换言之，dimA[MCDNN_SEQDATA_VECT_DIM] 元素的第一个连续组属于第一个向量，后面跟着第二个向量的元素，依此类推。mcdnnSetSeqDataDescriptor() 函数中的 axes 参数要复杂一些。此数组应与 dimA[] 具有相同的容量。axes[] 数组指定 GPU 内存中的实际数据布局。在此函数中，布局描述如下: 当我们通过增加元素指针从内存中的一个向量元素移动到另一个元素时，我们遇到的 VECT、TIME、BATCH 和 BEAM 维度的顺序是什么? 假设我们要定义以下数据布局：

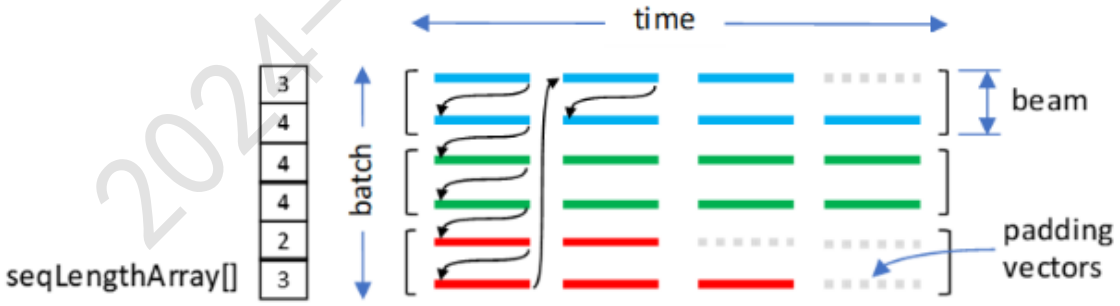

图 4. 对应于张量维度的 mcdnnSetSeqDataDescriptor() 的数据布局示例：

```
int dimA[MCDNN_SEQDATA_DIM_COUNT];
dimA[MCDNN_SEQDATA_TIME_DIM]   = 4;
dimA[MCDNN_SEQDATA_BATCH_DIM]  = 3;
dimA[MCDNN_SEQDATA_BEAM_DIM]   = 2;
dimA[MCDNN_SEQDATA_VECT_DIM]   = 5;
```

现在，我们初始化 axes[] 数组。请注意，最内维度由 axes[] 的最后一个活动元素描述。这里只有一个有效的配置，因为我们总是先遍历一个完整的向量。因此，我们需要在 axes[] 的最后一个活动元素中写入 MCDNN_SEQDATA_VECT_DIM。mcdnnSeqDataAxis_t axes[MCDNN_SEQDATA_DIM_COUNT];

```
axes[3] = MCDNN_SEQDATA_VECT_DIM;    // 3 = nbDims-1
```

现在，我们来研究 axes[] 的其余三个元素。当我们到达第一个向量的末尾时，我们跳到下一个束，因此：

```
axes[2] = MCDNN_SEQDATA_BEAM_DIM;
```

当我们接近第二个向量的末尾时，我们将移到下一个 batch，因此：

```
axes[1] = MCDNN_SEQDATA_BATCH_DIM;
```

最后一个（最外侧）维度为 TIME：

```
axes[0] = MCDNN_SEQDATA_TIME_DIM;
```

axes[] 数组的四个值完全描述了图中所示的数据布局。序列数据描述符允许用户选择 3!=6 种不同的数据布局或 BEAM、BATCH、TIME 维度的排列。多头注意力 API 支持所有六种布局。

参数

seqDataDesc

输出。指向已创建的序列数据描述符的指针。

dataType

输入。序列数据缓冲区的数据类型（MCDNN_DATA_HALF、MCDNN_DATA_FLOAT 或 MCDNN_DATA_DOUBLE）。

nbDims

输入。必须为 4。dimA[] 和 axes[] 数组中的活动维数。两个数组都应声明至少包含 MCDNN_SEQDATA_DIM_COUNT 个元素。

dimA[]

输入。包含序列数据维度的整数数组。使用 mcdnnSeqDataAxis_t 枚举类型以索引所有活动的 dimA[] 元素。

axes[]

输入。mcdnnSeqDataAxis_t 数组，用于定义内存中序列数据的布局。axes[] 的第一个 nbDims 元素应使用 axes[0] 的最外维度和 axes[nbDims-1] 的最内维度进行初始化。

seqLengthArraySize

输入。序列长度数组 seqLengthArray[] 中的元素数。

seqLengthArray[]

输入。整数数组，定义容器的所有序列长度。

paddingFill

输入。必须为 NULL。指向一个 dataType 值的指针，该值用于填充超出每个序列有效长度的输出向量，或者指向 NULL 以忽略此设置。

返回值

MCDNN_STATUS_SUCCESS

所有输入参数均已验证，序列数据描述符已成功更新。

MCDNN_STATUS_BAD_PARAM

找到一个无效的输入参数。例如:

- seqDataDesc=NULL

- dateType 不是有效的 mcdnnDataType_t 类型
- nbDims 为负值或零
- seqLengthArraySize 与期望的长度不匹配
- seqLengthArray[] 的某些元素无效

MCDNN_STATUS_NOT_SUPPORTED

输入参数不受支持。例如:

- nbDims 不等于 4
- paddingFill 不为 NULL

MCDNN_STATUS_ALLOC_FAILED

无法为序列数据描述符对象分配存储。

7 mcdnn_adv_train

此实体包含对应于 mcdnn_adv_infer 的训练相关功能与算法。mcdnn_adv_train 库依赖于 mcdnn_ops_infer，mcdnn_ops_train 和 mcdnn_adv_infer。

7.1 数据类型参考

以下为 mcdnn_adv_train.h 中的数据类型参考。

7.1.1 枚举类型

以下为 mcdnn_adv_train.h 中的枚举类型。

7.1.1.1 mcdnnLossNormalizationMode_t

mcdnnLossNormalizationMode_t 是一种枚举类型，用于控制损失函数（Loss function）的输入归一化模式。此类型可与 mcdnnSetCTCLossDescriptorEx() 一起使用。

值

MCDNN_LOSS_NORMALIZATION_NONE

mcdnnCTCLoss() 函数的输入 prob 应是归一化概率，而输出梯度是非归一化概率相关的损失梯度。

MCDNN_LOSS_NORMALIZATION_SOFTMAX

mcdnnCTCLoss() 函数的输入 prob 应是上一层的非归一化激活，而输出梯度是激活相关的梯度。概率是由 softmax 归一化内部计算得到的。

7.1.1.2 mcdnnWgradMode_t

mcdnnWgradMode_t 是一种枚举类型，用于选择如何更新缓冲区，该缓冲区保存损失函数的梯度，根据可训练参数计算所得。目前，此类型仅支持 mcdnnMultiHeadAttnBackwardWeights() 和 mcdnnRNNBackwardWeights_v8() 函数使用。

值

MCDNN_WGRAD_MODE_ADD

与新 batch 输入对应的权重梯度组件将添加到已求值的权重梯度中。使用此模式之前，将保存权重梯度的缓冲区初始化为零。或者，输出到未初始化缓冲区的第一个 API 调用应使用 MCDNN_WGRAD_MODE_SET 选项。

MCDNN_WGRAD_MODE_SET

与新 batch 输入对应的权重梯度组件会覆盖输出缓冲区中已存储的权重梯度。

7.2 API 参考

7.2.1 API 函数

以下为 mcdnn_adv_train.h 中的 API 函数。

7.2.1.1 mcdnnAdvTrainVersionCheck()

此函数检查库的 AdvTrain 子集的版本是否与其他子库一致。

```
mcdnnStatus_t mcdnnAdvTrainVersionCheck(void)
```

返回值

MCDNN_STATUS_SUCCESS

版本与其他子库一致。

MCDNN_STATUS_VERSION_MISMATCH

AdvTrain 的版本与其他子库不一致。用户应检查并确保所有子组件安装版本一致。

7.2.1.2 mcdnnCreateCTCLossDescriptor()

此函数创建 CTC 损失函数描述符。

```
mcdnnStatus_t mcdnnCreateCTCLossDescriptor(
    mcdnnCTCLossDescriptor_t* ctcLossDesc)
```

参数

ctcLossDesc

输出。要设置的 CTC 损失描述符。更多信息，参见 mcdnnCTCLossDescriptor_t。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

MCDNN_STATUS_BAD_PARAM

传入函数的 CTC 损失描述符无效。

MCDNN_STATUS_ALLOC_FAILED

此 CTC 损失描述符的内存分配失败。

7.2.1.3 mcdnnCTCLoss()

给定概率和标签，该函数返回 CTC 成本和梯度。

```
mcdnnStatus_t mcdnnCTCLoss(
    mcdnnHandle_t                       handle,
    const   mcdnnTensorDescriptor_t     probsDesc,
    const   void                       *probs,
    const   int                         hostLabels[],
    const   int                         hostLabelLengths[],
```

(下页继续)

(续上页)

```
    const   int                             hostInputLengths[],
    void                                   *costs,
    const   mcdnnTensorDescriptor_t         gradientsDesc,
    const   void                           *gradients,
    mcdnnCTCLossAlgo_t                      algo,
    const   mcdnnCTCLossDescriptor_t        ctcLossDesc,
    void                                   *workspace,
    size_t                                 *workSpaceSizeInBytes)
```

注 解: 根据所选的 mcdnnLossNormalizationMode_t（使用 mcdnnSetCTCLossDescriptionEx() 绑定到 mcdnnCTCLossDescriptor_t），此函数的接口可能不一致。对于 MCDNN_LOSS_NORMALIZATION_NONE，此函数的接口不一致，例如，probs 输入是由 softmax 归一化的概率，但梯度输出与非归一化激活有关。但是，对于 MCDNN_LOSS_NORMALIZATION_SOFTMAX，函数的接口一致；所有值均由 softmax 归一化。

参数

handle 输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

probsDesc

输入。已初始化的概率张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

probs

输入。指向已初始化的概率张量的指针。这些输入概率由 softmax 归一化。

hostLabels

输入。指向 CPU 内存中已初始化的标签列表的指针。

hostLabelLengths

输入。指向 CPU 内存中已初始化的长度列表的指针，以遍历上述标签列表。

hostInputLengths

输入。指向 CPU 内存中已初始化列表的指针，该列表为每个 batch 中时间步长的列表。

costs

输出。指向 CTC 计算成本的指针。

gradientsDesc

输入。已初始化的梯度张量描述符的句柄。

gradients

输出。指向 CTC 计算梯度的指针。这些计算的梯度输出与非归一化激活有关。

algo

输入。指定所选 CTC 损失算法的枚举。更多信息，参见 mcdnnCTCLossAlgo_t。

ctcLossDesc

输入。已初始化的 CTC 损失描述符的句柄。更多信息，参见 mcdnnCTCLossDescriptor_t。

workspace

输入。指针，指向执行指定算法所需的 GPU 内存工作空间。

sizeInBytes

输入。作为工作空间所需的 GPU 内存量，以便能够使用指定的算法执行 CTC 损失计算。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- probsDesc 的维度与 gradientsDesc 的维度不匹配。
- inputLengths 与 probsDesc 的第一维不一致。
- workSpaceSizeInBytes 不足。
- labelLengths 大于 255。

MCDNN_STATUS_NOT_SUPPORTED

选择了浮点以外的计算或数据类型，或者选择了未知算法类型。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

7.2.1.4 mcdnnCTCLoss_v8()

给定概率和标签，该函数返回 CTC 成本和梯度。

```
mcdnnStatus_t mcdnnCTCLoss_v8(
    mcdnnHandle_t                    handle,
    mcdnnCTCLossAlgo_t               algo,
    const   mcdnnCTCLossDescriptor_t ctcLossDesc,
    const   mcdnnTensorDescriptor_t  probsDesc,
    const   void                    *probs,
    const   int                      labels[],
    const   int                      labelLengths[],
    const   int                      inputLengths[],
    void                            *costs,
    const   mcdnnTensorDescriptor_t  gradientsDesc,
    const   void                    *gradients,
    size_t                          *workSpaceSizeInBytes,
    void                            *workspace)
```

注 解: 根据所选的 mcdnnLossNormalizationMode_t（使用 mcdnnSetCTCLossDescriptionEx() 绑定到 mcdnnCTCLossDescriptor_t），此函数的接口可能不一致。对于 MCDNN_LOSS_NORMALIZATION_NONE，此函数的接口不一致，例如，probs 输入是由 softmax 归一化的概率，但梯度输出与非归一化激活有关。但是，对于 MCDNN_LOSS_NORMALIZATION_SOFTMAX，函数的接口一致；所有值均由 softmax 归一化。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

algo

输入。指定所选 CTC 损失算法的枚举。更多信息，参见 mcdnnCTCLossAlgo_t。

ctcLossDesc

输入。已初始化的 CTC 损失描述符的句柄。要使用此 _v8 函数，必须使用 mcdnnSetCTCLossDescriptor_v8() 设置此描述符。更多信息，参见 mcdnnCTCLossDescriptor_t。

probsDesc

输入。已初始化的概率张量描述符的句柄。更多信息，参见 mcdnnTensorDescriptor_t。

probs

输入。指向已初始化的概率张量的指针。这些输入概率由 softmax 归一化。

labels

输入。指向 GPU 内存中已初始化的标签列表的指针。

labelLengths

输入。指向 GPU 内存中已初始化的长度列表的指针，以遍历上述标签列表。

inputLengths

输入。指向 GPU 内存中已初始化列表的指针，该列表为每个 batch 中时间步长的列表。

costs

输出。指向 CTC 计算成本的指针。

gradientsDesc

输入。已初始化的梯度张量描述符的句柄。

gradients

输出。指向 CTC 计算梯度的指针。这些计算的梯度输出与非归一化激活有关。

workspace

输入。指针，指向执行指定算法所需的 GPU 内存工作空间。

sizeInBytes

输入。作为工作空间所需的 GPU 内存量，以便能够使用指定的算法执行 CTC 损失计算。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- probsDesc 的维度与 gradientsDesc 的维度不匹配。
- inputLengths 与 probsDesc 的第一维不一致。
- workSpaceSizeInBytes 不足。
- labelLengths 大于 256。

MCDNN_STATUS_NOT_SUPPORTED

选择了浮点以外的计算或数据类型，或者选择了未知算法类型。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

7.2.1.5 mcdnnDestroyCTCLossDescriptor()

此函数用于销毁 CTC 损失函数描述符。

```
mcdnnStatus_t mcdnnDestroyCTCLossDescriptor(
    mcdnnCTCLossDescriptor_t        ctcLossDesc)
```

参数

ctcLossDesc

输入。要销毁的 CTC 损失函数描述符。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

7.2.1.6 mcdnnFindRNNBackwardDataAlgorithmEx()

此函数保留 API，不再单独实现。此函数使用用户分配的 GPU 内存尝试 mcdnnRNNBackwardData() 的所有可用 mcDNN 算法。它将影响算法性能的参数输出到用户分配的 mcdnnAlgorithmPerformance_t 数组。这些参数指标以一种有序的方式写入，其中第一个元素的计算时间最短。

```
mcdnnStatus_t mcdnnFindRNNBackwardDataAlgorithmEx(
    mcdnnHandle_t                    handle,
    const mcdnnRNNDescriptor_t       rnnDesc,
    const int                        seqLength,
    const mcdnnTensorDescriptor_t    *yDesc,
    const void                       *y,
    const mcdnnTensorDescriptor_t    *dyDesc,
    const void                       *dy,
    const mcdnnTensorDescriptor_t    dhyDesc,
    const void                       *dhy,
    const mcdnnTensorDescriptor_t    dcyDesc,
    const void                       *dcy,
    const mcdnnFilterDescriptor_t    wDesc,
    const void                       *w,
    const mcdnnTensorDescriptor_t    hxDesc,
    const void                       *hx,
    const mcdnnTensorDescriptor_t    cxDesc,
    const void                       *cx,
    const mcdnnTensorDescriptor_t    *dxDesc,
    void                             *dx,
    const mcdnnTensorDescriptor_t    dhxDesc,
    void                             *dhx,
    const mcdnnTensorDescriptor_t    dcxDesc,
    void                             *dcx,
    const float                      findIntensity,
    const int                        requestedAlgoCount,
    int                              *returnedAlgoCount,
    mcdnnAlgorithmPerformance_t      *perfResults,
    void                             *workspace,
    size_t                           workSpaceSizeInBytes,
    const void                       *reserveSpace,
    size_t                           reserveSpaceSizeInBytes)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

seqLength

输入。要展开的迭代次数。seqLength 的值不能超过 mcdnnGetRNNWorkspaceSize() 函数中用于查询执行 RNN 所需的工作空间大小。

yDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代输出（一个迭代一个描述符）。张量的第二维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第二维应与 hiddenSize 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第二维应是 hiddenSize 参数的 2 倍。

张量 n 的第一维必须与 dyDesc 中张量 n 的第一维匹配。

y

输入。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。

dyDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代输出梯度（一个迭代一个描述符）。张量的第二维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第二维应与 hiddenSize 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第二维应是 hiddenSize 参数的 2 倍。

张量 n 的第一维必须与 dxDesc 中张量 n 的第二维匹配。

dy

输入。数据指针，指向与 dyDesc 数组中张量描述符关联的 GPU 内存。

dhyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终隐藏状态梯度。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 dxDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

dhy

输入。数据指针，指向与张量描述符 dhyDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态梯度将初始化为零。

dcyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终单元状态梯度。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 dxDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

dcy

输入。数据指针，指向与张量描述符 dcyDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态梯度将初始化为零。

wDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 dxDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。

cxDesc

输入。完全压缩的的张量描述符，描述 LSTM 网络的初始单元（cell）状态。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 dxDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

cx

输入。数据指针，指向与张量描述符 cxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态将初始化为零。

dxDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代输入梯度（一个迭代一个描述符）。张量的第一维（批大小）可能从元素 n 减少到元素 n+1，但可能不会增加。每个张量描述符必须具有相同的第二维（向量长度）。

dx

输出。数据指针，指向与 dxDesc 数组中张量描述符关联的 GPU 内存。

dhxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态梯度。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 dxDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

dhx

输出。数据指针，指向与张量描述符 dhxDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会设置网络的隐藏输入梯度。

dcxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始单元状态梯度。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 dxDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

dcx

输出。数据指针，指向与张量描述符 dcxDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会设置网络的单元输入梯度。

findIntensity

输入。此输入在 mcDNN 之前的版本中未使用。在之后的版本中，它通过选择要搜索的空间占大的笛卡尔积空间的百分比，来控制 RNN 查找算法的总体运行时。

- 在 (0,1.] 范围内设置 findIntensity，则是设置要搜索的空间占整个 RNN 搜索空间的百分比。当 findIntensity 设置为 1.0 时，将对所有 RNN 参数执行完整搜索。
- 当 findIntensity 设置为 0.0f 时，将执行快速，最小的搜索。此设置可获得最佳运行时。但是，在这种情况下，此函数返回的参数不会获得算法的最佳性能；更大范围的搜索可能会发现更好的性能参数。此选项将最多执行三个已配置的 RNN problem 的实例。运行时会随 RNN problem size 而成比例地变化，在其他情况下也是如此，因此不能保证明确的时间限制。
- 在 [-1.,0) 范围内设置 findIntensity，则是设置要搜索的空间占归约笛卡尔积空间的百分比。为获得良好的性能，已启发式地选择此归约搜索空间。设置为-1.0 表示在此归约搜索空间上进行完整搜索。
- [-1,1] 范围以外的值被截断到 [-1,1] 范围中，然后按照上述说明进行解释。
- 此函数对大参数空间上的单个 RNN 执行进行乘积—一个参数组合一次执行。此函数返回的时间是运行总时间。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

reserveSpace

输入/输出。数据指针，指向要用作此调用预留空间的 GPU 内存。

reserveSpaceSizeInBytes

输入。指定已提供的 reserveSpace 的大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- dhxDesc、wDesc、hxDesc、cxDesc、dcxDesc、dhyDesc 或 dcyDesc 描述符中或者 yDesc、dxdesc、dydesc 描述符中至少有一个是无效的。

- yDesc、dxDesc、dyDesc、dhxDesc、wDesc、hxDesc、cxDesc、dcxDesc、dhyDesc 或 dcyDesc 的步幅或维度不正确。
- workSpaceSizeInBytes 太小。
- reserveSpaceSizeInBytes 太小。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

7.2.1.7 mcdnnFindRNNBackwardWeightsAlgorithmEx()

此函数保留 API，不再单独实现。此函数使用用户分配的 GPU 内存尝试 mcdnnRNNBackwardWeights() 的所有可用 mcDNN 算法。它将影响算法性能的参数输出到用户分配的 mcdnnAlgorithmPerformance_t 数组。这些参数指标以一种有序的方式写入，其中第一个元素的计算时间最短。

```
mcdnnStatus_t mcdnnFindRNNBackwardWeightsAlgorithmEx(
    mcdnnHandle_t                   handle,
    const mcdnnRNNDescriptor_t      rnnDesc,
    const int                       seqLength,
    const mcdnnTensorDescriptor_t   *xDesc,
    const void                      *x,
    const mcdnnTensorDescriptor_t   hxDesc,
    const void                      *hx,
    const mcdnnTensorDescriptor_t   *yDesc,
    const void                      *y,
    const float                     findIntensity,
    const int                       requestedAlgoCount,
    int                             *returnedAlgoCount,
    mcdnnAlgorithmPerformance_t     *perfResults,
    const void                      *workspace,
    size_t                          workSpaceSizeInBytes,
    const mcdnnFilterDescriptor_t   dwDesc,
    void                            *dw,
    const void                      *reserveSpace,
    size_t                          reserveSpaceSizeInBytes)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

seqLength

输入。要展开的迭代次数。seqLength 的值不能超过 mcdnnGetRNNWorkspaceSize() 函数中用于查询执行 RNN 所需的工作空间大小。

xDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代的输入（一个迭代一个描述符）。张量的第一维（批大小）可能从元素 n 减少到元素 n+1，但可能不会增加。每个张量描述符必须具有相同的第二维（向量长度）。

x

输入。数据指针，指向与 xDesc 数组中张量描述符关联的 GPU 内存。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。

yDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代输出（一个迭代一个描述符）。张量的第二维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第二维应与 hiddenSize 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第二维应是 hiddenSize 参数的 2 倍。

张量 n 的第一维必须与 dyDesc 中张量 n 的第一维匹配。

y

输入。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。

findIntensity

输入。此输入在 mcDNN 之前的版本中未使用。在之后的版本中，它通过选择要搜索的空间占大的笛卡尔积空间的百分比，来控制 RNN 查找算法的总体运行时。

- 在 (0,1.] 范围内设置 findIntensity，则是设置要搜索的空间占整个 RNN 搜索空间的百分比。当 findIntensity 设置为 1.0 时，将对所有 RNN 参数执行完整搜索。
- 当 findIntensity 设置为 0.0f 时，将执行快速，最小的搜索。此设置可获得最佳运行时。但是，在这种情况下，此函数返回的参数不会获得算法的最佳性能；更大范围的搜索可能会发现更好的性能参数。此选项将最多执行三个已配置的 RNN problem 的实例。运行时会随 RNN problem size 而成比例地变化，在其他情况下也是如此，因此不能保证明确的时间限制。
- 在 [-1.,0) 范围内设置 findIntensity，则是设置要搜索的空间占归约笛卡尔积空间的百分比。为获得良好的性能，已启发式地选择此归约搜索空间。设置为-1.0 表示在此归约搜索空间上进行完整搜索。
- [-1,1] 范围以外的值被截断到 [-1,1] 范围中，然后按照上述说明进行解释。
- 此函数对大参数空间上的单个 RNN 执行进行乘积—一个参数组合一次执行。此函数返回的时间是运行总时间。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

dwDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重梯度。

dw

输入/输出。数据指针，指向与卷积核描述符 dwDesc 关联的 GPU 内存。

reserveSpace

输入。数据指针，指向要用作此调用预留空间的 GPU 内存。

reserveSpaceSizeInBytes

输入。指定已提供的 reserveSpace 的大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- hxDesc、dwDesc 描述符中或者 xDesc、yDesc 描述符中至少有一个是无效的。
- xDesc、hxDesc、yDesc、dwDesc 中任一描述符的步幅或维度不正确。
- workSpaceSizeInBytes 太小。
- reserveSpaceSizeInBytes 太小。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

7.2.1.8 mcdnnFindRNNForwardTrainingAlgorithmEx()

此函数保留 API，不再单独实现。此函数使用用户分配的 GPU 内存尝试 mcdnnRNNForwardTraining() 的所有可用 mcDNN 算法。它将影响算法性能的参数输出到用户分配的 mcdnnAlgorithmPerformance_t 数组。这些参数指标以一种有序的方式写入，其中第一个元素的计算时间最短。

```
mcdnnStatus_t mcdnnFindRNNForwardTrainingAlgorithmEx(
    mcdnnHandle_t                    handle,
    const mcdnnRNNDescriptor_t       rnnDesc,
    const int                        seqLength,
    const mcdnnTensorDescriptor_t   *xDesc,
    const void                      *x,
    const mcdnnTensorDescriptor_t    hxDesc,
    const void                      *hx,
    const mcdnnTensorDescriptor_t    cxDesc,
    const void                      *cx,
    const mcdnnFilterDescriptor_t    wDesc,
```

(下页继续)

(续上页)

```
    const void                          *w,
    const mcdnnTensorDescriptor_t       *yDesc,
    void                                *y,
    const mcdnnTensorDescriptor_t        hyDesc,
    void                                *hy,
    const mcdnnTensorDescriptor_t        cyDesc,
    void                                *cy,
    const float                         findIntensity,
    const int                           requestedAlgoCount,
    int                                 *returnedAlgoCount,
    mcdnnAlgorithmPerformance_t         *perfResults,
    void                                *workspace,
    size_t                               workSpaceSizeInBytes,
    void                                *reserveSpace,
    size_t                               reserveSpaceSizeInBytes)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

xDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代的输入（一个迭代一个描述符）。张量的第一维（批大小）可能从元素 n 减少到元素 n+1，但可能不会增加。每个张量描述符必须具有相同的第二维（向量长度）。

seqLength

输入。要展开的迭代次数。seqLength 的值不能超过 mcdnnGetRNNWorkspaceSize() 函数中用于查询执行 RNN 所需的工作空间大小。

x

输入。数据指针，指向与 xDesc 数组中张量描述符关联的 GPU 内存。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。

cxDesc

输入。完全压缩的的张量描述符，描述 LSTM 网络的初始单元（cell）状态。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 中描述的张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

cx

输入。数据指针，指向与张量描述符 cxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态将初始化为零。

wDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

yDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代输出（一个迭代一个描述符）。张量的第二维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第二维应与 hiddenSize 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第二维应是 hiddenSize 参数的 2 倍。

张量 n 的第一维必须与 xDesc 中张量 n 的第一维匹配。

y

输出。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。

hyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终隐藏状态。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hy

输出。数据指针，指向与张量描述符 hyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终隐藏状态。

cyDesc

输入。完全压缩的张量描述符，描述 LSTM 网络的最终单元状态。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

cy

输出。数据指针，指向与张量描述符 cyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终单元状态。

findIntensity

输入。此输入之前未使用。在之后的版本中，它通过选择要搜索的空间占大的笛卡尔积空间的百分比，来控制 RNN 查找算法的总体运行时。

- 在 (0,1.] 范围内设置 findIntensity，则是设置要搜索的空间占整个 RNN 搜索空间的百分比。当 findIntensity 设置为 1.0 时，将对所有 RNN 参数执行完整搜索。

- 当 findIntensity 设置为 0.0f 时，将执行快速，最小的搜索。此设置可获得最佳运行时。但是，在这种情况下，此函数返回的参数不会获得算法的最佳性能；更大范围的搜索可能会发现更好的性能参数。此选项将最多执行三个已配置的 RNN problem 的实例。运行时会随 RNN problem size 而成比例地变化，在其他情况下也是如此，因此不能保证明确的时间限制。
- 在 [-1.,0) 范围内设置 findIntensity，则是设置要搜索的空间占归约笛卡尔积空间的百分比。为获得良好的性能，已启发式地选择此归约搜索空间。设置为-1.0 表示在此归约搜索空间上进行完整搜索。
- [-1,1] 范围以外的值被截断到 [-1,1] 范围中，然后按照上述说明进行解释。
- 此函数对大参数空间上的单个 RNN 执行进行乘积—一个参数组合一次执行。此函数返回的时间是运行总时间。

requestedAlgoCount

输入。要存储在 perfResults 中的最大元素数。

returnedAlgoCount

输出。存储在 perfResults 中的输出元素数。

perfResults

输出。用户分配的数组，用于存储按计算时间升序排序的性能指标。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

reserveSpace

输入/输出。数据指针，指向要用作此调用预留空间的 GPU 内存。

reserveSpaceSizeInBytes

输入。指定已提供的 reserveSpace 的大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- hxDesc、cxDesc、wDesc、hyDesc、cyDesc 描述符中或者 xDesc、yDesc 描述符中至少有一个是无效的。
- xDesc、hxDesc、cxDesc、wDesc、yDesc、hyDesc、cyDesc 中任一描述符的步幅或维度不正确。
- workSpaceSizeInBytes 太小。
- reserveSpaceSizeInBytes 太小。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

7.2.1.9 mcdnnGetCTCLossDescriptor()

该函数返回传入的 CTC 损失函数描述符的配置。

```
mcdnnStatus_t mcdnnGetCTCLossDescriptor(
    mcdnnCTCLossDescriptor_t          ctcLossDesc,
    mcdnnDataType_t*                  compType)
```

参数

ctcLossDesc

输入。传入的 CTC 损失函数描述符，用以从中检索配置。

compType

输出。与此 CTC 损失函数描述符关联的计算类型。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

MCDNN_STATUS_BAD_PARAM

传入的输入 ctcLossDesc 描述符无效。

7.2.1.10 mcdnnGetCTCLossDescriptorEx()

该函数返回传入的 CTC 损失函数描述符的配置。

```
mcdnnStatus_t mcdnnGetCTCLossDescriptorEx(
    mcdnnCTCLossDescriptor_t          ctcLossDesc,
    mcdnnDataType_t                  *compType,
    mcdnnLossNormalizationMode_t     *normMode,
    mcdnnNanPropagation_t            *gradMode)
```

参数

ctcLossDesc

输入。传入的 CTC 损失函数描述符，用以从中检索配置。

compType

输出。与此 CTC 损失函数描述符关联的计算类型。

normMode

输出。此 CTC 损失函数描述符的输入归一化类型。有关更多信息，请参见 mcdnnLossNormalizationMode_t。

gradMode

输出。此 CTC 损失函数描述符的 NaN 传播类型。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

MCDNN_STATUS_BAD_PARAM

传入的输入 ctcLossDesc 描述符无效。

7.2.1.11 mcdnnGetCTCLossDescriptor_v8()

该函数返回传入的 CTC 损失函数描述符的配置。

```
mcdnnStatus_t mcdnnGetCTCLossDescriptor_v8(
    mcdnnCTCLossDescriptor_t          ctcLossDesc,
    mcdnnDataType_t                  *compType,
    mcdnnLossNormalizationMode_t     *normMode,
    mcdnnNanPropagation_t            *gradMode,
    int                              *maxLabelLength)
```

参数

ctcLossDesc

输入。传入的 CTC 损失函数描述符，用以从中检索配置。

compType

输出。与此 CTC 损失函数描述符关联的计算类型。

normMode

输出。此 CTC 损失函数描述符的输入归一化类型。有关更多信息，请参见 mcdnnLossNormalizationMode_t。

gradMode

输出。此 CTC 损失函数描述符的 NaN 传播类型。

maxLabelLength

输出。此 CTC 损失函数描述符的最大标签长度。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

MCDNN_STATUS_BAD_PARAM

传入的输入 ctcLossDesc 描述符无效。

7.2.1.12 mcdnnGetCTCLossWorkspaceSize()

此函数返回用户应分配的 GPU 内存工作空间量，以便能够使用指定算法来调用 mcdnnCTCLoss() 函数。然后，分配的工作空间将传入 mcdnnCTCLoss() 函数。

```
mcdnnStatus_t mcdnnGetCTCLossWorkspaceSize(
    mcdnnHandle_t                       handle,
    const   mcdnnTensorDescriptor_t     probsDesc,
    const   mcdnnTensorDescriptor_t     gradientsDesc,
    const   int                        *labels,
    const   int                        *labelLengths,
    const   int                        *inputLengths,
    mcdnnCTCLossAlgo_t                  algo,
    const   mcdnnCTCLossDescriptor_t    ctcLossDesc,
    size_t                             *sizeInBytes)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

probsDesc

输入。已初始化的概率张量描述符的句柄。

gradientsDesc

输入。已初始化的梯度张量描述符的句柄。

labels

输入。指向已初始化的标签列表的指针。

labelLengths

输入。指向已初始化的长度列表的指针，以遍历上述标签列表。

inputLengths

输入。指向已初始化列表的指针，该列表为每个 batch 中时间步长的列表。

algo

输入。指定所选 CTC 损失算法的枚举。

ctcLossDesc

输入。已初始化的 CTC 损失描述符的句柄。

sizeInBytes

输出。作为工作空间所需的 GPU 内存量，以便能够使用指定的算法执行 CTC 损失计算。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- probsDesc 的维度与 gradientsDesc 的维度不匹配。
- inputLengths 与 probsDesc 的第一维不一致。
- workSpaceSizeInBytes 不足。
- labelLengths 大于 256。

MCDNN_STATUS_NOT_SUPPORTED

选择了浮点以外的计算或数据类型，或者选择了未知算法类型。

7.2.1.13 mcdnnGetCTCLossWorkspaceSize_v8()

此函数返回用户应分配的 GPU 内存工作空间量，以便能够使用指定算法来调用 mcdnnCTCLoss_v8() 函数。然后，分配的工作空间将传入 mcdnnCTCLoss_v8() 函数。

```
mcdnnStatus_t mcdnnGetCTCLossWorkspaceSize_v8(
    mcdnnHandle_t                       handle,
    mcdnnCTCLossAlgo_t                  algo,
    const   mcdnnCTCLossDescriptor_t    ctcLossDesc,
    const   mcdnnTensorDescriptor_t     probsDesc,
    const   mcdnnTensorDescriptor_t     gradientsDesc,
    size_t                             *sizeInBytes)
```

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

algo

输入。指定所选 CTC 损失算法的枚举。

ctcLossDesc

输入。已初始化的 CTC 损失描述符的句柄。

probsDesc

输入。已初始化的概率张量描述符的句柄。

gradientsDesc

输入。已初始化的梯度张量描述符的句柄。

sizeInBytes

输出。作为工作空间所需的 GPU 内存量，以便能够使用指定的算法执行 CTC 损失计算。

返回值

MCDNN_STATUS_SUCCESS

查询成功。

MCDNN_STATUS_BAD_PARAM

需满足以下条件：

- probsDesc 的维度与 gradientsDesc 的维度不匹配。

MCDNN_STATUS_NOT_SUPPORTED

选择了浮点以外的计算或数据类型，或者选择了未知算法类型。

7.2.1.14 mcdnnMultiHeadAttnBackwardData()

此函数根据其输入来计算多头注意力块的精确一阶导数：**Q、K、V**。如果 **y** = F (**x**) 是代表多头注意力层的向量值（vector-valued）函数，它将某个向量 $w \epsilon \mathbb{R}^n$ 作为输入（所有其他参数和输入均为常量），输出向量 $y \epsilon \mathbb{R}^m$，然后 mcdnnMultiHeadAttnBackwardData() 计算 $(\partial y_i / \partial x_j)^T \delta_{out}$ 的结果，其中 δ_{out} 是关于多头注意力输出的损失函数 $m \times 1$ 梯度。δ_{out} 梯度通过深度学习模型的前层（prior layers）进行反向传播。$\partial y_i / \partial x_j$ 是 F (x) 的 $m \times n$ 雅可比矩阵。通过 dout 参数提供输入，**Q、K、V** 的梯度结果写入 dqueries、dkeys 和 dvalues 缓冲区。

```
mcdnnStatus_t mcdnnMultiHeadAttnBackwardData(
    mcdnnHandle_t handle,
    const mcdnnAttnDescriptor_t attnDesc,
    const int loWinIdx[],
    const int hiWinIdx[],
    const int devSeqLengthsDQDO[],
    const int devSeqLengthsDKDV[],
    const mcdnnSeqDataDescriptor_t doDesc,
    const void *dout,
    const mcdnnSeqDataDescriptor_t dqDesc,
    void *dqueries,
    const void *queries,
    const mcdnnSeqDataDescriptor_t dkDesc,
    void *dkeys,
    const void *keys,
    const mcdnnSeqDataDescriptor_t dvDesc,
    void *dvalues,
    const void *values,
```

(下页继续)

(续上页)

```
    size_t weightSizeInBytes,
    const void *weights,
    size_t workSpaceSizeInBytes,
    void *workSpace,
    size_t reserveSpaceSizeInBytes,
    void *reserveSpace);
```

mcdnnMultiHeadAttnBackwardData() 函数不输出残差连接的偏导数，因为该结果等于 δ_{out} 。如果多头注意力模型启用直接来自 Q 的残差连接，则需要将 dout 张量添加到 dqueries 中，以获得后者的正确结果。此操作在 mcDNN

必须在 mcdnnMultiHeadAttnForward() 之后调用 mcdnnMultiHeadAttnBackwardData() 函数。oWinIdx[]、hiWinIdx[]、queries、keys、values、weights、reserveSpace 参数应与 mcdnnMultiHeadAttnForward() 调用中的参数相同。devSeqLengthsDQDO[] 和 devSeqLengthsDKDV[] 设备数组应包含与前向函数调用中的 devSeqLengthsQO[] 和 devSeqLengthsKV[] 数组相同的起始和结束注意力窗口索引。

注解: mcdnnMultiHeadAttnBackwardData() 不验证存储在 devSeqLengthsDQDO[] 和 devSeqLengthsDKDV[] 中的序列长度是否包含与相应序列数据描述符中的 seqLengthArray[] 相同的设置。

参数

handle

输入。当前的 mcDNN 上下文句柄。

attnDesc

输入。已初始化的注意力描述符。

loWinIdx[], hiWinIdx[]

输入。两个主机整数组，指定每个 Q 时间步的注意力窗口的起始和结束索引。包含 K，V 集合中的起始索引，而不包含结束索引。

devSeqLengthsDQDO[]

输入。设备数组，包含 dqDesc 或 doDesc 序列数据描述符中序列长度数组的拷贝。

devSeqLengthsDKDV[]

输入。设备数组，包含 dkDesc 或 dvDesc 序列数据描述符中序列长度数组的拷贝。

doDesc

输入。δ_{out} 梯度的描述符（关于多头注意力输出的损失函数偏导数向量）。

dout

指向设备内存中 δ_{out} 梯度数据的指针。

dqDesc

输入。queries 和 dqueries 序列数据的描述符。

dqueries

输出。设备指针，指向关于 queries 向量的计算损失函数梯度。

queries

输入。指向设备内存中 queries 数据的指针。这与 mcdnnMultiHeadAttnForward() 中的输入相同。

dkDesc

输入。keys 和 dkeys 序列数据的描述符。

dkeys

输出。设备指针，指向关于 keys 向量的计算损失函数梯度。

keys

输入。指向设备内存中 keys 数据的指针。这与 mcdnnMultiHeadAttnForward() 中的输入相同。

dvDesc

输入。values 和 dvalues 序列数据的描述符。

dvalues

输出。设备指针，指向关于 values 向量的计算损失函数梯度。

values

输入。指向设备内存中 values 数据的指针。这与 mcdnnMultiHeadAttnForward() 中的输入相同。

weightSizeInBytes

输入。权重缓冲区的大小（以字节为单位），存储了所有多头注意力可训练参数。

weights

输入。设备内存中权重缓冲区的地址。

workSpaceSizeInBytes

输入。用于临时 API 存储的工作空间缓冲区大小（以字节为单位）。

workSpace

输入/输出。设备内存中工作空间缓冲区的地址。

reserveSpaceSizeInBytes

输入。用于在正向和反向（梯度）API 调用之间进行数据交换的预留空间缓冲区大小（以字节为单位）。

reserveSpace

输入/输出。设备内存中预留空间缓冲区的地址。

返回值

MCDNN_STATUS_SUCCESS

处理 API 输入参数和启动 GPU 内核时，未检测到错误。

MCDNN_STATUS_BAD_PARAM

遇到一个无效或不兼容的输入参数。

MCDNN_STATUS_EXECUTION_FAILED

启动 GPU 内核的过程返回错误，或之前的内核未成功完成。

MCDNN_STATUS_INTERNAL_ERROR

内部状态不一致。

MCDNN_STATUS_NOT_SUPPORTED

请求的选项或输入参数组合不受支持。

MCDNN_STATUS_ALLOC_FAILED

共享内存不足，无法启动 GPU 内核。

7.2.1.15 mcdnnMultiHeadAttnBackwardWeights()

此函数根据其可训练参数（投影权重和投影 bias）来计算多头注意力块的精确一阶导数。如果 y = F (w) 是代表多头注意力层的向量值（vector-valued）函数，它将某个平展权重或 bias 的向量 $x\epsilon\mathbb{R}^n$ 作为输入（所有其他参数和输入均为固定的），输出向量 $y\epsilon\mathbb{R}^m$ ，然后 mcdnnMultiHeadAttnBackwardWeights() 计算 $(\partial y_i/\partial x_j)^T \delta_{out}$ 的结果，其中 δ_{out} 是关于多头注意力输出的损失函数 $m \times 1$ 梯度。δ_{out} 梯度通过深度学习模型的前层（prior layers）进行反向传播。$\partial y_i/\partial x_j$ 雅可比矩阵。δ_{out} 输入通过 dout 参数提供。

```
mcdnnStatus_t mcdnnMultiHeadAttnBackwardWeights(
    mcdnnHandle_t handle,
    const mcdnnAttnDescriptor_t attnDesc,
    mcdnnWgradMode_t addGrad,
    const mcdnnSeqDataDescriptor_t qDesc,
    const void *queries,
    const mcdnnSeqDataDescriptor_t kDesc,
    const void *keys,
    const mcdnnSeqDataDescriptor_t vDesc,
    const void *values,
    const mcdnnSeqDataDescriptor_t doDesc,
    const void *dout,
    size_t weightSizeInBytes,
    const void *weights,
    void *dweights,
    size_t workSpaceSizeInBytes,
    void *workSpace,
    size_t reserveSpaceSizeInBytes,
    void *reserveSpace);
```

有关权重和 bias 的所有梯度结果都写入 dweights 缓冲区。dweights 缓冲区的大小和组织与包含多头注意力权重和 bias 的 weights 缓冲区相同。mcDNN

关于权重或 bias 的损失函数梯度通常在多 batch 上计算。在这种情况下，应将每个 batch 中计算的结果相加。addGrad 参数指定是否应将当前 batch 的梯度添加到已计算的结果中，或者是否应使用新结果覆盖 dweights 缓冲区。

必须在 mcdnnMultiHeadAttnBackwardData() 之后调用 mcdnnMultiHeadAttnBackwardWeights() 函数。queries、keys、values、weights、reserveSpace 参数应与 mcdnnMultiHeadAttnForward() 和 mcdnnMultiHeadAttnBackwardData() 调用中的参数相同。dout 参数应与 mcdnnMultiHeadAttnBackwardData() 中的参数相同。

参数

handle

输入。当前的 mcDNN 上下文句柄。

attnDesc

输入。已初始化的注意力描述符。

addGrad

输入。权重梯度输出模式。

qDesc

输入。queries 序列数据的描述符。

queries

输入。指向设备内存中 queries 序列数据的指针。

kDesc

输入。keys 序列数据的描述符。

keys

输入。指向设备内存中 keys 序列数据的指针。

vDesc

输入。values 序列数据的描述符。

values

输入。指向设备内存中 values 序列数据的指针。

doDesc

输入。δ_{out} 梯度的描述符（关于多头注意力输出的损失函数偏导数向量）。

dout

输入。指向设备内存中 δ_{out} 梯度数据的指针。

weightSizeInBytes

输入。weights 和 dweights 缓冲区的大小（以字节为单位）。

weights

输入。设备内存中权重缓冲区的地址。

dweights

输出。设备内存中权重梯度缓冲区的地址。

workSpaceSizeInBytes

输入。用于临时 API 存储的工作空间缓冲区大小（以字节为单位）。

workSpace

输入/输出。设备内存中工作空间缓冲区的地址。

reserveSpaceSizeInBytes

输入。用于在正向和反向（梯度）API 调用之间进行数据交换的预留空间缓冲区大小（以字节为单位）。

reserveSpace

输入/输出。设备内存中预留空间缓冲区的地址。

返回值

MCDNN_STATUS_SUCCESS

处理 API 输入参数和启动 GPU 内核时，未检测到错误。

MCDNN_STATUS_BAD_PARAM

遇到一个无效或不兼容的输入参数。

MCDNN_STATUS_EXECUTION_FAILED

启动 GPU 内核的过程返回错误，或之前的内核未成功完成。

MCDNN_STATUS_INTERNAL_ERROR

内部状态不一致。

MCDNN_STATUS_NOT_SUPPORTED

请求的选项或输入参数组合不受支持。

7.2.1.16 mcdnnRNNBackwardData()

使用 mcdnnRNNBackwardData_v8() 代替 mcdnnRNNBackwardData()。

```
mcdnnStatus_t mcdnnRNNBackwardData(
    mcdnnHandle_t                    handle,
    const mcdnnRNNDescriptor_t       rnnDesc,
    const int                        seqLength,
    const mcdnnTensorDescriptor_t   *yDesc,
    const void                      *y,
    const mcdnnTensorDescriptor_t   *dyDesc,
    const void                      *dy,
    const mcdnnTensorDescriptor_t    dhyDesc,
    const void                      *dhy,
    const mcdnnTensorDescriptor_t    dcyDesc,
    const void                      *dcy,
    const mcdnnFilterDescriptor_t    wDesc,
    const void                      *w,
    const mcdnnTensorDescriptor_t    hxDesc,
    const void                      *hx,
    const mcdnnTensorDescriptor_t    cxDesc,
    const void                      *cx,
    const mcdnnTensorDescriptor_t   *dxDesc,
    void                            *dx,
    const mcdnnTensorDescriptor_t    dhxDesc,
    void                            *dhx,
    const mcdnnTensorDescriptor_t    dcxDesc,
    void                            *dcx,
    void                            *workspace,
    size_t                           workSpaceSizeInBytes,
    const void                      *reserveSpace,
    size_t                           reserveSpaceSizeInBytes)
```

此函数执行 rnnDesc 描述的循环神经网络，输出梯度为 dy、dhy、dhc，权重为 w，输入梯度为 dx、dhx、dcx。需要工作空间用于中间存储。reserveSpace 中的数据必须已由 mcdnnRNNForwardTraining() 生成。如果将来在相同的输入数据上执行 mcdnnRNNBackwardWeights() 调用，则必须使用相同的 reserveSpace 数据。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。更多信息，参见 mcdnnHandle_t。

rnnDesc

输入。已初始化的 RNN 描述符。更多信息，参见 mcdnnRNNDescriptor_t。

seqLength

输入。要展开的迭代次数。seqLength 的值不能超过 mcdnnGetRNNWorkspaceSize() 函数中用于查询执行 RNN 所需的工作空间大小。

yDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代输出（一个迭代一个描述符）。更多信息，参见 mcdnnTensorDescriptor_t。张量的第二维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第二维应与 hiddenSize 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第二维应是 hiddenSize 参数的 2 倍。

张量 n 的第一维必须与 dyDesc 中张量 n 的第一维匹配。

y

输入。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。

dyDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代输出梯度（一个迭代一个描述符）。张量的第二维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第二维应与 hiddenSize 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第二维应是 hiddenSize 参数的 2 倍。

张量 n 的第一维必须与 dxDesc 中张量 n 的第一维匹配。

dy

输入。数据指针，指向与 dyDesc 数组中张量描述符关联的 GPU 内存。

dhyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终隐藏状态梯度。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

dhy

输入。数据指针，指向与张量描述符 dhyDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态梯度将初始化为零。

dcyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终单元状态梯度。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

dcy

输入。数据指针，指向与张量描述符 dcyDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态梯度将初始化为零。

wDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重。更多信息，参见 mcdnnFilterDescriptor_t。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第二维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。

cxDesc

输入。完全压缩的的张量描述符，描述 LSTM 网络的初始单元（cell）状态。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第二维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

cx

输入。数据指针，指向与张量描述符 cxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态将初始化为零。

dxDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代输入梯度（一个迭代一个描述符）。张量的第一维（批大小）可能从元素 n 减少到元素 n+1，但可能不会增加。每个张量描述符必须具有相同的第二维（向量长度）。

dx

输出。数据指针，指向与 dxDesc 数组中张量描述符关联的 GPU 内存。

dhxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态梯度。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

dhx

输出。数据指针，指向与张量描述符 dhxDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会设置网络的隐藏输入梯度。

dcxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始单元状态梯度。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

dcx

输出。数据指针，指向与张量描述符 dcxDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会设置网络的单元输入梯度。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

reserveSpace

输入/输出。数据指针，指向要用作此调用预留空间的 GPU 内存。

reserveSpaceSizeInBytes

输入。指定已提供的 reserveSpace 的大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- dhxDesc、wDesc、hxDesc、cxDesc、dcxDesc、dhyDesc 或 dcyDesc 描述符中或者 yDesc、dxdesc、dydesc 描述符中至少有一个是无效的。
- yDesc、dxDesc、dyDesc、dhxDesc、wDesc、hxDesc、cxDesc、dcxDesc、dhyDesc 或 dcyDesc 的步幅或维度不正确。
- workSpaceSizeInBytes 太小。
- reserveSpaceSizeInBytes 太小。

MCDNN_STATUS_INVALID_VALUE

在 RNN 描述符中选择 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，在当前函数之前未调用 mcdnnSetPersistentRNNPlan()。

MCDNN_STATUS_MAPPING_ERROR

GPU/MXMACA 资源（如纹理对象，共享内存或零拷贝内存）没有所需的空间，或者用户资源与 mcDNN 内部资源不匹配。例如，在调用 mcdnnSetStream() 时，可能会出现资源不匹配。调用 mcdnnCreate() 时，用户提供的 MXMACA 流与 mcDNN 句柄中实例化的内部 MXMACA 事件之间可能不匹配。

此错误状态与纹理尺寸，共享内存大小或零拷贝内存可用性相关时，可能无法纠正。如果 mcdnnSetStream() 返回 MCDNN_STATUS_MAPPING_ERROR，则通常是可纠正的，但是，这意味着 mcDNN 句柄是在一个 GPU 上创建的，且传入此函数的用户流与另一个 GPU 相关联。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

7.2.1.17 mcdnnRNNBackwardData_v8()

此函数根据 RNN 模型的输入计算精确的一阶导数：x、hx 和 LSTM 单元 typealsocx。如果 o = [y, hy, cy] = F(x, hx, cx) = F(z) 是表示整个 RNN 模型的向量值函数，它将向量 x（对于所有时间步）和向量 hx、cx（对于所有层）作为连接到 $z \epsilon \mathbb{R}^n$ 的输入（假设网络权重和 bias 为常量），并输出连接到向量 $o \epsilon \mathbb{R}^m$ 的向量 y、hy、cy，然后 mcdnnRNNBackwardData_v8() 计算 $(\partial o_i / \partial z_j)^T \delta_{out}$ 的结果，其中 δ_{out} 是关于所有 RNN 输出的损失函数 $m \times 1$ 梯度。δ_{out} 梯度通过深度学习模型的前层进行反向传播，从模型输出开始。$\partial o_i / \partial z_j$ 是 F (z) 的 $m \times n$ 雅可比矩阵。δ_{out} 输入通过 dy、dhy 和 dcy 参数提供，梯度结果 $(\partial o_i / \partial z_j)^T \delta_{out}$ 写入 dx、dhx 和 dcx 缓冲区。

```
mcdnnStatus_t mcdnnRNNBackwardData_v8(
    mcdnnHandle_t handle,
    mcdnnRNNDescriptor_t rnnDesc,
    const int32_t devSeqLengths[],
    mcdnnRNNDataDescriptor_t yDesc,
    const void *y,
    const void *dy,
    mcdnnRNNDataDescriptor_t xDesc,
    void *dx,
    mcdnnTensorDescriptor_t hDesc,
    const void *hx,
    const void *dhy,
    void *dhx,
    mcdnnTensorDescriptor_t cDesc,
    const void *cx,
    const void *dcy,
    void *dcx,
    size_t weightSpaceSize,
    const void *weightSpace,
    size_t workSpaceSize,
    void *workSpace,
    size_t reserveSpaceSize,
    void *reserveSpace);
```

多层 RNN 模型中的 x、y、hx、cx、hy、cy、dx、dy、dhx、dcx、dhy 和 dcy 信号的位置如下图所示。请注意，mcdnnRNNBackwardData_v8() 函数不会暴露内部 RNN 信号（在时间步之间和层之间）。

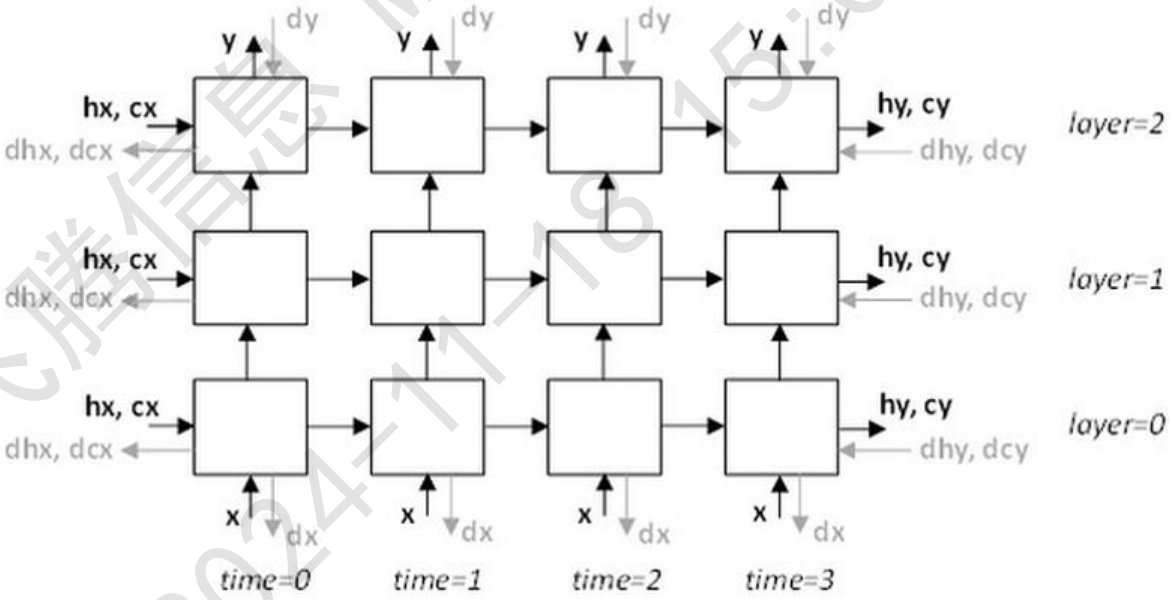

图 5. 多层 RNN 模型中的 x、y、hx、cx、hy、cy、dx、dy、dhx、dcx、dhy 和 dcy 信号的位置

主 RNN 输出 y 的内存地址，初始隐藏状态 hx 和初始单元状态 cx（仅限 LSTM）应指向与前面 mcdnnRNNForward() 调用相同的数据。dy 和 dx 指针不能为 NULL。

mcdnnRNNBackwardData_v8() 函数接受 dhy、dhx、dcy、dcx 缓冲区地址任意组合为 NULL。当 dhy 或 dcy 为 NULL 时，假定这些输入为零。当 dhx 或 dcx 指针为 NULL 时，则 mcdnnRNNBackwardData_v8() 不会写对应的结果。

当所有 hx、dhy、dhx 指针都为 NULL 时，相应的张量描述符 hDesc 也可以为 NULL。同样的规则适用于 cx，dcy，dcx 指针和 cDesc 张量描述符。

mcdnnRNNBackwardData_v8() 函数允许用户对输入 y、dy 和输出 dx 使用非压缩（填

充）布局。在压缩或非压缩布局（MCDNN_RNN_DATA_LAYOUT_SEQ_MAJOR_UNPACKED、MCDNN_RNN_DATA_LAYOUT_BATCH_MAJOR_UNPACKED）中，mini-batch 中的每个向量序列都有固定长度，是由 mcdnnSetRNNDataDescriptor() 函数中的 maxSeqLength 参数定义的。“非压缩”在此处指填充向量的存在，而不是连续向量之间未使用的地址范围。

每个已填充的固定长度序列都从一段有效向量开始。有效的向量计数存储在传入 mcdnnSetRNNDataDescriptor() 的 seqLengthArray 中，这样对于 mini-batch 中的所有序列，0 < seqLengthArray[i] <= maxSeqLength，即对于 i=0..batchSize-1。其余填充向量使组合序列长度等于 maxSeqLength。支持 sequence-major 和 batch-major 填充布局。

此外，用户可以选择 sequence-major 压缩布局：MCDNN_RNN_DATA_LAYOUT_SEQ_MAJOR_PACKED。在后一种布局中，mini-batch 中的向量序列根据序列长度按降序排序存储。首先，存储时间步为零的所有向量。然后存储时间步为 1 的向量，依此类推。此布局不使用填充矢量。

必须在 xDesc 和 yDesc 描述符中指定相同的布局类型。

RNN 数据描述符 xDesc 和 yDesc 中名为 seqLengthArray 的两个主机数组必须相同。此外，必须通过 devSeqLengths 参数传递设备内存中 seqLengthArray 的拷贝。此数组直接提供给 GPU 内核。mcdnnRNNBackwardData_v8() 函数不用于验证存储在 GPU 内存中 devSeqLengths 中的序列长度与 CPU 内存中 xDesc 和 yDesc 描述符中的是否相同。但是，会检查 xDesc 和 yDesc 描述符中的序列长度数组是否一致。

必须在 mcdnnRNNForward() 之后调用 mcdnnRNNBackwardData_v8() 函数。调用 mcdnnRNNForward() 函数时，应将 mcdnnRNNForward() 类型的 fwdMode 参数设置为 MCDNN_FWD_MODE_TRAINING。

参数

handle

输入。当前的 mcDNN 上下文句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

devSeqLengths

输入。RNN 数据描述符 xDesc 或 yDesc 中 seqLengthArray 的拷贝。devSeqLengths 数组必须存储在 GPU 内存中，因为该数组可能在 mcdnnRNNBackwardData_v8() 函数之后，被 GPU 内核异步访问。此参数不能为 NULL。

yDesc

输入。与 RNN 模型主输出对应的已初始化的描述符。dataType、layout、maxSeqLength、batchSize 和 seqLengthArray 必须与 xDesc 中的匹配。

y, dy

输入。指向 GPU 缓冲区的数据指针，该缓冲区包含 RNN 模型的主输出和梯度增量（关于 y 的损失函数梯度）。y 输出应由前面的 mcdnnRNNForward() 调用产生。y 和 dy 向量应根据 yDesc 指定的布局在内存中排列。张量中的元素（包括填充向量元素）必须紧密压缩。y 和 dy 参数不能为 NULL。

xDesc

输入。已初始化的 RNN 数据描述符，该描述符对应于与 RNN 主模型输入相关的损失函数梯度。dataType、layout、maxSeqLength、batchSize 和 seqLengthArray 必须与 yDesc 中的匹配。vectorSize 参数必须与传入 mcdnnSetRNNDescriptor_V8 函数的 inputSize 参数匹配。

dx

输出。指向 GPU 内存的数据指针，该内存中应存储与 RNN 主输入 x 相关的损失函数反向传播梯度。向量应根据 xDesc 指定的布局在内存中排列。张量中的元素（包括填充向量）必须紧密压缩。此参数不能为 NULL。

hDesc

输入。张量描述符，描述初始 RNN 隐藏状态 hx 和关于最终隐藏状态初始值的损失函数的梯度。隐藏状态数据和相应的梯度为完全压缩格式。张量的第一维取决于传入 mcdnnSetRNNDescriptor_v8() 函数的 dirMode 参数。

- 如果 dirMode 为 MCDNN_UNIDIRECTIONAL，则第一维应与传入 mcdnnSetRNNDescriptor_v8() 的 numLayers 参数匹配。
- 如果 dirMode 为 MCDNN_BIDIRECTIONAL，则第一维应是传入 mcdnnSetRNNDescriptor_v8() 的 numLayers 参数的 2 倍。

第二维必须与 xDesc 中描述的 batchSize 参数匹配。第三维取决于 RNN 模式是否为 MCDNN_LSTM 以及是否启用了 LSTM 投影。具体而言：

- 如果 RNN 模式为 MCDNN_LSTM 且启用了 LSTM 投影，则第三维必须与传入 mcdnnSetRNNDescriptor_v8() 调用的 projSize 参数匹配。
- 否则，第三维必须与传入 mcdnnSetRNNDescriptor_v8() 调用的 hiddenSize 参数匹配，其中 mcdnnSetRNNDescriptor_v8() 是用于初始化 rnnDesc。

hx, dhy

输入。包含 RNN 初始隐藏状态 hx 和梯度增量 dhy 的 GPU 缓冲区地址。数据维度由 hDesc 张量描述符描述。如果 hx 或 dhy 参数中传入了 NULL 指针，则假定相应的缓冲区全为零。

dhx

输出。指向 GPU 缓冲区的指针，该缓冲区应存储与初始隐藏状态变量对应的一阶导数。数据维度由 hDesc 张量描述符描述。如果将 NULL 指针分配给 dhx，则不会保存反向传播导数。

cDesc

输入。仅适用于 LSTM 网络。张量描述符，描述初始单元状态 cx 和关于最终单元状态初始值的损失函数的梯度。单元状态数据为完全压缩格式。张量的第一维取决于传入 mcdnnSetRNNDescriptor_v8() 函数的 dirMode 参数。

> 如果 dirMode 为 MCDNN_UNIDIRECTIONAL，则第一维应与传入 mcdnnSetRNNDescriptor_v8() 的 numLayers 参数匹配。如果 dirMode 为 MCDNN_BIDIRECTIONAL，则第一维应是传入 mcdnnSetRNNDescriptor_v8() 的 numLayers 参数的 2 倍。

第二维必须与 xDesc 中描述的 batchSize 参数匹配。否则，第三维必须与传入 mcdnnSetRNNDescriptor_v8() 调用的 hiddenSize 参数匹配。

cx, dcy

输入。仅适用于 LSTM 网络。包含初始 LSTM 状态数据 cx 和梯度增量 dcy 的 GPU 缓冲区地址。数据维度由 cDesc 张量描述符描述。如果 cx 或 dcy 参数中传入了 NULL 指针，则假定相应的缓冲区全为零。

dcx

输出。仅适用于 LSTM 网络。指向 GPU 缓冲区的指针，该缓冲区应存储与初始 LSTM 状态变量对应的一阶导数。数据维度由 cDesc 张量描述符描述。如果将 NULL 指针分配给 dcx，则不会保存反向传播导数。

weightSpaceSize

输入。指定已提供的权重空间缓冲区的大小（以字节为单位）。

weightSpace

输入。GPU 内存中权重空间缓冲区的地址。

workSpaceSize

输入。指定已提供的工作空间缓冲区的大小（以字节为单位）。

workSpace

输入/输出。GPU 内存中用于存储临时数据的工作空间缓冲区地址。

reserveSpaceSize

输入。指定预留空间缓冲区的大小（以字节为单位）。

reserveSpace

输入/输出。GPU 内存中预留空间缓冲区的地址。

返回值

MCDNN_STATUS_SUCCESS

处理 API 输入参数和启动 GPU 内核时，未检测到错误。

MCDNN_STATUS_NOT_SUPPORTED

需至少满足以下任一条件：

- 在指定 MCDNN_RNN_ALGO_PERSIST_STATIC 或 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，传入变量序列长度输入
- 在 pre-Pascal 设备上请求 MCDNN_RNN_ALGO_PERSIST_STATIC 或 MCDNN_RNN_ALGO_PERSIST_DYNAMICC
- “double”浮点类型用于输入/输出和 MCDNN_RNN_ALGO_PERSIST_STATIC 算法

MCDNN_STATUS_BAD_PARAM

遇到一个无效或不兼容的输入参数。例如：

- 一些输入描述符为 NULL
- rnnDesc、xDesc、yDesc、hDesc 或 cDesc 描述符中的设置无效
- weightSpaceSize、workSpaceSize 或 reserveSpaceSize 太小。

MCDNN_STATUS_MAPPING_ERROR

GPU/MXMACA 资源（如纹理对象，共享内存或零拷贝内存）没有所需的空间，或者用户资源与 mcDNN 内部资源不匹配。例如，在调用 mcdnnSetStream() 时，可能会出现资源不匹配。调用 mcdnnCreate() 时，用户提供的 MXMACA 流与 mcDNN 句柄中实例化的内部 MXMACA 事件之间可能不匹配。

此错误状态与纹理尺寸，共享内存大小或零拷贝内存可用性相关时，可能无法纠正。如果 mcdnnSetStream() 返回 MCDNN_STATUS_MAPPING_ERROR，则通常是可纠正的，但是，这意味着 mcDNN 句柄是在一个 GPU 上创建的，且传入此函数的用户流与另一个 GPU 相关联。

MCDNN_STATUS_EXECUTION_FAILED

启动 GPU 内核的过程返回错误，或之前的内核未成功完成。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配 CPU 内存。

7.2.1.18 mcdnnRNNBackwardDataEx()

使用 mcdnnRNNBackwardData_v8() 代替 mcdnnRNNBackwardDataEx()。

```
mcdnnStatus_t mcdnnRNNBackwardDataEx(
    mcdnnHandle_t                       handle,
    const mcdnnRNNDescriptor_t          rnnDesc,
    const mcdnnRNNDataDescriptor_t      yDesc,
    const void                          *y,
    const mcdnnRNNDataDescriptor_t      dyDesc,
    const void                          *dy,
    const mcdnnRNNDataDescriptor_t      dcDesc,
```

(下页继续)

(续上页)

```
    const void                          *dcAttn,
    const mcdnnTensorDescriptor_t       dhyDesc,
    const void                          *dhy,
    const mcdnnTensorDescriptor_t       dcyDesc,
    const void                          *dcy,
    const mcdnnFilterDescriptor_t       wDesc,
    const void                          *w,
    const mcdnnTensorDescriptor_t       hxDesc,
    const void                          *hx,
    const mcdnnTensorDescriptor_t       cxDesc,
    const void                          *cx,
    const mcdnnRNNDataDescriptor_t      dxDesc,
    void                                *dx,
    const mcdnnTensorDescriptor_t       dhxDesc,
    void                                *dhx,
    const mcdnnTensorDescriptor_t       dcxDesc,
    void                                *dcx,
    const mcdnnRNNDataDescriptor_t      dkDesc,
    void                                *dkeys,
    void                                *workSpace,
    size_t                              workSpaceSizeInBytes,
    void                                *reserveSpace,
    size_t                              reserveSpaceSizeInBytes)
```

该函数是 mcdnnRNNBackwardData() 函数的扩展版本。mcdnnRNNBackwardDataEx() 函数允许用户对输入 y 和输出 dx 使用非压缩（填充）布局。

在非压缩布局中，mini-batch 中的每个序列都被视为固定长度，由其对应的 RNNDataDescriptor 中的 maxSeqLength 指定。每个固定长度序列（例如，mini-batch 中的第 n 个序列）都由一个有效段（由其对应的 RNNDataDescriptor 中的 seqLengthArray[n] 指定）和一个填充段组成，使组合序列长度等于 maxSeqLength。

对于非压缩布局，同时支持序列主要（sequence-major）布局（即，时间主要）和批主要（batch-major）布局。为了向后兼容，支持压缩的序列主要布局。但是，与非扩展函数 mcdnnRNNBackwardData() 类似，mini-batch 中的序列需要根据长度按降序排序。

参数

handle

输入。已创建的上下文的句柄。此函数保留 API，不再单独实现。

rnnDesc

输入。已初始化的 RNN 描述符。

yDesc

输入。已初始化的 RNN 数据描述符。必须与已传入 mcdnnRNNForwardTrainingEx() 的描述符匹配或完全相同。

y

输入。数据指针，指向与 RNN 数据描述符 yDesc 关联的 GPU 内存。向量应根据 yDesc 指定的布局在内存中排布。张量中的元素（包括填充向量中的元素）必须紧密压缩，且不支持步幅。必须包含与 mcdnnRNNForwardTrainingEx() 已生成的完全相同的数据。

dyDesc

输入。已初始化的 RNN 数据描述符。dataType、layout、maxSeqLength、batchSize、vectorSize 和 seqLengthArray 需要与已传入 mcdnnRNNForwardTrainingEx() 的 yDesc 匹配。

dy

输入。数据指针，指向与 RNN 数据描述符 dyDesc 关联的 GPU 内存。向量应根据 dyDesc 指定的布局在内存中排布。张量中的元素（包括填充向量中的元素）必须紧密压缩，且不支持步幅。

dhyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终隐藏状态梯度。张量的第一维取决于初始化 rnnDesc 的方向参数。此外：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 中描述的 batchSize 参数匹配。第三维取决于 RNN 模式是否为 MCDNN_LSTM 以及是否启用了 LSTM 投影。此外：

- 如果 RNN 模式为 MCDNN_LSTM 且启用了 LSTM 投影，则第三维必须与传入 mcdnnSetRNNProjectionLayers() 调用的 recProjSize 参数匹配，其中 mcdnnSetRNNProjectionLayers() 是用于设置 rnnDesc 的。
- 否则，第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。

dhy

输入。数据指针，指向与张量描述符 dhyDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态梯度将初始化为零。

dcyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终单元状态梯度。张量的第一维取决于初始化 rnnDesc 的方向参数。此外：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

dcy

输入。数据指针，指向与张量描述符 dcyDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态梯度将初始化为零。

wDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。必须与已传入 mcdnnRNNForwardTrainingEx() 的描述符匹配或完全相同。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。必须包含与已传入 mcdnnRNNForwardTrainingEx() 的完全相同的数据，或者如果已将 NULL 传入 mcdnnRNNForwardTrainingEx()，则必须为 NULL。

cxDesc

输入。完全压缩的的张量描述符，描述 LSTM 网络的初始单元（cell）状态。必须与已传入 mcdnnRNNForwardTrainingEx() 的描述符匹配或完全相同。

cx

输入。数据指针，指向与张量描述符 cxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态将初始化为零。必须包含与已传入 mcdnnRNNForwardTrainingEx() 的完全相同的数据，或者如果已将 NULL 传入 mcdnnRNNForwardTrainingEx()，则必须为 NULL。

dxDesc

输入。已初始化的 RNN 数据描述符。dataType、layout、maxSeqLength、batchSize、vectorSize 和 seqLengthArray 需要与已传入 mcdnnRNNForwardTrainingEx() 的 xDesc 匹配。

dx

输出。数据指针，指向与 RNN 数据描述符 dxDesc 关联的 GPU 内存。向量应根据 dxDesc 指定的布局在内存中排布。张量中的元素（包括填充向量中的元素）必须紧密压缩，且不支持步幅。

dhxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态梯度。此描述符的设置必须与 dhyDesc 完全一致。

dhx

输出。数据指针，指向与张量描述符 dhxDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会设置网络的隐藏输入梯度。

dcxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始单元状态梯度。此描述符的设置必须与 dcyDesc 完全一致。

dcx

输出。数据指针，指向与张量描述符 dcxDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会设置网络的单元输入梯度。

dkDesc

保留。用户可以传入 NULL。

dkeys

保留。用户可以传入 NULL。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

reserveSpace

输入/输出。数据指针，指向要用作此调用预留空间的 GPU 内存。

reserveSpaceSizeInBytes

输入。指定已提供的 reserveSpace 的大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

需至少满足以下任一条件：

- 在使用 MCDNN_RNN_ALGO_PERSIST_STATIC 或 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，传入变量序列长度输入。
- 在 pre-Pascal 设备上使用 MCDNN_RNN_ALGO_PERSIST_STATIC 或 MCDNN_RNN_ALGO_PERSIST_DYNAMIC。

- 双输入/输出用于 MCDNN_RNN_ALGO_PERSIST_STATIC。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- yDesc、dxDesc、dyDesc、dhxDesc、wDesc、hxDesc、cxDesc、dcxDesc、dhyDesc、dcyDesc 中至少有一个描述符无效，或者步幅或维度不正确。
- workSpaceSizeInBytes 太小。
- reserveSpaceSizeInBytes 太小。

MCDNN_STATUS_INVALID_VALUE

在 RNN 描述符中选择 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，在当前函数之前未调用 mcdnnSetPersistentRNNPlan()。

MCDNN_STATUS_MAPPING_ERROR

GPU/MXMACA 资源（如纹理对象，共享内存或零拷贝内存）没有所需的空间，或者用户资源与 mcDNN 内部资源不匹配。例如，在调用 mcdnnSetStream() 时，可能会出现资源不匹配。调用 mcdnnCreate() 时，用户提供的 MXMACA 流与 mcDNN 句柄中实例化的内部 MXMACA 事件之间可能不匹配。

此错误状态与纹理尺寸，共享内存大小或零拷贝内存可用性相关时，可能无法纠正。如果 mcdnnSetStream() 返回 MCDNN_STATUS_MAPPING_ERROR，则通常是可纠正的，但是，这意味着 mcDNN 句柄是在一个 GPU 上创建的，且传入此函数的用户流与另一个 GPU 相关联。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

7.2.1.19 mcdnnRNNBackwardWeights()

使用 mcdnnRNNBackwardWeights_v8() 代替 mcdnnRNNBackwardWeights()。

```
mcdnnStatus_t mcdnnRNNBackwardWeights(
    mcdnnHandle_t                    handle,
    const mcdnnRNNDescriptor_t       rnnDesc,
    const int                        seqLength,
    const mcdnnTensorDescriptor_t   *xDesc,
    const void                      *x,
    const mcdnnTensorDescriptor_t    hxDesc,
    const void                      *hx,
    const mcdnnTensorDescriptor_t   *yDesc,
    const void                      *y,
    const void                      *workspace,
    size_t                           workSpaceSizeInBytes,
    const mcdnnFilterDescriptor_t    dwDesc,
    void                            *dw,
    const void                      *reserveSpace,
    size_t                           reserveSpaceSizeInBytes)
```

此函数通过输入 x、hx 和输出 y，从 rnnDesc 描述的循环神经网络中累积权重梯度 dw。在这种情况下，操作模式是累加的，计算出的权重梯度将添加到 dw 中已存在的梯度中。需要工作空间用于中间存储。reserveSpace 中的数据必须已由 mcdnnRNNBackwardData() 生成。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

seqLength

输入。要展开的迭代次数。seqLength 的值不能超过 mcdnnGetRNNWorkspaceSize() 函数中用于查询执行 RNN 所需的工作空间大小。

xDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代的输入（一个迭代一个描述符）。张量的第一维（批大小）可能从元素 n 减少到元素 n+1，但可能不会增加。每个张量描述符必须具有相同的第二维（向量长度）。

x

输入。数据指针，指向与 xDesc 数组中张量描述符关联的 GPU 内存。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。

yDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代输出（一个迭代一个描述符）。张量的第二维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第二维应与 hiddenSize 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第二维应是 hiddenSize 参数的 2 倍。

张量 n 的第一维必须与 dyDesc 中张量 n 的第一维匹配。

y

输入。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

dwDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重梯度。

dw

输入/输出。数据指针，指向与卷积核描述符 dwDesc 关联的 GPU 内存。

reserveSpace

输入。数据指针，指向要用作此调用预留空间的 GPU 内存。

reserveSpaceSizeInBytes

输入。指定已提供的 reserveSpace 的大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- hxDesc、dwDesc 描述符中或者 xDesc、yDesc 描述符中至少有一个是无效的。
- xDesc、hxDesc、yDesc、dwDesc 中任一描述符的步幅或维度不正确。
- workSpaceSizeInBytes 太小。
- reserveSpaceSizeInBytes 太小。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

7.2.1.20 mcdnnRNNBackwardWeights_v8()

此函数计算 RNN 模型中所有可训练参数（权重和 bias）的一阶精确导数。如果 o = [y, hy, cy] = F(w) 是代表多层 RNN 模型的向量值函数，它将某个平展权重或 bias 的向量 $w \epsilon \mathbb{R}^n$ 作为输入（所有其他数据输入为常量），输出向量 $o \epsilon \mathbb{R}^m$，mcdnnRNNBackwardWeights_v8() 计算 $(\partial o_i / \partial w_j)^T \delta_{out}$ 的结果，其中 δ_{out} 是关于所有 RNN 输出的损失函数 $m \times 1$ 梯度。δ_{out} 梯度通过深度学习模型的前层进行反向传播，从模型输出开始。$\partial o_i / \partial w_j$ 是 F(w) 的 $m \times n$ 雅可比矩阵。δ_{out} 输入由 mcdnnRNNBackwardData_v8() 函数中的 dy、dhy 和 dcy 参数提供。

```
mcdnnStatus_t mcdnnRNNBackwardWeights(
    mcdnnHandle_t                    handle,
    const mcdnnRNNDescriptor_t       rnnDesc,
    const int                        seqLength,
    const mcdnnTensorDescriptor_t   *xDesc,
    const void                      *x,
    const mcdnnTensorDescriptor_t    hxDesc,
    const void                      *hx,
    const mcdnnTensorDescriptor_t   *yDesc,
    const void                      *y,
    const void                      *workspace,
    size_t                           workSpaceSizeInBytes,
    const mcdnnFilterDescriptor_t    dwDesc,
    void                            *dw,
    const void                      *reserveSpace,
    size_t                           reserveSpaceSizeInBytes)
```

有关权重和 bias 的所有 $(\partial o_i/\partial w_j)^T \delta_{out}$ 梯度结果都写入 dweightSpace 缓冲区。dweightSpace 缓冲区的大小和组织与包含 RNN 权重和 bias 的 weightSpace 缓冲区相同。

关于权重和 bias 的损失函数梯度通常在多个 mini-batch 上计算。在这种情况下，应将每个 mini-batch 中计算的结果相加。addGrad 参数指定是否应将当前 mini-batch 的梯度添加到已计算的结果中（MCDNN_WGRAD_MODE_ADD），或者是否应使用新结果覆盖 dweightSpace 缓冲区（MCDNN_WGRAD_MODE_SET）。目前，mcdnnRNNBackwardWeights_v8() 函数仅支持 MCDNN_WGRAD_MODE_ADD 模式，因此用户应在首次调用函数之前将 dweightSpace 缓冲区归零。

必须在 xDesc 描述符和设备数组 devSeqLengths 中配置相同的序列长度。必须在 mcdnnRNNBackwardData() 之后调 mcdnnRNNBackwardWeights_v8() 函数。

参数

handle

输入。当前的 mcDNN 上下文句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

addGrad

输入。权重梯度输出模式。更多信息，参见 mcdnnWgradMode_t 枚举类型的说明。目前，mcdnnRNNBackwardWeights_v8() 函数仅支持 MCDNN_WGRAD_MODE_ADD 模式。

devSeqLengths

输入。RNN 数据描述符 xDesc 中 seqLengthArray 的拷贝。devSeqLengths 数组必须存储在 GPU 内存中，因为该数组可能在 mcdnnRNNBackwardWeights_v8() 函数之后，被 GPU 内核异步访问。

xDesc

输入。与 RNN 模型输入数据对应的已初始化的描述符。此描述符与前面的 mcdnnRNNForward() 和 mcdnnRNNBackwardData_v8() 调用中使用的 RNN 数据描述符相同。

x

输入。指针，指向存储主 RNN 输入的 GPU 缓冲区。此缓冲地址 x 应与前面的 mcdnnRNNForward() 和 mcdnnRNNBackwardData_v8() 调用中提供的相同。

hDesc

输入。描述 RNN 初始隐藏状态的张量描述符。隐藏状态数据为完全压缩格式。此描述符与前面的 mcdnnRNNForward() 和 mcdnnRNNBackwardData_v8() 调用中使用的张量描述符相同。

hx

输入。指针，指向具有 RNN 初始隐藏状态的 GPU 缓冲区。此缓冲地址 hx 应与前面的 mcdnnRNNForward() 和 mcdnnRNNBackwardData_v8() 调用中提供的相同。

yDesc

输入。与 RNN 模型输出数据对应的已初始化的描述符。此描述符与前面的 mcdnnRNNForward() 和 mcdnnRNNBackwardData_v8() 调用中使用的 RNN 数据描述符相同。

y

输出。指向 GPU 缓冲区的指针，该缓冲区中主 RNN 输出由先前的 mcdnnRNNForward() 调用生成。y 缓冲区中的数据由 yDesc 描述符描述。y 张量中的元素（包括填充向量元素）必须紧密压缩。

weightSpaceSize

输入。指定已提供的权重空间缓冲区的大小（以字节为单位）。

dweightSpace

输出。GPU 内存中权重空间缓冲区的地址。

workSpaceSize

输入。指定已提供的工作空间缓冲区的大小（以字节为单位）。

workSpace

输入/输出。GPU 内存中用于存储临时数据的工作空间缓冲区地址。

reserveSpaceSize

输入。指定预留空间缓冲区的大小（以字节为单位）。

reserveSpace

输入/输出。GPU 内存中预留空间缓冲区的地址。

返回值

MCDNN_STATUS_SUCCESS

处理 API 输入参数和启动 GPU 内核时，未检测到错误。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

遇到一个无效或不兼容的输入参数。例如：

- 一些输入描述符为 NULL
- rnnDesc、xDesc、yDesc 或 hDesc 描述符中的设置无效
- weightSpaceSize，workSpaceSize 或 reserveSpaceSize 的值太小。
- addGrad 参数与 MCDNN_WGRAD_MODE_ADD 不相等

MCDNN_STATUS_EXECUTION_FAILED

启动 GPU 内核的过程返回错误，或之前的内核未成功完成。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配 CPU 内存。

7.2.1.21 mcdnnRNNBackwardWeightsEx()

使用 mcdnnRNNBackwardWeights_v8() 代替 mcdnnRNNBackwardWeightsEX()。

```
mcdnnStatus_t mcdnnRNNBackwardWeightsEx(
    mcdnnHandle_t                   handle,
    const mcdnnRNNDescriptor_t      rnnDesc,
    const mcdnnRNNDataDescriptor_t  xDesc,
    const void                      *x,
    const mcdnnTensorDescriptor_t   hxDesc,
    const void                      *hx,
    const mcdnnRNNDataDescriptor_t  yDesc,
    const void                      *y,
    void                            *workSpace,
    size_t                          workSpaceSizeInBytes,
    const mcdnnFilterDescriptor_t   dwDesc,
    void                            *dw,
    void                            *reserveSpace,
    size_t                          reserveSpaceSizeInBytes)
```

该函数是 mcdnnRNNBackwardWeights() 函数的扩展版本。mcdnnRNNBackwardWeightsEx() 函数允许用户对输入 x 和输出 dw 使用非压缩（填充）布局。

在非压缩布局中，mini-batch 中的每个序列都被视为固定长度，由其对应的 RNNDataDescriptor 中的 maxSeqLength 指定。每个固定长度序列（例如，mini-batch 中的第 n 个序列）都由一个有效段（由其对应的 RNNDataDescriptor 中的 seqLengthArray[n] 指定）和一个填充段组成，使组合序列长度等于 maxSeqLength。

对于非压缩布局，同时支持序列主要（sequence-major）布局（即，时间主要）和批主要（batch-major）布局。为了向后兼容，支持压缩的序列主要布局。但是，与非扩展函数 mcdnnRNNBackwardWeights() 类似，mini-batch 中的序列需要根据长度按降序排序。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

xDesc

输入。已初始化的 RNN 数据描述符。必须与已传入 mcdnnRNNForwardTrainingEx() 的描述符匹配或完全相同。

x

输入。数据指针，指向与 xDesc 数组中张量描述符关联的 GPU 内存。必须包含与已传入 mcdnnRNNForwardTrainingEx() 的完全相同的数据。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。必须与已传入 mcdnnRNNForwardTrainingEx() 的描述符匹配或完全相同。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。必须包含与已传入 mcdnnRNNForwardTrainingEx() 的完全相同的数据，或者如果已将 NULL 传入 mcdnnRNNForwardTrainingEx()，则必须为 NULL。

yDesc

输入。已初始化的 RNN 数据描述符。必须与已传入 mcdnnRNNForwardTrainingEx() 的描述符匹配或完全相同。

y

输入。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。必须包含与 mcdnnRNN-ForwardTrainingEx() 已生成的完全相同的数据。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

dwDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重梯度。

dw

输入/输出。数据指针，指向与卷积核描述符 dwDesc 关联的 GPU 内存。

reserveSpace

输入。数据指针，指向要用作此调用预留空间的 GPU 内存。

reserveSpaceSizeInBytes

输入。指定已提供的 reserveSpace 的大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

此函数不支持已提供的配置。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- xDesc、yDesc、hxDesc、dwDesc 中至少一个描述符无效，或步幅或维度不正确。
- workSpaceSizeInBytes 太小。
- reserveSpaceSizeInBytes 太小。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

7.2.1.22 mcdnnRNNForwardTraining()

使用 mcdnnRNNForward() 代替 mcdnnRNNForwardTraining()。

```
mcdnnStatus_t mcdnnRNNForwardTraining(
    mcdnnHandle_t                    handle,
    const mcdnnRNNDescriptor_t       rnnDesc,
    const int                        seqLength,
    const mcdnnTensorDescriptor_t   *xDesc,
    const void                      *x,
    const mcdnnTensorDescriptor_t    hxDesc,
    const void                      *hx,
    const mcdnnTensorDescriptor_t    cxDesc,
    const void                      *cx,
    const mcdnnFilterDescriptor_t    wDesc,
    const void                      *w,
    const mcdnnTensorDescriptor_t   *yDesc,
    void                            *y,
    const mcdnnTensorDescriptor_t    hyDesc,
    void                            *hy,
    const mcdnnTensorDescriptor_t    cyDesc,
    void                            *cy,
    void                            *workspace,
    size_t                           workSpaceSizeInBytes,
    void                            *reserveSpace,
    size_t                           reserveSpaceSizeInBytes)
```

此函数使用输入 x、hx、cx，权重 w 和输出 y、hy、cy 来执行 rnnDesc 描述的循环神经网络。需要工作空间用于中间存储。reserveSpace 存储训练所需的数据。如果将来在相同的输入数据上执行 mcdnnRNNBackwardData() 和 mcdnnRNNBackwardWeights() 调用，则必须使用相同的 reserveSpace 数据。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rnnDesc**

输入。已初始化的 RNN 描述符。

seqLength

输入。要展开的迭代次数。seqLength 的值不能超过 mcdnnGetRNNWorkspaceSize() 函数中用于查询执行 RNN 所需的工作空间大小。

xDesc

输入。完全压缩的张量描述符的 seqLength 数组。数组中的每个描述符都应具有三个维度，这些维度将输入数据格式描述为一个循环迭代（一个 RNN 时间步一个描述符）。张量的第一维（批大小）可能从迭代元素 n 减少到迭代元素 n+1，但可能不会增加。每个张量描述符必须具有相同的第二维（RNN 输入向量长度 inputSize）。每个张量的第三维需要为 1。输入向量应按列主序排列，因此 xDesc 中的步幅应设置如下：

```
strideA[0]=inputSize, strideA[1]=1, strideA[2]=1
```

x

输入。数据指针，指向与张量描述符 xDesc 的数组关联的 GPU 内存。输入向量压缩会在迭代 n 的最后一个向量结束后直接从迭代（时间步）n+1 的第一个向量开始连续进行。换言之，所有 RNN 时间步的输入向量应填充在 GPU 内存的连续块中，且向量之间没有间隙。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。

cxDesc

输入。完全压缩的的张量描述符，描述 LSTM 网络的初始单元（cell）状态。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

cx

输入。数据指针，指向与张量描述符 cxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态将初始化为零。

wDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

yDesc

输入。完全压缩的张量描述符数组，描述每个循环迭代输出（一个迭代一个描述符）。张量的第二维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第二维应与 hiddenSize 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第二维应是 hiddenSize 参数的 2 倍。

张量 n 的第一维必须与 xDesc 中张量 n 的第一维匹配。

y

输出。数据指针，指向与输出张量描述符 yDesc 关联的 GPU 内存。

hyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终隐藏状态。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

hy

输出。数据指针，指向与张量描述符 hyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终隐藏状态。

cyDesc

输入。完全压缩的张量描述符，描述 LSTM 网络的最终单元状态。张量的第一维取决于初始化 rnnDesc 的方向参数：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

cy

输出。数据指针，指向与张量描述符 cyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终单元状态。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

reserveSpace

输入/输出。数据指针，指向要用作此调用预留空间的 GPU 内存。

reserveSpaceSizeInBytes

输入。指定已提供的 reserveSpace 的大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。

- hxDesc、cxDesc、wDesc、hyDesc、cyDesc 描述符中或者 xDesc、yDesc 描述符中至少有一个是无效的。
- xDesc、hxDesc、cxDesc、wDesc、yDesc、hyDesc、cyDesc 中任一描述符的步幅或维度不正确。
- workSpaceSizeInBytes 太小。
- reserveSpaceSizeInBytes 太小。

MCDNN_STATUS_INVALID_VALUE

在 RNN 描述符中选择 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，在当前函数之前未调用 mcdnnSetPersistentRNNPlan()。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

UDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

7.2.1.23 mcdnnRNNForwardTrainingEx()

使用 mcdnnRNNForward() 代替 mcdnnRNNForwardTrainingEx()。

```
mcdnnStatus_t mcdnnRNNForwardTrainingEx(
    mcdnnHandle_t                       handle,
    const mcdnnRNNDescriptor_t          rnnDesc,
    const mcdnnRNNDataDescriptor_t      xDesc,
    const void                          *x,
    const mcdnnTensorDescriptor_t       hxDesc,
    const void                          *hx,
    const mcdnnTensorDescriptor_t       cxDesc,
    const void                          *cx,
    const mcdnnFilterDescriptor_t       wDesc,
    const void                          *w,
    const mcdnnRNNDataDescriptor_t      yDesc,
    void                                *y,
    const mcdnnTensorDescriptor_t       hyDesc,
    void                                *hy,
    const mcdnnTensorDescriptor_t       cyDesc,
    void                                *cy,
    const mcdnnRNNDataDescriptor_t      kDesc,
    const void                          *keys,
    const mcdnnRNNDataDescriptor_t      cDesc,
    void                                *cAttn,
    const mcdnnRNNDataDescriptor_t      iDesc,
    void                                *iAttn,
    const mcdnnRNNDataDescriptor_t      qDesc,
    void                                *queries,
    void                                *workSpace,
    size_t                              workSpaceSizeInBytes,
    void                                *reserveSpace,
    size_t                              reserveSpaceSizeInBytes);
```

该函数是 mcdnnRNNForwardTraining() 函数的扩展版本。mcdnnRNNForwardTrainingEx() 允许用户对输入 x 和输出 y 使用非压缩（填充）布局。

在非压缩布局中，mini-batch 中的每个序列都被视为固定长度，由其对应的 RNNDataDescriptor 中的 maxSeqLength 指定。每个固定长度序列（例如，mini-batch 中的第 n 个序列）都由一个有效段（由其

对应的 RNNDataDescriptor 中的 seqLengthArray[n] 指定）和一个填充段组成，使组合序列长度等于 maxSeqLength。

对于非压缩布局，同时支持序列主要（sequence-major）布局（即，时间主要）和批主要（batch-major）布局。为了向后兼容，支持压缩的序列主要布局。但是，与非扩展函数 mcdnnRNNForwardTraining() 类似，mini-batch 中的序列需要根据长度按降序排序。

参数

handle

输入。已创建的 mcDNN 上下文的句柄。

rnnDesc

输入。已初始化的 RNN 描述符。

xDesc

输入。已初始化的 RNN 数据描述符。dataType、layout、maxSeqLength、batchSize 和 seqLengthArray 必须与 yDesc 中的匹配。

x

输入。数据指针，指向与 RNN 数据描述符 xDesc 关联的 GPU 内存。输入向量应根据 xDesc 指定的布局在内存中排列。张量中的元素（包括填充向量中的元素）必须紧密压缩，且不支持步幅。

hxDesc

输入。完全压缩的张量描述符，描述 RNN 的初始隐藏状态（initial hidden state）。

张量的第一维取决于初始化 rnnDesc 的方向参数。此外：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 中描述的 batchSize 参数匹配。第三维取决于 RNN 模式是否为 MCDNN_LSTM 以及是否启用了 LSTM 投影。此外：

- 如果 RNN 模式为 MCDNN_LSTM 且启用了 LSTM 投影，则第三维必须与传入 mcdnnSetRNNProjectionLayers() 调用的 recProjSize 参数匹配，其中 mcdnnSetRNNProjectionLayers() 是用于设置 rnnDesc 的。
- 否则，第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。

hx

输入。数据指针，指向与张量描述符 hxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始隐藏状态将初始化为零。

cxDesc

输入。完全压缩的的张量描述符，描述 LSTM 网络的初始单元（cell）状态。

张量的第一维取决于初始化 rnnDesc 的方向参数。此外：

- 如果方向为 MCDNN_UNIDIRECTIONAL，则第一维应与 numLayers 参数匹配。
- 如果方向为 MCDNN_BIDIRECTIONAL，则第一维应是 numLayers 参数的 2 倍。

第二维必须与 xDesc 所描述张量的第一维匹配。第三维必须与初始化 rnnDesc 的 hiddenSize 参数匹配。张量必须为完全压缩格式。

cx

输入。数据指针，指向与张量描述符 cxDesc 关联的 GPU 内存。如果传入 NULL 指针，则网络的初始单元状态将初始化为零。

wDesc

输入。已初始化的卷积核描述符的句柄，描述 RNN 权重。

w

输入。数据指针，指向与卷积核描述符 wDesc 关联的 GPU 内存。

yDesc

输入。已初始化的 RNN 数据描述符。dataType、layout、maxSeqLength、batchSize 和 seqLengthArray 必须与 dyDesc 和 dxDesc 中的相应参数匹配。vectorSize 参数取决于 RNN 模式是否设置为 MCDNN_LSTM，是否启用了 LSTM 投影以及网络是否为双向。具体而言：

- 对于单向网络，如果 RNN 模式为 MCDNN_LSTM 且启用了 LSTM 投影，则 vectorSize 参数必须与传入 mcdnnSetRNNProjectionLayers() 调用的 recProjSize 参数匹配，其中 mcdnnSetRNNProjectionLayers() 是用于设置 rnnDesc 的。如果网络是双向的，则将该值乘以 2。
- 否则，对于单向网络，vectorSize 参数必须与初始化 rnnDesc 的 hiddenSize 参数匹配。如果网络是双向的，则将该值乘以 2。

y

输出。数据指针，指向与 RNN 数据描述符 yDesc 关联的 GPU 内存。输入向量应根据 yDesc 指定的布局在内存中排列。张量中的元素（包括填充向量中的元素）必须紧密压缩，且不支持步幅。

hyDesc

输入。完全压缩的张量描述符，描述 RNN 的最终隐藏状态。此描述符的设置必须与 hxDesc 完全一致。

hy

输出。数据指针，指向与张量描述符 hyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终隐藏状态。

cyDesc

输入。完全压缩的张量描述符，描述 LSTM 网络的最终单元状态。此描述符的设置必须与 cxDesc 完全一致。

cy

输出。数据指针，指向与张量描述符 cyDesc 关联的 GPU 内存。如果传入 NULL 指针，将不会保存网络的最终单元状态。

kDesc

保留。用户可以传入 NULL。

keys

保留。用户可以传入 NULL。

cDesc

保留。用户可以传入 NULL。

cAttn

保留。用户可以传入 NULL。

iDesc

保留。用户可以传入 NULL。

iAttn

保留。用户可以传入 NULL。

qDesc

保留。用户可以传入 NULL。

queries

保留。用户可以传入 NULL。

workspace

输入。数据指针，指向要用作此调用工作空间的 GPU 内存。

workSpaceSizeInBytes

输入。指定已提供的 workspace 大小（以字节为单位）。

reserveSpace

输入/输出。数据指针，指向要用作此调用预留空间的 GPU 内存。

reserveSpaceSizeInBytes

输入。指定已提供的 reserveSpace 的大小（以字节为单位）。

返回值

MCDNN_STATUS_SUCCESS

此函数启用成功。

MCDNN_STATUS_NOT_SUPPORTED

需至少满足以下任一条件：

- 在使用 MCDNN_RNN_ALGO_PERSIST_STATIC 或 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，传入变量序列长度输入。
- 在 pre-Pascal 设备上使用 MCDNN_RNN_ALGO_PERSIST_STATIC 或 MCDNN_RNN_ALGO_PERSIST_DYNAMIC。
- 双输入/输出用于 MCDNN_RNN_ALGO_PERSIST_STATIC。

MCDNN_STATUS_BAD_PARAM

需至少满足以下任一条件：

- rnnDesc 描述符无效。
- xDesc、yDesc、hxDesc、cxDesc、wDesc、hyDesc 或 cyDesc 无效，或者步幅或维度不正确。
- workSpaceSizeInBytes 太小。
- reserveSpaceSizeInBytes 太小。

MCDNN_STATUS_INVALID_VALUE

在 RNN 描述符中选择 MCDNN_RNN_ALGO_PERSIST_DYNAMIC 时，在当前函数之前未调用 mcdnnSetPersistentRNNPlan()。

MCDNN_STATUS_EXECUTION_FAILED

此函数在 GPU 上启用失败。

MCDNN_STATUS_ALLOC_FAILED

此函数不能用来分配内存。

7.2.1.24 mcdnnSetCTCLossDescriptor()

此函数设置 CTC 损失函数描述符。另请参见扩展版本 mcdnnSetCTCLossDescriptorEx() 以设置输入归一化模式。

```
mcdnnStatus_t mcdnnSetCTCLossDescriptor(
    mcdnnCTCLossDescriptor_t        ctcLossDesc,
    mcdnnDataType_t                 compType)
```

当扩展版本 mcdnnSetCTCLossDescriptorEx() 的 normMode 设置为 MCDNN_LOSS_NORMALIZATION_NONE，gradMode 设置为 MCDNN_NOT_PROPAGATE_NAN 时，则它与当前函数 mcdnnSetCTCLossDescriptor() 相同，即：

```
mcdnnSetCtcLossDescriptor(*) = mcdnnSetCtcLossDescriptorEx(*,
↪normMode=MCDNN_LOSS_NORMALIZATION_NONE,
    gradMode=MCDNN_NOT_PROPAGATE_NAN)
```

参数

ctcLossDesc

输出。要设置的 CTC 损失描述符。

compType

输入。此 CTC 损失函数的计算类型。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

MCDNN_STATUS_BAD_PARAM

传入的输入参数至少有一个是无效的。

7.2.1.25 mcdnnSetCTCLossDescriptorEx()

此函数是 mcdnnSetCTCLossDescriptor() 的扩展版本。此函数提供了一个额外的接口 normMode 来设置 CTC 损失函数的输入归一化模式，并提供 gradMode 来控制 NaN 传播类型。

```
mcdnnStatus_t mcdnnSetCTCLossDescriptorEx(
    mcdnnCTCLossDescriptor_t        ctcLossDesc,
    mcdnnDataType_t                 compType,
    mcdnnLossNormalizationMode_t    normMode,
    mcdnnNanPropagation_t           gradMode)
```

当 mcdnnSetCTCLossDescriptorEx() 的 normMode 设置为 MCDNN_LOSS_NORMALIZATION_NONE，gradMode 设置为 MCDNN_NOT_PROPAGATE_NAN 时，则它与 mcdnnSetCTCLossDescriptor() 相同，即：

```
mcdnnSetCtcLossDescriptor(*) = mcdnnSetCtcLossDescriptorEx(*,
↪normMode=MCDNN_LOSS_NORMALIZATION_NONE,
    gradMode=MCDNN_NOT_PROPAGATE_NAN)
```

参数

ctcLossDesc

输出。要设置的 CTC 损失描述符。

compType

输入。此 CTC 损失函数的计算类型。

normMode

输入。此 CTC 损失函数的输入归一化类型。更多信息，参见 mcdnnLossNormalization-Mode_t。

gradMode

输入。此 CTC 损失函数的 NaN 传播类型。对于序列长度 L，序列中重复的字母数 R，序列数据的长度 T，以下情况适用：当梯度计算过程中遇到 L+R > T 的样本时，如果 gradMode 设置为 MCDNN_PROPAGATE_NAN（请参见 mcdnnNanPropagation_t），则 CTC 损失函数不会写入该样本的梯度缓冲区。而是保留当前值，即使不是有限值。如果 gradMode 设置为 MCDNN_NOT_PROPAGATE_NAN，则该样本的梯度设置为零。这保证了梯度为有限值。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

MCDNN_STATUS_BAD_PARAM

传入的输入参数至少有一个是无效的。

7.2.1.26 mcdnnSetCTCLossDescriptor_v8()

许多 CTC API 函数在 mcDNN 中已更新以支持 MXMACA 图形。为此，需要一个新参数 maxLabelLength。假定标签和输入数据在 GPU 内存中，否则此信息不易获取。

```
mcdnnStatus_t mcdnnSetCTCLossDescriptorEx(
    mcdnnCTCLossDescriptor_t        ctcLossDesc,
    mcdnnDataType_t                 compType,
    mcdnnLossNormalizationMode_t    normMode,
    mcdnnNanPropagation_t           gradMode,
    int                             maxLabelLength)
```

参数

ctcLossDesc

输出。要设置的 CTC 损失描述符。

compType

输入。此 CTC 损失函数的计算类型。

normMode

输入。此 CTC 损失函数的输入归一化类型。更多信息，参见 mcdnnLossNormalization-Mode_t。

gradMode

输入。此 CTC 损失函数的 NaN 传播类型。对于序列长度 L，序列中重复的字母数 R，序列数据的长度 T，以下情况适用：当梯度计算过程中遇到 L+R > T 的样本时，如果 gradMode 设置为 MCDNN_PROPAGATE_NAN（请参见 mcdnnNanPropagation_t），则 CTC 损失函数不会写入该样本的梯度缓冲区。而是保留当前值，即使不是有限值。如果 gradMode 设置为 MCDNN_NOT_PROPAGATE_NAN，则该样本的梯度设置为零。这保证了梯度为有限值。

maxLabelLength

输入。标签数据的最大标签长度。

返回值

MCDNN_STATUS_SUCCESS

此函数返回成功。

MCDNN_STATUS_BAD_PARAM

传入的输入参数至少有一个是无效的。

8 mcdnn_backend

本章介绍了 mcDNN 中引入的 mcdnnBackend API 的当前实现行为。用户可以指定计算用例，为其设置执行计划，并通过多个描述符执行计算。具有属性的描述符的典型使用模式由以下 API 调用序列组成：

1. mcdnnBackendCreateDescriptor() 创建指定类型的描述符。
2. mcdnnBackendSetAttribute() 为描述符设置可设置属性的值。必须在下一步之前设置所有必需的属性。
3. mcdnnBackendFinalize() 终结描述符。
4. mcdnnBackendGetAttribute() 从终结的描述符中获取属性值。

枚举类型 mcdnnBackendDescriptorType_t 枚举 mcDNN 后端描述符有效类型的列表。枚举类型 mcdnnBackendAttributeName_t 枚举有效属性的列表。mcdnnBackendDescriptorType_t 中的每个描述符类型都具有 mcdnnBackendAttributeName_t 有效属性值的不相交子集。每个描述符类型及其属性的完整描述，请参见后端描述符类型。

8.1 数据类型参考

以下为 mcDNN 后端 API 中的数据类型参考。

8.1.1 枚举类型

以下为 mcDNN 后端 API 的枚举类型。

8.1.1.1 mcdnnBackendAttributeName_t

mcdnnBackendAttributeName_t 是一种枚举类型，表示可以使用 mcdnnBackendSetAttribute() 和 mcdnnBackendGetAttribute() 函数设置或获取的后端描述符属性。属性的名称前缀标识其所属的后端描述符。

```
typedef enum {
    MCDNN_ATTR_POINTWISE_MODE                   = 0,
    MCDNN_ATTR_POINTWISE_MATH_PREC              = 1,
    MCDNN_ATTR_POINTWISE_NAN_PROPAGATION        = 2,
    MCDNN_ATTR_POINTWISE_RELU_LOWER_CLIP        = 3,
    MCDNN_ATTR_POINTWISE_RELU_UPPER_CLIP        = 4,
    MCDNN_ATTR_POINTWISE_RELU_LOWER_CLIP_SLOPE = 5,
    MCDNN_ATTR_POINTWISE_ELU_ALPHA              = 6,
    MCDNN_ATTR_POINTWISE_SOFTPLUS_BETA          = 7,
    MCDNN_ATTR_POINTWISE_SWISH_BETA             = 8,
    MCDNN_ATTR_POINTWISE_AXIS                   = 9,
```

(下页继续)

(续上页)

```
    MCDNN_ATTR_CONVOLUTION_COMP_TYPE      = 100,
    MCDNN_ATTR_CONVOLUTION_CONV_MODE      = 101,
    MCDNN_ATTR_CONVOLUTION_DILATIONS      = 102,
    MCDNN_ATTR_CONVOLUTION_FILTER_STRIDES = 103,
    MCDNN_ATTR_CONVOLUTION_POST_PADDINGS  = 104,
    MCDNN_ATTR_CONVOLUTION_PRE_PADDINGS   = 105,
    MCDNN_ATTR_CONVOLUTION_SPATIAL_DIMS   = 106,

    MCDNN_ATTR_ENGINEHEUR_MODE            = 200,
    MCDNN_ATTR_ENGINEHEUR_OPERATION_GRAPH = 201,
    MCDNN_ATTR_ENGINEHEUR_RESULTS         = 202,

    MCDNN_ATTR_ENGINECFG_ENGINE            = 300,
    MCDNN_ATTR_ENGINECFG_INTERMEDIATE_INFO = 301,
    MCDNN_ATTR_ENGINECFG_KNOB_CHOICES      = 302,

    MCDNN_ATTR_EXECUTION_PLAN_HANDLE                     = 400,
    MCDNN_ATTR_EXECUTION_PLAN_ENGINE_CONFIG              = 401,
    MCDNN_ATTR_EXECUTION_PLAN_WORKSPACE_SIZE             = 402,
    MCDNN_ATTR_EXECUTION_PLAN_COMPUTED_INTERMEDIATE_UIDS = 403,
    MCDNN_ATTR_EXECUTION_PLAN_RUN_ONLY_INTERMEDIATE_UIDS = 404,

    MCDNN_ATTR_INTERMEDIATE_INFO_UNIQUE_ID           = 500,
    MCDNN_ATTR_INTERMEDIATE_INFO_SIZE                = 501,
    MCDNN_ATTR_INTERMEDIATE_INFO_DEPENDENT_DATA_UIDS  = 502,
    MCDNN_ATTR_INTERMEDIATE_INFO_DEPENDENT_ATTRIBUTES = 503,

    MCDNN_ATTR_KNOB_CHOICE_KNOB_TYPE  = 600,
    MCDNN_ATTR_KNOB_CHOICE_KNOB_VALUE = 601,

    MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_ALPHA         = 700,
    MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_BETA          = 701,
    MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_CONV_DESC     = 702,
    MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_W             = 703,
    MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_X             = 704,
    MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_Y             = 705,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_ALPHA        = 706,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_BETA         = 707,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_CONV_DESC    = 708,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_W            = 709,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_DX           = 710,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_DY           = 711,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_ALPHA      = 712,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_BETA       = 713,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_CONV_DESC  = 714,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_DW         = 715,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_X          = 716,
    MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_DY         = 717,
    MCDNN_ATTR_OPERATION_POINTWISE_PW_DESCRIPTOR           = 750,
    MCDNN_ATTR_OPERATION_POINTWISE_XDESC                   = 751,
    MCDNN_ATTR_OPERATION_POINTWISE_BDESC                   = 752,
    MCDNN_ATTR_OPERATION_POINTWISE_YDESC                   = 753,
    MCDNN_ATTR_OPERATION_POINTWISE_ALPHA1                  = 754,
    MCDNN_ATTR_OPERATION_POINTWISE_ALPHA2                  = 755,
    MCDNN_ATTR_OPERATION_POINTWISE_DXDESC                  = 756,
```

(下页继续)

(续上页)

```
    MCDNN_ATTR_OPERATION_POINTWISE_DYDESC                  = 757,
    MCDNN_ATTR_OPERATION_POINTWISE_TDESC                   = 758,

    MCDNN_ATTR_OPERATION_GENSTATS_MODE                     = 770,
    MCDNN_ATTR_OPERATION_GENSTATS_MATH_PREC                = 771,
    MCDNN_ATTR_OPERATION_GENSTATS_XDESC                    = 772,
    MCDNN_ATTR_OPERATION_GENSTATS_SUMDESC                  = 773,
    MCDNN_ATTR_OPERATION_GENSTATS_SQSUMDESC                = 774,

    MCDNN_ATTR_OPERATION_BN_FINALIZE_STATS_MODE             = 780,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_MATH_PREC              = 781,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_Y_SUM_DESC             = 782,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_Y_SQ_SUM_DESC          = 783,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_SCALE_DESC             = 784,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_BIAS_DESC              = 785,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_PREV_RUNNING_MEAN_DESC = 786,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_PREV_RUNNING_VAR_DESC  = 787,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_UPDATED_RUNNING_MEAN_DESC = 788,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_UPDATED_RUNNING_VAR_DESC  = 789,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_SAVED_MEAN_DESC        = 790,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_SAVED_INV_STD_DESC     = 791,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_EQ_SCALE_DESC          = 792,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_EQ_BIAS_DESC           = 793,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_ACCUM_COUNT_DESC       = 794,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_EPSILON_DESC           = 795,
    MCDNN_ATTR_OPERATION_BN_FINALIZE_EXP_AVERATE_FACTOR_DESC   = 796,

    MCDNN_ATTR_OPERATIONGRAPH_HANDLE           = 800,
    MCDNN_ATTR_OPERATIONGRAPH_OPS              = 801,
    MCDNN_ATTR_OPERATIONGRAPH_ENGINE_GLOBAL_COUNT = 802,

    MCDNN_ATTR_TENSOR_BYTE_ALIGNMENT     = 900,
    MCDNN_ATTR_TENSOR_DATA_TYPE          = 901,
    MCDNN_ATTR_TENSOR_DIMENSIONS         = 902,
    MCDNN_ATTR_TENSOR_STRIDES            = 903,
    MCDNN_ATTR_TENSOR_VECTOR_COUNT       = 904,
    MCDNN_ATTR_TENSOR_VECTORIZED_DIMENSION = 905,
    MCDNN_ATTR_TENSOR_UNIQUE_ID          = 906,
    MCDNN_ATTR_TENSOR_IS_VIRTUAL         = 907,
    MCDNN_ATTR_TENSOR_IS_BY_VALUE        = 908,
    MCDNN_ATTR_TENSOR_REORDERING_MODE    = 909,

    MCDNN_ATTR_VARIANT_PACK_UNIQUE_IDS    = 1000,
    MCDNN_ATTR_VARIANT_PACK_DATA_POINTERS = 1001,
    MCDNN_ATTR_VARIANT_PACK_INTERMEDIATES = 1002,
    MCDNN_ATTR_VARIANT_PACK_WORKSPACE     = 1003,

    MCDNN_ATTR_LAYOUT_INFO_TENSOR_UID = 1100,
    MCDNN_ATTR_LAYOUT_INFO_TYPES      = 1101,

    MCDNN_ATTR_KNOB_INFO_TYPE          = 1200,
    MCDNN_ATTR_KNOB_INFO_MAXIMUM_VALUE = 1201,
    MCDNN_ATTR_KNOB_INFO_MINIMUM_VALUE = 1202,
    MCDNN_ATTR_KNOB_INFO_STRIDE        = 1203,

    MCDNN_ATTR_ENGINE_OPERATION_GRAPH = 1300,
```

(下页继续)

(续上页)

```
    MCDNN_ATTR_ENGINE_GLOBAL_INDEX     = 1301,
    MCDNN_ATTR_ENGINE_KNOB_INFO        = 1302,
    MCDNN_ATTR_ENGINE_NUMERICAL_NOTE   = 1303,
    MCDNN_ATTR_ENGINE_LAYOUT_INFO      = 1304,
    MCDNN_ATTR_ENGINE_BEHAVIOR_NOTE    = 1305,

    MCDNN_ATTR_MATMUL_COMP_TYPE        = 1500,

    MCDNN_ATTR_OPERATION_MATMUL_ADESC                          = 1520,
    MCDNN_ATTR_OPERATION_MATMUL_BDESC                          = 1521,
    MCDNN_ATTR_OPERATION_MATMUL_CDESC                          = 1522,
    MCDNN_ATTR_OPERATION_MATMUL_DESC                           = 1523,
    MCDNN_ATTR_OPERATION_MATMUL_IRREGULARLY_STRIDED_BATCH_COUNT = 1524,
    MCDNN_ATTR_OPERATION_MATMUL_GEMM_M_OVERRIDE_DESC           = 1525,
    MCDNN_ATTR_OPERATION_MATMUL_GEMM_N_OVERRIDE_DESC           = 1526,
    MCDNN_ATTR_OPERATION_MATMUL_GEMM_K_OVERRIDE_DESC           = 1527,


    MCDNN_ATTR_REDUCTION_OPERATOR  = 1600,
    MCDNN_ATTR_REDUCTION_COMP_TYPE = 1601,

    MCDNN_ATTR_OPERATION_REDUCTION_XDESC = 1610,
    MCDNN_ATTR_OPERATION_REDUCTION_YDESC = 1611,
    MCDNN_ATTR_OPERATION_REDUCTION_DESC  = 1612,

    MCDNN_ATTR_OPERATION_BN_BWD_WEIGHTS_MATH_PREC        = 1620,
    MCDNN_ATTR_OPERATION_BN_BWD_WEIGHTS_MEAN_DESC        = 1621,
    MCDNN_ATTR_OPERATION_BN_BWD_WEIGHTS_INVSTD_DESC      = 1622,
    MCDNN_ATTR_OPERATION_BN_BWD_WEIGHTS_BN_SCALE_DESC    = 1623,
    MCDNN_ATTR_OPERATION_BN_BWD_WEIGHTS_X_DESC           = 1624,
    MCDNN_ATTR_OPERATION_BN_BWD_WEIGHTS_DY_DESC          = 1625,
    MCDNN_ATTR_OPERATION_BN_BWD_WEIGHTS_DBN_SCALE_DESC   = 1626,
    MCDNN_ATTR_OPERATION_BN_BWD_WEIGHTS_DBN_BIAS_DESC    = 1627,
    MCDNN_ATTR_OPERATION_BN_BWD_WEIGHTS_EQ_DY_SCALE_DESC = 1628,
    MCDNN_ATTR_OPERATION_BN_BWD_WEIGHTS_EQ_X_SCALE_DESC  = 1629,
    MCDNN_ATTR_OPERATION_BN_BWD_WEIGHTS_EQ_BIAS          = 1630,

    MCDNN_ATTR_RESAMPLE_MODE          = 1700,
    MCDNN_ATTR_RESAMPLE_COMP_TYPE     = 1701,
    MCDNN_ATTR_RESAMPLE_SPATIAL_DIMS  = 1702,
    MCDNN_ATTR_RESAMPLE_POST_PADDINGS = 1703,
    MCDNN_ATTR_RESAMPLE_PRE_PADDINGS  = 1704,
    MCDNN_ATTR_RESAMPLE_STRIDES       = 1705,
    MCDNN_ATTR_RESAMPLE_WINDOW_DIMS   = 1706,
    MCDNN_ATTR_RESAMPLE_NAN_PROPAGATION = 1707,
    MCDNN_ATTR_RESAMPLE_PADDING_MODE  = 1708,

    MCDNN_ATTR_OPERATION_RESAMPLE_FWD_XDESC   = 1710,
    MCDNN_ATTR_OPERATION_RESAMPLE_FWD_YDESC   = 1711,
    MCDNN_ATTR_OPERATION_RESAMPLE_FWD_IDXDESC = 1712,
    MCDNN_ATTR_OPERATION_RESAMPLE_FWD_ALPHA   = 1713,
    MCDNN_ATTR_OPERATION_RESAMPLE_FWD_BETA    = 1714,
    MCDNN_ATTR_OPERATION_RESAMPLE_FWD_DESC    = 1716,

    MCDNN_ATTR_OPERATION_RESAMPLE_BWD_DXDESC  = 1720,
    MCDNN_ATTR_OPERATION_RESAMPLE_BWD_DYDESC  = 1721,
```

(下页继续)

(续上页)

```
MCDNN_ATTR_OPERATION_RESAMPLE_BWD_IDXDESC = 1722,
MCDNN_ATTR_OPERATION_RESAMPLE_BWD_ALPHA   = 1723,
MCDNN_ATTR_OPERATION_RESAMPLE_BWD_BETA    = 1724,
MCDNN_ATTR_OPERATION_RESAMPLE_BWD_DESC    = 1725,
MCDNN_ATTR_OPERATION_RESAMPLE_BWD_XDESC   = 1726,
MCDNN_ATTR_OPERATION_RESAMPLE_BWD_YDESC   = 1727,

MCDNN_ATTR_OPERATION_CONCAT_AXIS          = 1800,
MCDNN_ATTR_OPERATION_CONCAT_INPUT_DESCS   = 1801,
MCDNN_ATTR_OPERATION_CONCAT_INPLACE_INDEX = 1802,
MCDNN_ATTR_OPERATION_CONCAT_OUTPUT_DESC   = 1803,

MCDNN_ATTR_OPERATION_SIGNAL_MODE     = 1900,
MCDNN_ATTR_OPERATION_SIGNAL_FLAGDESC = 1901,
MCDNN_ATTR_OPERATION_SIGNAL_VALUE    = 1902,
MCDNN_ATTR_OPERATION_SIGNAL_XDESC    = 1903,
MCDNN_ATTR_OPERATION_SIGNAL_YDESC    = 1904,

MCDNN_ATTR_OPERATION_NORM_FWD_MODE                     = 2000,
MCDNN_ATTR_OPERATION_NORM_FWD_PHASE                    = 2001,
MCDNN_ATTR_OPERATION_NORM_FWD_XDESC                    = 2002,
MCDNN_ATTR_OPERATION_NORM_FWD_MEAN_DESC                = 2003,
MCDNN_ATTR_OPERATION_NORM_FWD_INV_VARIANCE_DESC        = 2004,
MCDNN_ATTR_OPERATION_NORM_FWD_SCALE_DESC               = 2005,
MCDNN_ATTR_OPERATION_NORM_FWD_BIAS_DESC                = 2006,
MCDNN_ATTR_OPERATION_NORM_FWD_EPSILON_DESC             = 2007,
MCDNN_ATTR_OPERATION_NORM_FWD_EXP_AVG_FACTOR_DESC      = 2008,
MCDNN_ATTR_OPERATION_NORM_FWD_INPUT_RUNNING_MEAN_DESC  = 2009,
MCDNN_ATTR_OPERATION_NORM_FWD_INPUT_RUNNING_VAR_DESC   = 2010,
MCDNN_ATTR_OPERATION_NORM_FWD_OUTPUT_RUNNING_MEAN_DESC = 2011,
MCDNN_ATTR_OPERATION_NORM_FWD_OUTPUT_RUNNING_VAR_DESC  = 2012,
MCDNN_ATTR_OPERATION_NORM_FWD_YDESC                    = 2013,
MCDNN_ATTR_OPERATION_NORM_FWD_PEER_STAT_DESCS          = 2014,

MCDNN_ATTR_OPERATION_NORM_BWD_MODE              = 2100,
MCDNN_ATTR_OPERATION_NORM_BWD_XDESC             = 2101,
MCDNN_ATTR_OPERATION_NORM_BWD_MEAN_DESC         = 2102,
MCDNN_ATTR_OPERATION_NORM_BWD_INV_VARIANCE_DESC = 2103,
MCDNN_ATTR_OPERATION_NORM_BWD_DYDESC            = 2104,
MCDNN_ATTR_OPERATION_NORM_BWD_SCALE_DESC        = 2105,
MCDNN_ATTR_OPERATION_NORM_BWD_EPSILON_DESC      = 2106,
MCDNN_ATTR_OPERATION_NORM_BWD_DSCALE_DESC       = 2107,
MCDNN_ATTR_OPERATION_NORM_BWD_DBIAS_DESC        = 2108,
MCDNN_ATTR_OPERATION_NORM_BWD_DXDESC            = 2109,
MCDNN_ATTR_OPERATION_NORM_BWD_PEER_STAT_DESCS   = 2110,

MCDNN_ATTR_OPERATION_RESHAPE_XDESC = 2200,
MCDNN_ATTR_OPERATION_RESHAPE_YDESC = 2201,

MCDNN_ATTR_RNG_DISTRIBUTION                  = 2300,
MCDNN_ATTR_RNG_NORMAL_DIST_MEAN              = 2301,
MCDNN_ATTR_RNG_NORMAL_DIST_STANDARD_DEVIATION = 2302,
MCDNN_ATTR_RNG_UNIFORM_DIST_MAXIMUM          = 2303,
MCDNN_ATTR_RNG_UNIFORM_DIST_MINIMUM          = 2304,
MCDNN_ATTR_RNG_BERNOULLI_DIST_PROBABILITY    = 2305,
```

(下页继续)

(续上页)

```
    MCDNN_ATTR_OPERATION_RNG_YDESC        = 2310,
    MCDNN_ATTR_OPERATION_RNG_SEED         = 2311,
    MCDNN_ATTR_OPERATION_RNG_DESC         = 2312,
    MCDNN_ATTR_OPERATION_RNG_OFFSET_DESC = 2313,

} mcdnnBackendAttributeName_t;
```

8.1.1.2 mcdnnBackendAttributeType_t

枚举类型 mcdnnBackendAttributeType_t 指定 mcDNN 后端描述符属性的数据类型。它用于指定 mcdnnBackendSetAttribute() 和 mcdnnBackendGetAttribute() 的 void *arrayOfElements 参数所指向的数据类型。

```
typedef enum{
    MCDNN_TYPE_HANDLE = 0,
    MCDNN_TYPE_DATA_TYPE,
    MCDNN_TYPE_BOOLEAN,
    MCDNN_TYPE_INT64,
    MCDNN_TYPE_FLOAT,
    MCDNN_TYPE_DOUBLE,
    MCDNN_TYPE_VOID_PTR,
    MCDNN_TYPE_CONVOLUTION_MODE,
    MCDNN_TYPE_HEUR_MODE,
    MCDNN_TYPE_KNOB_TYPE,
    MCDNN_TYPE_NAN_PROPOGATION,
    MCDNN_TYPE_NUMERICAL_NOTE,
    MCDNN_TYPE_LAYOUT_TYPE,
    MCDNN_TYPE_ATTRIB_NAME,
    MCDNN_TYPE_POINTWISE_MODE,
    MCDNN_TYPE_BACKEND_DESCRIPTOR,
    MCDNN_TYPE_GENSTATS_MODE,
    MCDNN_TYPE_BN_FINALIZE_STATS_MODE,
    MCDNN_TYPE_REDUCTION_OPERATOR_TYPE,
    MCDNN_TYPE_BEHAVIOR_NOTE,
    MCDNN_TYPE_TENSOR_REORDERING_MODE,
    MCDNN_TYPE_RESAMPLE_MODE,
    MCDNN_TYPE_PADDING_MODE,
    MCDNN_TYPE_INT32,
    MCDNN_TYPE_CHAR,
    MCDNN_TYPE_SIGNAL_MODE,
    MCDNN_TYPE_FRACTION,
    MCDNN_TYPE_NORM_MODE,
    MCDNN_TYPE_NORM_FWD_PHASE,
    MCDNN_TYPE_RNG_DISTRIBUTION
} mcdnnBackendAttributeType_t;
```

mcdnnBackendAttributeType_t 的属性类型

mcdnnBackendAttributeType_t	属性类型
MCDNN_TYPE_HANDLE	mcdnnHandle_t
MCDNN_TYPE_DATA_TYPE	mcdnnDataType_t
MCDNN_TYPE_BOOLEAN	bool
MCDNN_TYPE_INT64	int64_t
MCDNN_TYPE_FLOAT	float

下页继续

表 8.1 – 续上页

mcdnnBackendAttributeType_t	属性类型
MCDNN_TYPE_DOUBLE	double
MCDNN_TYPE_VOID_PTR	void *
MCDNN_TYPE_CONVOLUTION_MODE	mcdnnConvolutionMode_t
MCDNN_TYPE_HEUR_MODE	mcdnnBackendHeurMode_t
MCDNN_TYPE_KNOB_TYPE	mcdnnBackendKnobType_t
MCDNN_TYPE_NAN_PROPOGATION	mcdnnNanPropagation_t
MCDNN_TYPE_NUMERICAL_NOTE	mcdnnBackendNumericalNote_t
MCDNN_TYPE_LAYOUT_TYPE	mcdnnBackendLayoutType_t
MCDNN_TYPE_ATTRIB_NAME	mcdnnBackendAttributeName_t
MCDNN_TYPE_POINTWISE_MODE	mcdnnPointwiseMode_t
MC DNN_TYPE_BACKEND_DESCRIPTOR	mcdnnBackendDescriptor_t
MCDNN_TYPE_GENSTATS_MODE	mcdnnGenStatsMode_t
MCD NN_TYPE_BN_FINALIZE_STATS_MODE	mcdnnBnFinalizeStats Mode_t
MCDN N_TYPE_REDUCTION_OPERATOR_TYPE	mcdnnReduc eTensorOp_t
MCDNN_TYPE_BEHAVIOR_NOTE	mcdnnBackendBehaviorNote_t
MCD NN_TYPE_TENSOR_REORDERING_MODE	mcdnnBackendTenso rReordering_t
MCDNN_TYPE_RESAMPLE_MODE	mcdnnResampleMode_t
MCDNN_TYPE_PADDING_MODE	mcdnnPaddingMode_t
MCDNN_TYPE_INT32	int32_t
MCDNN_TYPE_CHAR	char

8.1.1.3 mcdnnBackendBehaviorNote_t

mcdnnBackendBehaviorNote_t 是一种枚举类型，用于表示引擎的可查询行为注释。用户可以使用 mcdnnBackendGetAttribute() 函数从 MCDNN_BACKEND_ENGINE_DESC 查询一系列行为注释。

```
typedef enum{
    MCDNN_BEHAVIOR_NOTE_RUNTIME_COMPILATION = 0,
    MCDNN_BEHAVIOR_NOTE_REQUIRES_FILTER_INT8x32_REORDER = 1,
    MCDNN_BEHAVIOR_NOTE_REQUIRES_BIAS_INT8x32_REORDER = 2,
    MCDNN_BEHAVIOR_NOTE_TYPE_COUNT,
} mcdnnBackendBehaviorNote_t;
```

8.1.1.4 mcdnnBackendDescriptorType_t

mcdnnBackendDescriptorType_t 是一种枚举类型，用于表示描述符。用户可以使用 mcdnnBackend-CreateDescriptor() 函数创建。

```
typedef enum{
    MCDNN_BACKEND_POINTWISE_DESCRIPTOR = 0,
    MCDNN_BACKEND_CONVOLUTION_DESCRIPTOR,
    MCDNN_BACKEND_ENGINE_DESCRIPTOR,
    MCDNN_BACKEND_ENGINECFG_DESCRIPTOR,
    MCDNN_BACKEND_ENGINEHEUR_DESCRIPTOR,
    MCDNN_BACKEND_EXECUTION_PLAN_DESCRIPTOR,
    MCDNN_BACKEND_INTERMEDIATE_INFO_DESCRIPTOR,
    MCDNN_BACKEND_KNOB_CHOICE_DESCRIPTOR,
    MCDNN_BACKEND_KNOB_INFO_DESCRIPTOR,
    MCDNN_BACKEND_LAYOUT_INFO_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_CONVOLUTION_FORWARD_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_CONVOLUTION_BACKWARD_FILTER_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_CONVOLUTION_BACKWARD_DATA_DESCRIPTOR,
```

(下页继续)

(续上页)

```
    MCDNN_BACKEND_OPERATION_POINTWISE_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_GEN_STATS_DESCRIPTOR,
    MCDNN_BACKEND_OPERATIONGRAPH_DESCRIPTOR,
    MCDNN_BACKEND_VARIANT_PACK_DESCRIPTOR,
    MCDNN_BACKEND_TENSOR_DESCRIPTOR,
    MCDNN_BACKEND_MATMUL_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_MATMUL_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_BN_FINALIZE_STATISTICS_DESCRIPTOR,
    MCDNN_BACKEND_REDUCTION_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_REDUCTION_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_BN_BWD_WEIGHTS_DESCRIPTOR,
    MCDNN_BACKEND_RESAMPLE_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_RESAMPLE_FWD_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_RESAMPLE_BWD_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_CONCAT_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_SIGNAL_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_NORM_FORWARD_DESCRIPTOR,
    MCDNN_BACKEND_OPERATION_NORM_BACKWARD_DESCRIPTOR,
} mcdnnBackendDescriptorType_t;
```

8.1.1.5 mcdnnBackendHeurMode_t

mcdnnBackendHeurMode_t是一种枚举类型,用于表示 MCDNN_BACKEND_ENGINEHEUR_DESCRIPTOR 的操作模式。

```
typedef enum{
    MCDNN_HEUR_MODE_INSTANT  = 0,
    MCDNN_HEUR_MODE_B        = 1,
    MCDNN_HEUR_MODE_FALLBACK = 2,
    MCDNN_HEUR_MODE_A        = 3
} mcdnnBackendHeurMode_t;
```

值

MCDNN_HEUR_MODE_A & MCDNN_HEUR_MODE_INSTANT

MCDNN_HEUR_MODE_A 提供与 MCDNN_HEUR_MODE_INSTANT 完全相同的功能。此重命名的目的是更好地匹配 MCDNN_HEUR_MODE_B 的命名。将 MCDNN_HEUR_MODE_INSTANT 的视为已弃用；应使用 MCDNN_HEUR_MODE_A。

MCDNN_HEUR_MODE_A 利用启发式决策树（heuristic decision tree），与 MCDNN_HEUR_MODE_B 相比，它在 CPU 上提供了最佳推理时间。

MCDNN_HEUR_MODE_A 和 MCDNN_HEUR_MODE_INSTANT 支持以下操作节点或操作图：

- ConvolutionFwd
- ConvlutionBwFilter
- ConvolutionBwData
- ConvBNfprop
- ConvBNwgrad
- ConvBiasAct
- ConvScaleBiasAct
- DgradDreluBNBwdWeight
- 运行时融合引擎支持的模式

不支持所有其他操作图。

MCDNN_HEUR_MODE_B

与 MCDNN_HEUR_MODE_INSTANT 相比，可以利用基于启发式的神经网络来提高泛化性能（Generalization Performance）。在使用神经网络的情况下，与 MCDNN_HEUR_MODE_INSTANT 相比，CPU 上的推理时间将增加 10-100 倍。以下情况都不支持这些神经网络启发式:

- 3D 卷积
- 分组卷积（groupCount 大于 1）
- 空洞卷积（任意空间维度的任意扩张大于 1)

此外，只有在曦云系列 GPU 上运行 mcDNN 时，x86 平台才会启用神经网络。如果神经网络不受支持，MCDNN_HEUR_MODE_B 也将回退到 MCDNN_HEUR_MODE_INSTANT。如果 MCDNN_HEUR_MODE_B 的开销将会降低总体网络性能，则 MCDNN_HEUR_MODE_B 将回退到 MCDNN_HEUR_MODE_INSTANT。

MCDNN_HEUR_MODE_FALLBACK

此启发式模式旨在查找可提供功能支持的回退选项（不期望提供最佳 GPU 性能)。

8.1.1.6 mcdnnBackendKnobType_t

mcdnnBackendKnobType_t 是一种枚举类型，用于表示性能旋钮（performance knob）的类型。性能旋钮是引擎的运行时设置，它会影响引擎的性能。用户可以使用 mcdnnBackendGetAttribute() 函数从 MCDNN_BACKEND_ENGINE_DESCRIPTOR 查询一系列性能旋钮及它们的有效值范围。用户可以使用带有 MCDNN_BACKEND_KNOB_CHOICE_DESCRIPTOR 的 mcdnnBackendSetAttribute() 函数对每个旋钮进行设置。

```
typedef enum{
    MCDNN_KNOB_TYPE_SPLIT_K           = 0,
    MCDNN_KNOB_TYPE_SWIZZLE           = 1,
    MCDNN_KNOB_TYPE_TILE_SIZE         = 2,
    MCDNN_KNOB_TYPE_USE_TEX           = 3,
    MCDNN_KNOB_TYPE_EDGE              = 4,
    MCDNN_KNOB_TYPE_KBLOCK            = 5,
    MCDNN_KNOB_TYPE_LDGA              = 6,
    MCDNN_KNOB_TYPE_LDGB              = 7,
    MCDNN_KNOB_TYPE_CHUNK_K           = 8,
    MCDNN_KNOB_TYPE_SPLIT_H           = 9,
    MCDNN_KNOB_TYPE_WINO_TILE         = 10,
    MCDNN_KNOB_TYPE_MULTIPLY          = 11,
    MCDNN_KNOB_TYPE_SPLIT_K_BUF       = 12,
    MCDNN_KNOB_TYPE_TILEK             = 13,
    MCDNN_KNOB_TYPE_STAGES            = 14,
    MCDNN_KNOB_TYPE_REDUCTION_MODE    = 15,
    MCDNN_KNOB_TYPE_CTA_SPLIT_K_MODE  = 16,
    MCDNN_KNOB_TYPE_SPLIT_K_SLC       = 17,
    MCDNN_KNOB_TYPE_IDX_MODE          = 18,
    MCDNN_KNOB_TYPE_SLICED            = 19,
    MCDNN_KNOB_TYPE_SPLIT_RS          = 20,
    MCDNN_KNOB_TYPE_SINGLEBUFFER      = 21,
    MCDNN_KNOB_TYPE_LDGC              = 22,
    MCDNN_KNOB_TYPE_SPECFILT          = 23,
    MCDNN_KNOB_TYPE_KERNEL_CFG        = 24,
    MCDNN_KNOB_TYPE_WORKSPACE         = 25,
```

(下页继续)

(续上页)

```
    MCDNN_KNOB_TYPE_COUNTS = 26,
} mcdnnBackendKnobType_t;
```

8.1.1.7 mcdnnBackendLayoutType_t

mcdnnBackendLayoutType_t 是一种枚举类型，用于表示引擎的可查询布局要求。用户可以使用 mcdnnBackendGetAttribute() 函数从 MCDNN_BACKEND_ENGINE_DESC 描述符中查询布局要求。

```
typedef enum{
    MCDNN_LAYOUT_TYPE_PREFERRED_NCHW    = 0,
    MCDNN_LAYOUT_TYPE_PREFERRED_NHWC    = 1,
    MCDNN_LAYOUT_TYPE_PREFERRED_PAD4CK = 2,
    MCDNN_LAYOUT_TYPE_PREFERRED_PAD8CK = 3,
    MCDNN_LAYOUT_TYPE_COUNT             = 4,
} mcdnnBackendLayoutType_t;
```

8.1.1.8 mcdnnBackendNormFwdPhase_t

mcdnnBackendNormFwdPhase_t 是一种枚举类型，用于区分归一化前向操作的推理和训练阶段。

```
typedef enum{
    MCDNN_NORM_FWD_INFERENCE = 0,
    MCDNN_NORM_FWD_TRAINING  = 1,
} mcdnnBackendNormFwdPhase_t;
```

8.1.1.9 mcdnnBackendNormMode_t

mcdnnBackendNormMode_t 是一种枚举类型，用于表示后端归一化前向和反向操作中的归一化模式。

以下供参考：

- 层归一化（Layer Normalization，LN）的定义，可参见 Layer Normalization。
- 实例归一化（Instance Normalization，IN）的定义，可参见 Instance Normalization: The Missing Components for Fast Stylization。
- 批量归一化的定义，可参见 Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift。

目前暂不支持 MCDNN_GROUP_NORM。如果尝试使用它，mcDNN 将返回 MCDNN_STATUS_INTERNAL_ERROR 错误。

```
typedef enum{
    MCDNN_LAYER_NORM    = 0,
    MCDNN_INSTANCE_NORM = 1,
    MCDNN_BATCH_NORM    = 2,
    MCDNN_GROUP_NORM    = 3,
} mcdnnBackendNormMode_t;
```

8.1.1.10 mcdnnBackendNumericalNote_t

mcdnnBackendNumericalNot_t 是一种枚举类型，用于表示引擎的可查询数值属性。用户可以使用 mcdnnBackendGetAttribute() 函数从 MCDNN_BACKEND_ENGINE_DESC 查询一系列数值注释。

```
typedef enum{
    MCDNN_NUMERICAL_NOTE_TENSOR_CORE = 0,
    MCDNN_NUMERICAL_NOTE_DOWN_CONVERT_INPUTS,
    MCDNN_NUMERICAL_NOTE_REDUCED_PRECISION_REDUCTION,
    MCDNN_NUMERICAL_NOTE_FFT,
    MCDNN_NUMERICAL_NOTE_NONDETERMINISTIC,
    MCDNN_NUMERICAL_NOTE_WINOGRAD,
    MCDNN_NUMERICAL_NOTE_WINOGRAD_TILE_4x4,
    MCDNN_NUMERICAL_NOTE_WINOGRAD_TILE_6x6,
    MCDNN_NUMERICAL_NOTE_WINOGRAD_TILE_13x13,
    MCDNN_NUMERICAL_NOTE_TYPE_COUNT,
} mcdnnBackendNumericalNote_t;
```

8.1.1.11 mcdnnBackendTensorReordering_t

mcdnnBackendTensorReordering_t 是一种枚举类型，用来表示张量描述符的张量重排序属性。用户可以通过 mcdnnBackendSetAttribute() 和 mcdnnBackendGetAttribute() 函数在 MCDNN_BACKEND_TENSOR_DESCRIPTOR 中获取和设置该属性。

```
typedef enum{
    MCDNN_TENSOR_REORDERING_NONE    = 0,
    MCDNN_TENSOR_REORDERING_INT8x32 = 1,
    MCDNN_TENSOR_REORDERING_F16x16  = 2,
} mcdnnBackendTensorReordering_t;
```

8.1.1.12 mcdnnBnFinalizeStatsMode_t

mcdnnBnFinalizeStatsMode_t 是一种枚举类型，用于暴露不同的数学运算模式，这些模式是将 BN 统计数据和训练的 scale 和 bias 转换为等效 scale 和 bias，以便在下一个归一阶段应用于推理和训练用例。

```
typedef enum{
    MCDNN_BN_FINALIZE_STATISTICS_TRAINING  = 0,
    MCDNN_BN_FINALIZE_STATISTICS_INFERENCE = 1,
} mcdnnBnFinalizeStatsMode_t;
```

mcdnnBnFinalizeStatsMode_t 的 BN 统计

BN 统计模式	说明
MCDNN_BN_FINALIZE_STATISTICS_TRAINING	从 ySum、ySqSum 和已学习的 scale、bias 计算等效 scale 和 bias (可选) 更新移动统计数据并生成保存的统计数据，以实现与 mcdnnBatchNormaliza tionBackward()，mcdnnBatchNormalization BackwardEx() 或 mcdnnNormalization Backward() 的互操作性
MCDNN_BN_FINALIZE_STATISTICS_INFERENCE	从已学习的移动统计数据和已学习 scale、bias，计算等效 scale 和 bias

8.1.1.13 mcdnnFraction_t

mcdnnFraction_t 是允许用户定义 int64_t 的结构。

```
typedef enum{
    int64_t numerator;
    int64_t denominator;
} mcdnnFraction_t;
```

8.1.1.14 mcdnnGenStatsMode_t

mcdnnGenStatsMode_t 是一种枚举类型，用于表示后端统计信息生成操作中的统计信息模式。

值

MCDNN_GENSTATS_SUM_SQSUM

在这种模式下，会计算并写出输入张量沿指定维度的和与平方和。当前每个通道对归约维度的支持受限，但可根据请求添加额外支持。

8.1.1.15 mcdnnPaddingMode_t

mcdnnPaddingMode_t 是一种枚举类型，用于表示后端重采样操作中的填充模式。

```
typedef enum{
    MCDNN_ZERO_PAD      = 0,
    MCDNN_NEG_INF_PAD   = 1,
    MCDNN_EDGE_VAL_PAD  = 2,
} mcdnnPaddingMode_t ;
```

8.1.1.16 mcdnnPointwiseMode_t

mcdnnPointwiseMode_t 是一种枚举类型，用于表示后端逐点运算（pointwise operation）描述符中描述的逐点数学运算。

值 MCDNN_POINTWISE_ADD

在此模式下，计算两个张量之间的逐点相加（pointwise addition）。

MCDNN_POINTWISE_ADD_SQUARE

在此模式下，计算第一张量与第二张量平方的逐点相加。

MCDNN_POINTWISE_DIV

在此模式下，计算第一张量被第二张量逐点真除（true division）。

MCDNN_POINTWISE_MAX

在此模式下，获取两个张量之间的逐点最大值（pointwise maximum）。

MCDNN_POINTWISE_MIN

在此模式下，获取两个张量之间的逐点最小值（pointwise minimum）。

MCDNN_POINTWISE_MOD

在此模式下，计算第一张量除以第二张量后的逐点浮点余数。

MCDNN_POINTWISE_MUL

在此模式下，计算两个张量之间的逐点相乘（pointwise multiplication）。

MCDNN_POINTWISE_POW

在此模式下，计算从第一张量到第二张量的幂的逐点值。

MCDNN_POINTWISE_SUB

在此模式下，计算两个张量之间的逐点相减（pointwise subtraction）。

MCDNN_POINTWISE_ABS

在此模式下，计算输入张量的逐点绝对值。

MCDNN_POINTWISE_CEIL

在此模式下，计算输入张量的逐点 ceiling。

MCDNN_POINTWISE_COS

在此模式下，计算输入张量的逐点三角余弦（cosine）。

MCDNN_POINTWISE_EXP

在此模式下，计算输入张量的逐点指数。

MCDNN_POINTWISE_FLOOR

在此模式下，计算输入张量的逐点 floor。

MCDNN_POINTWISE_LOG

在此模式下，计算输入张量的逐点自然对数。

MCDNN_POINTWISE_NEG

在此模式下，计算输入张量的逐点负值。

MCDNN_POINTWISE_RSQRT

在此模式下，计算输入张量平方根的逐点倒数。

MCDNN_POINTWISE_SIN

在此模式下，计算输入张量的逐点三角正弦（sine）。

MCDNN_POINTWISE_SQRT

在此模式下，计算输入张量的逐点平方根。

MCDNN_POINTWISE_TAN

在此模式下，计算输入张量的逐点三角正切（tangent）。

MCDNN_POINTWISE_ERF

在此模式下，计算逐点 Error 函数。

MCDNN_POINTWISE_IDENTITY

在此模式下，不执行任何计算。与其他逐点模式一样，此模式通过将输入张量的数据类型指定为一种类型，将输出张量的数据类型指定为另一种类型，来提供隐式转换。

MCDNN_POINTWISE_RELU_FWD

在此模式下，计算输入张量的逐点 ReLU 激活函数。

MCDNN_POINTWISE_TANH_FWD

在此模式下，计算输入张量的逐点 tanh 激活函数。

MCDNN_POINTWISE_SIGMOID_FWD

在此模式下，计算输入张量的逐点 sigmoid 激活函数。

MCDNN_POINTWISE_ELU_FWD

在此模式下，计算输入张量的逐点指数线性（Exponential Linear Unit，ELU）激活函数。

MCDNN_POINTWISE_GELU_FWD

在此模式下，计算输入张量的逐点高斯误差线性（Gaussian Error Linear Unit，GELU）激活函数。

MCDNN_POINTWISE_SOFTPLUS_FWD

在此模式下，计算输入张量的逐点 softplus 激活函数。

MCDNN_POINTWISE_SWISH_FWD

在此模式下，计算输入张量的逐点 swish 激活函数。

MCDNN_POINTWISE_GELU_APPROX_TANH_FWD

在此模式下，计算输入张量 GELU 激活函数的逐点 tanh 逼近（pointwise tanh approximation）。有关详细信息，请参见 GAUSSIAN ERROR LINEAR UNIT (GELUS) 。

MCDNN_POINTWISE_RELU_BWD

在此模式下，计算输入张量 ReLU 激活函数的逐点一阶导数。

MCDNN_POINTWISE_TANH_BWD

在此模式下，计算输入张量 tanh 激活函数的逐点一阶导数。

MCDNN_POINTWISE_SIGMOID_BWD

在此模式下，计算输入张量 sigmoid 激活函数的逐点一阶导数。

MCDNN_POINTWISE_ELU_BWD

在此模式下，计算输入张量 ELU 激活函数的逐点一阶导数。

MCDNN_POINTWISE_GELU_BWD

在此模式下，计算输入张量 GELU 激活函数的逐点一阶导数。

MCDNN_POINTWISE_SOFTPLUS_BWD

在此模式下，计算输入张量 softplus 激活函数的逐点一阶导数。

MCDNN_POINTWISE_SWISH_BWD

在此模式下，计算输入张量 swish 激活函数的逐点一阶导数。

MCDNN_POINTWISE_GELU_APPROX_TANH_BWD

在此模式下，计算输入张量 GELU 激活函数的 tanh 逼近的逐点一阶导数。

MCDNN_POINTWISE_CMP_EQ

在此模式下，计算第一个张量（与第二个张量相等）的逐点真假值（truth value）。

MCDNN_POINTWISE_CMP_NEQ

在此模式下，计算第一个张量（与第二个张量不相等）的逐点真假值。

MCDNN_POINTWISE_CMP_GT

在此模式下，计算第一个张量（大于第二个张量）的逐点真假值。

MCDNN_POINTWISE_CMP_GE

在此模式下，计算第一个张量（大于等于第二个张量）的逐点真假值。

MCDNN_POINTWISE_CMP_LT

在此模式下，计算第一个张量（小于第二个张量）的逐点真假值。

MCDNN_POINTWISE_CMP_LE

在此模式下，计算第一个张量（小于等于第二个张量）的逐点真假值。

MCDNN_POINTWISE_LOGICAL_AND

在此模式下，计算第一个张量 logical AND 第二个张量的逐点真假值。

MCDNN_POINTWISE_LOGICAL_OR

在此模式下，计算第一个张量 logical OR 第二个张量的逐点真假值。

MCDNN_POINTWISE_LOGICAL_NOT

在此模式下，计算输入张量 logical NOT 的逐点真假值。

MCDNN_POINTWISE_GEN_INDEX

在此模式中，输入张量的逐点索引值沿给定轴生成。

MCDNN_POINTWISE_BINARY_SELECT

在此模式下，根据给定的谓词（predicate）张量在两个输入张量中选择逐点值。

8.1.1.17 mcdnnResampleMode_t

mcdnnResampleMode_t 是一种枚举类型，用于表示后端重采样操作中的重采样模式。

```
typedef enum{
    MCDNN_RESAMPLE_NEAREST                  = 0,
    MCDNN_RESAMPLE_BILINEAR                 = 1,
    MCDNN_RESAMPLE_AVGPOOL                  = 2,
    MCDNN_RESAMPLE_AVGPOOL_INCLUDE_PADDING = 2,
    MCDNN_RESAMPLE_AVGPOOL_EXCLUDE_PADDING = 4,
    MCDNN_RESAMPLE_MAXPOOL                  = 3,
} mcdnnResampleMode_t;
```

8.1.1.18 mcdnnRngDistribution_t

mcdnnRngDistribution_t 是一种枚举类型，用于表示要在后端 Rng（随机数生成器）操作中使用的分布。

```
typedef enum{
    MCDNN_RNG_DISTRIBUTION_BERNOULLI,
    MCDNN_RNG_DISTRIBUTION_UNIFORM,
    MCDNN_RNG_DISTRIBUTION_NORMAL,
} mcdnnRngDistribution_t;
```

值

MCDNN_RNG_DISTRIBUTION_BERNOULLI

在此模式下,伯努利(Bernoulli)分布用于生成随机数。MCDNN_ATTR_RNG_BERNOULLI_DIST_PROBABILITY 属性可用于指定生成 1 的概率。

MCDNN_RNG_DISTRIBUTION_UNIFORM

在此模式下，正态分布用于生成随机数。

MMCDNN_ATTR_RNG_NORMAL_DIST_MEAN 和 MCDNN_ATTR_RNG_NORMAL_DIST_STANDARD_DEVIATION 属性可用于指定随机数生成器的均值和标准偏差。

8.1.1.19 mcdnnSignalMode_t

mcdnnSignalMode_t 是一种枚举类型，用于表示后端信号操作中的信令模式。

```
typedef enum{
    MCDNN_SIGNAL_SET  = 0,
    MCDNN_SIGNAL_WAIT = 1,
} mcdnnSignalMode_t;
```

值

MCDNN_SIGNAL_SET

在此模式下，flag 变量使用提供的信号值以原子方式进行更新。

MCDNN_SIGNAL_WAIT

在此模式下，操作将阻塞，直到 flag 变量与提供的信号值相等。

8.1.1.20 mcdnnBackendDescriptor_t

mcdnnBackendDescriptor_t 是一个 typedef void 指针，指向一个不透明描述符结构。它所指向的结构类型由 mcdnnBackendCreateDescriptor() 为不透明结构分配内存时使用的参数决定。

描述符的属性可以使用 mcdnnBackendSetAttribute() 进行设置。设置完描述符的所有必需属性后，可以使用 mcdnnBackendFinalize() 终结该描述符。从终结的描述符中，可以使用 mcdnnBackendGetAttribute() 查询其可查询的属性。最后，可以使用 mcdnnBackendDestroyDescriptor() 释放为描述符分配的内存。

8.2 API 参考

8.2.1 API 函数

以下为 mcDNN 后端 API 函数。

8.2.1.1 mcdnnBackendCreateDescriptor()

此函数在描述符中为给定的描述符类型分配内存，并分配在描述符所指向的位置。

```
mcdnnStatus_t mcdnnBackendCreateDescriptor(mcdnnBackendDescriptorType_t
↪descriptorType,
                                           mcdnnBackendDescriptor_t
↪*descriptor)
```

注解: mcdnnBackendDescriptor_t 是指向 void * 的指针。

参数

descriptorType

输入。枚举的 mcdnnBackendDescriptorType_t 之一。

descriptor

输入。指针，指向要创建的 mcdnnBackendDescriptor_t 实例。

返回值

MCDNN_STATUS_SUCCESS

创建成功。

MCDNN_STATUS_NOT_SUPPORTED

不支持给定类型的描述符创建。

MCDNN_STATUS_ALLOC_FAILED

内存分配失败。

其他返回值取决于使用的参数，如 mcDNN 后端 API 中所述。

8.2.1.2 mcdnnBackendDestroyDescriptor()

此函数用于销毁已使用 mcdnnBackendCreateDescriptor() 创建的 mcdnnBackendDescriptor_t 实例。

```
mcdnnStatus_t mcdnnBackendDestroyDescriptor(mcdnnBackendDescriptor_t
↪descriptor)
```

参数

descriptor

输入。已由 mcdnnBackendCreateDescriptor() 创建的 mcdnnBackendDescriptor_t 实例。

返回值

MCDNN_STATUS_SUCCESS

内存销毁成功。

MCDNN_STATUS_ALLOC_FAILED

内存销毁失败。

未定义行为

描述符在 Create 和 Destroy 描述符之间被更改。

未定义

在内存释放完成后，描述符所指向的值将为未定义值。

其他返回值取决于使用的参数，如 mcDNN 后端 API 中所述。

8.2.1.3 mcdnnBackendExecute()

此函数用于执行 VariantPack 上给定的引擎配置计划（Engine Configuration Plan），和数据上的最终 ExecutionPlan。数据和工作空间封装在 VariantPack 中。

```
mcdnnStatus_ mcdnnBackendExecute(mcdnnHandle_t handle,
                                 mcdnnBackendDescriptor_t executionPlan,
                                 mcdnnBackendDescriptor_t varianPack)
```

参数

executionPlan

输入。指向要销毁的 mcDNN 句柄的指针。

variantPack

输入。指向终结的 VariantPack 的指针，包括：

- 执行计划中所设置操作的每个非虚拟指针的数据指针。
- 指向全局内存中用户分配的工作空间的指针，其大小至少与从 MCDNN_BACKEND_. 查询到的相同。

返回值

MCDNN_STATUS_SUCCESS

ExecutionPlan 已成功执行。

MCDNN_STATUS_BAD_PARAM

遇到不正确或不一致的值。例如：

- 所需的数据指针无效。

MCDNN_STATUS_INTERNAL_ERROR

遇到一些内部错误。

MCDNN_STATUS_EXECUTION_FAILED

使用 VariantPack 执行计划时遇到错误。

其他返回值取决于使用的参数，如 mcDNN 后端 API 中所述。

8.2.1.4 mcdnnBackendFinalize()

此函数终结描述符所指向的内存。终结的类型取决于 descriptorType 参数，具有该参数的描述符是使用 mcdnnBackendCreateDescriptor() 创建的，或使用 mcdnnBackendInitialize() 初始化的。

```
mcdnnStatus_t mcdnnbBackendFinalize(mcdnnBackendDescriptor descriptor)
```

mcdnnBackendFinalize() 还会检查在创建/初始化和终结阶段之间设置的所有属性。如果成功，mcdnnBackendFinalize() 将返回 MCDNN_STATUS_SUCCESS，并且描述符的终结状态设置为 true。在此状态下，不允许使用 mcdnnBackendSetAttribute() 设置属性。仅当描述符的终结状态为 true 时，才允许使用 mcdnnBackendGetAttribute() 获取属性。

参数

descriptor

输入。要终结的 mcdnnBackendDescriptor_t 实例。

返回值

MCDNN_STATUS_SUCCESS

描述符已成功终结。

MCDNN_STATUS_BAD_PARAM

遇到无效的描述符属性值或其组合。

MCDNN_STATUS_NOT_SUPPORTED

遇到当前版本 mcDNN 不支持的描述符属性值或其组合。

MCDNN_STATUS_INTERNAL_ERROR

遇到一些内部错误。

其他返回值取决于使用的参数，如 mcDNN 后端 API 中所述。

8.2.1.5 mcdnnBackendGetAttribute()

此函数用于检索描述符属性的值。attributeName 是请求的属性名。attributeType 是属性类型。requestsedElementCount 是要检索的元素数。请求的属性的元素数存储在 elementCount 中。检索到的值存储在 arrayOfElements 中。当属性具有单个值时，arrayOfElements 可以是指向输出值的指针。如果描述符已成功终结，则此函数将返回 MCDNN_STATUS_NOT_INTIALIZED。

```
mcdnnStatus_t mcdnnBackendGetAttribute(
    mcdnnBackendDescriptor_t descriptor,
    mcdnnBackendAttributeName_t attributeName,
    mcdnnBackendAttributeType_t attributeType,
    int64_t requestedElementCount,
    int64_t *elementCount,
    void *arrayOfElements);
```

参数

descriptor

输入。用户要检索其属性的 mcdnnBackendDescriptor_t 实例。

attributeName

输入。从描述符中获取的属性名称。

attributeType

输入。属性类型。

requestedElementCount

输入。要输出到 arrayOfElements 的元素数。

elementCount

输入。描述符属性具有的元素数的输出指针。请注意，mcdnnBackendGetAttribute() 只会将最少的 elementCount 和 requestedElementCount 元素写入 arrayOfElements。

arrayOfElements

输入。attributeType 数据类型的元素数组。attributeType 的数据类型列在 mcdnnBackendAttributeType_t 映射表中。

返回值

MCDNN_STATUS_SUCCESS

attributeName 已成功提供给描述符。

MCDNN_STATUS_BAD_PARAM

遇到一个或多个无效或不一致的参数值。例如：

- attributeName 不是描述符的有效属性。
- attributeType 不是属性的有效类型之一。

MCDNN_STATUS_NOT_INITIALIZED

未使用 mcdnnBackendFinalize() 成功终结描述符。

其他返回值取决于使用的参数，如 mcDNN 后端 API 中所述。

8.2.1.6 mcdnnBackendInitialize()

此函数将大小为 sizeInByte 的描述符所指向的预分配内存重新用于类型为 descriptorType 的后端描述符。描述符的终结状态设置为 false。

```
mcdnnStatus_t mcdnnBackendInitialize(
    mcdnnBackendDescriptor_t descriptor,
    mcdnnBackendDescriptorType_t descriptorType,
    size_t sizeInBytes)
```

参数

descriptor

输入。要初始化的 mcdnnBackendDescriptor_t 实例。

descriptorType

输入。mcDNN 后端描述符类型的枚举值。

sizeInBytes

输入。描述符指向的内存大小。

返回值

MCDNN_STATUS_SUCCESS

内存初始化成功。

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的参数值。例如：

- 描述符是一个 nullptr
- sizeInBytes 小于描述符类型所需的大小

其他返回值取决于使用的参数，如 mcDNN 后端 API 中所述。

8.2.1.7 mcdnnBackendSetAttribute()

此函数将描述符的属性值设为作为指针时提供的值。descriptor 是要设置的描述符。attributeName 是要设置的属性名称。attributeType 是属性的类型。arrayOfElements 指向属性要设置的值。元素数由 elementCount 给定。如果描述符已使用 mcdnnBackendFinalize() 成功终结，则此函数将返回 MCDNN_STATUS_NOT_INTIALIZED。

```
mcdnnStatus_t mcdnnBackendSetAttribute(
    mcdnnBackendDescriptor_t descriptor,
    mcdnnBackendAttributeName_t attributeName,
    mcdnnBackendAttributeType_t attributeType,
    int64_t elementCount,
    void *arrayOfElements);
```

参数

descriptor

输入。正在设置其属性的 mcdnnBackendDescriptor_t 实例。

attributeName

输入。在描述符上所设置属性的名称。

attributeType

输入。属性类型。

elementCount

输入。设置的元素数。

arrayOfElements

输入。数组的起始位置，从中读取值。数组元素的数据类型应是 attributeType。attributeType 的数据类型列在 mcdnnBackendAttributeType_t 映射表中。

返回值

MCDNN_STATUS_SUCCESS

描述符中已设置 attributeName。

MCDNN_STATUS_NOT_INITIALIZED

描述符所指向的后端描述符已处于终结状态。

MCDNN_STATUS_BAD_PARAM

调用该函数所使用参数的值无效。可能的原因包括：

- attributeName 不是描述符的可设置属性
- 此 attributeName 的 attributeType 不正确。

- elementCount 值无效。
- arrayOfElements 包含对 attributeType 无效的值。

MCDNN_STATUS_NOT_SUPPORTED

当前版本的 mcDNN 不支持设置的属性值。

其他返回值取决于使用的参数，如 mcDNN 后端 API 中所述。

8.2.2 后端描述符类型

此部分枚举各种描述符的所有有效属性。

8.2.2.1 MCDNN_BACKEND_CONVOLUTION_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_CONVOLUTION_DESCRIPTOR,&desc)创建的 mcDNN 后端卷积描述符，为前向和反向传播的卷积运算符指定参数：计算数据类型、卷积模式、卷积核扩张和步幅、填充。

属性

mcDNN 后端卷积描述符的属性是 mcdnnBackendAttributeName_t 枚举类型的值（前缀为 MCDNN_ATTR_CONVOLUTION_）：

MCDNN_ATTR_CONVOLUTION_COMP_TYPE

卷积运算符的计算类型。

- MCDNN_TYPE_DATA_TYPE；一个元素。
- 必要属性

MCDNN_ATTR_CONVOLUTION_CONV_MODE

卷积或交叉相关模式。

- MCDNN_TYPE_CONVOLUTION_MODE；一个元素。
- 必要属性

MCDNN_ATTR_CONVOLUTION_DILATIONS

卷积核扩张。

- MCDNN_TYPE_INT64；一个或多个，但最多为 MCDNN_MAX_DIMS 个元素。
- 必要属性

MCDNN_ATTR_CONVOLUTION_FILTER_STRIDES

卷积核步幅。

- MCDNN_TYPE_INT64；一个或多个，但最多为 MCDNN_MAX_DIMS 个元素。
- 必要属性

MCDNN_ATTR_CONVOLUTION_PRE_PADDINGS

在每个空间维度起始处填充。

- MCDNN_TYPE_INT64；一个或多个，但最多为 MCDNN_MAX_DIMS 个元素。
- 必要属性

MCDNN_ATTR_CONVOLUTION_POST_PADDINGS

在每个空间维度结尾处填充。

- MCDNN_TYPE_INT64；一个或多个，但最多为 MCDNN_MAX_DIMS 个元素。

- 必要属性

MCDNN_ATTR_CONVOLUTION_SPATIAL_DIMS

卷积中的空间维数。

- MCDNN_TYPE_INT64，一个元素。
- 必要属性

终结

带有 MCDNN_BACKEND_CONVOLUTION_DESCRIPTOR 的 mcdnnBackendFinalize() 可以具有以下返回值：

MCDNN_STATUS_BAD_PARAM

用于设置 `MCDNN_ATTR_CONVOLUTION_DILATIONS`、`MCDNN_ATTR_CONVOLUTION_FILTER_STRIDES`、`MCDNN_ATTR_CONVOLUTION_PRE_PADDINGS` 和 `MCDNN_ATTR_CONVOLUTION_POST_PADDINGS` 的 elementCount 参数，与 `MCDNN_ATTR_CONVOLUTION_SPATIAL_DIMS` 的设置值不相等。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.2 MCDNN_BACKEND_ENGINE_DESCRIPTOR

使用描述符类型值 MCDNN_BACKEND_ENGINE_DESCRIPTOR 创建的 mcDNN 后端引擎描述符，描述用于计算操作图的引擎。引擎是一组具有类似计算和数值属性的内核。

属性

mcDNN 后端卷积描述符的属性是 mcdnnBackendAttributeName_t 枚举类型的值（前缀为 MCDNN_ATTR_ENGINE_）：

MCDNN_ATTR_ENGINE_OPERATION_GRAPH

要计算的操作图。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_OPERATIONGRAPH_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_ENGINE_GLOBAL_INDEX

引擎的索引。

- MCDNN_TYPE_INT64；一个元素。
- 有效值介于 0 和 MCDNN_ATTR_OPERATIONGRAPH_ENGINE_GLOBAL_COUNT - 1 之间
- 必要属性

MCDNN_ATTR_ENGINE_KNOB_INFO

引擎性能旋钮的描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_KNOB_INFO_DESCRIPTOR 的一个元素。
- 只读属性。

MCDNN_ATTR_ENGINE_NUMERICAL_NOTE

引擎的数值属性。

- MCDNN_TYPE_NUMERICAL_NOTE; zero or more elements.
- 只读属性。

MCDNN_ATTR_ENGINE_LAYOUT_INFO

引擎的首选张量布局。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_LAYOUT_INFO_DESCRIPTOR 的一个元素。
- 只读属性。

终结

MCDNN_STATUS_SUCCESS

描述符已成功终结。

MCDNN_STATUS_NOT_SUPPORTED

当前版本的 mcDNN 不支持设置的属性值。例如:

- MCDNN_ATTR_ENGINE_GLOBAL_INDEX 的值不在有效范围内。

MCDNN_STATUS_BAD_PARAM

描述符属性集不一致或无效。例如:

- 操作图描述符集尚未终结。

8.2.2.3 MCDNN_BACKEND_ENGINECFG_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_ENGINECFG_DESCRIPTOR, &desc) 创建的 mcDNN 后端引擎配置描述符，由引擎描述符和一组旋钮选择描述符组成。用户可以从引擎配置中查询中间信息：可在执行之间重复使用计算中间结果。

属性

MCDNN_ATTR_ENGINECFG_ENGINE

后端引擎。

- MCDNN_TYPE_BACKEND_DESCRIPTOR：一个元素，类型 MCDNN_BACKEND_ENGINE_DESCRIPTOR 的后端描述符。
- 必要属性

MCDNN_ATTR_ENGINECFG_KNOB_CHOICES

引擎调优旋钮和选择。

- MCDNN_TYPE_BACKEND_DESCRIPTOR：零个或多个元素，类型为 MCDNN_BACKEND_KNOB_CHOICE_DESCRIPTOR 的后端描述符。

MCDNN_ATTR_ENGINECFG_INTERMEDIATE_INFO

此引擎配置的计算中间信息。

- MCDNN_TYPE_BACKEND_DESCRIPTOR：一个元素，类型 MCDNN_BACKEND_INTERMEDIATE_INFO_DESCRIPTOR 的后端描述符。
- 只读属性。
- 当前不支持。用于将来实现的占位符。

终结

MCDNN_STATUS_SUCCESS

描述符已成功终结。

MCDNN_STATUS_NOT_SUPPORTED

当前版本的 mcDNN 不支持设置的属性值。例如:

- 值旋钮。

8.2.2.4 MCDNN_BACKEND_ENGINEHEUR_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_ENGINEHEUR_DESCRIPTOR, &desc) 创建的 mcDNN 后端引擎启发式描述符，允许用户获取操作图引擎配置描述符，操作图引擎配置描述符根据 mcDNN 的启发式按性能排序。

属性

MCDNN_ATTR_ENGINEHEUR_OPERATION_GRAPH

启发式结果进行查询的操作图。

MCDNN_TYPE_BACKEND_DESCRIPTOR

一个元素。

- 必要属性

MCDNN_ATTR_ENGINEHEUR_MODE

查询结果的启发式模式。

- MCDNN_TYPE_HEUR_MODE；一个元素。
- 必要属性

MCDNN_ATTR_ENGINEHEUR_RESULTS

启发式查询的结果。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_ENGINECFG_DESCRIPTOR 的零个或多个元素。
- 只读属性。

终结

返回 mcdnnBackendFinalize(desc) 的值，其中 desc 是 mcDNN 后端引擎启发式描述符：

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.5 MCDNN_BACKEND_EXECUTION_PLAN_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_EXECUTION_PLAN_DESCRIPTOR, &desc) 创建的 mcDNN 后端执行计划描述符，允许用户指定执行计划，该计划由 mcDNN 句柄，引擎配置以及（可选）一组要计算的中间结果组成。

属性

MCDNN_ATTR_EXECUTION_PLAN_HANDLE

mcDNN 句柄。

- MCDNN_TYPE_HANDLE；一个元素。
- 必要属性

MCDNN_ATTR_EXECUTION_PLAN_ENGINE_CONFIG

要执行的引擎配置。

- MCDNN_BACKEND_ENGINECFG_DESCRIPTOR；一个元素。
- 必要属性

MCDNN_ATTR_EXECUTION_PLAN_RUN_ONLY_INTERMEDIATE_UIDS

要计算的中间结果的唯一标识符。

- MCDNN_TYPE_INT64；零个或多个元素。
- 可选属性。如果设置此属性，执行计划将只计算指定的中间结果，而不计算引擎配置中操作图上的任何输出张量。

MCDNN_ATTR_EXECUTION_PLAN_COMPUTED_INTERMEDIATE_UIDS

预计算中间结果的唯一标识符。

- MCDNN_TYPE_INT64；零个或多个元素。
- 可选属性。如果设置此属性，计划将在执行过程中使用 variant pack 描述符中每个中间结果的指针。
- 当前不支持：用于将来实现的占位符。

MCDNN_ATTR_EXECUTION_PLAN_WORKSPACE_SIZE

执行该计划所需的工作空间缓冲区的大小。

- MCDNN_TYPE_INT64；一个元素。
- 只读属性。

MCDNN_ATTR_EXECUTION_PLAN_JSON_REPRESENTATION

序列化执行计划的 JSON 表达形式。序列化和反序列化可以分别通过获取和设置此属性来完成。

- MCDNN_TYPE_CHAR；许多元素，数量与执行计划的 json 表达形式的 NULL 结尾字符串的大小相同。

终结

返回 mcdnnBackendFinalize(desc) 的值，其中 desc 是 mcDNN 后端执行计划描述符：

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.6 MCDNN_BACKEND_INTERMEDIATE_INFO_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_INTERMEDIATE_INFO_DESCRIPTOR, &desc) 创建的 mcDNN 后端中间结果描述符，是一个只读描述符，包含有关执行中间结果的信息。执行中间结果是对设备内存中引擎配置进行的一些中间计算，可在计划执行之间重复使用，以摊销内核。每个中间结果都由唯一的 ID 标识。用户可以查询中间结果的设备内存大小。中间结果可以依赖于由张量 UID 标识的一个或多个张量的数据或操作图的另一个属性的数据。

此描述符是一个只读描述符。用户无法设置此描述符属性或终结此描述符。用户从引擎配置描述符查询终结的描述符。

属性

MCDNN_ATTR_INTERMEDIATE_INFO_UNIQUE_ID

中间结果的唯一标识符。

- MCDNN_TYPE_INT64；一个元素。
- 只读属性。

MCDNN_ATTR_INTERMEDIATE_INFO_SIZE

中间结果所需的设备内存大小。

- MCDNN_TYPE_INT64；一个元素。
- 只读属性。

MCDNN_ATTR_INTERMEDIATE_INFO_DEPENDENT_DATA_UIDS

中间结果所依赖张量的 UID。

- MCDNN_TYPE_INT64；零个或多个元素。
- 只读属性。

MCDNN_ATTR_INTERMEDIATE_INFO_DEPENDENT_ATTRIBUTES

用于将来实现的占位符。

终结

用户未终结此描述符。带有后端中间描述符的 mcdnnBackendFinalize(desc) 返回 MCDNN_STATUS_NOT_SUPPORTED。

8.2.2.7 MCDNN_BACKEND_KNOB_CHOICE_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_KNOB_CHOICE_DESCRIPTOR, &desc) 创建的 mcDNN 后端旋钮选择描述符，由要设置的旋钮类型和要设置旋钮的值组成。

属性

MCDNN_ATTR_KNOB_CHOICE_KNOB_TYPE

要设置的旋钮类型。

- MCDNN_TYPE_KNOB_TYPE：一个元素。
- 必要属性

MCDNN_ATTR_KNOB_CHOICE_KNOB_VALUE

- MCDNN_TYPE_INT64；一个元素。
- 必要属性

终结

返回 mcdnnBackendFinalize(desc) 的值，其中 desc 是 mcDNN 后端旋钮选择描述符：

MCDNN_STATUS_SUCCESS

旋钮选择描述符已成功终结。

8.2.2.8 MCDNN_BACKEND_KNOB_INFO_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_INFO_DESCRIPTOR, &desc) 创建的 mcDNN 后端旋钮信息描述符，由引擎性能旋钮的类型和有效值范围组成。有效值范围根据有效值的最小值，最大值和步幅给定。这是一个纯粹的信息描述符类型。不支持设置描述符属性。用户从终结的后端描述符中获取一组终结的描述符，每个旋钮类型对应一个描述符。

属性

MCDNN_ATTR_KNOB_INFO_TYPE

性能旋钮的类型。

- MCDNN_TYPE_KNOB_TYPE: one element.
- 只读属性。

MCDNN_ATTR_KNOB_INFO_MAXIMUM_VALUE

此旋钮的最小有效选择值。

- MCDNN_TYPE_INT64；一个元素。
- 只读属性。

MCDNN_ATTR_KNOB_INFO_MINIMUM_VALUE

此旋钮的最大有效选择值。

- MCDNN_TYPE_INT64；一个元素。
- 只读属性。

MCDNN_ATTR_KNOB_INFO_STRIDE

此旋钮有效选择值的步幅。

- MCDNN_TYPE_INT64；一个元素。
- 只读属性。

终结

此描述符为只读描述符；它是从 mcDNN 后端引擎配置描述符中检索和终结的。用户无法进行设置或终结。

8.2.2.9 MCDNN_BACKEND_LAYOUT_INFO_DESCRIPTOR

mcDNN 后端布局信息描述符（使用描述符类型值 MCDNN_BACKEND_LAYOUT_INFO_DESCRIPTOR 创建），提供有关张量首选布局的信息。

属性

MCDNN_ATTR_LAYOUT_INFO_TENSOR_UID

张量的 UID。

- MCDNN_TYPE_INT64；一个元素。
- 只读属性。

MCDNN_ATTR_LAYOUT_INFO_TYPES

张量的首选布局。

- MCDNN_TYPE_LAYOUT_TYPE：0 或者多个元素按 mcdnnBackendLayoutType_t 排列。
- 只读属性。

终结

此描述符为只读描述符；它是从 mcDNN 后端引擎配置描述符中检索和终结的。用户无法对其进行属性设置或终结。

8.2.2.10 MCDNN_BACKEND_MATMUL_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_MATMUL_DESCRIPTOR, &desc) 创建的 mcDNN 后端 matmul 描述符，指定 matmul 操作所需的任意元数据。

属性

MCDNN_ATTR_MATMUL_COMP_TYPE

用于 matmul 操作的计算精度。

- MCDNN_TYPE_DATA_TYPE；一个元素。
- 必要属性

终结

返回 mcdnnBackendFinalize(desc) 的值，其中 desc 是 mcDNN 后端 matmul 描述符：

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.11 MCDNN_BACKEND_OPERATION_CONCAT_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_CONCAT_DESCRIPTOR, &desc) 创建的 mcDNN 后端连接操作描述符，指定一个操作节点，用于沿给定连接轴连接张量的给定向量。

此操作还支持就地模式，其中假定一个输入张量在输出张量中的正确位置，即它们共享相同的设备缓冲区。

属性

mcDNN 后端连接操作描述符的属性是 mcdnnBackendAttributeName_t 枚举类型的值（前缀为 MCDNN_ATTR_OPERATION_CONCAT_）：

MCDNN_ATTR_OPERATION_CONCAT_AXIS

连接的张量维度。

- 类型：MCDNN_TYPE_INT64
- 必要属性

MCDNN_ATTR_OPERATION_CONCAT_INPUT_DESCS

输入张量描述符的向量，按该向量中提供的顺序进行连接。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个或多个元素。
- 必要属性

MCDNN_ATTR_OPERATION_CONCAT_INPLACE_INDEX

输入张量描述符向量中的输入张量索引，该索引已就地存在于输出张量中。

- 类型：MCDNN_TYPE_INT64
- 可选属性。

MCDNN_ATTR_OPERATION_CONCAT_OUTPUT_DESC

描述输入张量连接结果的输出张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

终结

带有 MCDNN_BACKEND_OPERATION_CONCAT_DESCRIPTOR() 的 mcdnnBackendFinalize() 可以具有以下返回值：

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- 操作中涉及的张量在所有维度中都应具有相同的形状，它们所连接的维度除外。
- 连接维度中的输出张量形状应等于同一维度中所有输入张量的形状总和。
- 连接轴应为有效的张量维度。
- 如果提供就地输入张量索引，其应是输入张量描述符向量中的有效索引。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.12 MCDNN_BACKEND_OPERATION_CONVOLUTION_BACKWARD _DATA_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_CONVOLUTION_BACKWARD _DATA_DESCRIPTOR, &desc) 创建的 mcDNN 后端卷积反向数据操作描述符，指定卷积反向数据的操作节点，以使用卷积核张量 w，响应梯度 dy，输出缩放比例 α 和残差相加缩放比例 β，来计算输入数据梯度 dx。即公式 $dx = \alpha \left(w \overset{-}{*} dy\right) + \beta dx$，其中 $\overset{-}{*}$ 表示反向卷积数据运算符。

属性

mcDNN 后端卷积描述符的属性是 mcdnnBackendAttributeName_t 枚举类型的值（前缀为 MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_）：

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_ALPHA

alpha 值。

- MCDNN_TYPE_FLOAT 或 MCDNN_TYPE_DOUBLE；一个或多个元素。
- 必要属性

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_BETA

beta 值。

- MCDNN_TYPE_FLOAT 或 MCDNN_TYPE_DOUBLE；一个或多个元素。
- 必要属性

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_CONV_DESC

卷积运算符描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_CONVOLUTION_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_W

卷积卷积核张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_DX

图像梯度张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_DATA_DY

响应梯度张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

终结

在终结卷积操作时，张量 DX，W 和 DY 的张量维度将根据 MCDNN_BACKEND_OPERATION_CONVOLUTION_FORWARD_DESCRIPTOR 中描述的 X，W 和 Y 张量维度的相同解释进行绑定。

带有 MCDNN_BACKEND_OPERATION_CONVOLUTION_BACKWARD_DATA_DESCRIPTOR 的 mcdnnBackendFinalize() 可以具有以下返回值：

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- 在卷积运算符下，DX，W 和 DY 张量不构成有效的卷积操作。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.13 MCDNN_BACKEND_OPERATION_CONVOLUTION_BACKWARD_FILTER _DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_CONVOLUTION_BACKWARD_FILTER_DESCRIPTOR, &desc) 创建的 mcDNN 后端卷积反向卷积核操作描述符，指定卷积反向卷积核的操作节点，以使用图像张量 w，响应梯度 dy，输出缩放比例 α 和残差相加缩放比例 β，来计算卷积核梯度 dw。即公式：$dw = \alpha\ (x *\tilde{} dy) + \beta dw$，其中 $*\tilde{}$ 表示反向卷积卷积核运算符。

属性

mcDNN 后端卷积描述符的属性是 mcdnnBackendAttributeName_t 枚举类型的值（前缀为 MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_）：

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_ALPHA

alpha 值。

- MCDNN_TYPE_FLOAT or MCDNN_TYPE_DOUBLE; one or more elements.
- 必要属性需要在终结前进行设置。

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_BETA

beta 值。

- MCDNN_TYPE_FLOAT 或 MCDNN_TYPE_DOUBLE；一个或多个元素。
- 必要属性需要在终结前进行设置。

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_CONV_DESC

卷积运算符描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_CONVOLUTION_DESCRIPTOR 的一个元素。
- 必要属性需要在终结前进行设置。

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_DW

卷积卷积核张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性需要在终结前进行设置。

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_X

图像梯度张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性需要在终结前进行设置。

MCDNN_ATTR_OPERATION_CONVOLUTION_BWD_FILTER_DY

响应梯度张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性需要在终结前进行设置。

终结

在终结卷积操作时，张量 X，DW 和 DY 的张量维度将根据 MCDNN_BACKEND_OPERATION_CONVOLUTION_FORWARD_DESCRIPTOR 中描述的 X，W 和 Y 张量维度的相同解释进行绑定。

带有 MCDNN_BACKEND_OPERATION_CONVOLUTION_BACKWARD_FILTER_DESCRIPTOR 的 mcdnnBackendFinalize() 可以具有以下返回值：

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- 在卷积运算符下，X，DW 和 DY 张量不构成有效的卷积操作。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.14 MCDNN_BACKEND_OPERATION_CONVOLUTION_FORWARD_ DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_CONVOLUTION_FORWARD_DESCRIPTOR, &desc) 创建的 mcDNN 后端卷积正向操作描述符，指定正向卷积的操作节点，以使用卷积核张量 w，输出缩放比例 α 和残差相加缩放比例 β，来卷积计算图像张量 x 的响应张量 y。即公式 $dw = \alpha\ (w * x)\ + \beta y$，其中 * 是正向的卷积运算符。

属性

mcDNN 后端卷积描述符的属性是 mcdnnBackendAttributeName_t 枚举类型的值（前缀为 MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_）：

MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_ALPHA

alpha 值。

- MCDNN_TYPE_FLOAT 或 MCDNN_TYPE_DOUBLE；一个或多个元素。
- 需要在终结前进行设置。

MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_BETA

beta 值。

- MCDNN_TYPE_FLOAT 或 MCDNN_TYPE_DOUBLE；一个或多个元素。
- 必要属性

MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_CONV_DESC

卷积运算符描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_CONVOLUTION_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_W

卷积卷积核张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_X

图像张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_Y

响应张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_CONVOLUTION_SPATIAL_DIMS

卷积中的空间维数。

- MCDNN_TYPE_INT64，一个元素。
- 必要属性

终结

在终结卷积操作时，张量 X，W 和 Y 的张量维度将根据以下解释进行绑定：

MCDNN_ATTR_OPERATION_CONVOLUTION_FORWARD_CONV_DESC 的 MCDNN_ATTR_CONVOLUTION_SPATIAL_DIMS 属性是卷积的空间维数。张量 X，W 和 Y 的维数必须比空间维数大 2 或 3，具体取决于用户选择如何指定卷积张量。

如果张量维数为空间维数 +2：

- X 张量维度和步幅数组为 [N，GC，…]
- W 张量维度和步幅数组为 [KG，C，…]
- Y 张量维度和步幅数组为 [N，GK，…]

其中省略号…是每个张量空间维度的缩略，G 是卷积组数，C 和 K 是每个组的输入和输出特征图数。在这种解释中，假定每个组的内存布局都是压缩的。mcdnnBackendFinalize() 声明张量维度和步幅与此解释一致，否则返回 MCDNN_STATUS_BAD_PARAM。

如果张量维数为空间维数 +3：

- X 张量维度和步幅数组为 [N，G，C，…]
- W 张量维度和步幅数组为 [G，K，C，…]
- Y 张量维度和步幅数组为 [N，G，K，…]

其中省略号…是每个张量空间维度的缩略，G 是卷积组数，C 和 K 是每个组的输入和输出特征图数。在这种解释中，用户可以指定未压缩的组步幅。mcdnnBackendFinalize() 声明张量维度和步幅与此解释一致，否则返回 MCDNN_STATUS_BAD_PARAM。

带有 MCDNN_BACKEND_OPERATION_CONVOLUTION_FORWARD_DESCRIPTOR 的 mcdnnBackendFinalize() 可以具有以下返回值：

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- 在卷积运算符下，X，W 和 Y 张量不构成有效的卷积操作。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.15 MCDNN_BACKEND_OPERATION_GEN_STATS_DESCRIPTOR

表示会生成每个通道统计信息的操作。描述符中的 MCDNN_ATTR_OPERATION_GENSTATS_MODE 属性决定生成特定的统计信息。目前，MCDNN_ATTR_OPERATION_GENSTATS_MODE 仅支持 MCDNN_GENSTATS_SUM_SQSUM。它将生成输入张量 x 的每个通道元素的总和与平方和。除 C 维度外，其他输出维度应都为 1。此外，输出的 C 维应等于输入的 C 维。此不透明结构可以使用 mcdnnBackendCreateDescriptor() （MCDNN_BACKEND_OPERATION_GEN_STATS_DESCRIPTOR）来创建。

属性

MCDNN_ATTR_OPERATION_GENSTATS_MODE

设置操作的 MCDNN_TYPE_GENSTATS_MODE。必要属性。

MCDNN_ATTR_OPERATION_GENSTATS_MATH_PREC

计算的数学精度。必要属性。

MCDNN_ATTR_OPERATION_GENSTATS_XDESC

设置输入张量 X 的描述符。此属性为必要属性。

MCDNN_ATTR_OPERATION_GENSTATS_SUMDESC

设置输出张量总和的描述符。必要属性。

MCDNN_ATTR_OPERATION_GENSTATS_SQSUMDESC

设置输出张量平方和的描述符。必要属性。

终结

在终结阶段，对属性进行交叉检查，以确保没有冲突。可能会返回以下状态：

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- 输入和输出张量之间的维数不匹配。
- 输入/输出张量维度与上述描述不一致。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.16 MCDNN_BACKEND_OPERATION_MATMUL_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_MATMUL_DESCRIPTOR, &desc) 创建的 mcDNN 后端 matmul 操作描述符，指定 matmul 操作节点，通过矩阵 A 和矩阵 B 相乘来计算矩阵乘积 C，如以下公式所示：C=AB

使用 matmul 操作时，矩阵至少应为秩-2 张量。最后两个维度应对应于 M、K 或 N。前面所有维度均解释为 batch 维度。如果 batch 维度为零，则要求如下：

零 Batch 维度的 MCDNN_BACKEND_OPERATION_MATMUL_DESCRIPTOR

Case	Matrix A	Matrix B	Matrix C
Single matmul	M x K	K x N	M x N

对于单个 batch 维度，要求以下：

单 Batch 维度的 MCDNN_BACKEND_OPERATION_MATMUL_DESCRIPTOR

Case	Matrix A	Matrix B	Matrix C
Single matmul	1 x M x K	1 x K x N	1 x M x N
Batch matmul	B x M x K	B x K x N	B x M x N
Broadcast A	1 x M x K	B x K x N	
Broadcast B	B x M x K	1 x K x N	

其中：

- B 表示批大小
- M 是矩阵 A 的行数
- K 是输入矩阵 A 的列数（与输入矩阵 B 的行数相同）
- N 是输入矩阵 B 的列数

如果矩阵 A 或 B 的批大小设置为 1，则表示矩阵将在 batch matmul 中广播。输出矩阵 C 将为 B x M x N 的张量。

上述广播规则适用于所有 batch 维度。具体而言，适用于具有 3 个 batch 维度的张量：

3 Batch 维度的 MCDNN_BACKEND_OPERATION_MATMUL_DESCRIPTOR

Case	Matrix A	Matrix B	Matrix C
Multiple batched matmul	B1 x 1 x B3 x M x K	1 x B2 x B3 x K x N	B1 x B2 x B3 x M x N

具有多个 batch 维度的功能支持此布局：batch 没有以单一步幅进行压缩。这种情况在多头注意力机制中尤其常见。

可以使用张量描述符中的步幅指定给定张量中矩阵元素的寻址。步幅表示每个张量维度的元素之间的间距。矩阵张量 A（B x M x N）具有步幅 [BS，MS，NS]，它表示实际矩阵元素 A[x，y，z] 位于为张量 A 分配的线性内存空间的（A_base_address + x * BS + y * MS + z * NS）中。目前，必须对最内维度进行压缩，需要 MS=1 或 NS=1。否则，对于如何在张量描述符中指定步幅，没有其他技术限制，因为它应该遵循上述寻址公式和用户指定的步幅。

这种表示为一些常见的用法提供了支持，例如前导维度和矩阵转置，我们将通过以下示例进行解释。

1. 最常见的情况是一个满载的行主 batch 矩阵，不考虑任何前导维度和转置。在这种情况下，BS = M*N，MS = N，NS = 1。
2. 矩阵转置可以通过使用步幅交换内部和外部维度来实现。即：
   - 指定非转置矩阵：BS = M*N, MS = N and NS = 1
   - 指定矩阵转置：BS = M*N, MS = 1 and NS = M
3. 前导维度是类 BLAS API 中广泛使用的概念，它描述了 2D 数组内存分配的内部维度（与概念上的矩阵维度相对）。它在某种程度上类似于步幅，定义了外部维度中元素之间的间距。显示与矩阵内部维度之间差异的最典型用例，是当矩阵仅是已分配内存中数据的一部分时，对子矩阵或对齐内存分配中的矩阵进行寻址。因此，列主矩阵 A 中的前导维度 LDA 必须满足 LDA >= M，而在行主矩阵 A 中，它必须满足 LDA >= N。要从前导维度概念过渡到使用步幅，必须满足 MS >= N 且 NS = 1，或 MS = 1 且 NS >= M。请记住，虽然这是一些实际使用案例，但这些不等式不会对可接受的步幅规格施加技术限制。

其他常用的 GEMM 功能，如 alpha/beta 输出混合，也可以使用 matmul 操作和其他逐点操作来实现。

属性

mcDNN 后端 matmul 描述符的属性是 mcdnnBackendAttributeName_t 枚举类型的值（前缀为 MCDNN_ATTR_OPERATION_MATMUL_）：

MCDNN_ATTR_OPERATION_MATMUL_ADESC

矩阵 A 描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。

- 必要属性

MCDNN_ATTR_OPERATION_MATMUL_BDESC

矩阵 B 描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_MATMUL_CDESC

矩阵 C 描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_MATMUL_IRREGULARLY_STRIDED_BATCH_COUNT

要在矩阵的 batch 中执行的 matmul 操作数。Default = 1.

- MCDNN_TYPE_INT64；一个元素。
- 默认值为 1。

MCDNN_ATTR_OPERATION_MATMUL_GEMM_M_OVERRIDE_DESC

gemm_m_override 张量描述符。允许通过此张量覆盖 batch 矩阵乘法的 M 维。仅在运行时融合引擎中记录的运行时融合引擎的模式 6 中受支持。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_MATMUL_GEMM_N_OVERRIDE_DESC

gemm_n_override 张量描述符。允许通过此张量覆盖 batch 矩阵乘法的 N 维。仅在运行时融合引擎中记录的运行时融合引擎的模式 6 中受支持。

- MCDNN_TYPE_BACKEND_DESCRIPTOR; one element of descriptor type MCDNN_BACKEND_TENSOR_DESCRIPTOR.
- 可选属性。

MCDNN_ATTR_OPERATION_MATMUL_GEMM_K_OVERRIDE_DESC

gemm_k_override 张量描述符。支持通过此张量覆盖 batch 矩阵乘法的 K 维。仅在运行时融合引擎中记录的运行时融合引擎的模式 6 中受支持。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_MATMUL_DESC

matmul 操作描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_MATMUL_DESCRIPTOR 的一个元素。
- 必要属性

终结

在 matmul 操作的终结过程中，将检查矩阵 A，B 和 C 的张量维度，以确保满足矩阵乘法的要求:

带有 MCDNN_BACKEND_OPERATION_MATMUL_DESCRIPTOR 的 mcdnnBackendFinalize() 可以具有以下返回值:

MCDNN_STATUS_NOT_SUPPORTED

遇到不支持的属性值。可能的原因包括：

- 不是所有的矩阵 A，B 和 C 都至少是秩-2 张量。

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- MCDNN_ATTR_OPERATION_MATMUL_IRREGULARLY_STRIDED_BATCH_COUNT 的指定值为负。
- MCDNN_ATTR_OPERATION_MATMUL_IRREGULARLY_STRIDED_BATCH_COUNT 和矩阵 A，B 和 C 的一个或多个批大小不等于 1。即有一个冲突，指定了不规则的和规则的跨步批处理矩阵乘法（strided batched matrix multiplication），这是一个无效用例。
- 矩阵 A，B 和 C 的维度不满足矩阵乘法的要求。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.17 MCDNN_BACKEND_OPERATION_NORM_BACKWARD_DESCRI PTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_NORM_BACKWARD_ DESCRIPTOR, &desc) 创建的 mcDNN 后端反向归一化操作，指定一个用于反向归一化的节点，该节点将用于输入梯度张量 dY，输出梯度张量 dX、权重梯度 dScale 和 dBias。使用 MCDNN_ATTR_OPERATION_NORM_BWD_MODE 属性设置归一化模式。

注解： 在 mcDNN 中，对该操作的支持仅限于实验性多 GPU 批量归一化后端模式，并具有一组功能限制。在将来的版本中，将扩展以支持不同的模式。

属性

MCDNN_ATTR_OPERATION_NORM_BWD_MODE

为反向归一化操作选择归一化模式。

- MCDNN_TYPE_NORM_MODE；一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_BWD_XDESC

输入张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_BWD_MEAN_DESC

保存的均值输入张量描述符，用于在训练阶段正向计算期间重复使用计算的均值。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_NORM_BWD_INV_VARIANCE_DESC

保存的逆方差输入张量描述符，用于在训练阶段正向计算期间重复使用计算的均值。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。

- 可选属性。

MCDNN_ATTR_OPERATION_NORM_BWD_DYDESC

梯度张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_NORM_BWD_DYDESC

梯度张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_BWD_SCALE_DESC

归一化 scale 描述符。请注意，反向传递不需要 bias 描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_BWD_EPSILON_DESC

epsilon 值的标量输入张量描述符。仅当保存的均值和方差未作为输入传递到操作时，才需要 epsilon 值。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_NORM_BWD_DSCALE_DESC

scale 梯度张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_BWD_DBIAS_DESC

bias 梯度张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_BWD_DXDESC

输入梯度张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_BWD_PEER_STAT_DESCS

多 GPU 归一化中使用的通信缓冲区的张量描述符向量。通常，为节点中的每个 GPU 提供一个缓冲区。这是一个可选属性，仅用于多 GPU 张量统计数据归约。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个或多个元素。

- 可选属性。

终结

在终结阶段，对属性进行检查，以确保没有冲突。

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- 梯度张量 dY，dX 和输入张量 X 的维度不匹配。
- 均值，scale 和 inv_variance 张量的通道计数 C 不匹配。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.18 MCDNN_BACKEND_OPERATION_NORM_FORWARD_DESCRI PTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_NORM_FORWARD_DESCRIPTOR, &desc) 创建的 mcDNN 后端正向归一化操作，指定一个用于正向归一化的节点，该节点将用于输入张量 X，并生成归一化输出张量 Y，其归一化模式通过 MCDNN_ATTR_OPERATION_NORM_FWD_MODE 属性设置。该操作支持可选的运行统计计算，并允许存储计算的均值和方差，以便根据 MCDNN_ATTR_OPERATION_NORM_FWD_PHASE 属性的设置在反向计算中重用。

注解： 在 mcDNN 中，对该操作的支持仅限于实验性多 GPU 批量归一化后端模式，并具有一组功能限制。在将来的版本中，将扩展以支持不同的模式。

属性

MCDNN_ATTR_OPERATION_NORM_FWD_MODE

为正向归一化操作选择归一化模式。

- MCDNN_TYPE_NORM_MODE；一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_FWD_PHASE

为正向归一化操作选择训练或推理阶段。

- MCDNN_TYPE_NORM_FWD_PHASE；一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_FWD_XDESC

输入张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_FWD_MEAN_DESC

推理阶段的预估均值输入张量描述符和训练阶段的计算均值输出张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_NORM_FWD_INV_VARIANCE_DESC

推理阶段的预估逆方差输入张量描述符和训练阶段的计算逆方差输出张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_NORM_FWD_SCALE_DESC

归一化 scale 输入张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_FWD_BIAS_DESC

归一化 bias 输入张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_FWD_EPSILON_DESC

归一化计算中使用的 epsilon 值的标量输入张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_FWD_EXP_AVG_FACTOR_DESC

运行统计计算时使用的指数平均系数值的标量输入张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_NORM_FWD_INPUT_RUNNING_MEAN_DESC

训练阶段中运行统计计算的输入运行均值张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_NORM_FWD_INPUT_RUNNING_VAR_DESC

训练阶段中运行统计计算的输入运行方差张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_NORM_FWD_OUTPUT_RUNNING_MEAN_DESC

训练阶段中运行统计计算的输出运行均值张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_NORM_FWD_OUTPUT_RUNNING_VAR_DESC

训练阶段中运行统计计算的输出运行方差张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。

- 可选属性。

MCDNN_ATTR_OPERATION_NORM_FWD_YDESC

归一化操作输出的张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_NORM_FWD_PEER_STAT_DESCS

多 GPU 归一化中使用的通信缓冲区的张量描述符向量。通常，为节点中的每个 GPU 提供一个缓冲区。这是一个可选属性，仅用于多 GPU 张量统计数据归约。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个或多个元素。
- 可选属性。

终结

在终结阶段，对属性进行检查，以确保没有冲突。

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- 输出张量维度与输入张量维度不匹配。
- 均值，scale，bias 和 inv_variance 张量的通道计数 C 不匹配。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.19 MCDNN_BACKEND_OPERATION_POINTWISE_DESCRIPTOR

表示根据操作类型实现公式 $Y = op\ (alpha1 * X)$ 或 $Y = op\ (alpha1 * X, alpha2 * B)$ 的逐点操作。上述 op() 表示的实际操作类型取决于描述符中的 MCDNN_ATTR_OPERATION_POINTWISE_PW_DESCRIPTOR 属性。此操作描述符支持单输入单输出操作。

有关支持的操作列表，请参见 mcdnnPointwiseMode_t。

对于双输入逐点操作，当其中一个张量的维度为 1 而其他张量的相应维度不为 1 时，执行广播。

对于三输入单输出逐点操作，不支持在任何张量中广播。

此不透明结构可以使用 mcdnnBackendCreateDescriptor()

(MCDNN_BACKEND_OPERATION_POINTWISE_DESCRIPTOR) 来创建。

属性

MCDNN_ATTR_OPERATION_POINTWISE_PW_DESCRIPTOR

设置描述符，该描述符包含逐点操作的数学设置。必要属性。

MCDNN_ATTR_OPERATION_POINTWISE_XDESC

设置输入张量 x 的描述符。逐点数学函数或激活正向传播计算需要此属性。

MCDNN_ATTR_OPERATION_POINTWISE_BDESC

如果操作需要 2 个输入，例如加或乘，此属性将设置第二个输入张量 β。如果操作只需要 1 个输入，则此字段不应设置和使用。

MCDNN_ATTR_OPERATION_POINTWISE_YDESC

设置输出张量 T 的描述符。逐点数学函数或激活正向传播计算需要此属性。

MCDNN_ATTR_OPERATION_POINTWISE_TDESC

设置张量 T 的描述符。MCDNN_ATTR_POINTWISE_MODE 设置为 MCDNN_POINTWISE_BINARY_SELECT 时需要此属性，并作为完成选择所基于的掩码。

MCDNN_ATTR_OPERATION_POINTWISE_ALPHA1

设置公式中的标量 alpha1 值。可以是 float 或 half。此属性是可选的，如果未设置，则默认值为 1.0。

MCDNN_ATTR_OPERATION_POINTWISE_ALPHA2

如果操作需要 2 个输入，例如加或乘，此属性将设置公式中的标量 alpha2 值。可以是 float 或 half。此属性是可选的，如果未设置，则默认值为 1.0。如果操作只需要 1 个输入，则此字段不应设置和使用。

MCDNN_ATTR_OPERATION_POINTWISE_DXDESC

设置输出张量 dX 的描述符。逐点激活反向传播计算需要此属性。

MCDNN_ATTR_OPERATION_POINTWISE_DYDESC

设置输入张量 dY 的描述符。逐点激活反向传播计算需要此属性。

终结

在终结阶段，对属性进行交叉检查，以确保没有冲突。可能会返回以下状态：

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- 输入和输出张量之间的维数不匹配。
- 输入/输出张量维度与上述描述的自动广播规则不一致。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.20 MCDNN_BACKEND_OPERATION_REDUCTION_DESCRIPTOR

mcDNN 后端归约操作描述符表示一个操作节点，该操作节点在一个或多个维度中实现输入张量 X 的归约值，以获取输出张量 Y。用于归约张量值的数学运算和计算数据类型通过 MCDNN_ATTR_OPERATION_REDUCTION_DESC 指定。

可以使用以下内容创建此操作描述符

```
mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_REDUCTION_DESCRIPTOR,
↪&desc);
```

输出张量 Y 的大小应与输入张量 X 的大小相同，但大小为 1 的维度除外。

属性

mcDNN 后端归约描述符的属性是 mcdnnBackendAttributeName_t 枚举类型的值（前缀为 MCDNN_ATTR_OPERATION_REDUCTION_）：

MCDNN_ATTR_OPERATION_REDUCTION_XDESC

矩阵 X 描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_REDUCTION_YDESC

矩阵 Y 描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_REDUCTION_DESC

归约操作描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_REDUCTION_DESCRIPTOR 的一个元素。
- 必要属性

终结

在归约操作的终结过程中，将检查张量 X 和 Y 的维度，以确保满足归约操作的要求。

带有 MCDNN_BACKEND_OPERATION_REDUCTION_DESCRIPTOR 的 mcdnnBackendFinalize() 可以具有以下返回值：

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- 张量 X 和 Y 的维度不满足归约操作的要求。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.21 MCDNN_BACKEND_OPERATION_RESAMPLE_BWD_DESCRI PTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_RESAMPLE_BWD_ DESCRIPTOR, &desc) 创建的 mcDNN 后端重采样反向操作描述符，指定一个操作节点，用于反向重采样。根据 MCDNN_ATTR_RESAMPLE_MODE 进行反向重采样，从输出张量梯度 dy 计算输入张量梯度 dx，输出缩放比例为 α，残差相加缩放比例为 β。

属性

MCDNN_ATTR_OPERATION_RESAMPLE_BWD_DESC

包含操作元数据的重采样操作描述符（MCDNN_BACKEND_RESAMPLE_DESCRIPTOR）实例。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_RESAMPLE_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_RESAMPLE_BWD_DXDESC

输入张量梯度描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_RESAMPLE_BWD_DYDESC

输出张量梯度描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_RESAMPLE_BWD_IDXDESC

张量, 包含要在 backprop 中使用的 maxpool 或最近邻重采样索引。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

MCDNN_ATTR_OPERATION_RESAMPLE_BWD_ALPHA

设置混合中使用的 alpha 参数。

- MCDNN_TYPE_DOUBLE or MCDNN_TYPE_FLOAT; one element.
- 可选属性。
- 默认值为 1.0。

MCDNN_ATTR_OPERATION_RESAMPLE_BWD_BETA

设置混合中使用的 beta 参数。

- MCDNN_TYPE_DOUBLE 或 MCDNN_TYPE_FLOAT；一个元素。
- 可选属性。
- 默认值为 0.0。

MCDNN_ATTR_OPERATION_RESAMPLE_BWD_XDESC

输入张量 X 描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。
- NCHW 布局所需。

MCDNN_ATTR_OPERATION_RESAMPLE_BWD_YDESC

输入张量 Y 描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。
- NCHW 布局所需。

终结

在终结阶段，对属性进行交叉检查，以确保没有冲突。可能会返回以下状态：

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- 基于填充和步幅计算的输出形状与给定的输出张量维度不匹配。
- YDESC 和 IDXDESC（如果提供）的形状不匹配。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.22 MCDNN_BACKEND_OPERATION_RESAMPLE_FWD_DESCRI PTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_RESAMPLE_FWD_ DESCRIPTOR, &desc) 创建的 mcDNN 后端重采样正向操作描述符，指定一个操作节点，用于正向重采样。根据 MCDNN_ATTR_RESAMPLE_MODE 进行重采样，计算图像张量 x 的输出张量 y，输出缩放比例为 α，残差相加缩放比例为 β。

属性

MCDNN_ATTR_OPERATION_RESAMPLE_FWD_DESC

包含操作元数据的重采样操作描述符（MCDNN_BACKEND_RESAMPLE_DESCRIPTOR）实例。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_RESAMPLE_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_RESAMPLE_FWD_XDESC

输入张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_RESAMPLE_FWD_YDESC

输出张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_RESAMPLE_FWD_IDXDESC

张量, 包含要在 backprop 中使用的 maxpool 或最近邻重采样索引。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性（主要用于涉及训练的用例）。

MCDNN_ATTR_OPERATION_RESAMPLE_FWD_ALPHA

设置混合中使用的 alpha 参数。

- MCDNN_TYPE_DOUBLE 或 MCDNN_TYPE_FLOAT；一个元素。
- 可选属性。
- 默认值为 1.0。

MCDNN_ATTR_OPERATION_RESAMPLE_FWD_BETA

设置混合中使用的 beta 参数。

- MCDNN_TYPE_DOUBLE 或 MCDNN_TYPE_FLOAT；一个元素。
- 可选属性。
- 默认值为 0.0。

终结

在终结阶段，对属性进行交叉检查，以确保没有冲突。可能会返回以下状态：

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。可能的原因包括：

- 基于填充和步幅计算的输出形状与给定的输出张量维度不匹配。
- YDESC 和 IDXDESC（如果提供）的形状不匹配。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.23 MCDNN_BACKEND_OPERATION_RNG_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_RNG_DESCRIPTOR, &desc 创建的 mcDNN 后端 Rng 操作描述符，指定一个操作节点，用于根据 Rng 描述符中指定的概率分布使用随机数生成张量。

随机数是使用 Philox 随机数生成器 (RNG) 生成的，如 Pytorch 中所述。Philox 对象携带一个种子值，一个用于开始生成的子序列，以及子序列的偏移。可以使用属性设置种子和偏移。子序列在内部设置，以确保独立的随机数。

属性

MCDNN_ATTR_OPERATION_RNG_DESC

包含操作元数据的 Rng 描述符（MCDNN_BACKEND_RNG_DESCRIPTOR）实例。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_RNG_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_RNG_YDESC

输出张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_RNG_SEED

为创建 Y 张量的随机数生成器设置种子。它可以是一个主机 INT64 值，或是一个绑定到设备的一个值上的后端描述符。仅支持所有维度和步幅都设置为 1 的张量。

- MCDNN_TYPE_INT64；MCDNN_TYPE_BACKEND_DESCRIPTOR 的一个元素；描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。
- 默认值为 0。

MCDNN_ATTR_OPERATION_RNG_OFFSET_DESC

RNG Philox 对象中使用的偏移的张量描述符。仅支持所有维度和步幅都设置为 1 的张量。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

终结

在终结阶段，对属性进行交叉检查，以确保没有冲突。可能会返回以下状态：

MCDNN_STATUS_BAD_PARAM

没有将 MCDNN_ATTR_OPERATION_RNG_OFFSET_DESC 或 MCDNN_ATTR_OPERATION_RNG_SEED 的所有维度和步幅设置为 1。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.24 MCDNN_BACKEND_OPERATION_SIGNAL_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_OPERATION_SIGNAL_DESCRIPTOR, &desc) 创建的 mcDNN 后端信号操作描述符，指定一个操作节点，用于更新或等待一个标记变量。信令操作可用于 mcDNN 操作图之间的通信，即使是另一个 GPU 中的操作图。

此操作要连接到图形中的其他节点，还有一个透传输入张量，不在此张量上执行操作，仅将其传递到输出张量。这种强制的透传输入张量有助于确定应在哪个前驱节点之后执行信号操作。可选的输出张量有助于确定应在哪个后继节点之前完成信号操作执行。对于作为输出张量的非虚拟张量，也保证在操作更新信号值之前，张量的所有写入都已发生。

属性

MCDNN_ATTR_OPERATION_SIGNAL_MODE

要使用的信令模式。

- MCDNN_TYPE_SIGNAL_MODE;
- 必要属性

MCDNN_ATTR_OPERATION_SIGNAL_FLAGDESC

标记张量描述符。

MCDNN_ATTR_OPERATION_RESAMPLE_FWD_YDESC

输出张量描述符。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_SIGNAL_VALUE

用于比较或更新 flag 变量的标量值。

- MCDNN_TYPE_INT64
- 必要属性

MCDNN_ATTR_OPERATION_SIGNAL_XDESC

一个透传输入张量，用于将此信号操作连接到图形中的其他节点。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 必要属性

MCDNN_ATTR_OPERATION_SIGNAL_YDESC

透传输入张量的输出张量。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_TENSOR_DESCRIPTOR 的一个元素。
- 可选属性。

终结

在终结阶段，对属性进行交叉检查，以确保没有冲突。可能会返回以下状态：

MCDNN_STATUS_BAD_PARAM

遇到无效或不一致的属性值。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.25 MCDNN_BACKEND_OPERATIONGRAPH_DESCRIPTOR

使用描述符类型 MCDNN_BACKEND_OPERATIONGRAPH_DESCRIPTOR 创建的 mcDNN 后端操作图描述符，描述一个操作图，即通过虚拟张量连接的一个或多个操作的小型网络。操作图定义用户想要计算的用例或数学表达式。

属性

mcDNN 后端卷积描述符的属性是 mcdnnBackendAttributeName_t 枚举类型的值（前缀为 MCDNN_ATTR_OPERATIONGRAPH_）：

MCDNN_ATTR_OPERATIONGRAPH_HANDLE

mcDNN 句柄。

- MCDNN_TYPE_HANDLE；一个元素。
- 必要属性

MCDNN_ATTR_OPERATIONGRAPH_OPS

形成操作图的操作节点。

- MCDNN_TYPE_BACKEND_DESCRIPTOR;描述符类型 MCDNN_BACKEND_OPERATION_*_DESCRIPTOR() 的一个或多个元素。
- 必要属性

MCDNN_ATTR_OPERATIONGRAPH_ENGINE_GLOBAL_COUNT

支持操作图的引擎数。

- MCDNN_TYPE_INT64；一个元素。
- 只读属性。

MCDNN_ATTR_OPERATIONGRAPH_ENGINE_SUPPORTED_COUNT

支持操作图的引擎数。

- MCDNN_TYPE_INT64；一个元素。
- 只读属性；仅为占位符：当前不支持。

终结

MCDNN_STATUS_BAD_PARAM

遇到无效的属性值。例如：

- MCDNN_ATTR_OPERATIONGRAPH_OPS 中的一个后端描述符尚未终结。
- MCDNN_ATTR_OPERATIONGRAPH_HANDLE 值不是一个有效的 mcDNN 句柄。

MCDNN_STATUS_NOT_SUPPORTED

遇到不支持的属性值。例如：

- 不支持属性 MCDNN_ATTR_OPERATIONGRAPH_OPS 的操作组合。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.26 MCDNN_BACKEND_POINTWISE_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_POINTWISE_DESCRIPTOR, &desc) 创建的 mcDNN 后端逐点描述符，指定逐点运算符的参数，例如模式，数学精度，NaN 传播等。

属性

mcDNN 后端卷积描述符的属性是 mcdnnBackendAttributeName_t 枚举类型的值（前缀为 MCDNN_ATTR_POINTWISE_）：

MCDNN_ATTR_POINTWISE_MODE

逐点操作的模式。

- MCDNN_TYPE_POINTWISE_MODE；一个元素。

- 必要属性

MCDNN_ATTR_POINTWISE_MATH_PREC

计算的数学精度。

- MCDNN_TYPE_DATA_TYPE；一个元素。
- 必要属性

MCDNN_ATTR_POINTWISE_NAN_PROPAGATION

指定 NaN 传播的方式。

- MCDNN_TYPE_NAN_PROPOGATION；一个元素。
- 仅基于比较的逐点模式（如 ReLU）需要。
- 当前仅支持枚举值 MCDNN_PROPAGATE_NAN。
- 默认值：MCDNN_NOT_PROPAGATE_NAN.

MCDNN_ATTR_POINTWISE_RELU_LOWER_CLIP

设置 ReLU 的下限裁剪值。(value < lower_clip) value = lower_clip + lower_clip_slope * (value - lower_clip)；

- MCDNN_TYPE_DOUBLE 或 MCDNN_TYPE_FLOAT；一个元素。
- 默认值：0.0f.

MCDNN_ATTR_POINTWISE_RELU_UPPER_CLIP

设置 Relu 的上限裁剪值。(value > upper_clip) value = upper_clip；

- MCDNN_TYPE_DOUBLE 或 MCDNN_TYPE_FLOAT；一个元素。
- 默认值：最大数值限制。

MCDNN_ATTR_POINTWISE_RELU_LOWER_CLIP_SLOPE

设置 ReLU 的下限裁剪斜率值。(value < lower_clip) value = lower_clip + lower_clip_slope * (value - lower_clip)；

- MCDNN_TYPE_DOUBLE / MCDNN_TYPE_FLOAT; one element.
- 默认值：0.0f.

MCDNN_ATTR_POINTWISE_ELU_ALPHA

设置 ELU 的 alpha 值。(value < 0.0) value = alpha * (e^value - 1.0)；

- MCDNN_TYPE_DOUBLE 或 MCDNN_TYPE_FLOAT；一个元素。
- 默认值：1.0f.

MCDNN_ATTR_POINTWISE_SOFTPLUS_BETA

设置 softplus 的 beta 值。value = log (1 + e^(beta * value)) / beta

- MCDNN_TYPE_DOUBLE 或 MCDNN_TYPE_FLOAT；一个元素。
- 默认值：1.0f

MCDNN_ATTR_POINTWISE_SWISH_BETA

设置 swish 的 beta 值。value = value / (1 + e^(-beta * value))

- MCDNN_TYPE_DOUBLE 或 MCDNN_TYPE_FLOAT；一个元素。
- 默认值：1.0f.

MCDNN_ATTR_POINTWISE_AXIS

设置 GEN_INDEX 的轴值。将为此轴生成索引。

- MCDNN_TYPE_INT64；一个元素。

- 默认值：-1.
- 需要介于 [0,input_dim_size-1] 之间。例如，如果输入维度为 [N,C,H,W]，则轴可以设置为 [0,3] 中的任意值。

终结

带有 MCDNN_BACKEND_POINTWISE_DESCRIPTOR 的 mcdnnBackendFinalize() 可以具有以下返回值：

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.27 MCDNN_BACKEND_REDUCTION_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_REDUCTION_DESCRIPTOR, &desc) 创建的 mcDNN 后端归约描述符，指定归约操作所需的任意元数据，包括数学运算和计算数据类型。

属性

MCDNN_ATTR_REDUCTION_OPERATOR

用于归约操作的数学运算。

- MCDNN_TYPE_REDUCTION_OPERATOR_TYPE；一个元素。
- 必要属性

MCDNN_ATTR_REDUCTION_COMP_TYPE

用于归约操作的计算精度。

- MCDNN_TYPE_DATA_TYPE；一个元素。
- 必要属性

终结

返回 mcdnnBackendFinalize(desc) 的值，其中 desc 是 MCDNN_BACKEND_REDUCTION_DESCRIPTOR：

MCDNN_STATUS_NOT_SUPPORTED

遇到不支持的属性值。可能的原因包括：

- MCDNN_ATTR_REDUCTION_OPERATOR 不可以设置为

```
MCDNN_REDUCE_TENSOR_ADD, MCDNN_REDUCE_TENSOR_MUL, MCDNN_
↪REDUCE_TENSOR_MIN,MCDNN_REDUCE_TENSOR_MAX
```

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.28 MCDNN_BACKEND_RESAMPLE_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_RESAMPLE_DESCRIPTOR, &desc) 创建的 mcDNN 后端重采样描述符，指定正向和反向传播中重采样操作 (上采样或下采样) 的参数。

属性

MCDNN_ATTR_RESAMPLE_MODE

指定重采样模式，例如，平均池，最近邻等。

- MCDNN_TYPE_RESAMPLE_MODE；一个元素。
- 默认值为 MCDNN_RESAMPLE_NEAREST。

MCDNN_ATTR_RESAMPLE_COMP_TYPE

重采样运算符的计算数据类型。

- MCDNN_TYPE_DATA_TYPE；一个元素。
- 默认值为 MCDNN_DATA_FLOAT。

MCDNN_ATTR_RESAMPLE_NAN_PROPAGATION

指定 NaN 传播的方式。

- MCDNN_TYPE_NAN_PROPAGATION；一个元素。
- 默认值为 MCDNN_NOT_PROPAGATE_NAN。

MCDNN_ATTR_RESAMPLE_SPATIAL_DIMS

指定要执行重采样的空间维数。

- MCDNN_TYPE_INT64；一个元素。
- 必要属性

MCDNN_ATTR_RESAMPLE_PADDING_MODE

指定用于填充的值。

- MCDNN_TYPE_PADDING_MODE；一个元素。
- 默认值为 MCDNN_ZERO_PAD。

MCDNN_ATTR_RESAMPLE_STRIDES

内核/卷积核的每个维度的步幅。

- MCDNN_TYPE_INT64 或 MCDNN_TYPE_FRACTION；至多为 MCDNN_MAX_DIMS - 2.
- 必要属性

MCDNN_ATTR_RESAMPLE_PRE_PADDINGS

添加到每个维度中输入张量起始处的填充。

- MCDNN_TYPE_INT64 或 MCDNN_TYPE_FRACTION；至多为 MCDNN_MAX_DIMS - 2.
- 必要属性

MCDNN_ATTR_RESAMPLE_POST_PADDINGS

添加到每个维度中输入张量结尾处的填充。

- MCDNN_TYPE_INT64 或 MCDNN_TYPE_FRACTION；至多为 MCDNN_MAX_DIMS - 2.
- 必要属性

MCDNN_ATTR_RESAMPLE_WINDOW_DIMS

卷积核的空间维度。

- MCDNN_TYPE_INT64 或 MCDNN_TYPE_FRACTION；至多为 MCDNN_MAX_DIMS - 2.
- 必要属性

终结

使用 MCDNN_BACKEND_RESAMPLE_DESCRIPTOR 调用 mcdnnBackendFinalize() 时，返回值为：

MCDNN_STATUS_NOT_SUPPORTED

遇到不支持的属性值。可能的原因包括：

- 用于设置 MCDNN_ATTR_RESAMPLE_WINDOW_DIMS、MCDNN_ATTR_RESAMPLE_STRIDES、MCDNN_ATTR_RESAMPLE_PRE_PADDINGS 和 MCDNN_ATTR_RESAMPLE_POST_PADDINGS 的 elementCount 参数，与 MCDNN_ATTR_RESAMPLE_SPATIAL_DIMS 的设置值不相等。

- MCDNN_ATTR_RESAMPLE_MODE 设置为 MCDNN_RESAMPLE_BILINEAR，且没有 MCDNN_ATTR_RESAMPLE_WINDOW_DIMS 设置为 2。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.29 MCDNN_BACKEND_RNG_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_RNG_DESCRIPTOR, &desc) 创建的 mcDNN 后端 Rng 描述符，指定任意元数据，包括将用于生成张量的概率分布以及分布的相应参数。

属性

MCDNN_ATTR_RNG_DISTRIBUTION

用于 RNG 操作的概率分布。

- MCDNN_TYPE_RNG_DISTRIBUTION；一个元素。
- 默认值为 MCDNN_RNG_DISTRIBUTION_BERNOULLI。

MCDNN_ATTR_RNG_NORMAL_DIST_MEAN

正态分布的均值，在 MCDNN_ATTR_RNG_DISTRIBUTION = MCDNN_RNG_DISTRIBUTION_NORMAL 时使用。

- MCDNN_TYPE_DOUBLE；一个元素。
- 默认值为-1。

MCDNN_ATTR_RNG_NORMAL_DIST_STANDARD_DEVIATION

正态分布的标准偏差值，在 MCDNN_ATTR_RNG_DISTRIBUTION = MCDNN_RNG_DISTRIBUTION_NORMAL 时使用。

- MCDNN_TYPE_DOUBLE；一个元素。
- 默认值为-1。

MCDNN_ATTR_RNG_UNIFORM_DIST_MAXIMUM

在均匀分布中使用的范围最大值，在 MCDNN_ATTR_RNG_DISTRIBUTION = MCDNN_RNG_DISTRIBUTION_UNIFORM 时使用。

- MCDNN_TYPE_DOUBLE；一个元素。
- 默认值为-1。

MCDNN_ATTR_RNG_UNIFORM_DIST_MINIMUM

在均匀分布中使用的范围最小值，在 MCDNN_ATTR_RNG_DISTRIBUTION = MCDNN_RNG_DISTRIBUTION_UNIFORM 时使用。

- MCDNN_TYPE_DOUBLE；一个元素。
- 默认值为-1。

MCDNN_ATTR_RNG_BERNOULLI_DIST_PROBABILITY

在张量中生成 1 的概率，在 MCDNN_ATTR_RNG_DISTRIBUTION = MCDNN_RNG_DISTRIBUTION_BERNOULLI 时使用。

- MCDNN_TYPE_DOUBLE；一个元素。
- 默认值为-1。

终结

返回 mcdnnBackendFinalize(desc) 的值，其中 desc 是 MCDNN_BACKEND_RNG_DESCRIPTOR：

MCDNN_STATUS_BAD_PARAM

遇到无效的属性值。例如：

- 如果 MCDNN_ATTR_RNG_DISTRIBUTION = MCDNN_RNG_DISTRIBUTION_NORMAL 且提供的标准偏差为负值。
- 如果 MCDNN_ATTR_RNG_DISTRIBUTION = MCDNN_RNG_DISTRIBUTION_UNIFORM 且该范围的最大值小于最小值。
- 如果 MCDNN_ATTR_RNG_DISTRIBUTION = MCDNN_RNG_DISTRIBUTION_BERNOULLI 且提供的概率为负值。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.30 MCDNN_BACKEND_TENSOR_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_TENSOR_DESCRIPTOR, &desc) 创建的 mcDNN 后端张量，允许用户指定通用张量的内存存储。张量由唯一标识符标识，并通过其数据类型，数据字节对齐要求以及其维度的范围和步幅来描述。或者，张量元素可以是其任一维度中的向量。当张量是计算图中的中间变量且未映射到物理全局内存存储时，也可以将其设置为虚拟张量。

属性

mcDNN 后端张量描述符的属性是 mcdnnBackendAttributeName_t 枚举类型的值（前缀为 MCDNN_ATTR_TENSOR_）：

MCDNN_ATTR_TENSOR_UNIQUE_ID

用于标识张量的唯一整数。

- MCDNN_TYPE_INT64；一个元素。
- 必要属性

MCDNN_ATTR_TENSOR_DATA_TYPE

张量的数据类型。

- MCDNN_TYPE_DATA_TYPE；一个元素。
- 必要属性

MCDNN_ATTR_TENSOR_BYTE_ALIGNMENT

此张量的指针字节对齐。

- MCDNN_TYPE_INT64；一个元素。
- 必要属性

MCDNN_ATTR_TENSOR_DIMENSIONS

张量维度。

- MCDNN_TYPE_INT64；至多有 MCDNN_MAX_DIMS 个元素。
- 必要属性

MCDNN_ATTR_TENSOR_STRIDES

张量步幅。

- MCDNN_TYPE_INT64；至多有 MCDNN_MAX_DIMS 个元素。
- 必要属性

MCDNN_ATTR_TENSOR_VECTOR_COUNT

向量化大小。

- MCDNN_TYPE_INT64；一个元素。

- 默认值：1

MCDNN_ATTR_TENSOR_VECTORIZED_DIMENSION

向量化维度的索引。

- MCDNN_TYPE_INT64；一个元素。
- 如果 MCDNN_ATTR_TENSOR_VECTOR_COUNT 不是默认值，则需要在终结前设置该属性；否则忽略。

MCDNN_ATTR_TENSOR_IS_VIRTUAL

表示张量是否为虚拟张量。虚拟张量是操作图中的中间张量，以瞬态形式存在，不能从全局设备内存中读取或写入。

- MCDNN_TYPE_BOOL；一个元素。
- 默认值：false

终结

带有 MCDNN_BACKEND_CONVOLUTION_DESCRIPTOR 的 mcdnnBackendFinalize() 可以具有以下返回值：

MCDNN_STATUS_BAD_PARAM

遇到无效的属性值。例如：

- 任意张量维度或步幅都不是正值。
- 张量对齐属性的值不能被数据类型的大小整除。

MCDNN_STATUS_NOT_SUPPORTED

遇到不支持的属性值。例如：

- 数据类型属性是 MCDNN_DATA_INT8x4，MCDNN_DATA_UINT8x4 或 MCDNN_DATA_INT8x32。
- 数据类型属性为 MCDNN_DATA_INT8，且 MCDNN_ATTR_TENSOR_VECTOR_COUNT 值不是 1，4 或 32。

MCDNN_STATUS_SUCCESS

描述符已成功终结。

8.2.2.31 MCDNN_BACKEND_VARIANT_PACK_DESCRIPTOR

使用 mcdnnBackendCreateDescriptor(MCDNN_BACKEND_VARIANT_PACK_DESCRIPTOR, &desc) 创建的 mcDNN 后端 variant pack 计划，允许用户设置指向设备缓冲区的指针，该缓冲区保存操作图，工作空间和计算中间结果的各种非虚拟张量（由唯一标识符标识）。

属性

MCDNN_ATTR_VARIANT_PACK_UNIQUE_IDS

每个数据指针张量的唯一标识符。

- MCDNN_TYPE_INT64；零个或多个元素。
- 必要属性

MCDNN_ATTR_VARIANT_PACK_DATA_POINTERS

张量数据设备指针。

- MCDNN_TYPE_VOID_PTR；零个或多个元素。
- 必要属性

MCDNN_ATTR_VARIANT_PACK_INTERMEDIATES

中间设备指针。

- MCDNN_TYPE_VOID_PTR；零个或多个元素。
- 不支持设置的属性。用于将来实现的占位符。

MCDNN_ATTR_VARIANT_PACK_WORKSPACE

设备指针的工作空间。

- MCDNN_TYPE_VOID_PTR；一个元素。
- 必要属性

终结

使用 mcDNN 后端 variant pack 描述符调用 mcdnnBackendFinalize() 时，返回值为：

MCDNN_STATUS_SUCCESS

描述符已成功完成。