ICS 35.040
CCS L 71

团　　体　　标　　准

T/AI 131.2—2025

人工智能 算子接口
第 2 部分：神经网络类

Artificial intelligence — Operater interface—Part 2: Neural network operators

2025 - 04 - 27 发布　　2025 - 04 - 27 实施

中关村视听产业技术创新联盟　　发 布

目　　次

前　言

本文件按照GB/T 1.1-2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

本文件为T/AI 131《人工智能 算子接口》的第2部分。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由新一代人工智能产业技术创新战略联盟AI标准工作组提出。

本文件由中关村视听产业技术创新联盟归口。

本文件起草单位：北京大学、北京大学长沙计算与数字经济研究院、中国电子技术标准化研究院、鹏城实验室、中国科学院软件研究所、北京百度网讯科技有限公司、华为技术有限公司、长沙壹零壹壹科技有限公司、中科寒武纪科技股份有限公司、上海商汤智能科技有限公司、浪潮电子信息产业股份有限公司、北京交通大学。

本文件主要起草人：杨超、陈军、胡晓光、勾海鹏、马艳军、范睿博、李克森、段炼、鲍薇、杨雨泽、贾梦珠、李敏、黎子毅、于佃海、关贺、胡帅、赵海英、高翔、马珊珊、高铁柱、吴庚、王丽、郑若琳、唐轶男、李卉、宋文林、栾晓旭、陈恺、刘文枫、陈德良、王新民、沈芷月、高歌、刘伟、聂简荻、杨征、冯海军、汪群博。

引　言

近年来，我国人工智能行业发展呈现繁荣态势，相关软件和硬件均向多元化发展，云服务器、边缘设备、终端设备等不同类型的处理器层出不穷，与此同时，各类计算库、中间表示工具以及编程框架也呈现百花齐放的态势。AI软硬件的极大丰富，为应用的高效部署提供了诸多便利，但同时也带来了多样化、复杂化、碎片化的挑战。一方面，AI软件从业者需要考虑软件与各种不同主流人工智能处理器之间的适配，花费大量精力提升软件的可移植性，另一方面，每一款AI硬件的研发，都必须从底层对常用人工智能软件提供支持，否则难以融入现有的软件生态。这种M*N级别的软硬件映射关系，已经逐渐发展成桎梏人工智能应用发展的一大障碍。

人工智能算子是构建人工智能应用的基础运算，是相关硬件操作的封装，人工智能软件通过调用算子接口来使用硬件资源完成计算，算子接口是人工智能软硬件衔接的桥梁。制定T/AI 131《人工智能算子接口》，是对人工智能算子的核心数据结构、功能和接口参数的规范化和标准化，是降低人工智能软硬件适配难度、促进生态建设的基础性工作。

T/AI 131《人工智能 算子接口》拟由五个部分构成。

——第1部分：基础数学类。目的在于确立适用于人工智能算子接口的总则与核心数据结构，以及规范基础数学类算子接口的基本功能和参数的要求。

——第2部分：神经网络类。目的在于规范神经网络类算子的基本功能和参数的要求。

——第3部分：机器学习类。目的在于规范机器学习类算子的基本功能和参数的要求。

——第4部分：大模型类算子。目的在于规范大模型类算子的基本功能和参数的要求。

——第5部分：自动化测试框架。目的在于为算子接口提供自动化测试方法和参考实现，验证算子开发标准符合性。

人工智能　算子接口 第2部分：神经网络类

1　范围

本文件规定了面向人工智能领域的神经网络类算子接口的基本功能及参数要求。

本文件适用于人工智能神经网络类算子库的设计、开发与应用，以及相关软硬件及系统的研制。

2　规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 41867-2022 信息技术　人工智能　术语

3　术语和定义

GB/T 41867-2022、T/AI 131.1-2025中界定的以及下列术语和定义适用于本文件。

3.1

神经网络　neural network

由一层或多层神经元组成的网络，通过权值可调的加权连接，接收输入数据并产生输出。

[来源：GB/T 41867-2022，3.2.26]

3.2

神经网络模型　neural-network model

神经网络的抽象模型，它能用软件来模拟或作为神经计算机加以实现。

3.3

递归神经网络　recursive neural network

具有树状层次结构，网络节点按其连接顺序对输入信息进行递归的人工神经网络。

3.4

模型推理　machine learning inference

采用训练好的深度神经网络或概率统计模型处理数据，获取预测结果的过程。

3.5

过拟合　overfitting

创建的模型由于学习到了训练数据中和与任务无关的部分而无法泛化新数据。

4 缩略语

下列缩略语适用于本文件：

GRU:门控递归单元网络(Gated Recurrent Unit)

LSTM:长短期记忆网络(Long Short-Term Memory)

RNN:递归神经网络(Recursive Neural Network)

SGD:随机梯度下降（Stochastic Gradient Descent）

5 总则

本章内容应符合T/AI 131.1—20255《人工智能 算子接口 第1部分：基础数学类》中第5章要求。

6 数据结构

本章内容应符合T/AI 131.1—《人工智能 算子接口 第1部分：基础数学类》中第6章要求。

7 神经网络类算子接口

7.1 接口列表

神经网络类算子接口列表见表1。

表 1 神经网络类算子接口列表

类别	名称
激活函数	S型函数，对数S型函数，分段线性近似S型函数，归一化指数函数，对数归一化指数函数，线性整流单元，带阈值的线性整流单元，指数线性单元，带泄漏线性整流单元，带参数线性整流单元，扩展指数线性单元，双边整流线性单元，高斯误差线性单元，Softplus函数，Softsign函数，Swish函数，HardSwish函数，误差函数，HardShrink函数，TanhShrink函数，HardTanh函数
损失函数	L1损失函数，均方误差损失函数，交叉熵损失函数， 稀疏交叉熵损失函数，负对数损失函数，负对数似然损失函数， CTC损失函数，平滑L1损失函数，KL散度损失函数，软间隔损失函数，间隔排序损失函数
正则函数	随机失活函数，标签平滑函数
归一化函数	批量归一化操作，分组归一化操作，层归一化操作，实例归一化操作，局部相应归一化操作，L2归一化操作，Lp范数归一化操作，权重归一化操作，谱归一化操作

表 1　神经网络类算子接口列表（续）

池化函数	一维池化操作，二维池化操作，三维池化操作，自适应一维池化操作， 自适应二维池化操作，自适应三维池化操作
卷积函数	一维卷积操作，二维卷积操作，三维卷积操作，一维反卷积操作 二维反卷积操作，三维反卷积操作
评估函数	准确率函数，AUC函数
循环网络函数	简单循环网络基本单元，简单循环网络， 长短期记忆网络基本单元，长短期记忆网络， 门控循环单元网络基本，　门控循环单元网络
编码操作	词嵌入操作，独热编码
距离函数	余弦相似度
视觉函数	网格插值采样，仿射网格，像素重排
优化器	SGD优化器，Momentum优化器，AdaGrad优化器，AdaDelta优化器， RMSProp优化器，CenteredRMSProp优化器，Adam优化器，AdaMax优化器

7.2　接口操作和参数

7.2.1　激活函数

7.2.1.1　S 型函数

7.2.1.1.1　功能

计算输入张量的sigmoid值，见式（1）。

$$y_i = 1/(1+\exp(-x_i)) \quad \cdots\cdots (1)$$

式中：

x--表示输入张量；

y--表示输出张量；

$exp(*)$—表示以自然常数e为底的指数函数。

7.2.1.1.2　前向接口参数

S型函数前向接口应符合表2，C语言示例见A.2.1.1。

表 2　S 型函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
输出张量	输出	必选	表示计算结果

7.2.1.1.3　前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.1.4　后向接口参数

S型函数后向接口应符合表3，C语言示例见A.2.1.1。

表 3　S 型函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.1.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.2　对数 S 型函数

7.2.1.2.1　功能

对数S型激活函数。计算输入张量每个元素的log sigmoid值，见式（2）。

$$\log\_sigmoid(x) = \log\frac{1}{1+e^{-x}} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (2)$$

式中：
x--表示输入张量；
$log\_sigmoid(x)$--表示输出张量,其中张量中每个元素表示对应输入元素的sigmoid值；
$log(*)$—表示以10为底的对数。

7.2.1.2.2　前向接口参数

对数S型函数前向接口应符合表4，C语言示例见A.2.1.2。

表 4　对数 S 型函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
输出张量	输出	必选	表示计算结果

7.2.1.2.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.2.4　后向接口参数

对数S型函数后向接口应符合表5，C语言示例见A.2.1.2。

表 5　对数 S 型函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度

表 5　对数 S 型函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示前向接口中的输入张量
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.2.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.3　分段线性近似 S 型函数

7.2.1.3.1　功能

sigmoid的分段线性逼近激活函数。计算输入张量每个元素的hard_sigmoid值，见式（3）。

$$y_i = \max(0, \min(1, slope * x_i + offset)) \quad (3)$$

式中：
x--表示输入张量；
y--表示输出张量；
$slope$—表示斜率；
offset—表示偏移量；
max—表示取两个数之间的最大值min—表示取两个数之间的最小值。

7.2.1.3.2　前向接口参数

分段线性近似S型函数前向接口应符合表6，C语言示例见A.2.1.3。

表 6　分段线性近似 S 型函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
斜率	输入	可选	公式中的slope，必须为正数，缺省值可为0.2
偏移量	输入	可选	公式中的offset，缺省值可为0.5
输出张量	输出	必选	表示计算结果

7.2.1.3.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：其他参数不合法。

7.2.1.3.4　后向接口参数

分段线性近似S型函数后向接口应符合表7，C语言示例见A.2.1.3。

表 7　分段线性近似 S 型函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度

表 7　分段线性近似 S 型函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示前向接口中的输入张量
斜率	输入	可选	公式中的slope，必须为正数，缺省值可为0.2
偏移量	输入	可选	公式中的offset，缺省值可为0.5
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.3.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：其他参数不合法。

7.2.1.4　归一化指数函数

7.2.1.4.1　功能

沿着输入张量的计算轴 axis 计算输入张量的 softmax 值，见式（4）。

$$y_i = \frac{\exp(x_i)}{\sum_j \exp(x_i)} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (4)$$

式中：
x--表示输入张量；
y--表示输出张量；
$exp(*)$—表示以自然常数e为底的指数函数。

7.2.1.4.2　前向接口参数

归一化指数函数前向接口应符合表8，C语言示例见A.2.1.4。

表 8　归一化指数函数前向接口参数

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
计算轴	输入	可选	softmax操作所沿着的轴axis。 $-1 \leq axis \leq ndim - 1$。$axis = -1$表示对所有输入张量元素进行softmax操作，默认值可为-1
输出张量	输出	必选	表示计算结果

7.2.1.4.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
超出范围：axis超出输入张量维度。

7.2.1.4.4　后向接口参数

归一化指数函数后向接口应符合表9，C语言示例见A.2.1.4

表 9　归一化指数函数后向接口参数

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输出张量	输入	必选	表示前向接口中的输出张量
计算轴	输入	可选	softmax操作所沿着的轴axis $-1 \leq axis \leq ndim-1$。$axis=-1$表示对所有输入张量元素进行softmax操作，默认值可为-1
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.4.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
超出范围：axis超出输入张量维度。

7.2.1.5　对数归一化指数函数

7.2.1.5.1　功能

沿着输入张量axis轴计算输入张量的log softmax值，见式（5）。

$$\text{output}_i = \ln\left(\frac{\exp(x_i)}{\sum_j \exp(x_i)}\right) \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (5)$$

式中：
x--表示输入张量；
$output_i$--表示输出张量元素；
$exp(*)$—表示以自然常数e为底的指数函数；
$\ln(*)$—表示以自然常数e为底的对数。

7.2.1.5.2　前向接口参数

对数归一化指数函数前向接口应符合表10，C语言示例见A.2.1.5。

表 10　对数归一化指数函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
计算轴	输入	可选	log softmax操作所沿着的轴axis。 $-1 \leq axis \leq ndim-1$。$axis=-1$表示对所有输入张量元素进行log softmax操作，默认值可为-1
输出张量	输出	必选	表示计算结果

7.2.1.5.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
超出范围：axis超出输入张量维度。

7.2.1.5.4 后向接口参数

对数归一化指数函数后向接口应符合表11，C语言示例见A.2.1.5。

表 11 对数归一化指数函数后向接口参数列表

参数名	类型	可选/必选	描述
输出梯度	输入	必选	表示输出张量的梯度
输出张量	输入	必选	表示前向接口中的输出张量
计算轴	输入	可选	log softmax操作所沿着的轴axis $-1 \leq axis \leq ndim - 1$。$axis = -1$表示对所有输入张量元素进行log softmax操作，默认值可为-1
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.5.5 后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
超出范围：axis超出输入张量维度。

7.2.1.6 线性整流单元

7.2.1.6.1 功能

计算输入张量每个元素的ReLU(Rectified Linear Unit)值，见式（6）。

$$y_i = \max(0, x_i) \quad \text{（6）}$$

式中：
x--表示输入张量；
y--表示输出张量；
max—表示取最大值。

7.2.1.6.2 前向接口参数

线性整流单元函数前向接口应符合表12，C语言示例见A.2.1.6。

表 12 线性整流单元函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
输出张量	输出	必选	表示计算结果

7.2.1.6.3 前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.6.4 后向接口参数

线性整流单元函数后向接口应符合表13，C语言示例见A.2.1.6。

表 13　线性整流单元函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.6.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.7　带阈值的线性整流单元

7.2.1.7.1　功能

计算输入张量每个元素的thresholded relu值，见式（7）。

$$y_i = x_i > \text{threshold}\ ?\ x_i : 0 \quad \text{（7）}$$

式中：
x--表示输入张量；
y--表示输出张量；
thresholded—表示阈值。

7.2.1.7.2　前向接口参数

带阈值的线性整流单元函数前向接口应符合表14，C语言示例见A.2.1.7。

表 14　带阈值的线性整流单元函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
阈值	输入	可选	公式中的threshold，默认值可为1.0
输出张量	输出	必选	表示计算结果

7.2.1.7.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.7.4　后向接口返回值

带阈值的线性整流单元函数后向接口应符合表15，C语言示例见A.2.1.7。

表 15　带阈值的线性整流单元函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
阈值	输入	可选	公式中的threshold，默认值可为1.0
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.7.5 后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.8 指数线性单元

7.2.1.8.1 功能

计算输入张量每个元素的ELU(Exponential Linear Unit)值，见式（8）。

$$y_i = \max(0, x_i) + \min(0, \alpha * (\exp(x_i) - 1)) \quad \text{（8）}$$

式中：
x--表示输入张量；
y--表示输出张量；
max—表示取两个数之间的最大值；
min—表示取两个数之间的最小值；
$exp(*)$—表示以自然常数e为底的指数函数。

7.2.1.8.2 前向接口参数

指数线性单元函数前向接口应符合表16，C语言示例见A.2.1.8。

表 16 指数线性单元函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
α	输入	可选	公式中的α值，默认值可为1.0
输出张量	输出	必选	表示计算结果

7.2.1.8.3 前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.8.4 后向接口参数

指数线性单元函数后向接口应符合表17，C语言示例见A.2.1.8。

表 17 指数线性单元函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
α	输入	可选	公式中的α值，默认值可为1.0
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.8.5 后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.9 带泄露线性整流单元

7.2.1.9.1 功能

计算输入张量每个元素的leaky ReLU值，见式（9）。

$$y_i = \max(0, x_i) + \alpha * \min(0, x_i) \quad \text{（9）}$$

式中：

x--表示输入张量；

y--表示输出张量；

max—表示取两个数之间的最大值；

min—表示取两个数之间的最小值；

a—表示Leaky ReLU中的调整因子，默认值可为0.01。

7.2.1.9.2 前向接口参数

带泄露线性整流单元函数前向接口应符合表18，C语言示例见A.2.1.9。

表 18 带泄露线性整流单元函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
α	输入	可选	公式中的α值，默认值可为0.01
输出张量	输出	必选	表示计算结果

7.2.1.9.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.9.4 后向接口参数

带泄露线性整流单元函数后向接口应符合表19，C语言示例见A.2.1.9。

表 19 带泄露线性整流单元函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
α	输入	可选	公式中的α值，默认值可为0.01
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.9.5 后向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.10 带参数线性整流单元

7.2.1.10.1 功能

计算输入张量每个元素的PReLU(Parametric ReLU)值，见式（10）。

$$y_i = \max(0, x_i) + \alpha * \min(0, x_i) \quad (10)$$

式中：

x--表示输入张量；

y--表示输出张量；

max—表示取两个数之间的最大值；

min—表示取两个数之间的最小值；

a—表示PReLU中的调整因子。

7.2.1.10.2 前向接口参数

带参数线性整流单元函数前向接口应符合表20，C语言示例见A.2.1.10。

表 20 带参数线性整流单元函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
alpha数组首地址	输入	可选	表示公式中α值的数组。若输入有channel（默认为第2维度），并且希望每个channel拥有单独的α值，则需要设置长度与channel数相同的alpha_array数组；否则，应设置长度为1的alpha_array数组。默认值可为0.25
alpha数组长度	输入	可选	表示公式中α数组的长度，默认值可为1
输出张量	输出	必选	表示计算结果

7.2.1.10.3 前向接口返回值

没有错误：操作成功。

其他内部错误：内部调用操作出错。

非法参数：alpha 数组长度不合法。

7.2.1.10.4 后向接口参数

带参数线性整流单元函数后向接口应符合表21，C语言示例见A.2.1.10。

表 21 带参数线性整流单元函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
alpha数组首地址	输入	可选	表示公式中α值的数组，默认值可为0.25
alpha数组长度	输入	可选	表示公式中α数组的长度，默认值可为1
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.10.5 后向接口返回值

没有错误：操作成功。

其他内部错误：内部调用操作出错。

非法参数：axis 超出输入张量维度。

7.2.1.11 扩展指数线性单元

7.2.1.11.1 功能

计算输入张量每个元素的SELU(Scaled Exponential Linear Units)值，见式（11）。

$$y_i = \lambda * (x_i > 0 \ ? \ x_i : \alpha * (\exp(x_i) - 1)) \quad (11)$$

式中：

x--表示输入张量；

y--表示输出张量；

λ—表示缩放因子；

$exp(*)$—表示以自然常数e为底的指数函数；

a—表示SELU公式中的α值。

7.2.1.11.2 前向接口参数

扩展指数线性单元函数前向接口应符合表22，C语言示例见A.2.1.11。

表 22 扩展指数线性单元函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
缩放因子	输入	可选	SELU公式中的λ值，缺省值可为1.05070098
alpha数组	输入	可选	SELU公式中的α值，缺省值可为1.67326324
输出张量	输出	必选	表示计算结果

7.2.1.11.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.11.4 后向接口参数

扩展指数线性单元后向接口应符合表23，C语言示例见A.2.1.11。

表 23 扩展指数线性单元后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
缩放因子	输入	可选	SELU公式中的λ，缺省值可为1.05070098
alpha数组	输入	可选	SELU公式中的α值，缺省值可为1.67326324
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.11.5 后向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.12 双边整流线性单元

7.2.1.12.1 功能

计算输入张量每个元素的Bilateral ReLU值，见式（12）。

$$y_i = \min(\max(x_i, \text{low}), \text{high}) \quad \text{（12）}$$

式中：

x--表示输入张量；

y--表示输出张量；

max—表示取两个数之间的最大值；

min—表示取两个数之间的最小值；

low—表示BReLU结果的最小值；

high—表示BReLU结果的最大值。

7.2.1.12.2 前向接口参数

双边整流线性单元函数前向接口应符合表24，C语言示例见A.2.1.12。

表 24 双边整流线性单元函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
最小值	输入	可选	BReLU结果的最小值，缺省值可为0.0
最大值	输入	可选	BReLU结果的最大值，缺省值可为24.0
输出张量	输出	必选	表示计算结果

7.2.1.12.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.12.4 后向接口参数

双边整流线性单元函数后向接口应符合表25，C语言示例见A.2.1.12。

表 25 双边整流线性单元函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
最小值	输入	可选	BReLU结果的最小值，缺省值可为0.0
最大值	输入	可选	breluBReLU结果的最大值，缺省值可为24.0
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.12.5 后向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.13 高斯误差线性单元

7.2.1.13.1 功能

计算输入张量每个元素的GELU(Gaussian Error Linear Units)值，如果使用近似计算，则应符合式（13）。

$$y = 0.5 * x * (1 + \tanh(\sqrt{\frac{2}{\pi}} * (x + 0.044715 * x^3))) \quad (13)$$

式中：

x--表示输入张量；

y--表示输出张量；

$tanh$—表示双曲正切函数。

如果不使用近似计算，则应符合式（14）。

$$y = 0.5 * x * (1 + \mathrm{erf}(\frac{x}{\sqrt{2}})) \quad (14)$$

式中：

x--表示输入张量；

y--表示输出张量。

7.2.1.13.2 erf—表示误差函数。

7.2.1.13.3 前向接口参数

高斯误差线性单元前向接口应符合表26，C语言示例见A.2.1.13。

表 26 高斯误差线性单元前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
是否近似	输入	可选	布尔类型变量，表示是否使用近似计算，默认值可为否
输出张量	输出	必选	表示计算结果

7.2.1.13.4 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.13.5 后向接口参数

高斯误差线性单元后向接口应符合表27，C语言示例见A.2.1.13。

表 27 高斯误差线性单元后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
是否近似	输入	可选	布尔类型变量，表示是否使用近似计算，默认值可为否
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.13.6 后向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.14 Softplus 函数

7.2.1.14.1 功能

计算输入张量每个元素的Softplus值，见式（15）

$$y_i = \frac{1}{\beta}\ln(1+\exp(\beta x_i)) \quad (15)$$

式中：

x--表示输入张量；

y--表示输出张量；

β —表示Softplus公式中的beta参数；

$exp(*)$—表示以自然常数e为底的指数函数；

$ln(*)$—表示以自然常数e为底的对数。

7.2.1.14.2 前向接口参数

Softplus函数前向接口应符合表28，C语言示例见A.2.1.14。

表 28 Softplus 函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
beta参数	输入	可选	Softplus 公式中的beta参数。默认可为1.0
阈值	输入	可选	为了保证数值稳定性，当beta*x>阈值时，函数转变为线性函数x。默认为20
输出张量	输出	必选	表示计算结果

7.2.1.14.3 前向接口返回值

没有错误：操作成功。

其他内部错误：内部调用出错。

7.2.1.14.4 后向接口参数

Softplus函数后向接口应符合表29，C语言示例见A.2.1.14。

表 29 Softplus 函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
beta参数	输入	可选	Softplus 公式中的beta参数。默认可为1.0
阈值	输入	可选	为了保证数值稳定性，当beta*x>阈值时，函数转变为线性函数x。默认为20
输入张量梯度	输出	可选	表示输入张量的梯度

7.2.1.14.5 后向接口返回值

没有错误：操作成功。
其他内部错误：内部调用出错。

7.2.1.15 Softsign 函数

7.2.1.15.1 功能

计算输入张量每个元素的 Softsign 值，见式（16）。

$$y_i = \frac{x_i}{1+|x_i|} \quad \text{（16）}$$

式中：
x--表示输入张量；
y--表示输出张量；
$|*|$—表示对数值取绝对值。

7.2.1.15.2 前向接口参数

Softsign函数前向接口应符合表30，C语言示例见A.2.1.15。

表 30 Softsign 函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
输出张量	输出	必选	表示计算结果

7.2.1.15.3 前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.15.4 后向接口参数

Softsign函数后向接口应符合表31，C语言示例见A.2.1.15。

表 31 Softsign 函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.15.5 后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.16 Swish 函数

7.2.1.16.1 功能

计算输入张量每个元素的Swish值，见式（17）。

$$y_i = \frac{x_i}{1+\exp(-beta * x_i)} \quad (17)$$

式中：

x--表示输入张量；

y--表示输出张量；

$beta$—表示常量值，默认为1.0；

$exp(*)$—表示以自然常数e为底的指数函数。

7.2.1.16.2 前向接口参数

Swish函数前向接口应符合表32，C语言示例见A.2.1.16。

表 32 swish 函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
Beta	输入	可选	公式中的常量beta值，默认值可为1.0。
输出张量	输出	必选	表示计算结果

7.2.1.16.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.16.4 后向接口参数

Swish函数后向接口应符合表33，C语言示例见A.2.1.16。

表 33 Swish 函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
Beta	输入	可选	公式中的常量beta值，默认值可为1.0
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.16.5 后向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.17 HardSwish 函数

7.2.1.17.1 功能

计算输入张量每个元素的 HardSwish 值，见式（18）。

$$y_i = \frac{x_i * \min(\max(0, x_i + offset), threshold)}{scale} \quad (18)$$

式中：

x--表示输入张量；

y--表示输出张量；
max—表示取两个数之间的最大值；
min—表示取两个数之间的最小值；
$offset$—表示偏移量；
threshold—表示阈值；
scale—表示缩放因子。

7.2.1.17.2 前向接口参数

HardSwish函数前向接口应符合表34，C语言示例见A.2.1.17。

表 34 HardSwish 函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
阈值	输入	可选	公式中的threshold，默认值可为6.0
缩放因子	输入	可选	公式中的scale，默认值可为6.0
偏移量	输入	可选	公式中的offset，默认值可为3.0
输出张量	输出	必选	表示计算结果

7.2.1.17.3 前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：其他参数不合法。

7.2.1.17.4 后向接口参数

HardSwish函数后向接口应符合表35，C语言示例见A.2.1.35。

表 35 HardSwish 函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
阈值	输入	可选	公式中的threshold，默认值可为6.0
缩放因子	输入	可选	公式中的scale，默认值可为6.0
偏移量	输入	可选	公式中的offset，默认值可为3.0
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.17.5 后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：其他参数不合法。

7.2.1.18 误差函数

7.2.1.18.1 功能

计算输入张量每个元素的ErF(Error Function)值，见式（19）。

$$y_i = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-\eta^2) d\eta \quad \text{（19）}$$

式中：

x--表示输入张量；

y--表示输出张量；

$exp(*)$—表示以自然常数e为底的指数函数。

7.2.1.18.2 前向接口参数

误差函数前向接口应符合表36，C语言示例见A.2.1.18。

表 36 误差函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
输出张量	输出	必选	表示计算结果

7.2.1.18.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.18.4 后向接口参数

误差函数后向接口应符合表37，C语言示例见A.2.1.18。

表 37 误差函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.18.5 后向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.1.19 HardShrink 函数

7.2.1.19.1 功能

计算输入张量每个元素的HardShrink值，见式（20）。

$$y_i = \begin{cases} x_i, \ if \ x_i > \lambda \\ x_i, \ if \ x_i < -\lambda \\ 0, \ \ if \ others \end{cases} \quad \text{（20）}$$

式中：

x--表示输入张量；

y--表示输出张量；

λ—表示激活函数的阈值，默认为0.5。

7.2.1.19.2 前向接口参数

HardShrink函数前向接口应符合表38，C语言示例见A.2.1.19。

表 38 HardShrink 函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
阈值	输入	可选	激活函数的阈值λ，默认值可为0.5
输出张量	输出	必选	表示计算结果

7.2.1.19.3 前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.19.4 后向接口参数

HardShrink函数后向接口应符合表39，C语言示例见A.2.1.19。

表 39 HardShrink 函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
阈值	输入	可选	激活函数的阈值λ，默认值可为0.5
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.19.5 后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.20 TanhShrink 函数

7.2.1.20.1 功能

计算输入张量每个元素的TanhShrink值，见式（21）。

$$y_i = x_i - \tanh(x_i) \quad (21)$$

式中：
x--表示输入张量；
y--表示输出张量。

tanh—表示双曲正切函数。

7.2.1.20.2 前向接口参数

TanhShrink函数前向接口应符合表40，C语言示例见A.2.1.20。

表 40　TanhShrink 函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
输出张量	输出	必选	表示计算结果

7.2.1.20.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.20.4　后向接口参数

TanhShrink函数后向接口应符合表41，C语言示例见A.2.1.20。

表 41　TanhShrink 函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.20.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.21　HardTanh 函数

7.2.1.21.1　功能

HardTanh的分段线性逼近激活函数。计算输入张量每个元素的HardTanh值，见式（22）。

$$y_i = \begin{cases} \max, if\ x_i > max \\ min, if\ x_i < min \\ x_i,\ if\ others \end{cases} \quad \text{（22）}$$

式中：
x--表示输入张量；
y--表示输出张量；
max—表示最大阈值；
min—表示最小阈值。

7.2.1.21.2　前向接口参数

HardTanh函数前向接口应符合表42，C语言示例见A.2.1.21。

表 42　HardTanh 函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为整数、浮点数
min	输入	可选	HardTanh激活计算公式中的min值，默认值可为-1

表 42　HardTanh 函数前向接口参数列表（续）

参数名	类型	可选/必选	描述
max	输入	可选	HardTanh激活计算公式中的max值，默认值可为1
输出张量	输出	必选	表示计算结果

7.2.1.21.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.1.21.4　后向接口参数

HardTanh函数后向接口应符合表43，C语言示例见A.2.1.21。

表 43　HardTanh 函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示前向接口中的输入张量
min	输入	可选	HardTanh激活计算公式中的min值，默认值可为-1
max	输入	可选	HardTanh激活计算公式中的max值，默认值可为1
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.1.21.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2　损失函数

7.2.2.1　L1 损失函数

7.2.2.1.1　功能

计算实际值张量(input)和期望值张量(target)的L1损失函数值，见式（23）。

$$loss_i = |input_i - target_i| \quad \text{（23）}$$

式中：
$input_i$--表示第i个样本的实际结果；
$target_i$--表示第i个样本的期望结果；
$loss_i$--表示第i个样本的损失值。
其中i表示张量平铺后，遍历张量的下标。若选择 “none”，则对输出结果不进行归约；若选择“mean”，则$loss = \frac{1}{n}\sum_i^n loss_i$；若选择“sum”，则$loss = \sum_i^n loss_i$。其中n为张量总的元素数量。

7.2.2.1.2　前向接口参数

L1损失函数前向接口应符合表44，C语言示例见A.2.2.1。

表 44　L1 损失函数前向接口参数列表

参数名	类型	可选/必选	描述
实际值张量	输入	必选	表示实际值
期望值张量	输入	必选	表示期望值，采用one-hot编码，其形状与实际值张量形状一致，元素类型与实际值张量在计算上兼容
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
损失值	输出	必选	表示对实际值张量和期望值张量计算L1损失函数值后结果，若不进行归约，则形状与实际值张量一致

7.2.2.1.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.1.4　后向接口参数

L1损失函数后向接口应符合表45，C语言示例见A.2.2.1。

表 45　L1 损失函数后向接口参数列表

参数名	类型	可选/必选	描述
损失值张量梯度	输入	必选	表示损失值张量的梯度
实际值张量	输入	必选	表示实际值
期望值张量	输入	必选	表示期望值，采用one-hot编码，其形状与实际值张量形状一致，元素类型与实际值张量在计算上兼容
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
实际值张量梯度	输出	必选	表示实际值张量的梯度

7.2.2.1.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.2　均方误差损失函数

7.2.2.2.1　功能

计算实际值张量(input)和期望值张量(target)的均方误差值，见式（24）。

$$loss_i = (input_i - target_i)^2 \quad \text{（24）}$$

式中：
$input_i$--表示第i个样本的实际结果；
$target_i$--表示第i个样本的期望结果。$loss_i$--表示第i个样本的损失值。
其中i表示张量平铺后，遍历张量的下标。若选择“none”，则对输出结果不进行归约；若选择“mean”，则$loss = \frac{1}{n}\sum_i^n loss_i$；若选择“sum”，则$loss = \sum_i^n loss_i$。

7.2.2.2.2　前向接口参数

均方误差(MSE)损失函数前向接口应符合表46，C语言示例见A.2.2.2。

表 46　均方误差损失函数前向接口参数列表

参数名	类型	可选/必选	描述
实际值张量	输入	必选	表示实际值
期望值张量	输入	必选	表示期望值，其形状与实际值张量的形状一致
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
损失值	输出	必选	表示对实际值张量和预期值张量计算MSE损失函数值后的结果

7.2.2.2.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.2.4　后向接口参数

均方误差(MSE)损失函数后向接口应符合表47，C语言示例见A.2.2.2。

表 47　均方误差(MSE)损失函数后向接口参数列表

参数名	类型	可选/必选	描述
损失值张量梯度	输入	必选	表示损失值张量的梯度
实际值张量	输入	必选	表示实际值
期望值张量	输入	必选	期望值，其形状与实际值张量的形状一致，元素类型与实际值张量在计算上兼容
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
实际值张量梯度	输出	必选	表示实际值张量的梯度

7.2.2.2.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.3　交叉熵损失函数

7.2.2.3.1　功能

计算实际值张量(input)和期望值张量(target)的交叉熵。归约类型若选择“none”，则对输出结果不进行归约；若选择“mean”，则输出所有样本损失的均值；若选择“sum”，则输出所有样本损失的总和。

7.2.2.3.2　前向接口参数

交叉熵损失函数前向接口应符合表48，C语言示例见A.2.2.3。

表 48　交叉熵损失函数前向接口参数列表

参数名	类型	可选/必选	描述
实际值张量	输入	必选	表示实际分布，元素类型可为浮点数。形状$[n,c]$或者$[n,c,d_1,d_2,\ldots,d_k]$，其中$k \geq 1$，这里n表示batch大小，c表示类别数

表 48　交叉熵损失函数前向接口参数列表（续）

参数名	类型	可选/必选	描述
期望值张量	输入	必选	表示期望分布，对应每个样本的类别，采用one-hot编码，因此形状与实际值张量一致，元素类型可为整数
权重张量	输入	可选	指定各个样本的权重张量指针，张量其形状与期望值张量的形状一致，类型与实际值张量在计算上兼容，NULL表示相同权重，缺省值可为NULL
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
损失值	输出	必选	表示对实际值张量和预期值张量计算交叉熵损失函数值后的结果

7.2.2.3.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.3.4　后向接口参数

交叉熵损失函数后向接口应符合表49，C语言示例见A.2.2.3。

表 49　交叉熵损失函数后向接口参数列表

参数名	类型	可选/必选	描述
损失值张量梯度	输入	必选	表示损失值张量的梯度
实际值张量	输入	必选	表示实际分布。形状$[n,c]$或者$[n,c,d_1,d_2,\dots,d_k]$，其中$k \geq 1$，这里n表示batch大小，c表示类别数
期望值张量	输入	必选	表示期望分布，对应每个样本的类别，采用one-hot编码，因此形状与实际值张量一致，类型为整数
权重张量	输入	可选	指定各个样本的权重张量指针，张量形状与期望值张量的形状一致，类型与实际值张量在计算上兼容，NULL表示相同权重，缺省值可为NULL
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
实际值张量梯度	输出	必选	实际值张量的梯度

7.2.2.3.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.4　稀疏交叉熵损失函数

7.2.2.4.1　功能

计算实际值张量(input)和期望值张量(target)的交叉熵。如果归约类型选择的是“mean”，则输出所有样本损失的均值；若选择“sum”，则输出所有样本损失的总和。

7.2.2.4.2　前向接口参数

稀疏交叉熵损失函数前向接口应符合表50，C语言示例见A.2.2.4。

表 50　稀疏交叉熵损失函数前向接口参数列表

参数名	类型	可选/必选	描述
实际值张量	输入	必选	表示实际分布。形状$[n,c]$或者$[n,c,d_1,d_2,\ldots,d_k]$，其中$k\geq 1$，这里n表示batch大小，c表示类别数， 元素类型可为浮点数
期望值张量	输入	必选	期望分布，对应每个样本的实际类别，范围在[0,c)，形状为[n]或者[n,d_1,d_2,…,d_k]，其中k≥1，类型为整数
权重张量	输入	可选	指定各个样本的权重张量指针，张量形状与期望值张量的形状一致，类型与实际值张量在计算上兼容，NULL表示相同权重，缺省值可为NULL
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
损失值	输出	必选	表示对实际值张量和预期值张量计算交叉熵损失函数值后的结果

7.2.2.4.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.4.4　后向接口参数

稀疏交叉熵损失函数后向接口应符合表51，C语言示例见A.2.2.4。

表 51　稀疏交叉熵损失函数后向接口参数列表

参数名	类型	可选/必选	描述
损失值张量梯度	输入	必选	表示损失值张量的梯度
实际值张量	输入	必选	表示实际分布
期望值张量	输入	必选	期望分布，对应每个样本的实际类别，范围在[0,c)，形状为[n]或者[n,d_1,d_2,…,d_k]，其中k≥1，类型为整数
权重张量	输入	可选	指定各个样本的权重张量指针，张量形状与期望值张量的形状一致，类型与实际值张量在计算上兼容，NULL表示相同权重，缺省值可为NULL
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
实际值张量梯度	输出	必选	表示实际值张量的梯度

7.2.2.4.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.5　负对数损失函数

7.2.2.5.1　功能

计算负对数损失函数，见式（25）。

$$loss_i = -target_i * \log(input_i + \varepsilon) - (1 - target_i) * \log(1 - input_i + \varepsilon) \quad \ldots\ldots\ldots \quad (25)$$

式中：
$input_i$--表示第i个样本的实际结果；
$target_i$--表示第i个样本的期望结果；
$loss_i$--表示第i个样本的损失值；

$log(*)$--表示以10为底的对数；

ε --表示一个小数。

归约类型若选择“none”，则对输出结果不进行归约；若选择“mean”，则 $loss = \frac{1}{n}\sum_i^n loss_i$；若选择“sum”，则$loss = \sum_i^n loss_i$。

7.2.2.5.2 前向接口参数

负对数损失函数前向接口应符合表52，C语言示例见A.2.2.5。

表 52 负对数损失函数前向接口参数列表

参数名	类型	可选/必选	描述
实际值张量	输入	必选	表示实际分布，元素类型可为浮点数
期望值张量	输入	必选	表示期望标签值，每个元素的取值范围在是$[0,C)$，形状与实际值张量相同，元素类型与实际值张量在计算上兼容
小数	输入	可选	表示计算公式中的ε，默认值可为0.0001
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
损失值	输出	必选	形状与reduction参数有关，若reduction=“none”，则形状与input相同，否则损失值是一个标量张量

7.2.2.5.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.2.5.4 后向接口参数

负对数损失函数后向接口应符合表53，C语言示例见A.2.2.5。

表 53 负对数损失函数后向接口参数列表

参数名	类型	可选/必选	描述
损失值张量梯度	输入	必选	表示损失值张量的梯度
实际值张量	输入	必选	表示前向网络计算得到的实际标签值
期望值张量	输入	必选	表示期望标签值，每个元素的取值范围在是$[0,C)$，形状与实际值张量相同，元素类型与实际值张量在计算上兼容
小数	输入	可选	表示计算公式中的ε，默认值可为0.0001
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
实际值张量梯度	输出	必选	实际值张量的梯度

7.2.2.5.5 后向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.2.6 负对数似然损失函数

7.2.2.6.1 功能

计算实际值张量(input)和期望值张量(target)的负对数似然损失值。假设input形状为$[N,C]$，简单起见，用x表示input，y表示target，w表示weight。若reduction=“none”，则$loss_i=-\omega_{y_i}x_{i,y_i},n\in[0,N-1)$。若归约类型为“mean”或者 “sum”,则应符合式（26）。

$$loss=\begin{cases}\sum_{i=1}^{N}\frac{loss_i}{\sum_{i=1}^{N}w_{y_i}}, & \text{reduction}=\text{“mean”}\\ \sum_{i=1}^{N}loss_i, & \text{reduction}=\text{“sum”}\end{cases} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (26)$$

式中：

N—表示输入张量第一维度的大小；

w_{yi}—表示权重张量元素。

若input形状为$[N,C,d_1,d_2,\ldots,d_k]$，以图像为例，则对于同一batch，需要针对图像的每一个像素点求loss，归约前的loss形状为$[n,d_1,d_2,\ldots,d_k]$。

7.2.2.6.2 前向接口参数

负对数似然损失函数前向接口应符合表54，C语言示例见A.2.2.6。

表 54 负对数似然损失函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示实际输出概率。形状可为[N,C]或$[N,C,d_1,d_2,\ldots,d_k]$，N表示大小，C表示类别数，元素类型可为浮点数
标签张量	输入	必选	对应每个样本的实际类别，形状应为[N]或$[N,d_1,d_2,\ldots,d_k]$，元素值范围应在[0,C)，元素类型为整数
权重张量	输入	可选	指定各个类别的权重张量指针，形状为[C]，数据类型与输入张量相同，NULL表示相同权重，默认值为NULL
忽略标签值	输入	可选	指定一个忽略的标签值，默认值可为-100
归约类型	输入	可选	“none”、“mean”或“sum”，默认值为“mean”
损失值	输出	必选	表示计算负对数似然损失函数值后的结果

7.2.2.6.3 向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致.

其他内部错误：内部调用操作出错

7.2.2.6.4 后向接口参数

负对数似然损失函数后向接口应符合表55，C语言示例见A.2.2.6。

表 55 负对数似然损失函数后向接口参数列表

参数名	类型	可选/必选	描述
损失值张量梯度	输入	必选	表示损失值张量的梯度
实际值张量	输入	必选	表示实际输出概率。形状可为[N,C]或$[N,C,d_1,d_2,\ldots,d_k]$，N表示大小，C表示类别数，元素类型可为浮点数

表 55　负对数似然损失函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
标签张量	输入	必选	对应每个样本的实际类别，形状应为[N]或$[N,d_1,d_2,\ldots,d_k]$，元素值范围应在[0,C)，元素类型为整数
权重张量	输入	可选	指定各个类别的权重张量指针，形状为[C]，数据类型与输入张量相同，NULL表示相同权重，默认值为NULL
忽略标签值	输入	可选	指定一个忽略的标签值，默认值可为-100
归约类型	输入	可选	“none”、“mean”或“sum”，默认值为“mean”
实际值张量梯度	输出	必选	实际值张量的梯度

7.2.2.6.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
其他内部错误：内部调用操作出错。

7.2.2.7　CTC 损失函数

7.2.2.7.1　功能

计算CTC（ Connectionist Temporal Classification）损失，第i个样本的损失应符合式（27）。

$$loss_i = -ln\left(p(target_i \mid x_i)\right) \quad (27)$$

式中：
$target_i$—表示第i个样本的真实结果（Ground Truth）；
x_i—表示第i个样本的输入；
$p(target_i \mid x_i)$—表示输入第i个样本，输出对应真实结果的概率。

为方便描述，这里用 l 表示样本的真实结果，x表示样本的输入，假设下标从1开始，则对应的概率计算公式应符合式（28）。

$$p(l \mid x) = \sum_{s=1}^{|l'|} \frac{\alpha_t(s) * \beta_t(s)}{y^t_{l'_s}} \quad (28)$$

式中：

l' —表示在真实结果 l 的所有元素之间（包括开头和结尾）插入空格之后拓展成的字符串；

l'_s —表示样本第s个位置对应元素的值；

$y^t_{l'_s}$ —表示网络在t时刻输出l'_s 的概率（对应到函数参数input，即为$y^t_{l'_s}$ =input[t][i][l'_s]）；

$\alpha_t(s)$—表示l'第s个位置在第t时刻的前向概率；
$\beta_t(s)$—表示l'第s个位置在第t时刻的后向概率。

$\alpha_t(s)$和$\beta_t(s)$计算方式见式（29）- 式（30）。

$$\alpha_t(s)=\begin{cases} y_b^1, \text{ if } t=1 \text{ and } s=1 \\ y_{l_1}^1, \text{ if } t=1 \text{ and } s=2 \\ 0, \text{ if } t=1 \text{ and } s>2 \\ \left(\alpha_{t-1}(s)+\alpha_{t-1}(s-1)\right)y_{l'_s}^t, \text{ if } l'_s=blank \text{ or } l'_s=l'_{s-2} \\ \left(\alpha_{t-1}(s)+\alpha_{t-1}(s-1)+\alpha_{t-1}(s-2)\right)y_{l'_s}^t, \text{ otherwise} \end{cases} \quad \cdots\cdots\cdots\cdots \quad (29)$$

式中：

l' —表示在真实结果 l 的所有元素之间（包括开头和结尾）插入空格之后拓展成的字符串；

l_s—表示真实结果l第s个位置对应元素的值；
l'_s—表示样本第s个位置对应元素的值；
y_m^t—表示网络在t时刻输出m的概率；
$\alpha_t(s)$—表示l'第s个位置在第t时刻的前向概率。

$$\beta_t(s)=\begin{cases} y_b^T, \text{ if } t=T \text{ and } s=|l'| \\ y_{l_1}^T, \text{ if } t=T \text{ and } s=|l'|-1 \\ 0, \text{ if } t=T \text{ and } s<|l'|-1 \\ \left(\beta_{t+1}(s)+\beta_{t+1}(s+1)\right)y_{l'_s}^t, \text{ if } l'_s=blank \text{ or } l'_s=l'_{s+2} \\ \left(\beta_{t+1}(s)+\beta_{t+1}(s+1)+\beta_{t+1}(s+2)\right)y_{l'_s}^t, \text{ otherwise} \end{cases} \quad \cdots\cdots\cdots\cdots \quad (30)$$

式中：

l' —表示在真实结果 l 的所有元素之间（包括开头和结尾）插入空格之后拓展成的字符串；

l_s—表示真实结果l第s个位置对应元素的值；
l'_s—表示样本第s个位置对应元素的值；
y_m^t—表示网络在t时刻输出m的概率；

$\beta_t(s)$—表示l'第s个位置在第t时刻的后向概率。

7.2.2.7.2 前向接口参数

CTC损失函数前向接口应符合表56，C语言示例见A.2.2.7。

表 56 CTC 损失函数前向接口参数列表

参数名	类型	可选/必选	描述
实际值张量	输入	必选	表示模型输出经过softmax层后得到的每一类的概率，形状为[T,N,C]，其中T表示所有样本的最大时间，N表示batch大小，C表示类别总数（包含空格，用数字0表示），元素类型可为浮点数
期望值张量	输入	必选	表示期望值，形状为[N,L]，其中L表示所有target的最大长度，元素类型可为整数
实际值长度张量	输入	必选	表示实际值，表示每一个样本的时间序列长度，形状为[N]
期望值长度张量	输入	必选	表示每一个样本真实结果（Ground Truth）的长度，形状为[N]
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
是否训练	输入	必选	表示该接口是否用于训练。若值为“true”，则该接口用于训练；若值为“false”，则该接口用于推理

表 56　CTC 损失函数前向接口参数列表（续）

参数名	类型	可选/必选	描述
损失值	输出	必选	表示对实际值张量和预期值张量计算CTC损失函数值后的结果
前向概率对数张量	输出	必选	表示公式中前向概率$\alpha_t(s)$的对数值张量，可用于后向计算。推理阶段此参数不起作用

7.2.2.7.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：参数不合法。

7.2.2.7.4　后向接口参数

CTC损失函数后向接口应符合表57，C语言示例见A.2.2.7。

表 57　CTC 损失函数后向接口参数列表

参数名	类型	可选/必选	描述
损失值梯度	输入	必选	损失值张量的梯度
实际值张量	输入	必选	表示模型输出经过softmax层后得到的每一类的概率，形状为$[T,N,C]$，其中T表示所有样本的最大时间，N表示batch大小，C表示类别总数（包含空格，用数字0表示），元素类型可为浮点数
期望值张量	输入	必选	形状为$[N,L]$，其中L表示所有target的最大长度，元素类型可为整数
实际值长度张量	输入	必选	示每一个样本的时间序列长度，形状为$[N]$
期望值长度张量	输入	必选	表示每一个样本真实结果（Ground Truth）的长度，形状为$[N]$
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
前向概率对数张量	输入	必选	表示公式中$\alpha_t(s)$的对数值张量
实际值梯度	输出	必选	表示实际值张量的梯度

7.2.2.7.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：参数不合法。

7.2.2.8　平滑 L1 损失函数

7.2.2.8.1　功能

计算输入张量input和target的平滑L1损失函数值。其计算公式如下：

$$loss_i = \begin{cases} 0.5 * (input_i - target_i)^2, & if\ |input_i - target_i| < 1 \\ |input_i - target_i| - 0.5, & otherwise \end{cases} \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (31)$$

式中：
$input_i$--表示第i个样本的实际结果；
$target_i$--表示第i个样本的期望结果；
$loss_i$--表示第i个样本的损失值。

若选择 “none”，则对输出结果不进行归约；若选择“mean”，则$loss = \frac{1}{n}\sum_i^n loss_i$；若选择“sum”，则$loss = \sum_i^n loss_i$。其中n为input张量总的元素数量。

7.2.2.8.2 前向接口参数

平滑L1损失函数前向接口应符合表58，C语言示例见A.2.2.8。

表 58 平滑 L1 损失函数前向接口参数列表

参数名	类型	可选/必选	描述
实际值张量	输入	必选	表示输出实际值
期望值张量	输入	必选	期望值，其形状与实际值张量形状一致，元素类型与实际值张量在计算上兼容
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
损失值	输出	必选	表示对实际值张量和期望值张量计算平滑L1损失函数值后结果

7.2.2.8.3 前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.8.4 后向接口参数

平滑L1损失函数后向接口应符合表59，C语言示例见A.2.2.8。

表 59 平滑 L1 损失函数后向接口参数列表

参数名	类型	可选/必选	描述
损失值张量梯度	输入	必选	表示损失值张量的梯度
实际值张量	输入	必选	实际值
期望值张量	输入	必选	期望值，其形状与实际值张量形状一致，元素类型与实际值张量在计算上兼容
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
实际值张量梯度	输出	必选	实际值张量的梯度

7.2.2.8.5 后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.9 KL 散度损失函数

7.2.2.9.1 功能

计算输入张量input和target之间的Kullback-Leibler散度损失函数值，其中实际值张量input应为对数概率值，期望值张量应为概率值，见式（32）。

$$loss_i = target_i * (\log(target_i) - input_i) \quad (32)$$

式中：

$input_i$--表示第i个样本的实际结果；

$target_i$--表示第i个样本的期望结果；

$loss_i$--表示第i个样本的损失值；

$log(*)$—表示以10为底的对数。

归约类型若选择“none”，则对输出结果不进行归约；若选择“mean”，则 $loss = \frac{1}{n}\sum_i^n loss_i$；若选择“sum”，则$loss = \sum_i^n loss_i$。

7.2.2.9.2 前向接口参数

KL散度损失函数前向接口应符合表60，C语言示例见A.2.2.9。

表 60 KL 散度损失函数前向接口参数列表

参数名	类型	可选/必选	描述
实际值张量	输入	必选	表示前向网络计算得到的实际值。元素数据类型可为浮点数
期望值张量	输入	必选	表示期望值，其形状与实际值张量形状一致
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
损失值	输出	必选	表示对实际值张量和期望值张量计算KL散度损失函数值后结果

7.2.2.9.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.2.9.4 后向接口参数

KL散度损失函数后向接口应符合表61，C语言示例见A.2.2.9。

表 61 KL 散度损失函数后向接口参数列表

参数名	类型	可选/必选	描述
损失值张量梯度	输入	必选	损失值张量的梯度
实际值张量	输入	必选	前向网络计算得到的实际值
期望值张量	输入	必选	期望值，其形状与实际值张量形状一致
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
实际值张量梯度	输出	必选	实际值张量的梯度

7.2.2.9.5 后向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.2.10 软间隔损失函数

7.2.2.10.1 功能

通过创建一个标准，来优化输入张量input和target之间的二分类逻辑损失，见式（33）。

$$loss_i = \sum_j \frac{\log(1+\exp(-target_j * input_j))}{input\_element\_count} \quad \text{（33）}$$

式中：

$input_i$--表示第i个样本的实际结果；
$target_i$--表示第i个样本的期望结果，满足$target_i \in \{1,-1\}$；
$loss_i$--表示第i个样本的损失值；
$log(*)$—表示以10为底的对数；
exp(*)—表示以自然常数e为底的指数函数；
$input\_element\_count$—表示张量 input 的元素个数。

归约类型若选择“none”，则对输出结果不进行归约；若选择“mean”，则 $loss = \frac{1}{n}\sum_i^n loss_i$；若选择“sum”，则$loss = \sum_i^n loss_i$。

7.2.2.10.2 前向接口参数

软间隔损失函数前向接口应符合表62，C语言示例见A.2.2.10。

表 62 软间隔损失函数前向接口参数列表

参数名	类型	可选/必选	描述
实际值张量	输入	必选	表示实际值
期望值张量	输入	必选	表示期望值，形状与输入张量一致，元素值只能为1或-1。
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
损失值	输出	必选	表示对实际值张量和预期值张量计算soft_margin_loss后的结果

7.2.2.10.3 前向接口参数

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.10.4 后向接口参数

软间隔损失函数后向接口应符合表63，C语言示例见A.2.2.10。

表 63 软间隔损失函数后向接口参数列表

参数名	类型	可选/必选	描述
损失值梯度	输入	必选	表示损失值张量的梯度
实际值张量	输入	必选	表示实际值
期望值张量	输入	必选	表示期望值，形状与输入张量一致，元素值只能为1或-1
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
实际值梯度	输出	必选	实际值张量的梯度

7.2.2.10.5 后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.2.11 间隔排序损失函数

7.2.2.11.1 功能

计算两个输入张量（input1、input2）与期望值张量（target）之间的margin ranking loss，用于排序问题中，见式（34）。

$$loss_i = \max(0, -target_i * (input1_i - input2_i) + margin) \quad \text{（34）}$$

式中：

$input1_i$--表示第i个样本的第一个实际结果；

$input2_i$--表示第i个样本的第二个实际结果；

$target_i$--表示第i个样本的期望结果；

$loss_i$--表示第i个样本的损失值；

$max(*)$—表示取两个数之间的最大值；

$margin$--表示间隔值。

归约类型若选择“none”，则对输出结果不进行归约；若选择“mean”，则 $loss = \frac{1}{n}\sum_i^n loss_i$；若选择“sum”，则$loss = \sum_i^n loss_i$。

7.2.2.11.2 前向接口参数

间隔排序损失函数前向接口应符合表64，C语言示例见A.2.2.11。

表 64 间隔排序损失函数前向接口参数列表

参数名	类型	可选/必选	描述
第一个实际值张量	输入	必选	表示第一个排序得分输入张量，元素数据类型可为浮点型
第二个实际值张量	输入	必选	表示第二个排序得分输入张量，元素数据类型可为浮点型
期望值张量	输入	必选	表示期望值，第一个排序得分值大于第二个排序得分值时值为1，否则为-1
间隔值	输入	可选	表示两个得分之间的差异，默认值可为0
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
损失值	输出	必选	表示对实际值张量和预期值张量计算margin_ranking_loss后的结果

7.2.2.11.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.2.11.4 后向接口参数

间隔排序损失函数后向接口应符合65，C语言示例见A.2.2.11。

表 65 间隔排序损失函数后向接口参数列表

参数名	类型	可选/必选	描述
损失值梯度	输入	必选	表示损失值张量的梯度
第一个实际值张量	输入	必选	表示第一个排序得分输入张量

表 65　间隔排序损失函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
第二个实际值张量	输入	必选	表示第二个排序得分输入张量
期望值张量	输入	必选	表示期望值，第一个排序得分值大于第二个排序得分值时值为1，否则为-1
间隔值	输入	可选	表示两个得分之间的差异，默认值可为0
归约类型	输入	可选	“none”、“mean”或“sum”，默认值可为“mean”
第一个实际值张量梯度	输出	必选	表示前向接口中第一个实际值张量的梯度
第二个实际值张量梯度	输出	必选	表示前向接口中第二个实际值张量的梯度

7.2.2.11.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。

7.2.3　正则函数

7.2.3.1　随机失活函数

7.2.3.1.1　功能

让输入张量按概率随机失活，以达到防止模型过拟合目的。

7.2.3.1.2　前向接口参数

随机失活函数前向接口应符合表66，C语言示例见A.2.3.1。

表 66　随机失活函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示输入张量，元素类型可为整数、浮点数
概率	输入	可选	表示输入张量中元素变为0的概率rate，取值在0~1范围内，默认值可为0.5
计算轴	输入	可选	随机失活所沿着的轴axis，默认值可为-1
随机种子	输入	可选	表示生成随机数的种子。若种子为0，表示使用系统的随机种子；否则，使用种子作为随机数生成的种子。如果给定一个固定的整型种子，则会始终丢弃相同的输出单元。默认值可为0
是否训练	输入	必选	表示该接口是否用于训练。若值为“true”，则该接口用于训练；若值为“false”，则该接口用于推理
输出张量	输出	必选	表示输出张量
失活掩码张量	输出	必选	表示计算过程中决定输入元素是否失活的掩码。若该参数接收到空指针，则不输出掩码。推理阶段此参数不起作用

7.2.3.1.3　前向接口返回值

没有错误：操作成功。

对象非法：表示输入张量对象不合法。

非法参数：参数出错。

7.2.3.1.4 后向接口参数

随机失活函数后向接口应符合表67，C语言示例见A.2.3.1。

表 67 随机失活函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
概率	输入	可选	输入张量中元素变为0的概率rate，取值在0~1范围内，默认值可为0.5
计算轴	输入	可选	随机失活所沿着的轴axis，默认值可为-1
随机种子	输入	可选	表示生成随机数的种子，默认值可为0
失活掩码张量	输入	必选	表示计算过程中决定输入元素是否失活的掩码。若该参数接收到空张量，则根据其他参数重新计算出掩码
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.3.1.5 后向接口返回值

没有错误：操作成功。

对象非法：表示输入张量对象不合法。

非法参数：参数出错。

7.2.3.1.6 其他附加说明

增加参数int axis。参照TensorFlow、PyTorch和MXNet，随机失活函数dropout一般包含两种情况：（1）输入中的每个元素独立地以概率rate被置零；（2）输入中以channel为单位，各样本、各channel之间相互独立，以rate为概率，该channel的所有元素同时被置零。为了使情况（2）得以实现，应增加参数axis，使其指定channel所在的维度；或者令axis=-1，让每个元素独立地被置零。

7.2.3.2 标签平滑函数

7.2.3.2.1 功能

标签平滑是一种对分类器层进行正则化的机制。由于直接优化正确标签的对数似然可能会导致过拟合，降低模型的适应能力，因此通过标签平滑来降低模型的置信度，见式（35）。

$$output_i = (1-\varepsilon) * label_i + \varepsilon * \mu_i \quad \text{.....} \quad (35)$$

式中：

$label_i$--表示输入标签张量元素；

u_i--表示先验分布张量元素；

$output_i$--表示输出张量元素。

其中$1-\varepsilon$和ε分别是权重，μ通常使用均匀分布。

7.2.3.2.2 标签平滑函数参数

标签平滑函数接口应符合表68，C语言示例见A.2.3.2。

表 68　标签平滑函数参数列表

参数名	类型	可选/必选	描述
标签张量	输入	必选	包含标签数据的输入张量。标签数据使用one-hot表示，形状为[N_1,…,N_k,C]，其中C为标签类别数。元素类型可为整数
先验分布张量	输入	必选	用于平滑标签的先验分布，形状为[1,C]，数据类型与张量label在计算上兼容。如果未设置，则使用均匀分布
权重	输入	可选	用于混合原始真实分布和固定分布的权重。默认值可为0.1
输出张量数据类型	输入	必选	指定输出张量的数据类型
输出张量	输出	必选	经过平滑后的输出张量，形状与标签张量相同

7.2.3.2.3　标签平滑函数返回值

没有错误：操作成功。

对象非法：表示输入张量对象不合法。

7.2.3.2.4　其他附加说明

标签平滑正则化是对输入数据的标签进行一定变换，没有可训练参数，无后向接口。

7.2.4　归一化函数

7.2.4.1　批量归一化操作

7.2.4.1.1　功能

实现批量归一化(Batch Normalization)，见式（36）。

$$y = scale \times \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} + bias \quad \text{（36）}$$

式中：

x--表示输入张量；

y--表示输出张量；

$scale$--表示缩放张量；

$bias$--表示偏移张量；

u--表示样本均值张量；

σ--表示样本方差张量；

ε--表示防止除数为0的小数字。

假设输入张量形状为[n, c, h, w]，axis=1，那么scale和bias形状为[c]。详细语义请参考[2]。如果提前计算了样本平均值和方差，或希望使用全局统计数据，可以作为参数输入。作为输出，可以返回批量归一化后的平均值与方差。

7.2.4.1.2　前向接口参数

批量归一化操作函数前向接口应符合表69，C语言示例见A.2.4.1。

表 69　批量归一化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为浮点数，表示维度的形式为[N,C,D1,D2,…,Dn]的张量数据，其中N是批量大小，C是通道数
计算轴	输入	必选	表示批量归一化所沿的轴axis，将输入张量沿着第axis个维度进行批量归一化
缩放量张量	输入	必选	公式中的scale，表示形状为输入数据中C的缩放张量
偏移量张量	输入	必选	公式中的bias，表示形状为输入数据中C的偏置张量
样本均值张量	输入	可选	计算公式为$\mu = \frac{1}{n}\sum_i^n x_i$，其中n为样本数量，默认值为NULL
样本方差张量	输入	可选	计算公式为$\sigma^2 = \frac{1}{n}\sum_i^n (x_i - \mu)^2$，其中n样本数量，默认值为NULL
小数	输入	可选	公式中的ϵ，防止除数为0的小数字，默认值可为0.00001
是否训练	输入	必选	表示该接口是否用于训练。若值为“true”，则该接口用于训练；若值为“false”，则该接口用于推理
输出张量	输出	必选	表示计算结果
计算因子	输入	可选	表示momentum，用于计算运行平均值和方差的因子。默认值可为0.1
运行均值张量	输出	可选	批量归一化操作后的运行均值张量，默认值为NULL
运行方差张量	输出	可选	批量归一化操作后的运行方差张量，默认值为NULL

7.2.4.1.3　前向接口返回值

没有错误：操作成功。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。
其他内部错误：出现除数为0的情况等。

7.2.4.1.4　后向接口参数

批量归一化操作函数后向接口应符合表70，C语言示例见A.2.4.1。

表 70　批量归一化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示输入的张量
中间输出张量	输入	必选	输入张量经过shift之后得到的中间变量
计算轴	输入	必选	表示批量归一化所沿的轴axis，将输入张量沿着第axis个维度进行批量归一化
缩放量张量	输入	必选	公式中的scale，表示形状为输入数据中C的缩放张量
偏移量张量	输入	必选	公式中的bias，表示形状为输入数据中C的偏置张量
样本均值张量	输入	可选	计算公式为$\mu = \frac{1}{n}\sum_i^n x_i$，其中n为样本数量，默认值为NULL
样本方差张量	输入	可选	计算公式为$\sigma^2 = \frac{1}{n}\sum_i^n (x_i - \mu)^2$，其中n为样本数量，默认值为NULL
小数	输入	可选	公式中的ϵ，防止除数为0的小数字，默认值可为0.00001
缩放量梯度	输出	必选	表示缩放量张量的梯度
偏移量梯度	输出	必选	表示偏移量张量的梯度
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.4.1.5 后向接口返回值

没有错误：操作成功。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。
其他内部错误：出现除数为0的情况等。

7.2.4.2 分组归一化操作

7.2.4.2.1 功能

为了避免批量归一化中batch size对模型的影响，提出分组归一化（Group Normalization），其首先在指定维度分为多个组后，对每一组做归一化处理，假设输入张量形状为[n, c, h, w]，分组轴channel_axis=1，规约轴reduced_axes=[3, 4]，那么scale和bias的形状为[groups * n]，归约后输出张量的形状为[c/groups, h, w]，详细语义请参考[3]。如果提前计算了批量样本平均值和方差，或希望使用全局统计数据，可以作为参数输入。作为输出，可以返回批量归一化后的平均值与方差。

7.2.4.2.2 前向接口参数

分组归一化操作函数前向接口应符合表71，C语言示例见A.2.4.2。

表 71 分组归一化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为浮点数
分组数	输入	必选	指定归约维度分为多少个组
分组轴	输入	必选	表示需要进行分组处理的维度
归约轴	输入	必选	表示进行数据统计的维度
归约轴长度	输入	必选	表示归约轴数组的长度
缩放量张量	输入	必选	公式中的scale
偏移量张量	输入	必选	公式中的bias
样本均值张量	输入	可选	计算公式为$\mu = \frac{1}{n}\sum_i^n x_i$，其中n为样本数量，默认值为NULL
样本方差张量	输入	可选	计算公式为$\sigma^2 = \frac{1}{n}\sum_i^n (x_i - \mu)^2$，其中n为样本数量，默认值为NULL
小数	输入	可选	防止除数为0的小数字，默认值可为0.00001
输出张量	输出	必选	表示计算结果
运行均值张量	输出	可选	批量归一化操作后的运行均值张量，默认值为NULL
运行方差张量	输出	可选	批量归一化操作后的运行方差张量，默认值为NULL

7.2.4.2.3 前向接口返回值

没有错误：操作成功。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。
类型不匹配：张量的数据类型不一致。
其他内部错误：出现除数为0的情况等。

7.2.4.2.4 后向接口参数

分组归一化操作函数后向接口应符合表72，C语言示例见A.2.4.2。

表 72　分组归一化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示输入的张量
中间输出张量	输入	必选	输入张量经过shift之后得到的中间变量
分组数	输入	必选	指定归约维度分为多少个组
分组轴	输入	必选	表示需要进行分组处理的维度
归约轴	输入	必选	表示进行数据统计的维度
归约轴长度	输入	必选	表示归约轴数组的长度
缩放量张量	输入	必选	公式中的scale
偏移量张量	输入	必选	公式中的bias
样本均值张量	输入	可选	计算公式为$\mu = \frac{1}{n}\sum_i^n x_i$，其中n为样本数量，默认值为NULL
样本方差张量	输入	可选	计算公式为$\sigma^2 = \frac{1}{n}\sum_i^n (x_i - \mu)^2$，其中n样本数量，默认值为NULL
小数	输入	可选	防止除数为0的小数字，默认值可为0.00001
缩放量梯度	输出	必选	表示缩放量张量的梯度
偏移量梯度	输出	必选	表示偏移量张量的梯度
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.4.2.5　后向接口返回值

没有错误：操作成功。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。
类型不匹配：张量的数据类型不一致。
其他内部错误：出现除数为0的情况等。

7.2.4.3　层归一化操作

7.2.4.3.1　功能

层归一化可以应用于小批量输入数据。在图像应用中，通常会对整个样本进行归一化，即假设输入张量形状为[N, C, H, W]或[N, H, W, C]，均设置axis=1，scale和bias的形状为[C*H*W]。层归一化作用于整个样本，且经常用于NLP应用中。详细语义请参考[22]。

7.2.4.3.2　前向接口参数

层归一化操作函数前向接口应符合表73，C语言示例见A.2.4.3。

表 73　层归一化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为浮点数
计算轴	输入	必选	表示归一化的维度。输入张量x依据axis转换成一个形状为[D_0*…*D_(axis-1),D_axis*…*D_(k-1)]的2-D张量，然后沿第2个维度进行归一化。axis的取值必须和张量x中的布局信息一致
缩放量张量	输入	必选	表示缩放量，形状为$[D_{axis}*\ldots*D_{k-1}]$的1-D张量，数据类型与张量x相同。可以为空
偏移量张量	输入	必选	表示偏移量，形状为$[D_{axis}*\ldots*D_{k-1}]$的1-D张量，数据类型与张量x相同。可以为空
小数	输入	可选	防止除数为0的小数字。默认值可为0.00001
输出张量	输出	必选	表示计算结果
运行均值张量	输出	可选	训练期间用于加速梯度计算的保存平均值
运行方差张量	输出	可选	训练期间用于加速梯度计算的保存方差

7.2.4.3.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。
其他内部错误：出现除数为0的情况等。

7.2.4.3.4　后向接口参数

层归一化操作函数后向接口应符合表74，C语言示例见A.2.4.3。

表 74　层归一化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示输入的张量
中间输出张量	输入	必选	输入张量经过shift之后得到的中间变量
计算轴	输入	必选	表示归一化的维度。输入张量x依据axis转换成一个形状为[D_0*…*D_(axis-1),D_axis*…*D_(k-1)]的2-D张量，然后沿第2个维度进行归一化。axis的取值必须和张量x中的布局信息一致
缩放量张量	输入	必选	表示缩放量，形状为$[D_{axis}*\ldots*D_{k-1}]$的1-D张量，数据类型与张量x相同。可以为空
偏移量张量	输入	必选	表示偏移量，形状为$[D_{axis}*\ldots*D_{k-1}]$的1-D张量，数据类型与张量x相同。可以为空
小数	输入	可选	防止除数为0的小数字。默认值可为0.00001
缩放量梯度	输出	必选	表示缩放量张量的梯
偏移量梯度	输出	必选	表示偏移量张量的梯度
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.4.3.5 后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。
其他内部错误：出现除数为0的情况等。

7.2.4.4 实例归一化操作

7.2.4.4.1 功能

实例归一化，可以作用于卷积和全连接操作的计算结果，作用于图像通道上的数据，根据样本每个通道的均值和方差信息进行正则化。假设输入张量形状为[N, C, H, W]，通道所在的维度为1，则需设置axis=1。详细语义请参考[23]。

7.2.4.4.2 前向接口参数

实例归一化操作函数前向接口应符合表75，C语言示例见A.2.4.4。

表 75 实例归一化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为浮点数。张量的维度不小于2
计算轴	输入	必选	表示输入张量通道所在的维度axis，值为C，并且将沿这个维度进行归一化。axis的取值必须和输入张量中的布局信息一致
缩放量张量	输入	必选	表示缩放量，形状为[C]的1维度张量，数据类型与张量x相同。可以为空
偏移量张量	输入	必选	表示偏移量，形状为[C]的1-D张量，数据类型与张量x相同。可以为空
小数	输入	可选	防止除数为0的小数字。默认值可为0.00001
输出张量	输出	必选	表示计算结果

7.2.4.4.3 前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。
其他内部错误：出现除数为0的情况等。

7.2.4.4.4 后向接口参数

实例归一化操作函数后向接口应符合表76，C语言示例见A.2.4.4。

表 76 实例归一化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示输入的张量
中间输出张量	输入	必选	输入张量经过shift之后得到的中间变量

表 76　实例归一化操作函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
计算轴	输入	必选	表示输入张量通道所在的维度axis，值为C，并且将沿这个维度进行归一化。axis的取值必须和输入张量中的布局信息一致
缩放量张量	输入	必选	表示缩放量，形状为[C]的1-D张量，数据类型与张量x相同。可以为空。
偏移量张量	输入	必选	表示偏移量，形状为[C]的1-D张量，数据类型与张量x相同。可以为空。
小数	输入	可选	防止除数为0的小数字。默认值可为0.00001
缩放量梯度	输出	必选	表示缩放量张量的梯度
偏移量梯度	输出	必选	表示偏移量张量的梯度
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.4.4.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。
其他内部错误：出现除数为0的情况等。

7.2.4.5　局部响应归一化操作

7.2.4.5.1　功能

对输入张量进行局部响应归一化，其中输出张量的每个元素都是原张量中对应元素与若干个相邻通道的对应元素进行归一化的结果。为了方便表示，假设$x_{i,j,k}$表示输入张量中第i个样本中第j个通道的切片张量中的第k个元素，见式（37）。

$$output_{i,j,k} = x_{i,j,k}\left(alpha\sum_{j'=\max(0,\ j-radius)}^{\min(num\_channels\ -1,\ j+radius)}\left(x_{i,j',k}\right)^2 + bias\right)^{-beta} \quad \cdots\cdots\cdots \quad (37)$$

式中：
$x_{i,j,k}$--表示输入张量中第i个样本中第j个通道的切片张量中的第k个元素；
$output$—表示输出张量；
$num\_channels$—表示通道数；
$radius$—表示半径；
$bias$—表示偏移量；
$alpha$—表示相乘系数；
$beta$—表示指数系数。

7.2.4.5.2　前向接口参数

局部响应归一化操作函数前向接口应符合表77，C语言示例见A.2.4.5。

表 77　局部响应归一化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	形状为[batch_size, num_channels,d1, …, dn]，元素类型可以为浮点数

表 77　局部响应归一化操作函数前向接口参数列表（续）

参数名	类型	可选/必选	描述
半径	输入	必选	表示局部归一化的半径
相乘系数	输入	可选	表示相乘系数，默认值可为0.0001
指数系数	输入	可选	表示指数系数，默认值可为0.75
偏移量	输入	可选	表示偏置系数，默认值可为1.0
输出张量	输出	必选	表示计算结果

7.2.4.5.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.4.5.4　后向接口参数

局部响应归一化操作函数后向接口应符合表78，C语言示例见A.2.4.5。

表 78　局部响应归一化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示输入的张量
半径	输入	必选	表示局部归一化的半径
相乘系数	输入	可选	表示相乘系数，默认值可为0.0001
指数系数	输入	可选	表示指数系数，默认值可为0.75
偏移量	输入	可选	表示偏置系数，默认值可为1.0
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.4.5.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.4.6　L2 归一化操作

7.2.4.6.1　功能

L2归一化方法，通过输入张量的欧几里得距离之和，来对输入张量x进行归一化。对于1-D张量，见式（38）。

$$y_i = \frac{x_i}{\sqrt{\sum_j x_j^2 + epsilon}} \quad \text{（38）}$$

式中：
x--表示输入张量；
y--表示输出张量；

epsilon--表示小数，防止除0。

对于多维输入张量的情况，则对axis维度轴上的每个1-D张量切片单独归一化。

7.2.4.6.2 前向接口参数

L2归一化操作函数前向接口应符合表79，C语言示例见A.2.4.6。

表 79 L2 归一化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为浮点数
计算轴	输入	必选	表示归一化的轴
小数	输入	可选	防止除数为0的小数字，默认值可为0.00001
输出张量	输出	必选	表示计算结果

7.2.4.6.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

其他内部错误：出现除数为0的情况。

非法参数：表示参数出错。

对象未初始化：输入张量对象未初始化。

7.2.4.6.4 后向接口参数

L2归一化操作函数后向接口应符合表80，C语言示例见A.2.4.6。

表 80 L2 归一化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度。
输入张量	输入	必选	表示输入的张量
计算轴	输入	必选	表示归一化的轴，从0开始计数。
小数	输入	可选	防止除数为0的小数字，默认值可为0.00001
输入张量梯度	输出	必选	表示输入张量的梯度。

7.2.4.6.5 后向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

其他内部错误：出现除数为0的情况。

非法参数：表示参数出错。

对象未初始化：输入张量对象未初始化。

7.2.4.7 Lp 范数归一化操作

7.2.4.7.1 功能

使用L_p范数沿指定维度对输入张量进行归一化，见式（39）。

$$y_i = \frac{x_i}{\max(\|x\|_p, \epsilon)} \quad (39)$$

式中：

x--表示输入张量；

y--表示输出张量；

$max\ (*)$--表示取最大值；

ε --表示一个小数。

其中，$\|x\|_p = (\sum_j |x_j|^p)^{1/p}$，沿指定维度axis进行计算。

7.2.4.7.2 前向接口参数

Lp范数归一化操作函数前向接口应符合表81，C语言示例见A.2.4.7。

表 81 Lp 范数归一化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	元素类型可以为浮点数
范数指数	输入	必选	范数公式中的指数值，缺省值可以为2
计算轴	输入	必选	表示归一化的轴
小数	输入	可选	防止除数为0的小数字，默认值可为0.00001
输出张量	输出	必选	与输入张量具有相同的形状

7.2.4.7.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

其他内部错误：出现除数为0的情况。

非法参数：表示参数出错。

对象未初始化：输入张量对象未初始化。

7.2.4.7.4 后向接口参数

Lp范数归一化操作函数后向接口应符合表82，C语言示例见A.2.4.7。

表 82 Lp 归一化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示输入的张量
范数指数	输入	必选	范数公式中的指数值，缺省值可以为2
计算轴	输入	必选	表示归一化的轴
小数	输入	可选	防止除数为0的小数字，默认值可为0.00001
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.4.7.5 后向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

其他内部错误：出现除数为0的情况。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.4.8 权重归一化操作

7.2.4.8.1 功能

对权重参数进行归一化处理。权重归一化操作可以将神经网络中权重向量的长度与其方向解耦，将指定权重转换成两个张量，代表长度的向量g和代表方向的向量v。见式（40）。

$$w = g\frac{v}{\|v\|} \quad \text{（40）}$$

式中：
g--表示长度的向量；
v--表示方向的向量；
w--表示方向的向量。

7.2.4.8.2 权重归一化操作函数参数

接口应符合表83，C语言示例见A.2.4.8。

表 83 权重归一化操作函数参数列表

参数名	类型	可选/必选	描述
权重张量	输入	必选	输入权重张量
计算轴	输入	必选	归一化的维度
长度张量	输出	必选	代表长度的向量（1-D张量）
方向张量	输出	必选	代表方向的向量（1-D张量）

7.2.4.8.3 权重归一化操作函数返回值

没有错误：操作成功。
类型不匹配：输入的类型不一致。
非法参数：参数出错。

7.2.4.9 谱归一化操作

7.2.4.9.1 功能

计算全连接层、卷积层的权重参数的谱正则值。通过“幂迭代法“计算出每层参数矩阵的谱范数来重新标定权张量，从而达到在生成对抗网络中discriminators（critics）的稳定训练。其计算步骤如下：

步骤一，生成向量u和v，其中向量u的长度是输入权重张量的第axis个维度，向量v的长度是剩余维度的乘积；

步骤二，用u和v迭代计算指定轮数，迭代公式见式（41）-式（42）。

$$v = \frac{W^T u}{\|W^T u\|_2} \quad \text{（41）}$$

式中：

u--表示输入权重张量的第axis个维度向量；

w--表示权重张量。

$$u = \frac{W^T v}{\|W^T v\|_2} \quad (42)$$

式中：

v--表示输入权重张量除去第axis维度的乘积；

w--表示权重张量。

步骤三，计算σ(W)并计算特征值归一化后的权重，见式（43）-式（44）。

$$\sigma(W) = u^T W v \quad (43)$$

式中：

u--表示输入权重张量的第axis个维度向量；

v--表示输入权重张量除去第axis维度的乘积；

w--表示权重张量。

$$W = \frac{w}{\sigma(w)} \quad (44)$$

式中：

w--表示输入权重张量；

W—表示谱正则化后权重张量。

7.2.4.9.2 谱归一化操作函数参数

谱归一化操作函数接口应符合表84，C语言示例见A.2.4.9。

表 84 谱归一化操作函数参数列表

参数名	类型	可选/必选	描述
权重张量	输入	必选	输入的权重张量，一般是fc、conv层的权重，元素数据类型可以为浮点数
计算轴	输入	必选	归一化的维度
迭代轮数	输入	必选	迭代轮数
小数	输入	可选	防止除数为0的小数字，默认值可为0.00001
输出权重张量	输出	必选	谱正则化后权重张量，形状、数据类型与输入权重一样

7.2.4.9.3 谱归一化操作函数返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

非法参数：参数出错。

7.2.5 池化函数

7.2.5.1 一维池化操作

7.2.5.1.1 功能

对输入张量x进行池化处理。当ksize设置成L_{in}、padding为NULL时，执行全局池化操作。

7.2.5.1.2 前向接口参数

一维池化操作函数前向接口应符合表85，C语言示例见A.2.5.1。

表 85 一维池化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	形状为[N, C, L]或[N, L, C]，其中N是batch_size、C是通道数、L是特征长度。元素类型可以为浮点数， 这里可以将形状分为batch_size, num_channels, spatial_shape三个部分，其中spatial_shape一般由长度等组成，而池化只对spatial_shape上的维度进行处理
池化类型	输入	必选	可从“max”和“avg”二者中选其一。 若选择“max”，则进行max池化；若选择“avg”，则进行average池化
池化窗口大小	输入	必选	表示池化窗口大小，必须不为空
池化窗口大小数组长度	输入	可选	池化窗口大小的数组长度，可以等于1或者2。如果为1，则表示所有维度的窗口大小一样。默认值为1
池化步长	输入	可选	表示池化步长，默认值可为1
池化步长数组长度	输入	可选	池化步长的数组长度，可以等于1或者2。如果为1，则表示所有维度的跨步一样。默认值为1
填充元素个数	输入	可选	表示填充元素个数，如果为空，则表示所有维度不进行填充，默认值可为空
填充元素个数数组长度	输入	可选	填充元素个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的填充个数一样。默认值为0
池化膨胀个数	输入	可选	表示池化膨胀个数，如果为空，则表示所有维度不进行膨胀，默认值可为空
池化膨胀个数数组长度	输入	可选	池化膨胀个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的膨胀个数一样。默认值为0
是否使用ceil函数计算输出高度和宽度	输入	可选	是否使用ceil函数计算输出高度和宽度。默认值可为否
是否忽略填充值	输入	可选	是否忽略填充值。该参数只对”avg”池化方式有效，”max”池化方式始终忽略填充值。默认值可为是
输出张量	输出	必选	表示计算结果
是否自适应	输入	可选	表示是否使用自适应方式，默认为否

7.2.5.1.3 前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.5.1.4 后向接口参数

一维池化操作函数后向接口应符合表86，C语言示例见A.2.5.1。

表 86　一维池化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度。
输入张量	输入	必选	表示输入的张量
池化类型	输入	必选	可从“max”和“avg”二者中选其一。若选择“max”，则进行max池化；若选择“avg”，则进行average池化。
池化窗口大小	输入	必选	表示池化窗口大小，必须不为空
池化窗口大小数组长度	输入	可选	池化窗口大小的数组长度，可以等于1或者2。如果为1，则表示所有维度的窗口大小一样。默认值为1
池化步长	输入	可选	表示池化步长，默认值可为1
池化步长数组长度	输入	可选	池化步长的数组长度，可以等于1或者2。如果为1，则表示所有维度的跨步一样。默认值为1
填充元素个数	输入	可选	表示填充元素个数，如果为空，则表示所有维度不进行填充，默认值可为空
填充元素个数数组长度	输入	可选	填充元素个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的填充个数一样。默认值为0
池化膨胀个数	输入	可选	表示池化膨胀个数，如果为空，则表示所有维度不进行膨胀，默认值可为空
池化膨胀个数数组长度	输入	可选	池化膨胀个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的膨胀个数一样。默认值为0
是否使用ceil函数计算输出高度和宽度	输入	可选	是否使用ceil函数计算输出高度和宽度。默认值可为否
是否忽略填充值	输入	可选	是否忽略填充值。该参数只对”avg”池化方式有效，”max”池化方式始终忽略填充值。默认值可为是
是否自适应	输入	可选	表示是否使用自适应方式，默认为否
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.5.1.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.5.2　二维池化操作

7.2.5.2.1　功能

对输入张量x进行池化处理。当ksize设置成(H_{in},W_{in})、padding为空时，执行全局池化操作。

7.2.5.2.2　前向接口参数

二维池化操作函数前向接口应符合表87，C语言示例见A.2.5.2。

表 87　二维池化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	形状为[N, C, H, W]或[N, H, W, C]，其中N是batch_size、C是通道数、H是特征高度、W是特征宽度。池化只在H和W两个维度上进行处理。元素类型可为浮点数，这里可以将形状分为batch_size, num_channels, spatial_shape三个部分，其中spatial_shape一般由长度、宽度、深度等组成，而池化只对spatial_shape上的维度进行处理
池化类型	输入	必选	可从“max”和“avg”二者中选其一。若选择“max”，则进行max池化；若选择“avg”，则进行average池化
池化窗口大小	输入	必选	表示池化窗口大小，必须不为空
池化窗口大小数组长度	输入	可选	池化窗口大小的数组长度，可以等于1或者2。如果为1，则表示所有维度的窗口大小一样。默认值为1
池化步长	输入	可选	表示池化步长，默认值可为1
池化步长数组长度	输入	可选	池化步长的数组长度，可以等于1或者2。如果为1，则表示所有维度的跨步一样。默认值为1
填充元素个数	输入	可选	表示填充元素个数，如果为空，则表示所有维度不进行填充，默认值可为空
填充元素个数数组长度	输入	可选	填充元素个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的填充个数一样。默认值为0
池化膨胀个数	输入	可选	表示池化膨胀个数，如果为空，则表示所有维度不进行膨胀，默认值可为空
池化膨胀个数数组长度	输入	可选	池化膨胀个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的膨胀个数一样。默认值为0
是否使用ceil函数计算输出高度和宽度	输入	可选	是否使用ceil函数计算输出高度和宽度。默认值可为否
是否忽略填充值	输入	可选	是否忽略填充值。该参数只对”avg”池化方式有效，”max”池化方式始终忽略填充值。默认值可为是
输出张量	输出	必选	表示计算结果
是否自适应	输入	可选	表示是否使用自适应方式，默认为否

7.2.5.2.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.5.2.4　后向接口参数

二维池化操作函数后向接口应符合表88，C语言示例见A.2.5.2。

表 88　二维池化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度。
输入张量	输入	必选	表示输入的张量
池化类型	输入	必选	可从“max”和“avg”二者中选其一。若选择“max”，则进行max池化；若选择“avg”，则进行average池化。
池化窗口大小	输入	必选	表示池化窗口大小，必须不为空。
池化窗口大小数组长度		可选	池化窗口大小的数组长度，可以等于1或者2。如果为1，则表示所有维度的窗口大小一样。默认值为1
池化步长	输入	可选	表示池化步长，默认值可为1
池化步长数组长度	输入	可选	池化步长的数组长度，可以等于1或者2。如果为1，则表示所有维度的跨步一样。默认值为1
填充元素个数	输入	可选	表示填充元素个数，如果为空，则表示所有维度不进行填充，默认值可为空
填充元素个数数组长度	输入	可选	填充元素个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的填充个数一样。默认值为0
池化膨胀个数	输入	可选	表示池化膨胀个数，如果为空，则表示所有维度不进行膨胀，默认值可为空
池化膨胀个数数组长度	输入	可选	池化膨胀个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的膨胀个数一样。默认值为0
是否使用ceil函数计算输出高度和宽度	输入	可选	是否使用ceil函数计算输出高度和宽度。默认值可为否
是否忽略填充值	输入	可选	是否忽略填充值。该参数只对”avg”池化方式有效，”max”池化方式始终忽略填充值。默认值可为是
是否自适应	输入	可选	表示是否使用自适应方式，默认为否
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.5.2.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.5.3　三维池化操作

7.2.5.3.1　功能

对输入张量x进行池化处理。当ksize设置成(D_{in},H_{in},W_{in})、padding为空时，执行全局池化操作。

7.2.5.3.2　前向接口参数

三维池化操作函数前向接口应符合表89，C语言示例见A.2.5.3。

表 89　三维池化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示输入张量，形状为[N, C, D, H, W]或[N, D, H, W, C]，其中N是batch_size、C是通道数、D是特征深度、H是特征高度、W是特征宽度。池化只在D、H、W三个维度上进行处理。元素类型可以为浮点数，这里可以将形状分为batch_size, num_channels, spatial_shape三个部分，其中spatial_shape一般由长度、宽度、深度等组成，而池化只对spatial_shape上的维度进行处理。
池化类型	输入	必选	可从“max”和“avg”二者中选其一。若选择“max”，则进行max池化；若选择“avg”，则进行average池化。
池化窗口大小	输入	必选	表示池化窗口大小，必须不为空。
池化窗口大小数组长度	输入	可选	池化窗口大小的数组长度，可以等于1或者2。如果为1，则表示所有维度的窗口大小一样。默认值为1
池化步长	输入	可选	表示池化步长，默认值可为1
池化步长数组长度	输入	可选	池化步长的数组长度，可以等于1或者2。如果为1，则表示所有维度的跨步一样。默认值为1
填充元素个数	输入	可选	表示填充元素个数，如果为空，则表示所有维度不进行填充，默认值可为空
填充元素个数数组长度	输入	可选	填充元素个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的填充个数一样。默认值为0
池化膨胀个数	输入	可选	表示池化膨胀个数，如果为空，则表示所有维度不进行膨胀，默认值可为空
池化膨胀个数数组长度	输入	可选	池化膨胀个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的膨胀个数一样。默认值为0
是否使用ceil函数计算输出高度和宽度	输入	可选	是否使用ceil函数计算输出高度和宽度。默认值可为否
是否忽略填充值	输入	可选	是否忽略填充值。该参数只对”avg”池化方式有效，”max”池化方式始终忽略填充值。默认值可为是
输出张量	输出	必选	表示计算结果
是否自适应	输入	可选	表示是否使用自适应方式，默认为否

7.2.5.3.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.5.3.4　后向接口参数

三维池化操作函数后向接口应符合表90，C语言示例见A.2.5.3。

表 90 三维池化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度。
输入张量	输入	必选	表示输入的张量
池化类型	输入	必选	可从“max”和“avg”二者中选其一。若选择“max”，则进行max池化；若选择“avg”，则进行average池化。
池化窗口大小	输入	必选	表示池化窗口大小，必须不为空
池化窗口大小数组长度	输入	可选	池化窗口大小的数组长度，可以等于1或者2。如果为1，则表示所有维度的窗口大小一样。默认值为1
池化步长	输入	可选	表示池化步长，默认值可为1
池化步长数组长度	输入	可选	池化步长的数组长度，可以等于1或者2。如果为1，则表示所有维度的跨步一样。默认值为1
填充元素个数	输入	可选	表示填充元素个数，如果为空，则表示所有维度不进行填充，默认值可为空
填充元素个数数组长度	输入	可选	填充元素个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的填充个数一样。默认值为0
池化膨胀个数	输入	可选	表示池化膨胀个数，如果为空，则表示所有维度不进行膨胀，默认值可为空
池化膨胀个数数组长度	输入	可选	池化膨胀个数的数组长度，可以等于1或者2。如果为1，则表示所有维度的膨胀个数一样。默认值为0
是否使用ceil函数计算输出高度和宽度	输入	可选	是否使用ceil函数计算输出高度和宽度。默认值可为否
是否忽略填充值	输入	可选	是否忽略填充值。该参数只对”avg”池化方式有效，”max”池化方式始终忽略填充值。默认值可为是
是否自适应	输入	可选	表示是否使用自适应方式，默认为否
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.5.3.5 后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.5.4 自适应一维池化操作

7.2.5.4.1 功能

对输入张量x进行自适应一维池化处理。

7.2.5.4.2 前向接口参数

自适应一维池化操作函数前向接口应符合表91，C语言示例见A.2.5.4。

表 91 自适应一维池化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	形状为[N, C, L]或[N, L, C]，其中N是batch_size、C是通道数、L是特征长度。元素类型可以为浮点数。这里可以将形状分为batch_size, num_channels, spatial_shape三个部分，其中spatial_shape一般由长度组成，而池化只对spatial_shape上的维度进行处理
池化类型	输入	必选	可从“max”和“avg”二者中选其一。若选择“max”，则进行max池化；若选择“avg”，则进行average池化
特征图长度	输入	必选	输出特征的长度
输出张量	输出	必选	表示计算结果

7.2.5.4.3 前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.5.4.4 后向接口参数

自适应一维池化操作函数后向接口应符合表92，C语言示例见A.2.5.4。

表 92 自适应一维池化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示输入的张量
池化类型	输入	必选	可从“max”和“avg”二者中选其一。若选择“max”，则进行max池化；若选择“avg”，则进行average池化
特征图长度	输入	必选	输出特征的长度
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.5.4.5 后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.5.5 自适应二维池化操作

7.2.5.5.1 功能

对输入张量x进行自适应二维池化处理。

7.2.5.5.2 前向接口参数

自适应二维池化操作函数前向接口应符合表93，C语言示例见A.2.5.5。

表 93　自适应二维池化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	形状为[N, C, H, W]或[N, H, W, C]，其中N是batch_size、C是通道数、H是特征高度、W是特征宽度。元素类型可为浮点数。这里可以将形状分为batch_size, num_channels, spatial_shape三个部分，其中spatial_shape一般由长度、宽度等组成，而池化只对spatial_shape上的维度进行处理
池化类型	输入	必选	可从“max”和“avg”二者中选其一。若选择“max”，则进行max池化；若选择“avg”，则进行average池化
特征图长度	输入	必选	输出特征的长度
输出张量	输出	必选	表示计算结果

7.2.5.5.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.5.5.4　后向接口参数

自适应二维池化操作函数后向接口应符合表94，C语言示例见A.2.5.5。

表 94　自适应二维池化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度
输入张量	输入	必选	表示输入的张量
池化类型	输入	必选	可从“max”和“avg”二者中选其一。若选择“max”，则进行max池化；若选择“avg”，则进行average池化
特征图长度	输入	必选	输出特征的长度
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.5.5.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.5.6　自适应三维池化操作

7.2.5.6.1　功能

对输入张量x进行自适应三维池化处理。

7.2.5.6.2　前向接口参数

自适应三维池化操作函数前向接口应符合表95，C语言示例见A.2.5.6。

表 95　自适应三维池化操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	形状为[N, C, D, H, W]或[N, D, H, W, C]，其中N是batch_size、C是通道数、D是特征深度、H是特征高度、W是特征宽度。元素类型可以为浮点数，这里可以将形状分为batch_size, num_channels, spatial_shape三个部分，其中spatial_shape一般由长度、宽度、深度等组成，而池化只对spatial_shape上的维度进行处理
池化类型	输入	必选	可从“max”和“avg”二者中选其一。若选择“max”，则进行max池化；若选择“avg”，则进行average池化
特征图长度	输入	必选	输出特征的长度
输出张量	输出	必选	表示计算结果

7.2.5.6.3　前向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.5.6.4　后向接口参数

自适应三维池化操作函数后向接口应符合表96，C语言示例见A.2.5.6。

表 96　自适应三维池化操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度。
输入张量	输入	必选	表示输入的张量
池化类型	输入	必选	可从“max”和“avg”二者中选其一。若选择“max”，则进行max池化；若选择“avg”，则进行average池化
特征图长度	输入	必选	输出特征的长度
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.5.6.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
非法参数：表示参数出错。
对象未初始化：输入张量对象未初始化。

7.2.6　卷积函数

7.2.6.1　一维卷积操作

7.2.6.1.1　功能

将三维输入张量用对应的卷积核进行卷积运算。如果输入张量(input)的形状为$[N, C_{in}, L_{in}]$，其中N是批大小、C_{in}是通道数、L_{in}是特征长度。卷积核(filter)的形状为$[C_{out}, C_{in}, L_f]$，其中C_{out}是卷积核的个数（和输出张量的通道数相同），$[L_f]$是滤波器的大小。输出张量(output)形状为$[N, C_{out}, L_{out}]$，见式(38)。

$$L_{out} = \frac{(L_{in}+padding\times 2-(dilation\times(L_f-1)+1))}{stride}+1 \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (38)$$

式中：

$padding$——填充大小参数值；

$dilation$——空洞大小参数值；

$stride$——步长参数值；

L_f——卷积核的特征长度；

L_{in}——输入张量的特征长度；

L_{out}——输出张量的特征长度。

7.2.6.1.2 前向接口参数

一维卷积前向接口参数应符合表97，C语言示例见A.2.6.1。

表 97 一维卷积前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	待卷积张量，形状可以为[N, C_{in}, L_{in}]或[N, L_{in}, C_{in}]，元素类型可以为浮点数
卷积核	输入	必选	3维张量，数据类型、布局方式与输入张量相同
卷积步长	输入	可选	卷积步长组成的数组，如果为空，步长则为1，默认为空
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1
填充数	输入	可选	填充元素个数组成的数组，如果为空，则表示不填充，默认为空
填充数数组长度	输入	可选	表示填充元素数组长度，默认值为0
空洞大小	输入	可选	卷积膨胀个数组成的数组，如果为空，则空洞大小为0，默认为空
空洞数组长度	输入	可选	表示空洞数组长度，默认为1
分组卷积组数	输入	可选	进行分组卷积的组数。如果值为n，输入张量和卷积核分别根据通道数量平均分成n组，每组输入和卷积核分别进行卷积计算，默认为1
输出张量	输出	必选	卷积运算后的张量，形状可以为[N, C_{out}, L_{out}]或[N, L_{out}, C_{out}]

7.2.6.1.3 前向接口返回值

一维卷积前向接口返回值如下：

——没有错误：操作成功；

——对象未初始化：表示输入张量对象不合法；

——内存不足：表示输出张量分配空间不足；

——非法参数：表示其他参数不合法；

——其他内部错误：内部调用操作出错。

7.2.6.1.4 后向接口参数

一维卷积后向接口参数应符合表98，C语言示例见A.2.6.1。

表 98　一维卷积后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	输出张量的梯度。输出张量是形状为[N, Cout, Lout]或[N, Lout, Cout]的张量，即前向接口中输出张量
卷积核	输入	必选	3维张量，数据类型、布局方式与输入张量相同，即前向接口中的卷积核
卷积步长	输入	可选	卷积步长组成的数组，如果为空，步长则为1，默认为空，即前向接口中卷积步长
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1，即前向接口中步长数组长度
填充数	输入	可选	填充元素个数组成的数组，如果为空，则表示不填充，默认为空，即前向接口中填充数。
填充数数组长度	输入	可选	表示填充元素数组长度，默认值为0，即前向接口中填充数数组长度
空洞大小	输入	可选	卷积膨胀个数组成的数组，如果为空，则空洞大小为0，默认为空，即前向接口中空洞大小。
空洞数组长度	输入	可选	表示空洞数组长度，默认为1，即前向接口中空洞数组长度。
分组卷积组数	输入	可选	进行分组卷积的组数。如果值为n，输入张量和卷积核分别根据通道数量平均分成n组，每组输入和卷积核分别进行卷积计算，默认为1，即前向接口中分组卷积组数
输入张量梯度	输出	必选	输入张量的梯度。输入张量形状可以为[N, C_{in}, L_{in}]或[N, L_{in}, C_{in}]，元素类型可以为浮点数，即前向接口中的输入张量

7.2.6.1.5　后向接口返回值

一维卷积后向接口返回值如下：

——没有错误：操作成功；

——对象未初始化：表示输入张量对象不合法；

——内存不足：表示输出张量分配空间不足；

——非法参数：表示其他参数不合法；

——其他内部错误：内部调用操作出错。

7.2.6.1.6　后向接口参数

一维卷积后向接口参数应符合表99，C语言示例见A.2.6.1。

表 99　一维卷积后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	输出张量的梯度。输出张量是形状为[N, C_{out}, L_{out}]或[N, L_{out}, C_{out}]的张量，即前向接口中输出张量
输入张量	输入	必选	输入张量形状可以为[N, Cin, Lin]或[N, Lin, Cin]，元素类型可以为浮点数，即前向接口中的输入张量

表 99　一维卷积后向接口参数列表（续）

参数名	类型	可选/必选	描述
卷积步长	输入	可选	卷积步长组成的数组，如果为空，步长则为1，默认为空，即前向接口中卷积步长
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1，即前向接口中步长数组长度
填充数	输入	可选	填充元素个数组成的数组，如果为空，则表示不填充，默认为空，即前向接口中填充数。
填充数数组长度	输入	可选	表示填充元素数组长度，默认值为0，即前向接口中填充数数组长度
空洞大小	输入	可选	卷积膨胀个数组成的数组，如果为空，则空洞大小为0，默认为空，即前向接口中空洞大小。
空洞数组长度	输入	可选	表示空洞数组长度，默认为1，即前向接口中空洞数组长度。
分组卷积组数	输入	可选	进行分组卷积的组数。如果值为n，输入张量和卷积核分别根据通道数量平均分成n组，每组输入和卷积核分别进行卷积计算，默认为1，即前向接口中分组卷积组数
卷积核梯度	输出	必选	卷积核的梯度，卷积核是3维张量，数据类型、布局方式与输入张量相同，即前向接口中卷积核。

7.2.6.1.7　后向接口返回值

一维卷积后向接口返回值如下：
——没有错误：操作成功；
——对象未初始化：表示输入张量对象不合法；
——内存不足：表示输出张量分配空间不足；
——非法参数：表示其他参数不合法；
——其他内部错误：内部调用操作出错。

7.2.6.2　二维卷积操作

7.2.6.2.1　功能

将输入张量用对应的卷积核进行卷积运算。如果输入张量input形状为$[\mathrm{N},C_{in},H_{in},W_{in}]$，其中N是batch_size、$C_{in}$是通道数、$H_{in}$是特征高度、$W_{in}$是特征宽度。卷积核filter形状为$[C_{out},C_{in},H_f,W_f]$，其中$C_{out}$是卷积核的个数（和输出张量的通道数相同），$[H_f,W_f]$是滤波器的大小。输出张量output形状为$[\mathrm{N},C_{out},H_{out},W_{out}]$，见式（45）-式（46）。

$$H_{out}=\frac{(H_{in}+padding[0]*2-(dilation[0]*(H_f-1)+1))}{stride[0]}+1 \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (45)$$

式中：
H_{in}—表示是特征高度；
$padding[0]$—表示填充第一个维度大小；
$dilation[0]$—表示空洞第一个维度大小；
H_f--卷积核的高度；
$stride[0]$—表示步长第一个维度大小；
H_{out}—表示输出张量的高度。

$$W_{out} = \frac{(W_{in}+padding[1]*2-(dilation[1]*(W_f-1)+1))}{stride[1]} + 1 \quad \text{（46）}$$

式中：

W_{in}—表示是输入特征的宽度；

$padding[1]$—表示填充第二个维度大小；

$dilation[1]$—表示空洞第二个维度大小；

W_f--卷积核的宽度；

$stride[1]$—表示步长第二个维度大小；

W_{out}—表示输出张量的宽度。

7.2.6.2.2 前向接口参数

二维卷积操作函数前向接口应符合表100，C语言示例见A.2.6.2。

表 100 二维卷积操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	4维张量，支持不同的布局格式，[N,C,H,W]或[N,H,W,C]元素类型可以为浮点数
卷积核	输入	必选	表示卷积核，是一个4维张量，数据类型、布局方式与张量input相同
卷积步长	输入	可选	可以为单个数或者两个元素的数组，如果为空，步长则为1，默认为空
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1
填充数	输入	可选	表示填充元素个数，可以为单个数或两元素数组，如果为空，则表示不填充，默认为空
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空
卷积膨胀数组长度	输入	可选	表示空洞数组长度，默认为1
分组卷积组数	输入	可选	表示进行分组卷积的组数。当数值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1
输出张量	输出	必选	表示计算结果

7.2.6.2.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

对象未初始化：表示输入张量对象不合法。

非法参数：表示其他参数不合法。

7.2.6.2.4 后向接口参数

二维卷积操作函数后向接口应符合表101，C语言示例见A.2.6.2。

表 101　二维卷积操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出梯度	输入	必选	表示输出张量的梯度，输出张量是前向接口中的输出张量。
卷积核	输入	必选	表示卷积核，是一个4维张量，数据类型、布局方式与张量input相同，即前向接口中的卷积核。
卷积步长	输入	可选	可以为单个数或者两个元素的数组，如果为空，步长则为1，默认为空，即前向接口中的卷积步长
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1，即前向接口中的步长数组长度
填充数	输入	可选	表示填充元素个数，可以为单个数或两元素数组，如果为空，则表示不填充，默认为空，即前向接口中的填充数。
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0，即前向接口中的填充数组长度
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空，即前向接口中的卷积膨胀个数
卷积膨胀数组长度	输入	可选	表示膨胀数组长度，默认为1，即前向接口中的卷积膨胀数组长度
分组卷积组数	输入	可选	表示进行分组卷积的组数。当数值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1，即前向接口中的分组卷积组数
输入卷积核梯度	输出	必选	表示卷积核的梯度，卷积核是一个4维张量，数据类型、布局方式与输入张量相同，即前向接口中的卷积核

7.2.6.2.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
对象未初始化：表示输入张量对象不合法。
非法参数：表示其他参数不合法。

7.2.6.3　三维卷积操作

7.2.6.3.1　功能

将输入张量用对应的卷积核进行卷积运算。如果输入张量input形状为$[\mathrm{N},C_{in},D_{in},H_{in},W_{in}]$，其中N是batch_size、$C_{in}$是通道数、$D_{in}$是特征深度、$H_{in}$是特征高度、$W_{in}$是特征宽度。卷积核filter形状为$[C_{out},C_{in},D_f,H_f,W_f]$，其中$C_{out}$是卷积核的个数（和输出张量的通道数相同），$[D_f,H_f,W_f]$是滤波器的大小。输出张量output形状为$[\mathrm{N},C_{out},D_{out},H_{out},W_{out}]$，见式（47）-式（49）。

$$D_{out}=\frac{(D_{in}+padding[0]*2-(dilation[0]*(D_f-1)+1))}{stride[0]}+1 \quad \text{（47）}$$

式中：
D_{in}—表示是输入特征的深度；
$padding[0]$—表示填充第一个维度大小；
$dilation[0]$—表示空洞第一个维度大小；

D_f—滤波器的特征深度；
$stride[0]$—表示步长第一个维度大小；
D_{out}—表示输出特征的深度。

$$H_{out} = \frac{(H_{in}+padding[1]*2-(dilation[1]*(H_f-1)+1))}{stride[1]} + 1 \quad (48)$$

式中：
H_{in}—表示是特征高度；
$padding[1]$—表示填充第二个维度大小；
$dilation[1]$—表示空洞第二个维度大小；
H_f--卷积核的高度；
$stride[1]$—表示步长第二个维度大小；
H_{out}—表示输出张量的高度。

$$W_{out} = \frac{(W_{in}+padding[2]*2-(dilation[2]*(W_f-1)+1))}{stride[2]} + 1 \quad (49)$$

式中：
W_{in}—表示是输入特征的宽度；
$padding[2]$—表示填充第三个维度大小；
$dilation[2]$—表示空洞第三个维度大小；
W_f--卷积核的宽度；
$stride[2]$—表示步长第三个维度大小；
W_{out}—表示输出张量的宽度。

7.2.6.3.2 前向接口参数

三维卷积操作函数前向接口应符合表102，C语言示例见A.2.6.3。

表 102 三维卷积操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	5维张量，支持不同的布局格式，[N,C,D,H,W]或[N,D,H,W,C]，元素类型可为浮点型
卷积核	输入	必选	表示卷积核，是一个5维张量，数据类型、布局方式与张量input相同
卷积步长	输入	可选	可以为单个数或三元素数组，如果为空，步长则为1，默认为空
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1
填充数	输入	可选	表示填充元素个数，可以为单个数或三元素数组，如果为空，则表示所有维度不进行填充，默认值可为空
填充数数组长度	输入	可选	表示填充元素数组长度，默认值为0
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空
卷积膨胀数组长度	输入	可选	表示空洞数组长度，默认为1

表 102　三维卷积操作函数前向接口参数列表（续）

参数名	类型	可选/必选	描述
分组卷积组数	输入	可选	表示进行分组卷积的组数。当数值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1
输出张量	输出	必选	表示计算结果

7.2.6.3.3　前向接口返回值

没有错误：操作成功。
非法参数：表示其他参数不合法。
对象未初始化：表示输入张量对象不合法。
类型不匹配：张量的数据类型不一致。

7.2.6.3.4　后向接口参数

三维卷积操作函数后向接口应符合表103，C语言示例见A.2.6.3。

表 103　三维卷积操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度，输出张量为前向接口中的输出张量
卷积核	输入	必选	表示卷积核，是一个5维张量，数据类型、布局方式与张量input相同，即前向接口中的卷积核
卷积步长	输入	可选	可以为单个数或三元素数组，如果为空，步长则为1，默认为空。即前向接口中的卷积步长
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1，即前向接口中的步长数组长度
填充数	输入	可选	表示填充元素个数，可以为单个数或三元素数组，如果为空，则表示所有维度不进行填充，默认值可为空，即前向接口中的填充数
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0，即前向接口中的填充数组长度
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空，即前向接口中的卷积膨胀个数
卷积膨胀数组长度	输入	可选	表示膨胀数组长度，默认为1，即前向接口中的卷积膨胀数组长度
分组卷积组数	输入	可选	表示进行分组卷积的组数。当数值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1，即前向接口中的分组卷积数
输入张量梯度	输出	必选	表示输入张量的梯度，输入张量是5维张量，支持不同的布局格式，[N,C,D,H,W]或[N,D,H,W,C]，元素类型可为浮点型，即前向接口中的输入张量

7.2.6.3.5　后向接口返回值

没有错误：操作成功。
非法参数：表示其他参数不合法。

对象未初始化：表示输入张量对象不合法。
类型不匹配：张量的数据类型不一致。

7.2.6.3.6 后向接口参数

三维卷积操作函数后向接口应符合表104，C语言示例见A.2.6.3。

表 104 三维卷积操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度，输出张量为前向接口中的输出张量
输入张量	输入	必选	表示输入的5维度张量，输入张量支持不同的布局格式，[N,C,D,H,W]或[N,D,H,W,C]，元素类型可为浮点型，即前向接口中的输入张量
卷积步长	输入	可选	可以为单个数或三元素数组，如果为空，步长则为1，默认为空，即前向接口中的卷积步长
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1，即前向接口中的步长数组长度
填充数	输入	可选	表示填充元素个数，可以为单个数或三元素数组，如果为空，则表示所有维度不进行填充，默认值可为空，即前向接口中的填充数
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0，即前向接口中的填充数组长度
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空，即前向接口中的卷积膨胀个数
卷积膨胀数组长度	输入	可选	表示膨胀数组长度，默认为1，即前向接口中的卷积膨胀数组长度
分组卷积组数	输入	可选	表示进行分组卷积的组数。当数值为=n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1，即前向接口中的分组卷积数
输入卷积核梯度	输出	必选	表示卷积核的梯度，卷积核是一个5维张量，数据类型、布局方式与张量input相同，即前向接口中的卷积核

7.2.6.3.7 后向接口返回值

没有错误：操作成功。
非法参数：表示其他参数不合法。
对象未初始化：表示输入张量对象不合法。
类型不匹配：张量的数据类型不一致。

7.2.6.4 一维反卷积操作

7.2.6.4.1 功能

转置卷积又被称为反卷积，其计算过程相当于卷积的反向计算。如果输入张量input形状为$[N, C_{in}, L_{in}]$，其中N是batch_size、C_{in}是通道数、L_{in}是特征长度。卷积核filter形状为$[C_{in}, C_{cout}, L_f]$，其中C_{out}是卷积核的个数（和输出张量的通道数相同），$[L_f]$是滤波器的大小。那么输出张量output形状为$[N, C_{out}, L_{out}]$，见式（50）。

$$L_{out} = (L_{in} - 1) * stride[0] - padding[0] * 2 + dilation[0] * (L_f - 1) + 1 \quad \text{（50）}$$

式中：

L_{in}—表示是特征长度；
$padding[0]$—表示填充第一个维度大小；
$dilation[0]$—表示空洞第一个维度大小；
L_f--卷积核的长度；
$stride[0]$—表示步长第一个维度大小；
L_{out}—表示输出张量的长度。

7.2.6.4.2 前向接口参数

一维反卷积操作函数前向接口应符合表105，C语言示例见A.2.6.4。

表 105 一维反卷积操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	3维张量，支持不同的布局格式，[N,C,L]或[N,L,C]，元素类型可以为浮点数
卷积核	输入	必选	表示卷积核，是一个3维张量，数据类型、布局方式与张量input相同
卷积步长	输入	可选	可以为单个数或者两个元素的数组，默认值可为1
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1
填充数	输入	可选	表示填充元素个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行填充，默认值可为空
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空
卷积膨胀数组长度	输入	可选	表示膨胀数组长度，默认为1
分组卷积组数	输入	可选	表示进行分组卷积的组数。当数值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1
输出张量	输出	必选	表示计算结果

7.2.6.4.3 前向接口返回值

没有错误：操作成功。
非法参数：表示其他参数不合法。
对象未初始化：表示输入张量对象不合法。
类型不匹配：张量的数据类型不一致。

7.2.6.4.4 后向接口参数

一维反卷积操作函数后向接口应符合表106，C语言示例见A.2.6.4。

表 106　一维反卷积操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度。输出张量表示前向计算结果，即前向接口中输出张量
卷积核	输入	必选	表示卷积核，是一个3维张量，数据类型、布局方式与张量input相同，即前向接口中的卷积核
卷积步长	输入	可选	可以为单个数或者两个元素的数组，默认值可为1，即前向接口中的卷积步长
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1，即前向接口中的步长数组长度
填充数	输入	可选	表示填充元素个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行填充，默认值可为空，即前向接口中的填充数
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0，即前向接口中的填充数组长度
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空，即前向接口中的卷积膨胀个数
卷积膨胀数组长度	输入	可选	表示膨胀数组长度，默认为1，即前向接口中的卷积膨胀数组长度
分组卷积组数	输入	可选	表示进行分组卷积的组数。当数值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1，即前向接口中的分组卷积组数
输入张量梯度	输出	必选	表示输入张量的梯度，输入张量是3维张量，支持不同的布局格式，[N,C,L]或[N,L,C]，元素类型可以为浮点数，即前向接口中输入张量

7.2.6.4.5　后向接口返回值

没有错误：操作成功。
非法参数：表示其他参数不合法。
对象未初始化：表示输入张量对象不合法。
类型不匹配：张量的数据类型不一致。

7.2.6.5　二维反卷积操作

7.2.6.5.1　功能

转置卷积又被称为反卷积，其计算过程相当于卷积的反向计算。如果输入张量input形状为$[N,C_{in},H_{in},W_{in}]$，其中N是批尺寸、C_{in}是通道数、H_{in}是特征高度、W_{in}是特征宽度。卷积核filter形状为$[C_{in},C_{cout},H_f,W_f]$，其中$C_{out}$是卷积核的个数（和输出张量的通道数相同），$[H_f,W_f]$是滤波器的大小。输出张量output形状为$[N,C_{out},H_{out},W_{out}]$，见式（51）-式（52）。

$$H_{out}=(H_{in}-1)*stride[0]-padding[0]*2+dilation[0]*(H_f-1)+1 \quad \text{（51）}$$

式中：

H_{in}—表示是特征高度；

$padding[0]$—表示填充第一个维度大小；

$dilation[0]$—表示空洞第一个维度大小；

H_f--卷积核的高度；

$stride[0]$—表示步长第一个维度大小；

H_{out}—表示输出张量的高度。

$$W_{out} = (W_{in} - 1) * stride[1] - padding[1] * 2 + dilation[1] * (W_f - 1) + 1 \quad \cdots\cdots \quad (52)$$

式中：

W_{in}—表示是输入特征的宽度；

$padding[1]$—表示填充第二个维度大小；

$dilation[1]$—表示空洞第二个维度大小；

W_f--卷积核的宽度；

$stride[1]$—表示步长第二个维度大小；

W_{out}—表示输出张量的宽度。

7.2.6.5.2 前向接口参数

二维反卷积操作函数前向接口应符合表107，C语言示例见A.2.6.5。

表 107 二维反卷积操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	4维张量，支持不同的布局格式，[N,C,H,W]或[N,H,W,C]元素类型可以为浮点数
卷积核	输入	必选	表示卷积核，是一个4维张量，数据类型、布局方式与张量input相同
卷积步长	输入	可选	可以为单个数或者两个元素的数组，默认值可为1
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1
填充数	输入	可选	表示填充元素个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行填充，默认值可为空
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空
卷积膨胀数组长度	输入	可选	表示膨胀数组长度，默认为1
分组卷积组数	输入	可选	表示进行分组卷积的组数。当值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1
输出张量	输出	必选	表示计算结果

7.2.6.5.3 前向接口返回值

没有错误：操作成功。

非法参数：表示其他参数不合法。

对象未初始化：表示输入张量对象不合法。

类型不匹配：张量的数据类型不一致。

7.2.6.5.4 后向接口参数

二维反卷积操作函数后向接口应符合表108，C语言示例见A.2.6.5。

表 108　二维反卷积操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度，输出张量是前向接口的输出张量
卷积核	输入	必选	表示卷积核，是一个4维张量，数据类型、布局方式与张量input相同，即前向接口中的卷积核
卷积步长	输入	可选	可以为单个数或者两个元素的数组，默认值可为1 ，即前向接口中的卷积步长
卷积步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1，即前向接口中的卷积步长数组长度
填充数	输入	可选	表示填充元素个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行填充，默认值可为空，即前向接口中的填充数
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0，即前向接口中的填充数组长度
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空，即前向接口中的卷积膨胀个数
卷积膨胀数组长度	输入	可选	表示膨胀数组长度，默认为1，即前向接口中的卷积膨胀数组长度
分组卷积组数	输入	可选	表示进行分组卷积的组数。当值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1，即前向接口中的分组卷积组数
输入张量梯度	输出	必选	表示输入张量的梯度，输入张量是4维张量，支持不同的布局格式，[N,C,H,W]或[N,H,W,C]元素类型可以为浮点数，即前向接口中的输入张量

7.2.6.5.5　后向接口返回值

没有错误：操作成功。
非法参数：表示其他参数不合法。
对象未初始化：表示输入张量对象不合法。
类型不匹配：张量的数据类型不一致。

7.2.6.5.6　后向接口参数

二维反卷积操作函数后向接口应符合表109，C语言示例见A.2.6.5。

表 109　二维反卷积操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度，输出张量是前向接口的输出张量
输入张量	输入	必选	表示输入的张量，输入张量是4维张量，支持不同的布局格式，[N,C,H,W]或[N,H,W,C]元素类型可以为浮点数，即前向接口中的输入张量
卷积步长	输入	可选	可以为单个数或者两个元素的数组，默认值可为1，即前向接口中的卷积步长
卷积步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1，即前向接口中的卷积步长数组长度

表 109　二维反卷积操作函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
填充数	输入	可选	表示填充元素个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行填充，默认值可为空，即前向接口中的填充数
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0，即前向接口中的填充数组长度
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空，即前向接口中的卷积膨胀个数
卷积膨胀数组长度	输入	可选	表示膨胀数组长度，默认为1，即前向接口中的卷积膨胀数组长度
分组卷积组数	输入	可选	表示进行分组卷积的组数。当值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1，即前向接口中的分组卷积组数
输入卷积核梯度	输出	必选	表示卷积核的梯度，卷积核是一个4维张量，数据类型、布局方式与张量input相同，即前向接口中的卷积核

7.2.6.5.7　后向接口返回值

没有错误：操作成功。
非法参数：表示其他参数不合法。
对象未初始化：表示输入张量对象不合法。
类型不匹配：张量的数据类型不一致。

7.2.6.6　三维反卷积操作

7.2.6.6.1　功能

转置卷积又被称为反卷积，其计算过程相当于卷积的反向计算。如果输入张量input形状为$[\mathrm{N},C_{in},D_{in},H_{in},W_{in}]$，其中N是batch_size、$C_{in}$是通道数、$H_{in}$是特征高度、$W_{in}$是特征宽度。卷积核filter形状为$[C_{in},C_{cout},D_f,H_f,W_f]$，其中$C_{out}$是卷积核的个数（和输出张量的通道数相同），$[D_f,H_f,W_f]$是滤波器的大小。输出张量output形状为$[\mathrm{N},C_{out},D_{out},H_{out},W_{out}]$，见式（53）-式（55）。

$$D_{out}=(D_{in}-1)*stride[0]-padding[0]*2+dilation[0]*(D_f-1)+1 \quad \text{（53）}$$

式中：
D_{in}—表示是输入特征的深度；
$padding[0]$—表示填充第一个维度大小；
$dilation[0]$—表示空洞第一个维度大小；
D_f—滤波器的特征深度；
$stride[0]$—表示步长第一个维度大小；
D_{out}—表示输出特征的深度。

$$H_{out}=(H_{in}-1)*stride[1]-padding[1]*2+dilation[1]*(H_f-1)+1 \quad \text{（54）}$$

式中：
H_{in}—表示是特征高度；
$padding[1]$—表示填充第二个维度大小；
$dilation[1]$—表示空洞第二个维度大小；
H_f--卷积核的高度；

$stride[1]$—表示步长第二个维度大小；
H_{out}—表示输出张量的高度。

$$W_{out} = (W_{in} - 1) * stride[2] - padding[2] * 2 + dilation[2] * (W_f - 1) + 1 \quad (55)$$

式中：
W_{in}—表示是输入特征的宽度；
$padding[2]$—表示填充第三个维度大小；
$dilation[2]$—表示空洞第三个维度大小；
W_f--卷积核的宽度；
$stride[2]$—表示步长第三个维度大小；
W_{out}—表示输出张量的宽度。

7.2.6.6.2 前向接口参数

三维反卷积操作函数前向接口应符合表110，C语言示例见A.2.6.6。

表 110 三维反卷积操作函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	5维度张量，支持不同的布局格式，[N,C,D,H,W]或[N,D,H,W,C]，元素类型可以为整数
卷积核	输入	必选	表示卷积核，是一个维张量，数据类型、布局方式与张量input相同
卷积步长	输入	可选	可以为单个数或三元素数组，默认值可为1
步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1
填充数	输入	可选	表示填充元素个数，可以为单个数或三元素数组，如果为空，则表示所有维度不进行填充，默认值可为空
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空
卷积膨胀数组长度	输入	可选	表示膨胀数组长度，默认为1
分组卷积组数	输入	可选	表示进行分组卷积的组数。当值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1
输出张量	输出	必选	表示计算结果

7.2.6.6.3 前向接口返回值

没有错误：操作成功。
非法参数：表示其他参数不合法。
对象未初始化：表示输入张量对象不合法。
类型不匹配：张量的数据类型不一致。

7.2.6.6.4 后向接口参数

三维反卷积操作函数后向接口应符合表111，C语言示例见A.2.6.6。

表 111　三维反卷积操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示输出张量的梯度，输出张量是前向接口中的输出张量
卷积核	输入	必选	表示卷积核，是一个5维张量，数据类型、布局方式与张量input相同，即前向接口中的卷积核
卷积步长	输入	可选	可以为单个数或三元素数组，默认值可为1 ，即前向接口中的卷积步长
卷积步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1，即前向接口中的卷积步长数组长度
填充数	输入	可选	表示填充元素个数，可以为单个数或三元素数组，如果为空，则表示所有维度不进行填充，默认值可为空，即前向接口中的填充数
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0，即前向接口中的填充数组长度
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空，即前向接口中的卷积膨胀数组个数
卷积膨胀数组长度	输入	可选	表示膨胀数组长度，默认为1，即前向接口中的卷积膨胀数组长度
分组卷积组数	输入	可选	表示进行分组卷积的组数。当值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1，即前向接口中的分组卷积组数
输入张量梯度	输出	必选	表示输入张量的梯度，输入张量是5维度张量，支持不同的布局格式，[N,C,D,H,W]或[N,D,H,W,C]，元素类型可以为整数，即前向接口中的输入梯度

7.2.6.6.5　后向接口返回值

没有错误：操作成功。
非法参数：表示其他参数不合法。
对象未初始化：表示输入张量对象不合法。
类型不匹配：张量的数据类型不一致。

7.2.6.6.6　后向接口参数

三维反卷积操作函数后向接口应符合表112，C语言示例见A.2.6.6。

表 112　三维反卷积操作函数后向接口参数列表

参数名	类型	可选/必选	描述
输出梯度	输入	必选	表示输出张量的梯度，输出张量是前向接口中的输出张量
输入张量	输入	必选	表示输入的张量，输入张量是5维度张量，支持不同的布局格式，[N,C,D,H,W]或[N,D,H,W,C]，元素类型可以为整数，即前向接口中的输入梯度
卷积步长	输入	可选	可以为单个数或三元素数组，默认值可为1 ，即前向接口中的卷积步长
卷积步长数组长度	输入	可选	卷积步长数组长度，数值需大于0，默认为1 ，即前向接口中的卷积步长

表 112　三维反卷积操作函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
填充数	输入	可选	表示填充元素个数，可以为单个数或三元素数组，如果为空，则表示所有维度不进行填充，默认值可为空，即前向接口中的填充数
填充数组长度	输入	可选	表示填充元素数组长度，默认值为0，即前向接口中的填充数组长度
卷积膨胀个数	输入	可选	表示卷积膨胀个数，可以为单个数或两元素数组，如果为空，则表示所有维度不进行膨胀，默认值可为空，即前向接口中的卷积膨胀数组个数
卷积膨胀数组长度	输入	可选	表示膨胀数组长度，默认为1，即前向接口中的卷积膨胀数组长度
分组卷积组数	输入	可选	表示进行分组卷积的组数。当值为n，输入和滤波器分别根据通道数量平均分成n组，每组输入、滤波器分别进行卷积计算。如果为1，则表示正常卷积，默认值可为1，即前向接口中的分组卷积组数
输入卷积核梯度	输出	必选	表示卷积核的梯度，卷积核是一个维张量，数据类型、布局方式与张量input相同，即前向接口中的卷积核

7.2.6.6.7　后向接口返回值

没有错误：操作成功。
非法参数：表示其他参数不合法。
对象未初始化：表示输入张量对象不合法。
类型不匹配：张量的数据类型不一致。

7.2.7　评估函数

7.2.7.1　准确率函数

7.2.7.1.1　功能

使用输入和标签计算准确率。如果正确的标签在最大的k个预测值里，则计算结果加1。

7.2.7.1.2　接口参数

正确率函数接口应符合表113，C语言示例见A.2.7.1。

表 113　正确率函数参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示前向网络计算得到的预测概率值，形状为[D_0,…,D_(k-1),C]，其中C为类别数
标签	输入	必选	表示期望的标签值，形状为[D_0,…,D_(k-1)]，元素类型与张量input在计算上兼容
计算值个数	输入	必选	取每个类别中k个预测值用于计算
正确个数	输出	必选	表示正确预测值的个数的张量，形状为[1]
总共个数	输出	必须	表示总共预测值的个数的张量。
输出张量	输出	必选	表示正确率的张量，形状为[1]

7.2.7.1.3　接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.7.2 AUC 函数

7.2.7.2.1 功能

根据前向输出和标签计算AUC，广泛用于二分类场景。

7.2.7.2.2 接口参数

AUC函数接口应符合表114，C语言示例见A.2.7.2。

表 114 AUC 函数参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示前向网络计算得到的预测概率值，形状为[D_0,…,D_(k-1),C]，其中C为类别数且C=2。
标签	输入	必选	表示期望的标签值，形状为[D_0,…,D_(k-1)]，元素类型与张量input在计算上兼容
曲线类型	输入	必选	表示曲线类型，可设置为“ROC”（受试者工作特征曲线）或“PR”（准确率召回率曲线）
临界数值	输入	必选	将ROC曲线离散化时使用的临界数值
滑步	输入	必选	当计算批次AUC时，slide_steps=1表示用当前步，slide_steps=3表示用当前步和前两步，slide_steps=0表示用所有步。
输出张量	输出	必选	表示AUC结果的张量，形状为[1]

7.2.7.2.3 接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

7.2.8 循环网络函数

7.2.8.1 简单循环网络基本单元

7.2.8.1.1 功能

实现简单循环网络的基本单元Elman RNN Cell。假设激活函数是“tanh”，见式（56）。

$$h_t = tanh\left(weight_{i,h} \cdot x_t + bias_{i,h} + weight_{h,h} \cdot h_{t-1} + bias_{h,h}\right) \quad (56)$$

式中：

t—当前时间步；

x_t—输入张量；

$weight_{i,h}$—输入权重张量；

$bias_{i,h}$—输入偏置系数张量；

$weight_{h,h}$—隐藏状态权重张量；

$bias_{h,h}$—隐藏状态偏置系数张量；

h_{t-1}—隐藏状态张量；

h_t—下个隐藏状态张量。

7.2.8.1.2 前向接口参数

简单循环网络基本单元函数前向接口应符合表115，C语言示例见A.2.8.1。

表 115 简单循环网络基本单元函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示某个时间步的输入数据张量。形状为[batch_size, input_size]。元素类型可以为浮点数
隐藏状态张量	输入	必选	表示当前隐藏状态。形状为[batch_size, hidden_size]
输入权重张量	输入	必选	表示从输入张量到下个（时间步的）隐藏状态的权重。形状为[hidden_size, input_size]
输入偏置系数张量	输入	必选	表示从输入张量到下个隐藏状态的偏置系数。形状为[hidden_size]
隐藏状态权重张量	输入	必选	表示从当前隐藏状态到下个隐藏状态的权重。形状为[hidden_size, hidden_size]
隐藏状态偏置系数张量	输入	必选	表示从当前隐藏状态到下个隐藏状态的偏置系数。形状为[hidden_size]
激活函数	输入	可选	表示激活函数，常用的有线性整流单元(relu)和双曲线正切函数(tanh)，默认值可为空
是否训练	输入	必选	表示该接口是否用于训练。若值为“true”，则该接口用于训练；若值为“false”，则该接口用于推理
下个隐藏状态张量	输出	必选	表示下个隐藏状态。形状为[batch_size, hidden_size]

7.2.8.1.3 前向接口返回值

没有错误：操作成功。
对象未初始化：表示输入张量对象不合法。
非法参数：参数出错。
类型不匹配：表示参数的数据类型不一致。
维度不匹配：表示维度不匹配。

7.2.8.1.4 后向接口参数

简单循环网络基本单元函数后向接口应符合表116，C语言示例见A.2.8.1.。

表 116 简单循环网络基本单元函数后向接口参数列表

参数名	类型	可选/必选	描述
输出梯度	输入	必选	表示输出张量的梯度
隐藏状态张量	输入	必选	表示当前隐藏状态，形状为[batch_size, hidden_size]
输入权重张量	输入	必选	表示从输入张量到下个（时间步的）隐藏状态的权重。形状为[hidden_size, input_size]

表 116　简单循环网络基本单元函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
输入偏置系数张量	输入	必选	表示从输入张量到下个隐藏状态的偏置系数。形状为[hidden_size]
隐藏状态权重张量	输入	必选	表示从当前隐藏状态到下个隐藏状态的权重。形状为[hidden_size, hidden_size]
隐藏状态偏置系数张量	输入	必选	表示从当前隐藏状态到下个隐藏状态的偏置系数。形状为[hidden_size]
激活函数	输入	可选	表示激活函数，常用的有线性整流单元(relu)和双曲线正切函数(tanh)，默认值可为空
隐藏状态梯度	输出	必选	表示隐藏状态梯度
输入权重梯度	输出	必选	表示权重张量的梯度
输入偏置梯度	输出	必选	表示偏置的梯度
隐藏状态权重梯度	输出	必选	表示隐藏状态权重的梯度
隐藏状态偏置梯度	输出	必选	表示隐藏状态偏置的梯度
输入梯度	输出	必选	表示输入张量的梯度

7.2.8.1.5　后向接口返回值

没有错误：操作成功。
对象未初始化：表示输入张量对象不合法。
非法参数：参数出错。
类型不匹配：表示参数的数据类型不一致。
维度不匹配：表示维度不匹配。

7.2.8.1.6　其他附加说明

Elman RNN Cell结构见图1。

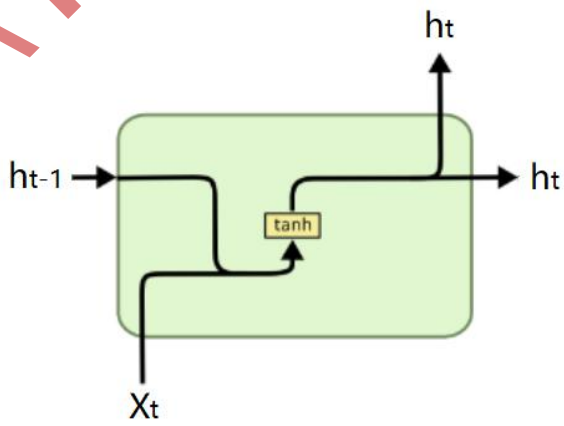

图 1　Elman RNN Cell 结构示意图

7.2.8.2　简单循环网络

7.2.8.2.1 功能

实现Elman RNN。每一层是一个独立的Elman RNN Cell，其中每个时间步隐藏状态的计算参考Elman RNN Cell接口。

7.2.8.2.2 前向接口参数

简单循环网络基本单元函数前向接口应符合表117，C语言示例见A.2.8.2。

表 117 简单循环网络函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示某个时间步的输入数据张量。形状为[sequence_length,batch_size, input_size]。元素类型可以为浮点数
隐藏状态张量数组	输入	必选	表示当前隐藏状态。数组长度为num_layers，每层形状为[num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同
输入权重张量数组	输入	必选	表示从输入张量到下个（时间步的）隐藏状态的权重张量数组。数组长度为num_layers，每层形状为[hidden_size, input_size]，其中每层的hidden_size不一定相同
输入偏置系数张量数组	输入	必选	表示从输入张量到下个隐藏状态的偏置系数。数组长度为num_layers，每层形状为[hidden_size]，其中每层的hidden_size不一定相同
隐藏状态权重张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的权重。数组长度为num_layers，每层形状为[hidden_size, hidden_size]，其中每层的hidden_size不一定相同
隐藏状态偏置系数张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的偏置系数。数组长度为num_layers，每层形状为[hidden_size]，其中每层的hidden_size不一定相同
循环网络层数	输入	必选	表示网络的层数
是否双向	输入	必选	表示网络是否为双向
随机失活率	输入	可选	表示随机失活率，如果为0，则随机失活失效；如果非零，则在每一层后加上随机失活率对应的随机失活层，默认值可为0
激活函数	输入	可选	表示激活函数，常用的有线性整流单元(relu)和双曲线正切函数(tanh)，默认值可为空
是否训练	输入	必选	表示该接口是否用于训练。若值为“true”，则该接口用于训练；若值为“false”，则该接口用于推理
随机失活掩码	输出	必选	只在随机失活率不为0时有效，保存对应掩码
最后一层的隐藏状态张量	输出	必选	表示网络最后一层所有时间步的隐藏状态张量。用于构建损失函数，其形状为[sequence_length, num_directions, batch_size, hidden_size]
下个隐藏状态张量数组	输出	必选	表示网络每层的最后时间步的隐藏状态张量数组。长度为num_layers，每层形状为[num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同

7.2.8.2.3 前向接口返回值

没有错误：操作成功。

类型不匹配：张量的数据类型不一致。

维度不匹配：参数的维度不一致。

7.2.8.2.4 后向接口参数

简单循环网络基本单元函数后向接口应符合表118，C语言示例见A.2.8.2。

表 118 简单循环网络函数后向接口参数列表

参数名	类型	可选/必选	描述
输出梯度	输入	必选	表示输出张量的梯度
隐藏状态张量数组	输入	必选	表示当前隐藏状态。数组长度为num_layers，每层形状为[num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同
输入权重张量数组	输入	必选	表示当前隐藏状态。数组长度为num_layers，每层形状为[num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同
输入偏置系数张量数组	输入	必选	表示从输入张量到下个隐藏状态的偏置系数。数组长度为num_layers，每层形状为[hidden_size]，其中每层的hidden_size不一定相同
隐藏状态权重张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的权重。数组长度为num_layers，每层形状为[hidden_size, hidden_size]，其中每层的hidden_size不一定相同
隐藏状态偏置系数张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的偏置系数。数组长度为num_layers，每层形状为[hidden_size]，其中每层的hidden_size不一定相同
随机失活掩码	输入	可选	随机失活层对应掩码，默认值可为空
循环网络层数	输入	必选	表示网络的层数
是否双向	输入	必选	表示网络是否为双向，即前向接口中的是否双向
激活函数	输入	可选	表示激活函数，常用的有线性整流单元(relu)和双曲线正切函数(tanh)，默认值可为空
下个隐藏状态张量数组	输入	必选	表示网络每层的最后时间步的隐藏状态张量数组。长度为num_layers，每层形状为[num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同
隐藏状态梯度数组	输出	必选	表示隐藏状态梯度数组
输入权重梯度数组	输出	必选	表示权重张量的梯度数组
输入偏置梯度数组	输出	必选	表示偏置的梯度数组
隐藏状态权重梯度数组	输出	必选	表示隐藏状态权重的梯度数组

表 118　简单循环网络函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
隐藏状态偏置梯度数组	输出	必选	隐藏状态偏置的梯度数组
输入梯度数组	输出	必选	表示输入张量的梯度数组

7.2.8.2.5　后向接口返回值

没有错误：操作成功。
类型不匹配：张量的数据类型不一致。
维度不匹配：参数的维度不一致。

7.2.8.2.6　其他附加说明

单向Elman RNN结构见图2。

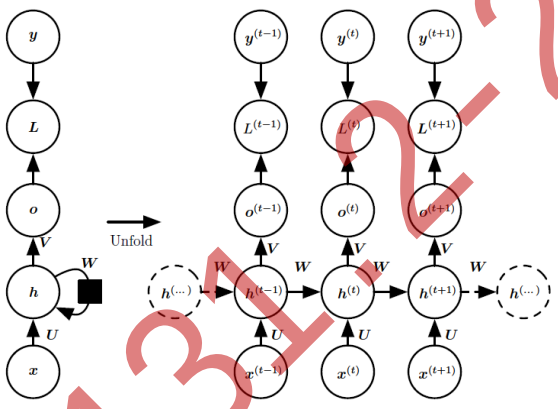

图 2　单向 Elman RNN 结构示意图

双向Elman RNN结构见图3。

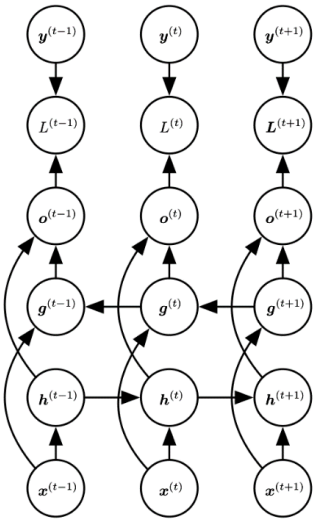

图 3　双向 Elman RNN 结构示意图

此接口不包含图中网络最后的o层和损失函数，输出仅为最后一层的隐藏状态g和h。此接口默认同一层的两个隐藏状态g和h的形状是相同的，因此最后的输出合并到同一个张量中。

7.2.8.3 长短期记忆网络基本单元

7.2.8.3.1 功能

实现长短期记忆网络的基本单元LSTM Cell。假设激活函数为“tanh”，循环激活函数为“sigmoid”，各个门的计算方法见式（57）-式（62）。

$$i_t = sigmoid(W_{ii}x_t + b_{xi} + W_{hi}h_{(t-1)} + b_{hi}) \quad (57)$$

式中：

t—当前时间步；

x_t—输入张量；

W_{ii}—输入门和输入门的权重；

W_{hi}—隐藏状态和输入门的权重；

$h_{(t-1)}$—第t-1个时间步的隐藏状态；

b_{xi}—输入张量和输入门的偏置值向量；

b_{hi}—隐藏状态和输入门的偏置值向量；

i_t—输入门。

$$f_t = sigmoid(W_{if}x_t + b_{xf} + W_{hf}h_{(t-1)} + b_{hf}) \quad (58)$$

式中：

t—当前时间步；

x_t—输入张量；

W_{if}—输入门和遗忘门的权重；

W_{hf}—隐藏状态和遗忘门的权重；

$h_{(t-1)}$—第t-1个时间步的隐藏状态；

b_{xf}—输入张量和遗忘门的偏置值向量；

b_{hf}—隐藏状态和遗忘门的偏置值向量；

f_t—遗忘门；

$$g_t = tanh(W_{ig}x_t + b_{xg} + W_{hg}h_{(t-1)} + b_{hg}) \quad (59)$$

式中：

t—当前时间步；

x_t—输入张量；

W_{ig}—输入门和胞元门的权重；

W_{hg}—隐藏状态和胞元门的权重；

$h_{(t-1)}$—第t-1个时间步的隐藏状态；

b_{xg}—输入张量和胞元门的偏置值向量；

b_{hg}—隐藏状态和胞元门的偏置值向量；

g_t—胞元门。

$$o_t = sigmoid(W_{io}x_t + b_{xo} + W_{ho}h_{(t-1)} + b_{ho}) \quad (60)$$

式中：

t—当前时间步；

x_t—输入张量；
W_{io}—输入门和输出门的权重；
W_{ho}—隐藏状态和输出门的权重；
$h_{(t-1)}$—第t-1个时间步的隐藏状态；
b_{xo}—输入张量和输出门的偏置值向量；
b_{ho}—隐藏状态和输出门的偏置值向量；
o_t—输出门。

$$c_t = f_t \odot c_{(t-1)} + i_t \odot g_t \quad (61)$$

式中：
t—当前时间步；
f_t—遗忘门；
$c_{(t-1)}$—第t-1个时间步的胞元状态；
i_t—输入门；
g_t—胞元门；
⊙—Hadamard积；
c_t—第t个时间步的胞元状态。

$$h_t = o_t \odot tanh(c_t) \quad (62)$$

式中：
t—当前时间步；
o_t—输出门；
c_t—第t个时间步的胞元状态；
⊙—Hadamard积；
h_t—第t个时间步的隐藏状态。

7.2.8.3.2 前向接口参数

长短期记忆单元函数前向接口应符合表119，C语言示例见A.2.8.3。

表 119 长短期记忆单元函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示某个时间步的输入数据张量。形状为[batch_size, input_size]。元素类型可以为浮点数
当前状态张量	输入	必选	表示当前状态。按顺序包括隐藏状态h_{t-1}和胞元状态c_{t-1}。其形状为[2, batch_size, hidden_size]
输入权重张量	输入	必选	表示各个门中从输入张量到下组状态的权重，按顺序包括输入门、遗忘门、胞元门和输出门，形状为[4, hidden_size, input_size]
输入偏置系数张量	输入	必选	表示各个门中从输入张量到下组状态的偏置值，按顺序包括输入门、遗忘门、胞元门和输出门，形状为[4, hidden_size]
状态权重张量	输入	必选	表示各个门中从当前状态到下组状态的权重，按顺序包括输入门、遗忘门、胞元门和输出门，形状为[4, hidden_size, hidden_size]

表 119　长短期记忆单元函数前向接口参数列表（续）

参数名	类型	可选/必选	描述
状态偏置系数张量	输入	必选	表示各个门中从当前状态到下组状态的偏置值，按顺序包括输入门、遗忘门、胞元门和输出门，形状为[4, hidden_size]
激活函数	输入	可选	表示激活函数，常用的有双曲线正切函数(tanh)等。默认值：空，表示无激活函数
循环激活函数	输入	可选	表示循环所用的激活函数。常用的函数有sigmoid等。默认值：空，表示无激活函数
是否训练	输入	必选	表示该接口是否用于训练。若值为“true”，则该接口用于训练；若值为“false”，则该接口用于推理
下组状态张量	输出	必选	表示下组状态。形状为[batch_size, hidden_size]

7.2.8.3.3　前向接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。
维度不匹配：参数的维度不一致。

7.2.8.3.4　后向接口参数

长短期记忆单元函数后向接口应符合表120，C语言示例见A.2.8.3。

表 120　长短期记忆单元函数后向接口参数列表

参数名	类型	可选/必选	描述
输出梯度	输入	必选	表示输出张量的梯度
当前状态张量	输入	必选	表示当前状态。按顺序包括隐藏状态h_{t-1}和胞元状态c_{t-1}。其形状为[2, batch_size, hidden_size]。即前向接口中当前状态张量
输入权重张量	输入	必选	表示各个门中从输入张量到下组状态的权重，按顺序包括输入门、遗忘门、胞元门和输出门，形状为[4, hidden_size, input_size]。即前向接口中输入权重张量
输入偏置系数张量	输入	必选	表示各个门中从输入张量到下组状态的偏置值，按顺序包括输入门、遗忘门、胞元门和输出门，形状为[4, hidden_size]。即前向接口中输入偏置系数张量
状态权重张量	输入	可选	表示各个门中从当前状态到下组状态的权重，按顺序包括输入门、遗忘门、胞元门和输出门，形状为[4, hidden_size, hidden_size]。即前向接口中状态权重张量
状态偏置系数张量	输入	可选	表示各个门中从当前状态到下组状态的偏置值，按顺序包括输入门、遗忘门、胞元门和输出门，形状为[4, hidden_size]。即前向接口中状态偏置系数张量
激活函数	输入	可选	表示激活函数，常用的有双曲线正切函数(tanh)等。即前向接口中激活函数。默认值：空，表示无激活函数
循环激活函数	输入	可选	表示循环所用的激活函数。常用的函数有sigmoid等。表示前向接口中循环激活函数。默认值：空，表示无循环激活函数

表 120　长短期记忆单元函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
当前状态张量梯度	输出	必选	表示表示前向接口中当前状态张量梯度
输入权重张量梯度	输出	必选	表示表示前向接口中权重张量的梯度
输入偏置系数张量梯度	输出	必选	表示表示前向接口中偏置系数张量的梯度
状态权重张量梯度	输出	必选	表示表示前向接口中状态权重张量的梯度
状态偏置系数张量梯度	输出	必选	表示前向接口中状态偏置系数张量的梯度
输入张量梯度	输出	必选	表示输入张量的梯度

7.2.8.3.5　后向接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。
维度不匹配：参数的维度不一致。

7.2.8.3.6　其他附加说明

在LSTM Cell里输入张量的权重中，有W_{ii}、W_{if}、W_{ig}、W_{io}，分别对应着输入门、遗忘门、胞元门、输出门，它们的形状是相同的，所以可以合并到一起作为一个输入，而不需要分开成4个输入。同理，状态的4个权重也是形状相同的，可以合并成一个输入。但是，输入张量的权重与状态的权重有不一样的形状，如W_{ii}和W_{hi}可能有不一样的形状，因此不能把两者的权重合并表示。

LSTM Cell结构见图4。

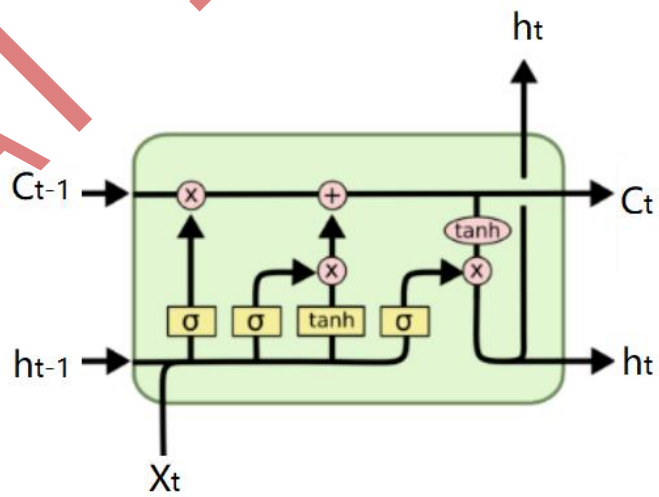

图 4　LSTM Cell 结构示意图

7.2.8.4　长短期记忆网络

7.2.8.4.1　功能

实现长短期记忆网络。每一层是一个独立的长短期记忆网络基本单元(LSTM Cell)，其中每个时间步各个状态的计算参考LSTM Cell接口。

7.2.8.4.2 前向接口参数

长短期记忆网络函数前向接口应符合表121，C语言示例见A.2.8.4。

表 121 长短期记忆网络函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示某个时间步的输入数据张量。形状为[sequence_length, batch_size, input_size]。元素类型可以为浮点数
初始隐藏状态张量数组	输入	必选	表示初始的状态张量数组，包括隐藏状态和胞元状态，长度为num_layers，其的形状为[2, num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同
输入权重张量数组	输入	必选	表示从输入张量到下个（时间步的）隐藏状态的权重张量数组。数组长度为num_layers，每层形状为[4, hidden_size, input_size] ，其中每层的hidden_size不一定相同
输入偏置系数张量数组	输入	必选	表示从输入张量到下个隐藏状态的偏置系数张量数组 。数组长度为num_layers，每层形状为[4, hidden_size, hidden_size] ，其中每层的hidden_size不一定相同
隐藏状态权重张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的权重。数组长度为num_layers，每层形状为[hidden_size, hidden_size]，其中每层的hidden_size不一定相同
隐藏状态偏置系数张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的偏置系数张量数组。数组长度为num_layers，每层形状为[4, hidden_size]，其中每层的hidden_size不一定相同
循环网络层数	输入	必选	表示网络的层数
是否双向	输入	必选	表示网络是否为双向
随机失活率	输入	可选	表示随机失活率，如果为0，则随机失活失效；如果非零，则在每一层后加上随机失活率对应的随机失活层，默认值可为0
激活函数	输入	可选	表示激活函数，常用的有双曲线正切函数(tanh)等。默认值：空，表示无激活函数
循环激活函数	输入	可选	表示用于循环的激活函数。默认值：空，表示无循环激活函数
是否训练	输入	必选	表示该接口是否用于训练。若值为“true”，则该接口用于训练；若值为“false”，则该接口用于推理
最后一层的隐藏状态	输出	必选	表示网络最后一层所有时间步的隐藏状态张量。用于构建损失函数，其形状为[sequence_length, 2, num_directions, batch_size, hidden_size]
随机失活掩码	输出	必选	只在随机失活率不为0时有效，保存对应掩码
下个隐藏状态张量数组	输出	必选	表示网络每层的最后时间步的隐藏状态张量数组。长度为num_layers，每层形状为[2, num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同

7.2.8.4.3 前向接口返回值

没有错误：表示操作成功。

类型不匹配：表示参数的数据类型不一致。

维度不匹配：参数的维度不一致。

7.2.8.4.4 后向接口参数

长短期记忆网络函数后向接口应符合表122，C语言示例见A.2.8.4。

表 122 长短期记忆网络函数后向接口参数列表

参数名	类型	可选/必选	描述
输出梯度	输入	必选	表示输出张量的梯度
初始隐藏状态张量数组	输入	必选	表示初始的状态张量数组，包括隐藏状态和胞元状态，长度为num_layers，其的形状为[2, num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同。即前向接口中的初始隐藏状态张量数组
输入权重张量数组	输入	必选	表示从输入张量到下个（时间步的）隐藏状态的权重张量数组。数组长度为num_layers，每层形状为[4, hidden_size, input_size] ，其中每层的hidden_size不一定相同。即前向接口中的输入权重张量数组
输入偏置系数张量数组	输入	必选	表示从输入张量到下个隐藏状态的偏置系数张量数组 。数组长度为num_layers，每层形状为[4, hidden_size, hidden_size] ，其中每层的hidden_size不一定相同。即前向接口中的输入偏置系数张量数组
隐藏状态权重张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的权重。数组长度为num_layers，每层形状为[hidden_size, hidden_size]，其中每层的hidden_size不一定相同。即前向接口中的隐藏状态权重张量数组
隐藏状态偏置系数张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的偏置系数张量数组。数组长度为num_layers，每层形状为[4, hidden_size]，其中每层的hidden_size不一定相同。即前向接口中的隐藏状态偏置系数张量数组
激活函数	输入	可选	表示激活函数，常用的有双曲线正切函数(tanh)等。即前向接口中的激活函数。默认值：空，表示无激活函数
循环激活函数	输入	可选	表示用于循环的激活函数。即前向接口中的循环激活函数。默认值：空，表示无循环激活函数
随机失活掩码	输入	必选	只在随机失活率不为0时有效，保存对应掩码。即前向接口中的随机失活掩码
下个隐藏状态张量数组	输入	必选	表示网络每层的最后时间步的隐藏状态张量数组。长度为num_layers，每层形状为[2, num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同。即前向接口中的下个隐藏状态张量数组
循环网络层数	输入	必选	表示网络的层数。即前向接口中的循环网络层数
是否双向	输入	必选	表示网络是否为双向。即前向接口中的是否双向
初始隐藏状态张量数组梯度	输出	必选	表示前向接口中初始隐藏状态张量数组的梯度
输入权重张量数组梯度	输出	必选	表示前向接口中输入权重张量数组数组的梯度

表 122　长短期记忆网络函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
输入偏置系数张量数组梯度	输出	必选	表示前向接口中输入偏置系数张量数组的梯度
隐藏状态权重权重张量数组梯度	输出	必选	表示前向接口中隐藏状态权重权重张量数组的梯度
隐藏状态偏置系数张量数组梯度	输出	必选	表示前向接口中隐藏状态偏置系数张量数组的梯度
输入梯度数组	输出	必选	表示输入梯度。

7.2.8.4.5　后向接口返回值

没有错误：表示操作成功。

类型不匹配：表示参数的数据类型不一致。

维度不匹配：参数的维度不一致。

7.2.8.4.6　其他附加说明

单向LSTM结构见图5。

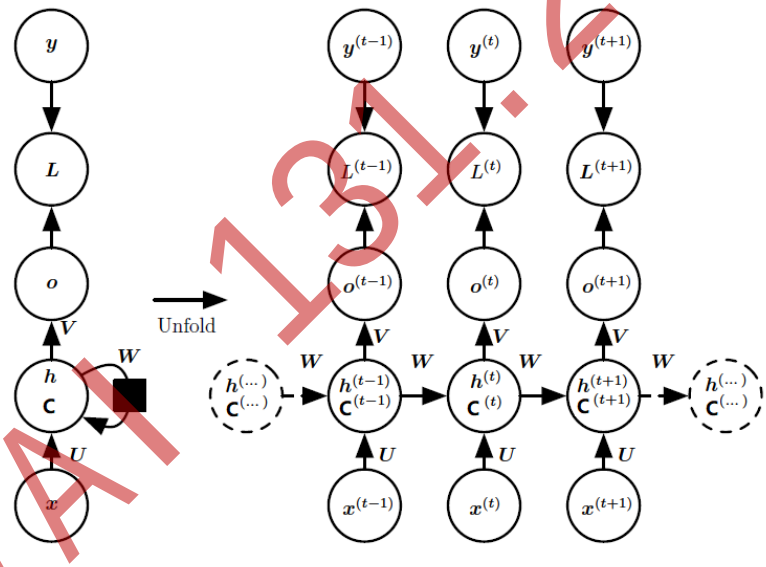

图 5　单向 LSTM 结构示意图

双向LSTM结构见图6。

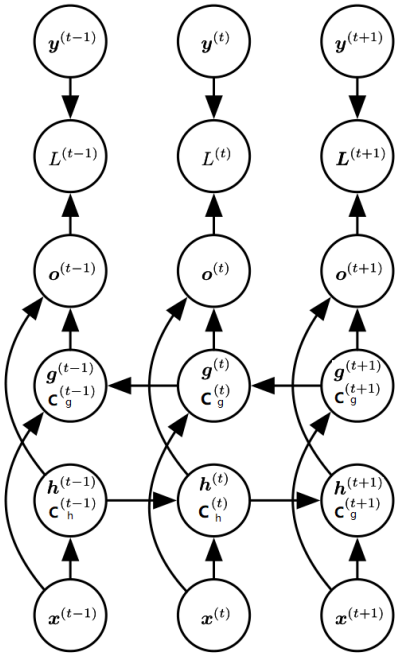

图 6　双向 LSTM 结构示意图

此接口不包含图中网络最后的o层和损失函数，输出仅为最后一层的隐藏状态g和h。此接口默认同一层的两个隐藏状态g和h的形状是相同的，因此最后的输出合并到同一个张量中。

7.2.8.5　门控循环单元网络基本单元

7.2.8.5.1　功能

实现门控循环单元网络的基本单元GRU Cell。假设激活函数为“tanh”，循环激活函数为“sigmoid”，各个门的计算方法见式（63）-式（66）。

$$r_t = sigmoid(W_{ir}x_t + b_{xr} + W_{hr}h_{(t-1)} + b_{hr}) \quad \text{（63）}$$

式中：

t—当前时间步；

x_t—输入张量；

W_{ir}—输入门和重置门的权重；

W_{hr}—隐藏状态和重置门的权重；

b_{xr}—输入张量和重置门的偏置系数张量；

b_{hr}—隐藏状态和重置门的偏置系数张量；

$h_{(t-1)}$—第t-1个时间步的隐藏状态；

r_t—重置门。

$$i_t = sigmoid(W_{ii}x_t + b_{xi} + W_{hi}h_{(t-1)} + b_{hi}) \quad \text{（64）}$$

式中：

t—当前时间步；

x_t—输入张量；

W_{ii}—输入门和输入门的权重；

W_{hi}—隐藏状态和输入门的权重；

b_{xi}—输入张量和输入门的偏置系数张量；

b_{hi}—隐藏状态和输入门的偏置系数张量；
$h_{(t-1)}$—第t-1个时间步的隐藏状态；
i_t—输入门。

$$n_t = tanh(W_{in}x_t + b_{xn} + r_t \odot (W_{hn}h_{(t-1)} + b_{hn})) \quad (65)$$

式中：
t—当前时间步；
x_t—输入张量；
W_{in}—输入门和更新门的权重；
W_{hn}—隐藏状态和更新门的权重；
b_{xn}—输入张量和更新门的偏置系数张量；
b_{hn}—隐藏状态和更新门的偏置系数张量；
$h_{(t-1)}$—第t-1个时间步的隐藏状态；
r_t—重置门；
⊙—Hadamard积；
n_t—更新门。

$$h_t = (1 - i_t) \odot n_t + i_t \odot h_{(t-1)} \quad (66)$$

式中：
t—当前时间步；
i_t—输入门；
n_t—更新门；
$h_{(t-1)}$—第t-1个时间步的隐藏状态；
⊙—Hadamard积；
h_t—第t个时间步的隐藏状态。

7.2.8.5.2 前向接口参数

门控循环单元网络基本单元函数前向接口应符合表123，C语言示例见A.2.8.5。

表 123 门控循环单元网络基本单元函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示某个时间步的输入数据张量。形状为[batch_size, input_size]。元素类型可以为浮点数
上一个状态张量	输入	必选	表示上一个隐藏状态h_{t-1}。其形状为[batch_size, hidden_size]
输入权重张量数组	输入	必选	表示各个门中从输入张量到下组状态的权重，按顺序包括重置门、输入门和更新门，形状为[3, hidden_size, input_size]
输入偏置系数张量数组	输入	必选	表示各个门中从输入张量到下组状态的偏置值，按顺序包括重置门、输入门和更新门，形状为[3, hidden_size]
状态权重张量数组	输入	必选	表示各个门中从当前状态到下组状态的权重，按顺序包括输入门、遗忘门、胞元门和输出门，形状为[4, hidden_size, hidden_size]

表 123　门控循环单元网络基本单元函数前向接口参数列表（续）

参数名	类型	可选/必选	描述
状态偏置系数张量数组	输入	必选	表示各个门中从当前状态到下组状态的偏置值，按顺序包括重置门、输入门和更新门，形状为[3, hidden_size, hidden_size]
激活函数	输入	可选	表示激活函数，常用的有双曲线正切函数(tanh)等。默认值：空，表示无激活函数
循环激活函数	输入	可选	表示循环所用的激活函数。常用的函数有sigmoid等。默认值：空，表示无循环激活函数
是否训练	输入	必选	表示该接口是否用于训练。若值为“true”，则该接口用于训练；若值为“false”，则该接口用于推理
输出张量	输出	必选	表示输出张量，即隐藏状态h_t。形状为[batch_size, hidden_size]

7.2.8.5.3　前向接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。
维度不匹配：表示维度不匹配。

7.2.8.5.4　后向接口参数

门控循环单元网络基本单元函数后向接口应符合表124，C语言示例见A.2.8.5。

表 124　门控循环单元网络基本单元函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示前向接口中输出张量的梯度。输出张量即隐藏状态h_t。形状为[batch_size, hidden_size]
上一个状态张量	输入	必选	表示上一个隐藏状态h_{t-1}。其形状为[batch_size, hidden_size]。即前向接口中上一个状态张量
输入权重张量数组	输入	必选	表示各个门中从输入张量到下组状态的权重，按顺序包括重置门、输入门和更新门，形状为[3, hidden_size, input_size]。即前向接口中输入权重张量数组
输入偏置系数张量数组	输入	必选	表示各个门中从输入张量到下组状态的偏置值，按顺序包括重置门、输入门和更新门，形状为[3, hidden_size]。即前向接口中输入偏置系数张量数组
状态权重张量数组	输入	必选	表示各个门中从当前状态到下组状态的权重，按顺序包括输入门、遗忘门、胞元门和输出门，形状为[4, hidden_size, hidden_size]。即前向接口中状态权重张量数组
状态偏置系数张量数组	输入	必选	表示各个门中从当前状态到下组状态的偏置值，按顺序包括重置门、输入门和更新门，形状为[3, hidden_size, hidden_size]。即前向接口中输入偏置系数张量数组
激活函数	输入	可选	表示激活函数，常用的有双曲线正切函数(tanh)等。即前向接口中激活函数。默认值：空，表示无激活函数

表 124　门控循环单元网络基本单元函数后向接口参数列表（续）

参数名	类型	可选/必选	描述
循环激活函数	输入	可选	表示循环所用的激活函数。常用的函数有sigmoid等。即前向接口中循环激活函数。默认值：空，表示无循环激活函数
上一个状态张量梯度	输出	必选	表示前向接口中上一个状态张量梯度的梯度。
输入权重张量数组梯度	输出	必选	表示前向接口中输入权重对应张量数组的梯度
输入偏置系数张量数组梯度	输出	必选	表示前向接口中输入偏置对应系数张量数组的梯度
状态权重张量数组梯度	输出	必选	表示前向接口中状态权重张量数组的梯度
状态偏置系数张量数组梯度	输出	必选	表示前向接口中状态偏置系数张量数组的梯度
输入张量梯度	输出	必选	表示前向接口中表示输入张量梯度

7.2.8.5.5　后向接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。
维度不匹配：表示维度不匹配。

7.2.8.5.6　其他附加说明

在GRU Cell里对应x的权重中，有W_{ir}、W_{ii}、W_{in}，分别对应着重置门、输入门和更新门，它们的形状是相同的，所以可以合并到一起作为一个输入，而不需要分开成3个输入。同理，对应hidden的3个权重也是形状相同的，可以合并成一个输入。但是，对应input的权重与对应hidden的权重有不一样的形状，如W_{ir}和W_{hr}可能有不一样的形状，因此不能把对应input的权重与对应hidden的权重合并。

GRU Cell结构见图7。

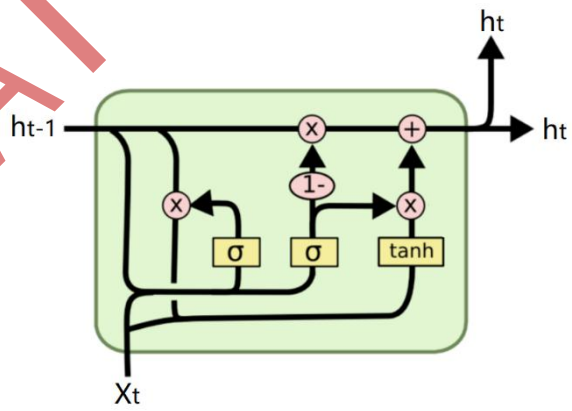

图 7　GRU Cell 结构示意图

7.2.8.6　门控循环单元网络

7.2.8.6.1　功能

实现门控循环单元网络。每一层是一个独立的GRU Cell，其中每个时间步各个状态的计算参考GRU Cell接口。

7.2.8.6.2 前向接口参数

门控循环单元网络函数前向接口应符合表125，C语言示例见A.2.8.6。

表 125 门控循环单元网络函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	表示某个时间步的输入数据张量。形状为[sequence_length, batch_size, input_size]。元素类型可以为浮点数
初始隐藏状态张量数组	输入	必选	表示初始的状态张量数组，每层形状为[num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同
输入权重张量数组	输入	必选	表示从输入张量到下个（时间步的）隐藏状态的权重张量数组。数组长度为num_layers，每层形状为[3, hidden_size, input_size]
输入偏置系数张量数组	输入	必选	表示从输入张量到下个隐藏状态的偏置系数张量数组 。数组长度为num_layers，每层形状为[3, hidden_size]
隐藏状态权重张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的权重。数组长度为num_layers，每层形状为[3, hidden_size]
隐藏状态偏置系数张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的偏置系数张量数组。数组长度为num_layers，每层形状为[3, hidden_size]
循环网络层数	输入	必选	表示网络的层数
是否双向	输入	可选	表示网络是否为双向，默认值可为否
随机失活率	输入	可选	表示随机失活率，如果为0，则随机失活失效；如果非零，则在每一层后加上随机失活率对应的随机失活层，默认值可为0
激活函数	输入	可选	表示激活函数，常用的有双曲线正切函数(tanh)等。默认值：空，表示无激活函数
循环激活函数	输入	可选	表示用于循环的激活函数。默认值：空，表示无循环激活函数
是否训练	输入	必选	表示该接口是否用于训练。若值为“true”，则该接口用于训练；若值为“false”，则该接口用于推理
随机失活掩码	输出	必选	只在随机失活率不为0时有效，保存对应掩码
最后一层的隐藏状态	输出	必选	表示网络最后一层所有时间步的隐藏状态张量。用于构建损失函数，其形状为[sequence_length, num_directions, batch_size, hidden_size]
下个隐藏状态张量数组	输出	必选	表示网络每层的最后时间步的隐藏状态张量数组。长度为num_layers，每层形状为[num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同

7.2.8.6.3 前向接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。
维度不匹配：表示维度不匹配。

7.2.8.6.4 后向接口参数

门控循环单元网络函数后向接口应符合表126，C语言示例见A.2.8.6。

表 126　门控循环单元网络函数后向接口参数列表

参数名	类型	可选/必选	描述
输出梯度	输入	必选	表示输出张量的梯度
初始隐藏状态张量数组	输入	必选	表示初始的状态张量数组，每层形状为[num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同。即前向接口中的初始隐藏状态张量数组
输入权重张量数组	输入	必选	表示从输入张量到下个（时间步的）隐藏状态的权重张量数组。数组长度为num_layers，每层形状为[3, hidden_size, input_size]。即前向接口中的输入权重张量数组
输入偏置系数张量数组	输入	必选	表示从输入张量到下个隐藏状态的偏置系数张量数组 。数组长度为num_layers，每层形状为[3, hidden_size]。即前向接口中的输入偏置系数张量数组
隐藏状态权重张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的权重。数组长度为num_layers，每层形状为[3, hidden_size]。即前向接口中的隐藏状态权重张量数组
隐藏状态偏置系数张量数组	输入	必选	表示从当前隐藏状态到下个隐藏状态的偏置系数张量数组。数组长度为num_layers，每层形状为[3, hidden_size]。即前向接口中的隐藏状态偏置系数张量数组
循环网络层数	输入	必选	表示网络的层数。即前向接口中的循环网络层数
下个隐藏状态张量数组	输入	必选	表示网络每层的最后时间步的隐藏状态张量数组。长度为num_layers，每层形状为[num_directions, batch_size, hidden_size]，其中每层的hidden_size不一定相同。即前向接口中的下个隐藏状态张量数组
是否双向	输入	可选	表示网络是否为双向。即前向接口中的是否双向。默认值可为否
随机失活掩码	输入	可选	表示随机失活的掩码。即前向接口中的随机失活掩码
激活函数	输入	可选	表示激活函数，常用的有双曲线正切函数(tanh)等。即前向接口中的激活函数。默认值：空，表示无激活函数
循环激活函数	输入	可选	表示用于循环的激活函数。即前向接口中的循环激活函数。默认值：空，表示无循环激活函数
初始隐藏状态张量梯度数组	输出	必选	表示初始状态对应的梯度
输入权重梯度数组	输出	必选	表示输入权重对应的梯度
输入偏置梯度数组	输出	必选	表示输入偏置对应的梯度
隐藏状态权重梯度数组	输出	必选	表示隐藏状态权重的梯度
隐藏状态偏置梯度数组	输出	必选	表示隐藏状态偏置的梯度
输入梯度	输出	必选	表示输入梯度

7.2.8.6.5　后向接口返回值

没有错误：表示操作成功。

类型不匹配：表示参数的数据类型不一致。
维度不匹配：表示维度不匹配。

7.2.8.6.6 其他附加说明

单向GRU结构见图8，双向GRU结构见图9。

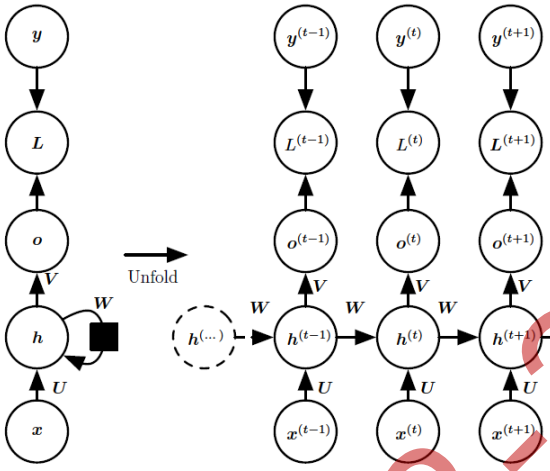

图 8 单向 GRU 结构示意图

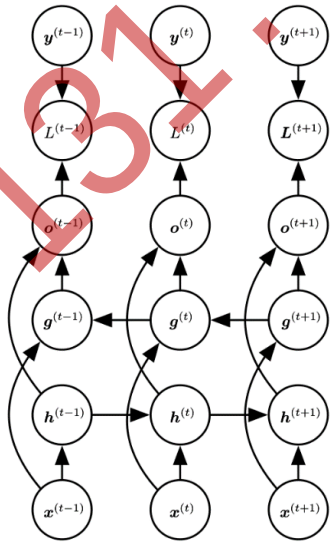

图 9 双向 GRU 结构示意图

此接口不包含图中网络最后的o层和损失函数，输出仅为最后一层的隐藏状态g和h。此接口默认同一层的两个隐藏状态g和h的形状是相同的，因此最后的输出合并到同一个张量中。

7.2.9 编码操作

7.2.9.1 词嵌入操作

7.2.9.1.1 功能

将输入表示单词索引的下标张量，通过对应的词向量权重，映射成最终输出的词向量表示

7.2.9.1.2 前向接口参数

词嵌入函数前向接口应符合表127，C语言示例见A.2.9.1。

表 127 词嵌入函数前向接口参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	输入的索引张量，元素类型可以为整数，约定其形状为[*]，任意形状。
权重张量	输入 输出	必选	embedding操作的权重，形状为[可能的最大索引数+ 1，嵌入大小]的张量。在有范数上界的约束时，会对原始输入权重进行修改，使得嵌入向量满足约束要求。
范数上界	输入	可选	规定每个嵌入向量的范数不应超过范数上界，并对范数大于该值的每个嵌入向量重新进行规范化。在范数为-1时，该参数不起作用，表示不对范数上界进行约束，默认值可为-1
范数	输入	可选	计算范数的参数，当该参数为-1时，表示不对范数上界进行约束，默认值可为-1
输出张量	输出	必选	嵌入表示的结果，与输入张量数据类型相同，形状为[*，嵌入大小]的张量。

7.2.9.1.3 前向接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。
维度不匹配：表示维度不匹配。

7.2.9.1.4 后向接口参数

词嵌入函数后向接口应符合表128，C语言示例见A.2.9.1。

表 128 词嵌入函数后向接口参数列表

参数名	类型	可选/必选	描述
输出张量梯度	输入	必选	表示前向接口中输出张量的梯度。输出张量是嵌入表示的结果，与输入张量数据类型相同，形状为[*，嵌入大小]的张量。
权重张量	输入	必选	embedding操作的权重，形状为[可能的最大索引数+ 1，嵌入大小]的张量。在有范数上界的约束时，会对原始输入权重进行修改，使得嵌入向量满足约束要求。即前向接口中权重张量数组
权重张量梯度	输出	必选	表示前向接口中权重张量的梯度。权重张量是embedding操作的权重，形状为[可能的最大索引数+ 1，嵌入大小]的张量。在有范数上界的约束时，会对原始输入权重进行修改，使得嵌入向量满足约束要求。即前向接口中权重张量数组
输入张量梯度	输出	必选	表示前向接口中输入张量的梯度。输入张量是词嵌入操作输入的索引张量，元素类型可以为整数，约定其形状为[*]，任意形状

7.2.9.1.5 后向接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。

维度不匹配：表示维度不匹配。

7.2.9.2 独热编码操作

7.2.9.2.1 功能

对输入的张量进行独热编码，输入的张量为标签张量，输出张量中最后一个维度的下标与输入张量值相同的位置对应的值为1，其余均为0。

7.2.9.2.2 接口参数

独热编码操作函数接口应符合表129，C语言示例见A.2.9.2。

表 129 独热编码操作函数参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	标签张量，元素类型可以整数，值非负，约定其形状为[*]，任意形状。
类别数	输入	必选	独热编码的最后一个维度的大小，不得小于输入标签张量中的最大值。
输出张量	输出	必选	与输入张量数据类型相同，形状为[*，类别数]的张量。

7.2.9.2.3 接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。
非法参数：表示类别数参数不满足要求或者输入张量有负值。

7.2.10 距离函数

7.2.10.1 余弦相似度函数

7.2.10.1.1 功能

沿指定维度计算两输入张量的余弦相似度，见式（67）。

$$\text{similarity} = \frac{x_1 \cdot x_2}{\max\left(\|x_1\|_2 \cdot \|x_2\|_2, \varepsilon\right)} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (67)$$

式中：
x_1—第一个输入张量；
x_2—第二个输入张量；
$\|x\|_2$—张量x的二范数；
ε—小数，防止除数为0，默认值可为1e-8；

$similarity$—余弦相似度。

7.2.10.1.2 接口参数

余弦相似度函数接口应符合表130，C语言示例见A.2.10.1。

表 130　余弦相似度函数参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	第一个输入张量，元素类型可以为浮点数，维度为$[*_1,C,*_2]$
输入张量	输入	必选	第二个输入张量，形状、数据类型与第一个输入张量相同
计算轴	输入	可选	指定维度，默认值可为1
小数	输入	可选	防止除数为0的小数字，默认值可为1e-8
输出张量	输出	必选	输出张量，数据类型与输入张量相同，维度为$[*_1,*_2]$

7.2.10.1.3　接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。
非法参数：表示计算轴参数不满足要求。

7.2.11　视觉函数

7.2.11.1　网格插值采样函数

7.2.11.1.1　功能

基于flow field网格，对输入张量进行插值采样。网格通常由放射变换函数生成， 形状为[N， H, W, 2]，是形状为[N, H, W]的采样点张量(x, y)的坐标。其中，x坐标是对输入张量的第四个维度(宽度维度)的索引，y坐标是对输入张量的第三个维度(高度维度)的索引，最终输出采样值为采样点周围的多个最接近的角点的插值结果，输出张量的形状为[N, C, H, W]。

7.2.11.1.2　接口参数

网格插值采样函数接口应符合表131，C语言示例见A.2.11.1。

表 131　网格插值采样函数参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	输入张量，形状为[N,C,H,W]，元素数据类型可以为浮点数
网格张量	输入	必选	输入网格数据张量，形状为[N,H,W,2]，元素数据类型可以为浮点数
插值方法	输入	可选	可为双线性插值、近邻插值等,默认值可为近邻插值
输出张量	输出	必选	输入张量基于输入网格的插值计算结果，维度为[N,C,H,W] 的4维张量

7.2.11.1.3　接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。
非法参数：表示参数出错。

7.2.11.2　仿射网格函数

7.2.11.2.1　功能

生成仿射变换前后的feature map的坐标映射关系。在视觉应用中，根据该算子得到的映射关系，可以将输入feature map的像素点变换到对应的坐标，就得到了经过仿射变换的feature map。

7.2.11.2.2 接口参数

仿射网格函数接口应符合表132，C语言示例见A.2.11.2。

表 132 仿射网格函数参数列表

参数名	类型	可选/必选	描述
变换矩阵张量	输入	必选	表示用于放射变换的变换矩阵张量，形状为[N, 2, 3]，表示N个2×3的变换矩阵。元素类型可以为浮点数
目标图像形状	输入	必选	表示目标图像的形状，取值为[N, C, H, W]
输出张量	输出	必选	形状为[N, H, W, 2]的4-D张量，表示仿射变换前后的坐标的映射关系

7.2.11.2.3 接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。

7.2.11.3 像素重排函数

7.2.11.3.1 功能

将一个形为[N, C, H, W]的张量重新排列成形为$[\mathrm{N}, {}^{C}/_{r^2}, \mathrm{H}*\mathrm{r}, \mathrm{W}*\mathrm{r}]$的张量，其中r是增大因子。这样做有利于实现步长（stride）为1/r的高效sub-pixel（亚像素）卷积。

7.2.11.3.2 接口参数

像素重排函数接口应符合表133，C语言示例见A.2.11.3。

表 133 像素重排函数参数列表

参数名	类型	可选/必选	描述
输入张量	输入	必选	输入张量，形状为$[N_1, N_2, \ldots, N_k, \mathrm{D}]$，其中最后一维D是类别数目。元素类型可以为浮点数
增大因子	输入	必选	上采样因子，表示增大空间分辨率的倍数
输出张量	输出	必选	根据新的维度信息进行重组的张量

7.2.11.3.3 接口返回值

没有错误：表示操作成功。
类型不匹配：表示参数的数据类型不一致。
非法参数：表示增大因子的平方不能整除输入的通道大小。

7.2.12 优化器

7.2.12.1 SGD 优化器

7.2.12.1.1 功能

SGD优化器，用于更新神经网络中各权重参数，见式(68)。

$$param_{out} = param - learning\_rate \times grad \quad (68)$$

式中：

$grad$——梯度张量；

$param$——待更新参数；

$learning\_rate$——学习率；

$param_{out}$——更新后的参数。

7.2.12.1.2 接口参数

SGD优化器接口参数应符合表134，C语言示例见A.2.12.1。

表 134 SGD 优化器参数列表

参数名	类型	可选/必选	描述
待更新参数	输入	必选	输入的待更新参数，元素类型可以为浮点数
梯度张量	输入	必选	神经网络训练回传的梯度，元素类型可以为浮点数
学习率	输入	必选	参数更新的步长
更新后的参数	输出	必选	更新后的参数张量

7.2.12.1.3 接口返回值

没有错误：表示操作成功。

类型不匹配：表示参数的数据类型不一致。

7.2.12.2 Momentum 优化器

7.2.12.2.1 功能

Momentum优化器，用于更新神经网络中各权重参数，见式(69)-式(71)。

$$momentum_{out} = momentum\_factor \times momentum + grad \quad (69)$$

式中：

$momentum\_factor$——动量因子；

$momentum$——动量；

$grad$——梯度张量；

$momentum_{out}$——更新后的动量。

如果使用赋能牛顿动量：

$$param_{out} = param - learning\_rate \times (grad + momentum\_factor \times momentum_{out}) \quad (70)$$

式中：

$param$——待更新参数；

$learning\_rate$——学习率；

$grad$——梯度张量；

$momentum\_factor$——动量因子；

$momentum_{out}$——更新后的动量；

$param_{out}$——更新后的参数。

如果不使用赋能牛顿动量：

$$param_{out} = param - learning\_rate \times momentum_{out} \quad (71)$$

式中：

$param$——待更新参数；

$learning\_rate$——学习率；

$momentum_{out}$——更新后的动量；

$param_{out}$——更新后的参数。

7.2.12.2.2 接口参数

Momentum优化器接口参数应符合表135，C语言示例见A.2.12.2。

表 135 Momentum 优化器参数列表

参数名	类型	可选/必选	描述
待更新参数	输入	必选	输入的待更新参数，元素类型可以为浮点数
梯度张量	输入	必选	神经网络训练回传的梯度，元素类型可以为浮点数
动量	输入	必选	训练过程中累加的动量
动量因子	输入	必选	计算动量时的衰减率
学习率	输入	必选	参数更新的步长
使用赋能牛顿动量	输入	必选	是否在参数更新过程中使用赋能牛顿动量
更新后的参数	输出	必选	更新后的参数张量
更新后的动量	输出	必选	更新后的动量

7.2.12.2.3 接口返回值

没有错误：表示操作成功；

类型不匹配：表示参数的数据类型不一致。

7.2.12.3 AdaGrad 优化器

7.2.12.3.1 功能

AdaGrad优化器，用于更新神经网络中各权重参数，见式(72)与(73)。

$$accum_{out} = accum + grad^2 \quad (72)$$

式中：

$accum$——梯度平方和；

$grad$——梯度张量；

$accum_{out}$——更新后的梯度平方和。

$$param_{out} = param - learning\_rate \times \frac{grad}{accum_{out}} \quad (73)$$

式中：

$param$——待更新参数；

$learning\_rate$——学习率；

$grad$——梯度张量；

$accum_{out}$——更新后的梯度平方和；

$param_{out}$——更新后的参数。

7.2.12.3.2 接口参数

AdaGrad优化器接口参数应符合表136，C语言示例见A.2.12.3。

表 136 AdaGrad 优化器参数列表

参数名	类型	可选/必选	描述
待更新参数	输入	必选	输入的待更新参数，元素类型可以为浮点数
梯度张量	输入	必选	神经网络训练回传的梯度，元素类型可以为浮点数
梯度平方和	输入	必选	训练过程中累加的梯度平方和
学习率	输入	必选	参数更新的步长
更新后的参数	输出	必选	更新后的参数张量
更新后的梯度平方和	输出	必选	更新后的梯度平方和

7.2.12.3.3 接口返回值

没有错误：表示操作成功；

类型不匹配：表示参数的数据类型不一致。

7.2.12.4 AdaDelta 优化器

7.2.12.4.1 功能

AdaDelta优化器，用于更新神经网络中各权重参数，见式(74)-(77)。

$$mean\_square_{out} = \alpha * mean\_square + (1-\alpha)grad^2 \quad (74)$$

式中：

α——衰减率；

$mean\_square$——梯度均方；

$grad$——梯度张量；

$mean\_square_{out}$——更新后的梯度均方。

$$update = \frac{\sqrt{accum\_update+\varepsilon}}{\sqrt{mean\_square_{out}+\varepsilon}} \times grad \quad (75)$$

式中：

$grad$——梯度张量；

$accum\_update$——参数更新量均方；

ε——平滑项；

$accum\_update_{out}$——更新后的参数更新量均方；

$update$——参数更新量。

$$accum\_update_{out} = \alpha \times accum\_update + (1-\alpha) \times update^2 \quad (76)$$

式中：

α——衰减率；
$accum\_update$——参数更新量均方；
$update$——参数更新量；
$accum\_update_{out}$——更新后的参数更新量均方。

$$param_{out} = param - learning\_rate \times update \quad (77)$$

式中：
$param$——待更新参数；
$learning\_rate$——学习率；
$update$——参数更新量；
$param_{out}$——更新后的参数。

7.2.12.4.2 接口参数

AdaDelta优化器接口参数应符合表137，C语言示例见A.2.12.4。

表 137 AdaDelta 优化器参数列表

参数名	类型	可选/必选	描述
待更新参数	输入	必选	输入的待更新参数，元素类型可以为浮点数
梯度张量	输入	必选	神经网络训练回传的梯度，元素类型可以为浮点数
梯度均方	输入	必选	训练过程中累加的梯度均方
衰减率	输入	必选	梯度均方和参数更新量均方累加时的衰减率系数
参数更新量均方	输入	必选	参数更新量均方
学习率	输入	必选	参数更新的步长
平滑项	输入	必选	为了避免除零而加在分母上的小常数
更新后的参数	输出	必选	更新后的参数张量
更新后的梯度均方	输出	必选	更新后的梯度均方
更新后的参数更新量均方	输出	必选	更新后的参数更新量均方

7.2.12.4.3 接口返回值

没有错误：表示操作成功；
类型不匹配：表示参数的数据类型不一致。

7.2.12.5 RMSProp 优化器

7.2.12.5.1 功能

RMSProp优化器，用于更新神经网络中各权重参数，见式(78)-(80)。

$$mean\_square_{out} = \alpha * mean\_square + (1-\alpha)grad^2 \quad (78)$$

式中：
α——衰减率；
$mean\_square$——梯度均方；
$grad$——梯度张量；

$mean\_square_{out}$——更新后的梯度均方。

$$momentum_{out} = momentum\_factor \times momentum + \frac{grad}{\sqrt{mean\_square_{out}+\varepsilon}} \quad \cdots\cdots\cdots\cdots (79)$$

式中：

$momentum\_factor$——动量因子；

$momentum$——动量；

$grad$——梯度张量；

$mean\_square_{out}$——更新后的梯度均方；

ε——平滑项；

$momentum_{out}$——更新后的动量。

$$param_{out} = param - learning\_rate \times momentum_{out} \quad \cdots\cdots\cdots\cdots\cdots\cdots (80)$$

式中：

$param$——待更新参数；

$learning\_rate$——学习率；

$momentum_{out}$——更新后的动量；

$param_{out}$——更新后的参数。

7.2.12.5.2 接口参数

RMSProp优化器接口参数应符合表138，C语言示例见A.2.12.5。

表 138 RMSProp 优化器参数列表

参数名	类型	可选/必选	描述
待更新参数	输入	必选	输入的待更新参数，元素类型可以为浮点数
梯度张量	输入	必选	神经网络训练回传的梯度，元素类型可以为浮点数
梯度均方	输入	必选	训练过程中累加的梯度均方
动量	输入	必选	训练过程中累加的动量
动量因子	输入	必选	计算动量时的衰减率
衰减率	输入	必选	梯度均方累加时历史梯度的系数
学习率	输入	必选	参数更新的步长
平滑项	输入	必选	为了避免除零而加在分母上的小常数
更新后的参数	输出	必选	更新后的参数张量
更新后的梯度均方	输出	必选	更新后的梯度均方
更新后的动量	输出	必选	更新后的动量

7.2.12.5.3 接口返回值

没有错误：表示操作成功；

类型不匹配：表示参数的数据类型不一致。

7.2.12.6 CenteredRMSProp 优化器

7.2.12.6.1 功能

CenteredRMSProp优化器，用于更新神经网络中各权重参数，见式(81)-(84)。

$$mean\_square_{out} = \alpha * mean\_square + (1-\alpha)grad^2 \quad (81)$$

式中：

α——衰减率；

$mean\_square$——梯度均方；

$grad$——梯度张量；

$mean\_square_{out}$——更新后的梯度均方。

$$mean\_grad_{out} = \alpha * mean\_grad + (1-\alpha)grad \quad (82)$$

式中：

α——衰减率；

$mean\_grad$——平均梯度；

$grad$——梯度张量；

$mean\_grad_{out}$——更新后的平均梯度。

$$momentum_{out} = momentum\_factor \times momentum + \frac{grad}{\sqrt{mean\_square_{out} - mean\_grad_{out}^2 + \varepsilon}} \quad (83)$$

式中：

$momentum\_factor$——动量因子；

$momentum$——动量；

$grad$——梯度张量；

$mean\_square_{out}$——更新后的梯度均方；

$mean\_grad_{out}$——更新后的平均梯度；

ε——平滑项；

$momentum_{out}$——更新后的动量。

$$param_{out} = param - learning\_rate \times momentum_{out} \quad (84)$$

式中：

$param$——待更新参数；

$learning\_rate$——学习率；

$momentum_{out}$——更新后的动量；

$param_{out}$——更新后的参数。

7.2.12.6.2 接口参数

CenteredRMSProp优化器接口参数应符合表139，C语言示例见A.2.12.6。

表 139 CenteredRMSProp 优化器参数列表

参数名	类型	可选/必选	描述
待更新参数	输入	必选	输入的待更新参数，元素类型可以为32位浮点数，64位浮点数
梯度张量	输入	必选	神经网络训练回传的梯度，元素类型可以为32位浮点数，64位浮点数
平均梯度	输入	必选	训练过程中累加的平均梯度
梯度均方	输入	必选	训练过程中累加的梯度均方
动量	输入	必选	训练过程中累加的动量

表 139　CenteredRMSProp 优化器参数列表（续）

参数名	类型	可选/必选	描述
动量因子	输入	必选	计算动量时的衰减率
衰减率	输入	必选	梯度均方累加时历史梯度的系数
学习率	输入	必选	参数更新的步长
平滑项	输入	必选	为了避免除零而加在分母上的小常数
更新后的参数	输出	必选	更新后的参数张量
更新后的平均梯度	输出	必选	更新后的平均梯度
更新后的梯度均方	输出	必选	更新后的梯度均方
更新后的动量	输出	必选	更新后的动量

7.2.12.6.3　接口返回值

没有错误：表示操作成功；
类型不匹配：表示参数的数据类型不一致。

7.2.12.7　Adam 优化器

7.2.12.7.1　功能

Adam优化器，用于更新神经网络中各权重参数，见式(85)-(91)。

$$m_{out} = \beta_1 \times m + (1 - \beta_1) \times grad \quad (85)$$

式中：
β_1——衰减率一；
m——动量一；
$grad$——梯度张量；
m_{out}——更新后的动量一。

$$v_{out} = \beta_2 \times v + (1 - \beta_2) \times grad^2 \quad (86)$$

式中：
β_2——衰减率二；
v——动量二；
$grad$——梯度张量；
v_{out}——更新后的动量二。

$$\beta\_power\_out_1 = \beta\_power_1 \times \beta_1 \quad (87)$$

式中：
β_power_1——衰减率一的幂；
β_1——衰减率一；
$\beta\_power\_out_1$——更新后的衰减率一的幂。

$$\beta\_power\_out_2 = \beta\_power_2 \times \beta_2 \quad (88)$$

式中：

$\beta\_power\_out_2$——更新后的衰减率二的幂；
β_2——衰减率二；
$\beta\_power\_out_2$——更新后的衰减率二的幂。

$$m_{bias} = \frac{m_{out}}{1-\beta\_power\_out_1} \quad (89)$$

式中：
m_{out}——更新后的动量一；
$\beta\_power\_out_1$——更新后的衰减率一的幂；
m_{bias}——动量一的偏差修正值。

$$v_{bias} = \frac{v_{out}}{1-\beta\_power\_out_2} \quad (90)$$

式中：
m_{out}——更新后的动量一；
$\beta\_power\_out_1$——更新后的衰减率一的幂；
v_{bias}——动量二的偏差修正值。

$$param_{out} = param - learning\_rate \times \frac{m_{bias}}{\sqrt{v_{bias}}+\varepsilon} \quad (91)$$

式中：
$param$——待更新参数；
$learning\_rate$——学习率；
m_{bias}——动量一的偏差修正值；
v_{bias}——动量二的偏差修正值；
ε——平滑项；
$param_{out}$——更新后的参数。

7.2.12.7.2 接口参数

Adam优化器接口参数应符合表140，C语言示例见A.2.12.7。

表 140 Adam 优化器参数列表

参数名	类型	可选/必选	描述
待更新参数	输入	必选	输入的待更新参数，元素类型可以为浮点数
梯度张量	输入	必选	神经网络训练回传的梯度，元素类型可以为浮点数
动量一	输入	必选	训练过程中累加的动量一
动量二	输入	必选	训练过程中累加的动量二
衰减率一	输入	必选	计算动量一时的衰减率
衰减率二	输入	必选	计算动量二时的衰减率
衰减率一的幂	输入	必选	衰减率一的幂
衰减率二的幂	输入	必选	衰减率二的幂

表 140　Adam 优化器参数列表（续）

参数名	类型	可选/必选	描述
学习率	输入	必选	参数更新的步长
平滑项	输入	必选	为了避免除零而加在分母上的小常数
更新后的参数	输出	必选	更新后的参数张量
更新后的动量一	输出	必选	更新后的动量一
更新后的动量二	输出	必选	更新后的动量二
更新后的衰减率一的幂	输出	必选	更新后的衰减率一的幂
更新后的衰减率二的幂	输出	必选	更新后的衰减率二的幂

7.2.12.7.3　接口返回值

没有错误：表示操作成功；
类型不匹配：表示参数的数据类型不一致。

7.2.12.8　AdaMax 优化器

7.2.12.8.1　功能

AdaMax优化器算子，用于更新神经网络中各权重参数，见式(92)-(95)。

$$mean\_grad_{out} = \alpha \times mean\_grad + (1 - \alpha) * grad \quad (92)$$

式中：
α——平均梯度衰减率；
$mean\_grad$——平均梯度；
$mean\_grad_{out}$——更新后的平均梯度。

$$\alpha\_power_{out} = \alpha \times \alpha\_power \quad (93)$$

式中：
α——平均梯度衰减率；
α_power——平均梯度衰减率的幂；
α_power_{out}——更新后的平均梯度衰减率的幂。

$$grad\_max_{out} = max\,(\beta \times grad\_max, |grad|) \quad (94)$$

式中：
β——梯度最大值衰减率；
$grad\_max$——梯度最大值；
$grad$——梯度张量；
$grad\_max_{out}$——更新后的梯度最大值。

$$param_{out} = param - \frac{mean\_grad_{out}}{(grad\_max_{out}+\varepsilon)\times(1-\alpha\_power_{out})} \times learning\_rate \quad (95)$$

式中：
$param$——待更新参数；

$mean\_grad_{out}$——更新后的平均梯度；
$grad\_max_{out}$——更新后的梯度最大值；
ε——平滑项；
α_power_{out}——更新后的平均梯度衰减率的幂；
$learning\_rate$——学习率；
$param_{out}$——更新后的参数。

7.2.12.8.2 接口参数

AdaMax优化器接口参数应符合表141，C语言示例见A.2.12.8。

表 141 AdaMax 优化器参数列表

参数名	类型	可选/必选	描述
待更新参数	输入	必选	输入的待更新参数，元素类型可以为32位浮点数，64位浮点数
梯度张量	输入	必选	神经网络训练回传的梯度，元素类型可以为32位浮点数，64位浮点数
平均梯度	输入	必选	训练过程中累加的平均梯度
平均梯度衰减率	输入	必选	计算平均梯度时使用的衰减率
平均梯度衰减率的幂	输入	必选	计算平均梯度时使用的衰减率的幂
梯度最大值衰减率	输入	必选	计算梯度最大值时使用的衰减率
梯度最大值	输入	必选	梯度最大值
学习率	输入	必选	参数更新的步长
平滑项	输入	必选	为了避免除零而加在分母上的小常数
更新后的参数	输出	必选	更新后的参数张量
更新后的平均梯度	输出	必选	更新后的平均梯度
更新后的平均梯度衰减率的幂	输出	必选	更新后的平均梯度衰减率的幂
更新后的梯度最大值	输出	必选	更新后的梯度最大值

7.2.12.8.3 接口返回值

没有错误：表示操作成功；
类型不匹配：表示参数的数据类型不一致。

7.3 算子接口最小集

为适用于深度学习编译器等技术，实现无限阶微分计算，以及降低硬件厂商和人工智能框架厂商适配成本，本文件抽象出了基础的神经网络算子接口集合，形成算子接口最小集，详见表142。人工智能框架应至少具备算子接口最小集中的算子并满足其接口标准。

表 142　算子接口最小集列表

序号	分类	标准章节号	标准定义的接口名称
1.	激活函数	7.2.1.1	S 型函数
2.		7.2.1.3	分段线性近似 S 型函数
3.		7.2.1.4	归一化指数函数
4.		7.2.1.5	对数归一化指数函数
5.		7.2.1.6	线性整流单元
6.		7.2.1.7	带阈值的线性整流单元
7.		7.2.1.8	指数线性单元
8.		7.2.1.9	带泄露线性整流单元
9.		7.2.1.10	带参数线性整流单元
10.		7.2.1.11	扩展指数线性单元
11.		7.2.1.14	Softplus 函数
12.		7.2.1.15	Softsign 函数
13.		7.2.1.17	Hardswish 函数
14.		7.2.1.18	误差函数
15.	正则函数	7.2.3.1	随机失活函数
16.	归一化函数	7.2.4.1	批量归一化
17.		7.2.4.2	分组归一化
18.		7.2.4.3	层归一化
19.		7.2.4.4	实例归一化
20.		7.2.4.7	Lp 范数归一化
21.	池化函数	7.2.5.1	一维池化
22.		7.2.5.2	二维池化
23.		7.2.5.3	三维池化
24.	卷积函数	7.2.6.1	二维卷积
25.		7.2.6.2	三维卷积
26.		7.2.6.4	二维转置卷积
27.		7.2.6.5	三维转置卷积
28.	编码操作	7.2.9.2	独热编码
29.	视觉函数	7.2.11.1	网格插值采样

附 录 A
（资料性）
神经网络类算子接口 C 语言参考定义示例

A.1 数据结构

本文件使用的C语言参考定义数据结构应符合T/AI 131.1-2025 《人工智能 算子接口 第1部分：基础数学类》中A.1的规定。

A.2 神经网络类算子接口

A.2.1 激活函数

A.2.1.1 S型函数

前向接口 C 语言：

```
Status op_sigmoid_forward(const Tensor x,
                          Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_sigmoid_backward(const Tensor grad_out,
                           const Tensor input,
                           Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [0, 2.2] */
Tensor y;
op_sigmoid_forward(x, &y);
/* y:  [0.5, 0.9002495108803148] */
```

A.2.1.2 对数S型函数

前向接口 C 语言：

```
Status op_log_sigmoid_forward(const Tensor x,
```

```
                                    Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量，元素类型可以为FLOAT32，FLOAT64等。

y (INOUT)：表示输出张量，形状、数据类型与输入张量相同。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_log_sigmoid_backward(const Tensor grad_out,
                               const Tensor input,
                               Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [1.0, 2.0, 3.0] */
op_log_sigmoid_forward(x, &y);
/* y:  [-0.313262, -0.126928 0.0485874] */
```

A.2.1.3 分段线性近似S型函数

前向接口 C 语言：

```
Status op_hard_sigmoid_forward (const Tensor x,
                                const float slope,
                                const float offset,
                                Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

slope(IN)：表示斜率。

offset(IN)：表示偏移量。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_INVALID_ARGUMENT：表示其他参数不合法。

后向接口 C 语言：

```
Status op_hard_sigmoid_ backward (const Tensor grad_out,
                                  const Tensor input,
                                  const float slope,
                                  const float offset,
```

```
                                    Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

slope(IN)：表示斜率 2。

offset(IN)：表示偏移量。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_INVALID_ARGUMENT：表示其他参数不合法。

示例：

```
/* x:  [0.604, 0.806, -0.324, 0.875] */
op_hard_sigmoid_forward(x, 0.2, 0.5, &y);
/* y:  [0.621, 0.661, 0.435, 0.675] */
```

A.2.1.4 归一化指数函数

前向接口 C 语言：

```
Status op_softmax_forward(const Tensor x,
                          const int axis,
                          Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

axis (IN)：表示计算 Softmax 所沿着的轴。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_OUT_OF_RANGE：表示 axis 超出输入张量维度。

后向接口 C 语言：

```
Status op_softmax_ backward (const Tensor grad_out,
                             const Tensor output,
                             const int axis,
                             Tensor * grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

output(IN)：表示前向接口中的输出张量。

axis (IN)：表示计算 Softmax 所沿着的轴。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_OUT_OF_RANGE：表示 axis 超出输入张量维度。

示例：

```
/* x:  [[0, 2.2, 3.5],
       [1,  2,  3]]  */
Tensor y;
op_softmax_forward(x, -1, &y);
/* y:  [[0.02318009, 0.20920065, 0.76761926],
       [0.09003057, 0.24472847, 0.66524095]] */
```

A.2.1.5 对数归一化指数函数

前向接口 C 语言：

```
Status op_log_softmax_forward (const Tensor logits,
                               const int axis,
                               Tensor *output);
```

前向接口参数：

logits (IN)：表示输入张量。

axis：表示计算 Log Softmax 值所沿着的轴。

output (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_OUT_OF_RANGE：表示 axis 超出输入张量维度。

后向接口 C 语言：

```
Status op_log_softmax_ backward (const Tensor grad_out,
                                 const Tensor output,
                                 const int axis,
                                 Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

output(IN)：表示前向接口中的输出张量。

axis：表示计算 Log Softmax 值所沿着的轴。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_OUT_OF_RANGE：表示 axis 超出输入张量维度。

示例：

```
/* x:  [[0, 2.2, 3.5],
       [1,  2,  3]]  */
Tensor y;
op_log_softmax_forward(x, 0, &y);
/* y:  [[-3.76446156, -1.56446144,-0.26446142],
       [-2.40760599, -1.40760597, -0.40760597]] */
```

A.2.1.6 线性整流单元

前向接口 C 语言：

```
Status op_relu_forward (const Tensor x,
                        Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_relu_ backward(const Tensor grad_out,
                         const Tensor input,
                         Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [-1.2, 1, 2.3] */
Tensor y;
op_relu_forward(x, &y);
/* y:  [0, 0, 2.3] */
```

A.2.1.7 带阈值的线性整流单元

前向接口 C 语言：

```
Status op_thresholded_relu_forward (const Tensor x,
                                    const float threshold,
                                    Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

threshold (IN)：激活函数的阈值。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_thresholded_relu_ backward forward (const Tensor grad_out,
                                              const Tensor input,
                                              const float threshold,
                                              Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

threshold (IN)：激活函数的阈值。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [-1.2, 1, 2.3] */
op_thresholded_relu_forward(x, 2.0, &y);
/* y:  [0, 0, 2.3] */
```

A.2.1.8 指数线性单元

前向接口 C 语言：

```
Status op_elu_forward (const Tensor x,
                       const float alpha,
                       Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

alpha(IN)：ELU 公式中的α值。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_elu_ backward (const Tensor grad_out,
                         const Tensor input,
                         const float alpha,
                         Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

alpha(IN)：ELU 公式中的α值。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [-1.2, 0, 1.3] */
Tensor y;
op_elu_forward(x, 1.0, &y);
/* y:  [-0.6988, 0, 1.3] */
```

A.2.1.9 带泄露线性整流单元

前向接口 C 语言：

```
Status op_leaky_relu_forward (const Tensor x,
                              const float alpha,
                              Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

alpha(IN)：leaky_relu 公式中的 α 值。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_leaky_relu_ backward (const Tensor grad_out,
                                const Tensor input,
                                const float alpha,
                                Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

alpha(IN)：leaky_relu 公式中的 α 值。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [-1.2, 0, 1.3] */
Tensor y;
op_leaky_relu_forward(x, 0.1, &y);
/* y:  [-0.12, 0, 1.3] */
```

A.2.1.10 带参数线性整流单元

前向接口 C 语言：

```
Status op_prelu_forward (const Tensor x,
                         const float *alpha_array,
                         const int alpha_array_len,
                         Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

alpha_array (IN)：prelu 公式中的 α 值数组。

alpha_array_len(IN)：数组 alpha_array 的长度。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INTERNAL_ERROR：表示内部调用操作出错。

STATUS_INVALID_ARGUMENT：alpha_array_len 不合法。

后向接口 C 语言：

```
Status op_prelu_ backward (const Tensor grad_out,
                           const Tensor input,
                           const float *alpha_array,
                           const int alpha_array_len,
                           Tensor *grad_in);
```

后前向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

alpha_array (IN)：prelu 公式中的 α 值数组。

alpha_array_len(IN)：数组 alpha_array 的长度。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INTERNAL_ERROR：表示内部调用操作出错。

STATUS_INVALID_ARGUMENT：alpha_array_len 不合法。

示例：

```
/* x:  [[[-1.2, 0, 1.3],
       [0, 1.5, -0.2]]] */
/* alpha_array:  [[0.3, 0.5] */
int alpha_array_len = 2;
Tensor y;
op_prelu_forward(x, alpha_array, alpha_array_len, &y);
/* y:  [[[-0.36, 0, 1.3],
       [0, 1.5, -0.1]]] */
```

A.2.1.11 扩展指数线性单元

前向接口 C 语言：

```
Status op_selu_forward (const Tensor x,
                        const float scale,
                        const float alpha,
                        Tensor *y);
```

前向接口参数：

x(IN)：表示输入张量。

scale (IN)：selu 公式中的缩放因子。

alpha (IN)：selu 公式中的 α 值。

y(INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_selu_ backward (const Tensor grad_out,
                          const Tensor input,
                              const float scale,
                              const float alpha,
                              Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

scale (IN): selu 公式中的缩放因子。

alpha (IN)：selu 公式中的 α 值。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [-0.218, 0.899, 0.706, -0.633] */
op_selu_forward(x, 1.05070098, 1.67326324, &y);
/* y:  [-0.344, 0.945, 0.741, -0.825] */
```

A.2.1.12 双边整流线性单元

前向接口 C 语言：

```
Status op_brelu_forward (const Tensor x,
                         const float low,
                         const float high,
                         Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

low (IN)：brelu 结果的最小值。

high (IN)：brelu 结果的最大值。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_brelu_ backward (const Tensor grad_out,
                           const Tensor input,
```

```
                                const float low,
                                const float high,
                                Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

low (IN)：brelu 结果的最小值。

high (IN)：brelu 结果的最大值。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [-1.2, 1, 2.3, 25.0] */
op_brelu_forward(x, 0.0, 24.0, &y);
/* y:  [0, 0, 2.3, 24.0] */
```

A.2.1.13 高斯误差线性单元

前向接口C语言：

```
 Status op_gelu_forward (const Tensor x,
                         const bool approximate,
                         Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

approximate (IN): 是否使用近似计算。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口C语言：

```
 Status op_gelu_ backward(const Tensor grad_out,
                          const Tensor input,
                          const bool approximate,
                          Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

approximate (IN): 是否使用近似计算。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [0.034, -0.611, 0.037, 0.823] */
op_gelu_forward(x, &y);
/* y:  [0.018, -0.165, 0.019, 0.654] */
```

A.2.1.14 softplus函数

前向接口 C 语言：

```
Status op_softplus_forward (const Tensor x,
                            const float beta,
                            const float threshold,
                            Tensor *y)
```

前向接口参数：

x (IN)：表示输入张量。

beta(IN)：计算公式中的常量 beta 值。

threshold(IN)：计算公式中的 threshold 值。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INTERNAL_ERROR: 表示内部调用操作出错。

后向接口 C 语言：

```
Status op_softplus_ backward(const Tensor grad_out,
                             const float beta,
                             const float threshold,
                             const Tensor input,
                             Tensor *grad_in)
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

beta(IN)：计算公式中的常量 beta 值。

threshold(IN)：计算公式中的 threshold 值。

input(IN)：表示前向接口的输入张量。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INTERNAL_ERROR: 表示内部调用操作出错。

示例：

```
/* x:  [-0.255, -0.222, -0.466, 0.271] */
op_softplus_forward(x, 1.0, 20, &y);
/* y:  [0.574, 0.588, 0.487, 0.838] */
```

A.2.1.15 softsign函数

前向接口 C 语言:

```
Status op_softsign_forward (const Tensor x,
                            Tensor *y);
```

前向接口参数：

x（IN）：表示输入张量。

y（INOUT）：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_softsign_ backward(const Tensor grad_out,
                             const Tensor input,
                             Tensor *grad_in);
```

后向接口参数：

grad_out（IN）：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

示例：

```
/* x:  [-0.998, 0.434, 0.279, 0.730] */
op_softsign_forward(x, &y);
/* y:  [-0.499, 0.302, 0.218, 0.422] */
```

A.2.1.16 swish函数

前向接口C语言：

```
Status op_swish_forward (const Tensor x,
                         const float beta,
                         Tensor *y);
```

前向接口参数：

x（IN）：表示输入张量。

beta（IN）：swish 计算公式中的常量 beta 值。

y（INOUT）：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

后向接口C语言：

```
Status op_swish_ backward(const Tensor grad_out,
                          const Tensor input,
                          const float beta,
                          Tensor *grad_in);
```

后向接口参数：

grad_out（IN）：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

beta（IN）：swish 计算公式中的常量 beta 值。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [0.320, -0.550, 0.082, -0.745] */
op_swish_forward(x, 1.0, &y);
/* y:  [0.185, -0.201, 0.043, -0.240] */
```

A.2.1.17 hard_swish函数

前向接口 C 语言：

```
Status op_hard_swish_forward (const Tensor x,
                              const float threshold,
                              const float scale,
                              const float offset,
                              Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

threshold (IN): 激活操作的阈值。

scale (IN): 激活操作的缩放因子。

offset (IN): 激活操作的位移。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_INVALID_ARGUMENT：表示其他参数不合法。

后前向接口 C 语言：

```
Status op_hard_swish_ backward(const Tensor grad_out,
                               const Tensor input,
                               const float threshold,
                               const float scale,
                               const float offset,
                               Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

threshold (IN): 激活操作的阈值。

scale (IN): 激活操作的缩放因子。

offset (IN): 激活操作的位移。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_INVALID_ARGUMENT：表示其他参数不合法。

示例：

```
/* x:  [-0.482, 0.746, 0.443, -0.008] */
op_hard_swish_forward(x, 6.0, 6.0, 3.0, &y);
/* y:  [-0.202, 0.466, 0.254, -0.004] */
```

A.2.1.18 误差函数

前向接口C语言：

```
Status op_erf_forward (const Tensor x,
                       Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量。

y (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口C语言：

```
Status op_erf_ backward(const Tensor grad_out,
                        const Tensor input,
                        Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [-0.287, -0.180, 0.779, -0.748] */
op_erf_forward(x, &y);
/* y:  [-0.315, -0.200, 0.730, -0.780] */
```

A.2.1.19 hard_shrink函数

前向接口 C 语言：

```
Status op_hard_shrink_forward(const Tensor x,
                              const float threshold,
                              Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量

threshold (IN)：激活函数的阈值λ，通常取 0.5。

y (INOUT)：表示输出张量，形状、数据类型与输入张量相同。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_hard_shrink_ backward(const Tensor grad_out,
                                const Tensor input,
                                const float threshold,
                                Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量

threshold (IN)：激活函数的阈值λ，通常取 0.5。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [-1, -0.3, 2.5] */
op_hard_shrink_forward(x, 0.5, &y);
/* y:  [-1, 0., 2.5] */
```

A.2.1.20 tanh_shrink函数

前向接口 C 语言：

```
Status op_tanh_shrink_forward(const Tensor x,
                              Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量

y (INOUT)：表示输出张量，形状、数据类型与输入张量相同。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_tanh_shrink_ backward(const Tensor grad_out,
                                const Tensor input,
                                Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [-0.4, -0.2, 0.1, 0.3] */
op_tanh_shrink_forward(x, &y);
```

```
/* y:  [-0.020051, -0.00262468, 0.000332005, 0.00868739] */
```

A.2.1.21 hard_tanh函数

前向接口 C 语言：

```
Status op_hard_tanh_forward(const Tensor x,
                            const float min.
                            const float max,
                            Tensor *y);
```

前向接口参数：

x (IN)：表示输入张量，元素类型可以为 FLOAT32，FLOAT64 等。

min (IN)：表示 hard_tanh 激活中的 min 值。

max (IN)：表示 hard_tanh 激活中的 max 值。

y (INOUT)：表示输出张量，形状、数据类型与输入张量相同。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_hard_tanh_ backward(const Tensor grad_out,
                              const Tensor input,
                              const float min.
                              const float max,
                              Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示前向接口的输入张量

min (IN)：表示 hard_tanh 激活中的 min 值。

max (IN)：表示 hard_tanh 激活中的 max 值。

grad_in(INOUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [-1.5, 0.3, 2.5] */
/* min: -1.0 */
/* max: 1.0 */
op_hard_tanh _forward(x, min, max, &y);
/* y:  [-1, 0.3, 1] */
```

A.2.2 损失函数

A.2.2.1 L1损失函数

前向接口 C 语言：

```
Status op_l1_loss_forward(const Tensor input,
                          const Tensor target,
```

```
                                const char *reduction,
                                Tensor *loss);
```

前向接口参数：

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

reduction (IN)：表示规约类型。

loss (OUT)：表示损失值。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_l1_loss_backward(const Tensor grad_loss,
                                const Tensor input,
                                const Tensor target,
                                const char *reduction,
                                Tensor *grad_in);
```

后向接口参数：

grad_loss (IN)：表示损失值梯度张量。

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

reduction (IN)：表示规约类型。

grad_in (OUT)：表示实际值梯度张量。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* input:  [0.1, 0.2, 0.7] */
/* target:  [0,  0,  1] */
/* reduction:  “sum”  */
Tensor loss;
op_l1_loss_forward(input, target, reduction, &loss);
/* loss:  0.6 */
```

A.2.2.2 均方误差(MSE)损失函数

前向接口 C 语言：

```
Status op_mse_loss_forward(const Tensor input,
                                const Tensor target,
                                const char *reduction,
                                Tensor *loss);
```

前向接口参数：

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

reduction (IN)：表示规约类型。

loss (OUT)：表示损失值。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_mse_loss_backward(const Tensor grad_loss,
                            const Tensor input,
                            const Tensor target,
                            const char *reduction,
                            Tensor *grad_in);
```

后向接口参数：

grad_loss (IN)：表示损失值张量的梯度。

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

reduction (IN) ：表示规约类型。

grad_in (OUT)：表示实际值张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* ipnut:  [0.1, 0.2, 0.7] */
/* target:  [0, 0, 1] */
/* reduction: “sum” */
Tensor loss;
op_mse_loss_forward(input, target, reduction, &loss);
/* loss:  0.14 */
```

A.2.2.3 交叉熵损失函数

前向接口 C 语言：

```
Status op_cross_entropy_forward(const Tensor input,
                                const Tensor target,
                                const Tensor *weight,
                                const char *reduction,
                                Tensor *loss);
```

前向接口参数：

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

weight (IN)：表示权重张量。

reduction (IN)：表示规约类型。

loss (OUT)：表示损失值。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_cross_entropy_backward(const Tensor grad_loss,
                                 const Tensor input,
                                 const Tensor target,
                                 const Tensor *weight,
                                 const char *reduction,
                                 Tensor *grad_in);
```

后向接口参数：

grad_loss (IN)：表示损失张量的梯度。

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

weight (IN)：表示权重张量。

reduction (IN)：表示规约类型。

grad_in (OUT)：表示实际值张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* input:  [0.1, 0.2, 0.7] */
/* target:  [0,  0,  1] */
/* weight: [1.0, 1.0, 1.0] */
/* reduction: “sum” */
Tensor loss;
op_cross_entropy_forward(input, target, weight, reduction, &loss);
/* loss:  0.356674944 */
```

A.2.2.4 稀疏交叉熵损失函数

前向接口 C 语言：

```
Status op_sparse_cross_entropy_forward(const Tensor input,
                                       const Tensor target,
                                       const Tensor *weight,
                                       const char *reduction,
                                       Tensor *loss);
```

前向接口参数：

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

weight (IN)：表示权重张量。

reduction (IN)：表示归约类型。

loss (OUT)：表示损失值。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_sparse_cross_entropy_backward(const Tensor grad_loss,
                                        const Tensor input,
                                        const Tensor target,
                                        const Tensor *weight,
                                        const char *reduction,
                                        Tensor *grad_in);
```

后向接口参数：

grad_loss (IN)：表示损失张量的梯度。

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

weight (IN)：表示权重张量。

reduction (IN)：表示归约类型。

grad_in (OUT)：表示实际值张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* input:  [0.1, 0.2, 0.7] */
/* target:  [2] */
/* weight: [1.0, 1.0, 1.0] */
/* reduction: “sum” */
Tensor loss;
op_cross_entropy_forward(ipnut, target, weight, reduction, &loss);
/* loss:  0.356674944 */
```

A.2.2.5 负对数损失函数

前向接口C语言：

```
Status op_log_loss_forward(const Tensor input,
                           const Tensor target,
                           const float epsilon,
                           const char *reduction,
                           Tensor *loss);
```

前向接口参数：

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

epsilon (IN)：计算公式中的 ε 。

reduction (IN)：表示归约类型。

loss (INOUT)：表示损失值。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口C语言：

```
Status op_log_loss_backward(const Tensor grad_loss,
```

```
                                const Tensor input,
                                const Tensor target,
                                const float epsilon,
                                const char *reduction,
                                Tensor *grad_in);
```

后向接口参数：

grad_loss (IN)：表示损失张量的梯度。

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

epsilon (IN)：计算公式中的 ε 。

reduction (IN)：表示归约类型。

grad_in (OUT)：表示实际值张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

A.2.2.6 负对数似然损失函数

前向接口 C 语言：

```
Status op_nll_loss_forward (const Tensor input,
                            const Tensor target,
                            const Tensor *weight,
                            const int ignore_index
                            const char* reduction,
                            Tensor *loss);
```

前向接口参数：

input (IN)：表示实际值张量

target (IN)：表示期望值张量。

weight(IN)：表示权重张量。

ignore_index(IN)：表示一个忽略的标签值。

reduction(IN)：表示归约类型。

loss(OUT)：表示损失值。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_INTERNAL_ERROR：表示内部的调用操作出错。

后向接口 C 语言：

```
Status op_nll_loss_backward(const Tensor grad_loss,
                            const Tensor input,
                            const Tensor target,
                            const Tensor *weight,
                            const int ignore_index
                            const char* reduction,
                            Tensor *grad_in);
```

后向接口参数：

grad_loss (IN)：表示损失张量的梯度。

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

weight(IN)：表示权重张量。

ignore_index(IN)：表示一个忽略的标签值。

reduction(IN)：表示归约类型。

grad_in (OUT)：表示实际值张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_INTERNAL_ERROR：表示内部的调用操作出错。

示例：

```
/* input:  [0.1, 0.2, 0.7] */
/* target:  [0, 0, 1] */
/* epsilon: 1e-8 */
/* reduction: "mean" */
Tensor loss;
op_log_loss_forward(input, target, epsilon, reduction, &loss);
/* loss:  0.35667494394 */
```

A.2.2.7 CTC损失函数

前向接口 C 语言：

```
Status op_ctc_loss_forward(const Tensor input,
                           const Tensor target,
                           const Tensor input_lengths,
                           const Tensor target_lengths,
                           const char *reduction,
                           const bool is_train,
                           Tensor *loss,
                           Tensor *log_alphas);
```

前向接口参数：

input (IN)：表示实际值张量。

target(IN)：表示期望值张量。

input_lengths (IN)：表示实际值长度张量。

target_lengths (IN)：表示期望值长度张量。

reduction(IN)：表示归约类型。

is_train(IN)：表示是否用于训练。

loss(OUT)：表示损失值。

log_alphas (IN)：表示前向概率对数张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGUMENT：表示参数不合法。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_ctc_loss_backward(const Tensor grad_loss,
                            const Tensor input,
                            const Tensor target,
                            const Tensor input_lengths,
                            const Tensor target_lengths,
                            const char *reduction,
                            Tensor *log_alphas,
                            Tensor *grad_in);
```

后向接口参数：

grad_loss (IN)：表示损失张量的梯度。

input (IN)：表示实际值张量。

target(IN)：表示期望值张量。

input_lengths (IN)：表示实际值长度张量。

target_lengths (IN)：表示期望值长度张量。

reduction(IN)：表示归约类型。

log_alphas (IN)：表示前向概率对数张量。

grad_in (OUT)：表示实际值张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGUMENT：表示参数不合法。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* input:  [[0.1, 0.2, 0.7],
            [0.3, 0.4, 0.3],
            [0.2, 0.5, 0.3],
            [0.4, 0.4, 0.2]] */
/* target: [1, 2, 2] */
/* input_lengths: [4] */
/* target_lengths: [3] */
/* reduction: "mean" */
/* is_train: true */
Tensor loss;
Tensor log_alphas;
op_ctc_loss_forward(input, target, input_lengths, target_lengths, reduction, is_train,
&loss, &log_alphas);
/* loss:  1.23456789 */
/* log_alphas: [[-0.8, -1.5, -1.2],
                [-1.1, -1.4, -1.3],
                [-1.2, -1.3, -1.4],
                [-1.3, -1.5, -1.2]] */
```

A.2.2.8 平滑L1损失函数

前向接口 C 语言：

```
Status op_smooth_l1_loss_forward(const Tensor input,
                                 const Tensor target,
                                 const char *reduction,
                                 Tensor *loss);
```

前向接口参数：

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

reduction(IN) ：表示规约类型。

loss (OUT)：表示损失值。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_smooth_l1_loss_backward(const Tensor grad_loss,
                                  const Tensor input,
                                  const Tensor target,
                                  const char *reduction,
                                  Tensor *grad_in);
```

后向接口参数：

grad_loss (IN)：表示损失张量的梯度。

input (IN)：表示实际值张量。

target (IN)：表示期望值张量。

reduction(IN) ：表示规约类型。

grad_in (OUT)：表示实际值张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* input:  [1, 0.5, 0.7] */
/* target:  [0.5, 1, 2] */
/* reduction: “none” */
Tensor loss;
op_smooth_l1_loss_forward(input, target, reduction, &loss);
/* loss:  [0.125, 0.125, 0.8] */
```

A.2.2.9 KL散度损失函数

前向接口 C 语言：

```
Status op_kldiv_loss_forward(const Tensor input,
                             const Tensor target,
                             const char *reduction,
                             Tensor *loss);
```

前向接口参数：

input (IN)：表示前向网络计算得到的实际值张量，元素数据类型为 FLOAT32、FLOAT64。

target (IN)：表示期望值张量，其形状与实际值张量形状一致，元素数据类型为 FLOAT32、FLOAT64。

reduction(IN) ：表示规约类型，从“none”，“mean”和“sum”三者间任选其一，默认为“mean” 。

loss (OUT)：表示损失值。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_kldiv_loss_backward(const Tensor grad_loss,
                              const Tensor input,
                              const Tensor target,
                              const char *reduction,
                              Tensor *grad_in);
```

后向接口参数：

grad_loss (IN)：表示损失张量的梯度。

input (IN)：表示前向网络计算得到的实际值张量，元素数据类型为 FLOAT32、FLOAT64。

target (IN)：表示期望值张量，其形状与实际值张量形状一致，元素数据类型为 FLOAT32、FLOAT64。

reduction(IN) ：表示规约类型，从“none”，“mean”和“sum”三者间任选其一，默认为“mean” 。

grad_in (OUT)：表示实际值张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* input:  [[0.3, 0.1, -0.5], [1.5, -3.0, 1.5]] */
/* target:  [[0.35, -0.1, 0.8], [-2.0, 0.0, 1.5]] */
/* reduction:  “none”  */
Tensor loss;
op_kldiv_loss_forward(input, target, reduction, &loss);
/* loss:  [[-0.4724, 0., 0.2215], [0., 0., -1.6418]] */
```

A.2.2.10 软间隔损失函数

前向接口 C 语言：

```
Status op_soft_margin_loss_forward(const Tensor input,
                                   const Tensor target,
                                   const char *reduction,
                                   Tensor *loss);
```

前向接口参数：

input (IN)：表示实际值张量。

target (IN)：表示期望值张量，形状与输入张量一致，元素值只能为 1 或-1。

reduction(IN) ：表示规约类型，从“none”，“mean”和“sum”三者间任选其一，默认为“mean” 。

loss (OUT)：表示损失值。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_soft_margin_loss_backward(const Tensor grad_loss,
                                    const Tensor input,
                                    const Tensor target,
                                    const char *reduction,
                                    Tensor *grad_in);
```

后向接口参数：

grad_loss (IN)：表示损失张量的梯度。

input (IN)：表示实际值张量。

target (IN)：表示期望值张量，形状与输入张量一致，元素值只能为 1 或-1。

reduction(IN) ：表示规约类型，从“none”，“mean”和“sum”三者间任选其一，默认为“mean”。

grad_in (OUT)：表示实际值张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* input:  [[0.2, -0.5], [-1.0, 0.0]] */
/* target:  [[1.0, 1.0], [1.0, -1.0]] */
/* reduction: “none” */
Tensor loss;
op_soft_margin_loss_forward(input, target, reduction, &loss);
/* loss:  [[0.5981, 0.9741], [1.3133, 0.6931]] */
```

A.2.2.11 间隔排序损失函数

前向接口 C 语言：

```
Status op_soft_margin_loss_forward(const Tensor input1,
                                   const Tensor input2,
                                   const Tensor target,
                                   const float margin,
                                   const char *reduction,
                                   Tensor *loss);
```

前向接口参数：

input1 (IN)：表示第一个排序得分输入张量，元素数据类型可以为 FLOAT32、FLOAT64。

input2 (IN)：表示第二个排序得分输入张量，元素数据类型可以为 FLOAT32、FLOAT64。

target (IN)：表示期望值张量，第一个排序得分值大于第二个排序得分值时值为 1，否则为-1，元素数据类型可以为 FLOAT32、FLOAT64。

margin(IN)：间隔值，表示两个得分之间的差异。

reduction(IN) ：表示规约类型，从“none”，“mean”和“sum”三者间任选其一，默认为“mean” 。

loss (OUT)：表示损失值。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_soft_margin_loss_backward(const Tensor grad_loss,
                                    const Tensor input1,
                                    const Tensor input2,
                                    const Tensor target,
                                    const float margin,
                                    const char *reduction,
                                    Tensor *grad_in1,
                                    Tensor *grad_in2);
```

后向接口参数：

grad_loss(IN)：表示第一个排序得分输入张量，元素数据类型可以为 FLOAT32、FLOAT64。

input1 (IN)：表示第一个排序得分输入张量，元素数据类型可以为 FLOAT32、FLOAT64。

input2 (IN)：表示第二个排序得分输入张量，元素数据类型可以为 FLOAT32、FLOAT64。

target (IN)：表示期望值张量，第一个排序得分值大于第二个排序得分值时值为 1，否则为-1，元素数据类型可以为 FLOAT32、FLOAT64。

margin(IN)：间隔值，表示两个得分之间的差异。

reduction(IN) ：表示规约类型，从“none”，“mean”和“sum”三者间任选其一，默认为“mean”。

grad_in1 (OUT)：表示第一个排序得分输入张量的梯度。

grad_in2 (OUT)：表示第二个排序得分输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* input1:  [[1.0, 2.0], [3.0, 4.0]] */
/* input2:  [[2.0, 1.0], [2.0, 4.0]] */
/* target:  [[1, -1], [-1, -1]] */
/* reduction: “none” */
Tensor loss;
op_margin_ranking_loss_forward(input1, input2, target, 0.0, reduction, &loss);
/* loss:  [[1, 1], [1, 0]] */
```

A.2.3 正则函数

A.2.3.1 随机失活函数

前向接口 C 语言：

```
Status op_dropout_forward (const Tensor input,
                           const double rate,
                           const int axis,
                           const int64_t seed,
                           const bool is_train,
```

```
                    Tensor *output,
                    Tensor *mask);
```

前向接口参数：

logits (IN)：表示输入张量。

rate (IN)：表示随即失活的失活比率。

axis (IN)：表示随机失活所沿着的轴。

seed (IN)：表示该操作计算中随机种子。

is_train (IN)：表示是否随机失活，True 则表示正在使用随机失活，False 则不会使用随机失活。

output (OUT)：表示输出张量。

mask (OUT)：表示输出掩码矩阵。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

后向接口 C 语言：

```
Status op_dropout_backward(const Tensor grad_out,
                           const double rate,
                           const int axis,
                           const int64_t seed,
                           Tensor mask,
                           Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

rate (IN) ：表示输入张量中元素变为 0 的概率。

axis (IN)：表示随机失活所沿着的轴。

seed (IN)：表示生成随机数的种子。

mask(IN)：表示失活掩码。

grad_in(OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

示例：

```
/* input: data = [[[5, 2, 3],
                   [2, 1, 4]],
                  [[7, 8, 9],
                   [5, 4, 6]]] */
/* rate = 0.5 */
/* axis = 1 */
/* seed = 0 */
/* is_train = true */
op_dropout_forward(input, rate, axis, seed, is_train, &output, &mask);
/* output: data = [[[5, 2, 0],
```

```
                [2, 0, 0]],
               [[0, 0, 0],
                [5, 4, 6]]] */
        /*mask:data=[[[1, 1, 0],
                [1, 0, 0]],
               [[0, 0, 0],
               [1, 1, 1]]]*/
```

A.2.3.2 标签平滑函数

C语言：

```
Status op_label_smooth(const Tensor label,
                       const Tensor prior_dist,
                       const float epsilon,
                       const DataType dtype,
                       Tensor *output);
```

参数：

label (IN)：表示标签张量。

prior_dist (IN)：先验分布张量。

epsilon (IN)：表示用于混合原始真实分布和固定分布的权重。

dtype (IN)：表示输出张量数据类型。

output (INOUT)：表示输出张量。

返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

示例：

```
/* label:  [0, 0, 1, 0] */
/* prior_dist: [0.1, 0.1, 0.1, 0.1] */
/* epsilon: 0.1 */
/* dtype: FLOAT32 */
Tensor output;
op_label_smooth(label, prior_dist, epsilon, dtype, &output);
/* output: [0.1, 0.1, 0.9, 0.1] */
```

A.2.4 归一化函数

A.2.4.1 批量归一化操作

前向接口 C 语言：

```
Status op_batch_norm_forward(const Tensor input,
                             const int axis,
                             const Tensor scale,
                             const Tensor bias,
                             const Tensor *mean,
                             const Tensor *variance,
```

```
                        const double epsilon,
                        const float momentum ,
                        const bool is_train,
                        Tensor *output,
                        Tensor *running_mean,
                        Tensor *running_var,);
```

前向接口参数：

input （IN）：表示维度的形式为（N x C x D1 x D2…Dn）的张量数据，其中 N 是批量大小，C 是通道数。

axis （IN）：表示批量归一化所沿的轴。

scale （IN）：表示形状为输入数据中 C 的缩放张量。

bias （IN）：表示形状为输入数据中 C 的偏置张量。

mean （IN）：表示形状为输入数据中 C 的运行或是测试中使用的均值张量。

variance （IN）：表示形状为输入数据中 C 的运行或是测试中使用的方差张量。

epsilon （IN）：表示为了数值稳定加在分母上的权重。

momentum （IN）：表示用于计算运行平均值和方差的因子。

is_train （IN）：表示批量归一化是否使用。

output （OUT）：表示输出张量。

running_mean （OUT）：表示批量归一化操作后的运行均值张量。

running_var （OUT）：表示批量归一化操作后的运行方差张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对未初始化。

STATUS_INTERNAL_ERROR：出现除数为 0 的情况等。

后向接口 C 语言：

```
Status op_batch_norm_backward(const Tensor grad_out,
                        const Tensor input,
                        const Tensor output_cache,
                        const int axis,
                        const Tensor scale,
                        const Tensor bias,
                        const Tensor *mean,
                        const Tensor *variance,
                        const double epsilon,
                        Tensor *grad_scale,
                        Tensor *grad_bias,
                        Tensor *grad_in);
```

后向接口参数：

grad_out （IN）：表示输出张量的梯度。

input(IN)：表示输入张量。

output_cache(IN)：表示中间结果张量。

axis (IN)：表示批量归一化维度。
scale (IN)：表示缩放量张量。
bias (IN)：表示偏移量张量。
mean (IN)：表示样本均值张量。
variance (IN)：表示样本方差张量。
epsilon (IN)：表示防止除数为 0 的小数字。
grad_scale(OUT)：缩放量的梯度。
grad_bias(OUT)：偏移量的梯度。
grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_INVALID_ARGRUMENT：表示参数出错。
STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法未初始化。
STATUS_INTERNAL_ERROR：出现除数为 0 的情况等。

示例：

```
/* input:  [[[1.0, 2.0, 3.0],
             [4.0, 5.0, 6.0]],
            [[7.0, 8.0, 9.0],
             [10.0, 11.0, 12.0]]] */
/* axis: 1 */
/* scale: [1.0, 1.0, 1.0] */
/* bias: [0.0, 0.0, 0.0] */
/* mean: [5.5, 6.5, 7.5] */
/* variance: [8.25, 8.25, 8.25] */
/* epsilon: 1e-5 */
/* momentum: 0.9 */
/* is_train: true */
Tensor output;
Tensor running_mean;
Tensor running_var;
op_batch_norm_forward(input, axis, scale, bias, &mean, &variance, epsilon, momentum,
is_train, &output, &running_mean, &running_var);
/* output: [[[ -1.264911, -0.632456, 0.        ],
             [ 0.632456, 1.264911, 1.897356]],
            [[-1.264911, -0.632456, 0.        ],
             [ 0.632456, 1.264911, 1.897356]]] */
/* running_mean: [5.95, 6.95, 7.95] */
/* running_var: [8.225, 8.225, 8.225] */
```

A.2.4.2 分组归一化操作

前向接口 C 语言：

```
Status op_group_norm_forward(const Tensor input,
                             const int channel_axis,
```

```
                                const int *reduced_axes,
                                const int reduced_axes_len,
                                const Tensor *mean,
                                const Tensor *variance,
                                       const Tensor scale,
                                       const Tensor bias,
                                       const double epsilon,
                                       const int groups,
                                       Tensor *output,
                                      Tensor *running_mean,
                                      Tensor *running_var);
```

前向接口参数：

input （IN）：表示维度的形式为（N x C x D1 x D2…Dn）的张量数据，其中 N 是批量大小，C 是通道数。

groups(IN)：指定归约维度分为多少个组。

channel_axis （IN）：表示分组轴。

reduced_axes （IN）：表示归约轴。

reduced_axes_len （IN）：表示归约轴 reduced_axes 数组长度。

scale （IN）：表示形状为输入数据中 C 的缩放张量。

bias （IN）：表示形状为输入数据中 C 的偏置张量。

mean （IN）: 表示样本均值张量。

variance （IN）：表示样本方差张量。

epsilon （IN）：表示避免除 0 操作的权重。

output （OUT）：表示输出张量。

running_mean （OUT）：表示分组归一化操作后的运行均值张量。

running_var （OUT）：表示分组归一化操作后的运行方差张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_INTERNAL_ERROR：出现除数为 0 的情况等。

后向接口 C 语言：

```
Status op_group_norm_backward(const Tensor grad_out,
                              const Tensor input,
                              const Tensor output_cache,
                              const int groups,
                              const int channel_axis,
                              const int *reduced_axes,
                              const int reduced_axes_len,
                              const Tensor scale,
                              const Tensor bias,
```

```
                                const Tensor  *mean,
                                const Tensor  *variance,
                                const double epsilon,
                                Tensor *grad_scale,
                                Tensor *grad_bias,
                                Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示输入张量。

output_cache(IN)：表示中间结果张量。

groups (IN)：表示分组数。

channel_axis (IN)：表示分组轴。

reduced_axes (IN)：表示归约轴。

reduced_axes_len (IN)：表示归约轴 reduced_axes 数组长度。

scale (IN)：表示缩放量张量。

bias (IN)：表示偏移量张量。

mean (IN)：表示样本均值张量。

variance (IN)：表示样本方差张量。

epsilon (IN)：表示防止除数为 0 的小数字。

grad_scale(OUT)：缩放量的梯度。

grad_bias(OUT)：偏移量的梯度。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_INTERNAL_ERROR：出现除数为 0 的情况等。

示例：

```
/* input:  [[[1.0, 2.0, 3.0],
             [4.0, 5.0, 6.0]],
            [[7.0, 8.0, 9.0],
             [10.0, 11.0, 12.0]]] */
/* channel_axis: 1 */
/* reduced_axes: [2, 3] */
/* reduced_axes_len: 2 */
/* mean: [3.5, 9.5] */
/* variance: [5.25, 5.25] */
/* scale: [1.0, 1.0, 1.0] */
/* bias: [0.0, 0.0, 0.0] */
/* epsilon: 1e-5 */
/* groups: 2 */
Tensor output;
```

```
Tensor running_mean;
Tensor running_var;
op_group_norm_forward(input, channel_axis, reduced_axes, reduced_axes_len, &mean,
&variance, scale, bias, epsilon, groups, &output, &running_mean, &running_var);
/* output: [[[ -1.264911, -0.632456, 0.        ],
             [ 0.632456, 1.264911, 1.897356]],
            [[-1.264911, -0.632456, 0.        ],
             [ 0.632456, 1.264911, 1.897356]]] */
/* running_mean: [3.5, 9.5] */
/* running_var: [5.25, 5.25] */
```

A.2.4.3 权重归一化操作

C 语言：

```
Status op_weight_norm(const Tensor weight,
                      const int axis,
                      Tensor *weight_g,
                      Tensor *weight_v);
```

参数：

weight (IN)：表示权重张量。
axis (IN)：表示归一化的维度。
weight_g (OUT)：表示代表长度的 1-D 张量。
weight_v (OUT)：表示代表方向的 1-D 张量。

返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。
STATUS_INVALID_ARGRUMENT：表示参数出错。

示例：

```
/* weight: [[0.5, 0.2, -0.3],
            [0.8, -0.1, 0.4]] */
/* axis: 0 */
Tensor weight_g;
Tensor weight_v;
op_weight_norm(weight, axis, &weight_g, &weight_v);
/* weight_g: [[0.5831, 0.5831, 0.5831],
              [0.5831, 0.5831, 0.5831]] */
/* weight_v: [[0.1, 0.2, -0.3],
              [0.8, -0.1, 0.4]] */
```

A.2.4.4 谱归一化操作

C 语言：

```
Status op_spectral_norm(const Tensor weight,
                        const int axis,
                        const int num_iters,
```

```
                const double epsilon,
                Tensor *output);
```

参数：

weight (IN)：表示权重张量，一般是 fc、conv 层的权重，元素数据类型可以为 FLOAT32、FLOAT64。

axis (IN)：表示归一化的维度。

num_iters (IN)：表示迭代轮数。

epsilon (IN)：表示防止除数为 0 的小数字。

output (OUT)：表示输出权重张量，形状、数据类型与输入权重一致。

返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

示例：

```
/* weight: [[0.5, 0.2],
            [0.8, -0.1],
            [-0.3, 0.4]] */
/* axis: 0 */
/* num_iters: 10 */
/* epsilon: 1e-12 */
Tensor output;
op_spectral_norm(weight, axis, num_iters, epsilon, &output);
/* output: [[0.4226, 0.1690],
            [0.6767, -0.0945],
            [-0.2476, 0.3118]] */
```

A.2.4.5 层归一化操作

前向接口 C 语言：

```
Status op_layer_norm_forward(const Tensor input,
                             const Tensor *scale,
                             const Tensor *bias,
                             const double epsilon,
                                 const int begin_norm_axis,
                                 Tensor *output,
                             Tensor *running_mean,
                             Tensor *running_var);
```

前向接口参数：

input (IN)：表示需要被归一化的输入张量。

scale (IN)：表示缩放张量。

bias (IN)：表示偏置张量。

begin_norm_axis (IN) ：表示从哪个维度开始归一化。

epsilon (IN)：表示避免除 0 操作的权重。

output (OUT)：表示输出张量。

running_mean (OUT)：表示训练期间用于加速梯度计算的保存平均值。

running_var(OUT)：表示训练期间用于加速梯度计算的保存方差。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_INTERNAL_ERROR：出现除数为 0 的情况等。

后向接口 C 语言：

```
Status op_layer_norm_backward(const Tensor grad_out,
                              const Tensor input,
                              const Tensor output_cache,
                              const Tensor scale,
                              const Tensor bias,
                              const int  begin_norm_axis,
                              const double epsilon,
                              Tensor *grad_scale,
                              Tensor *grad_bias,
                              Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示输入张量。

output_cache(IN)：表示中间结果张量。

scale (IN)：表示缩放量张量。

bias (IN)：表示偏移量张量。

begin_norm_axis (IN)：表示归一化的维度。

epsilon (IN)：表示防止除数为 0 的小数字。

grad_scale(OUT)：缩放量的梯度。

grad_bias(OUT)：偏移量的梯度。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_INTERNAL_ERROR：出现除数为 0 的情况。

示例：

```
/* input: [[[1.0, 2.0, 3.0],
            [4.0, 5.0, 6.0]],
           [[7.0, 8.0, 9.0],
            [10.0, 11.0, 12.0]]] */
/* scale: [1.0, 1.0, 1.0] */
/* bias: [0.0, 0.0, 0.0] */
/* epsilon: 1e-5 */
```

```
/* begin_norm_axis: 1 */
Tensor output;
Tensor running_mean;
Tensor running_var;
op_layer_norm_forward(input, &scale, &bias, epsilon, begin_norm_axis, &output,
&running_mean, &running_var);
/* output: [[[ -1.224744, -0.612372, 0.        ],
             [ 0.612372, 1.224744, 1.837117]],
            [[-1.224744, -0.612372, 0.        ],
             [ 0.612372, 1.224744, 1.837117]]] */
/* running_mean: [3.5, 9.5] */
/* running_var: [5.25, 5.25] */
```

A.2.4.6 实例归一化操作

前向接口 C 语言：

```
Status op_ instance_norm_forward(const Tensor input,
                        const int axis,
                        const Tensor scale,
                        const Tensor bias,
                        const double epsilon,
                        Tensor *output );
```

前向接口参数：

input (IN)：表示需要被归一化的输入张量。
axis (IN)：表示归一化的维度。
scale (IN)：表示缩放张量。
bias (IN)：表示偏置张量。
epsilon (IN)：表示避免除 0 操作的权重。
output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_INVALID_ARGRUMENT: 表示参数出错。
STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。
STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。
STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。
STATUS_INTERNAL_ERROR：出现除数为 0 的情况。

后向接口 C 语言：

```
Status op_instance_norm_backward(const Tensor grad_out,
                        const Tensor input,
                        const Tensor output_cache,
                        const Tensor scale,
                        const Tensor bias,
                        const int axis,
                        const double epsilon,
```

```
                                Tensor *grad_scale,
                                Tensor *grad_bias,
                                Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示输入张量。

output_cache(IN)：表示中间结果张量。

scale (IN)：表示缩放量张量。

bias (IN)：表示偏移量张量。

axis (IN)：表示归一化的维度。

epsilon (IN)：表示防止除数为 0 的小数字。

grad_scale(OUT)：缩放量的梯度。

grad_bias(OUT)：偏移量的梯度。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_INTERNAL_ERROR：出现除数为 0 的情况。

示例：

```
/* input: [[[1.0, 2.0, 3.0],
            [4.0, 5.0, 6.0]],
           [[7.0, 8.0, 9.0],
            [10.0, 11.0, 12.0]]] */
/* axis: 1 */
/* scale: [1.0, 1.0, 1.0] */
/* bias: [0.0, 0.0, 0.0] */
/* epsilon: 1e-5 */
Tensor output;
op_instance_norm_forward(input, axis, scale, bias, epsilon, &output);
/* output: [[[ -1.224744, -0.612372, 0.        ],
             [ 0.612372, 1.224744, 1.837117]],
            [[-1.224744, -0.612372, 0.        ],
             [ 0.612372, 1.224744, 1.837117]]] */
```

A.2.4.7 局部响应归一化操作

前向接口 C 语言：

```
Status op_lrn_norm_forward(const Tensor input,
                           const int radius,
                           const double alpha,
                           const double beta,
                           const double bias,
```

```
                        Tensor *output);
```

前向接口参数：

input (IN)：表示输入张量。

radius(IN)：表示局部归一化的半径。

alpha(IN)：表示相乘系数。

beta(IN)：表示指数系数。

bias(IN)：表示偏置系数。

output (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法未初始化。

后向接口 C 语言：

```
Status op_lrn_norm_backward(const Tensor grad_out,
                        const Tensor input,
                        const int radius,
                        const double alpha,
                        const double beta,
                        const double bias,
                        Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示输入张量。

radius(IN)：表示局部归一化的半径。

alpha(IN)：表示相乘系数。

beta(IN)：表示指数系数。

bias(IN)：表示偏置系数。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

示例：

```
/* input: [[[1.0, 2.0, 3.0],
            [4.0, 5.0, 6.0]],
           [[7.0, 8.0, 9.0],
            [10.0, 11.0, 12.0]]] */
/* radius: 1 */
/* alpha: 0.0001 */
/* beta: 0.75 */
/* bias: 2.0 */
```

```
Tensor output;
op_lrn_norm_forward(input, radius, alpha, beta, bias, &output);
/* output: [[[ 0.49988, 0.99972, 1.49954],
             [ 1.99940, 2.49929, 2.99919]],
            [[ 3.49913, 3.99907, 4.49902],
             [ 4.99897, 5.49893, 5.99890]]] */
```

A.2.4.8 L2归一化操作

前向接口 C 语言：

```
Status op_l2_norm_forward(const Tensor input,
                          const int axis,
                          const double epsilon,
                          Tensor *output);
```

前向接口参数：

input (IN)：表示输入张量。

axis (IN)：表示归一化的轴。

epsilon (IN)：表示防止除数为 0 的小数字。

output (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_INTERNAL_ERROR：出现除数为 0 的情况。

后向接口 C 语言：

```
Status op_l2_norm_backward(const Tensor grad_out,
                           const Tensor input,
                           const int axis,
                           const double epsilon,
                           Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input(IN)：表示输入张量。

axis (IN)：表示归一化的轴。

epsilon (IN)：表示防止除数为 0 的小数字。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_INTERNAL_ERROR：出现除数为 0 的情况。

示例：

```
/* input: [[1.0, 2.0, 3.0],
           [4.0, 5.0, 6.0]] */
/* axis: 1 */
/* epsilon: 1e-12 */
Tensor output;
op_l2_norm_forward(input, axis, epsilon, &output);
/* output: [[0.26726124, 0.53452248, 0.80178373],
            [0.45584231, 0.56980288, 0.68376346]] */
```

A.2.4.9 Lp范数归一化操作

前向接口 C 语言：

```
Status op_lp_norm_forward(const Tensor input,
                          const float p,
                          const int axis,
                          const double epsilon,
                          Tensor *output);
```

前向接口参数：

input (IN)：表示输入张量。

p (IN)：表示范数计算中的指数值，通常取 2。

axis (IN)：表示归一化的轴。

epsilon (IN)：表示防止除数为 0 的小数字。

output (INOUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_INTERNAL_ERROR：出现除数为 0 的情况。

后向接口 C 语言：

```
Status op_lp_norm_backward(const Tensor grad_out,
                           const Tensor input,
                           const float p,
                           const int axis,
                           const double epsilon,
                           Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input (IN)：表示输入张量。

p (IN)：表示范数计算中的指数值，通常取 2。

axis (IN)：表示归一化的轴。

epsilon (IN)：表示防止除数为 0 的小数字。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_INTERNAL_ERROR：出现除数为 0 的情况。

示例：

```
/* input: [[1.0, 2.0, 3.0],
           [4.0, 5.0, 6.0]] */
/* p: 3.0 */
/* axis: 1 */
/* epsilon: 1e-12 */
Tensor output;
op_lp_norm_forward(input, p, axis, epsilon, &output);
/* output: [[0.26726124, 0.53452248, 0.80178373],
            [0.45584231, 0.56980288, 0.68376346]] */
```

A.2.5 池化函数

A.2.5.1 一维池化操作

前向接口 C 语言：

```
Status op_pool1d_forward(const Tensor input,
                         const char *mode,
                         const int *ksize,
                         const int ksize_len,
                         const int *stride,
                         const int stride_len,
                         const int *padding,
                         const int padding_len,
                         const int *dilation,
                         const int dilation_len,
                         const bool ceil_mode,
                         const bool exclusive,
                         const bool adaptive
                         Tensor *output );
```

前向接口参数：

input (IN)：表示输入张量

mode (IN)：表示池化类型

ksize (IN)：表示池化窗口大小。

ksize_len (IN)：表示数组 ksize 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的窗口大小一样。

stride (IN)：表示池化步长。

stride_len (IN)：表示数组 stride 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的跨步一样。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的填充个数一样。

dilation(IN)：表示池化膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以等于 1 或者 2。

ceil_mode (IN)：是否使用 ceil 函数计算输出高度和宽度。

exclusive (IN)：是否忽略填充值。

adaptive (IN)：表示表示是否使用自适应方式，默认为 false。

output (OUT)：表示输出张量 。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_pool1d_backward(const Tensor grad_out,
                          const Tensor input,
                          const char *mode,
                          const int *ksize,
                          const int ksize_len,
                          const int *stride,
                          const int stride_len,
                          const int *padding,
                          const int padding_len,
                          const int *dilation,
                          const int dilation_len,
                          const bool ceil_mode,
                          const bool exclusive,
                          const bool adaptive,
                          Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input (IN)：表示输入张量。

mode (IN)：表示池化类型。

ksize (IN)：表示池化窗口大小。

ksize_len (IN)：表示数组 ksize 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的窗口大小一样。

stride (IN)：表示池化步长。

stride_len (IN)：表示数组 stride 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的跨步一样。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的填充个数一样。

dilation(IN)：表示池化膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以等于 1 或者 2。

ceil_mode (IN)：是否使用 ceil 函数计算输出高度和宽度。

adaptive (IN)：表示表示是否使用自适应方式，默认为 false。

exclusive (IN)：是否忽略填充值。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

示例：

```
/* x:  [[[1, 2, 3, 4, 5, 6],
       [1, 2, 3, 4, 5, 6],
       [1, 2, 3, 4, 5, 6]]] */
/* mode: “max” */
/* ksize: 2 */
/* stride: 2 */
/* padding: 0 */
/* dilation: 0 */
/* ceil_mode = false */
/* exclusive = true */
Tensor y;
op_pool1d_forward (x, mode, ksize, stride, padding, dilation, ceil_mode, exclusive,
&y, &indices);
/* y:  [[[2, 4, 6],
        [2, 4, 6],
        [2, 4, 6]]] */
```

A.2.5.2 二维池化操作

前向接口 C 语言：

```
Status op_pool2d_forward(const Tensor input,
                         const char *mode,
                         const int *ksize,
                         const int ksize_len,
                         const int *stride,
                         const int stride_len,
                         const int *padding,
                         const int padding_len,
                         const int *dilation,
                         const int dilation_len,
                         const bool ceil_mode,
                         const bool exclusive,
                         const bool adaptive
```

```
                    Tensor *output );
```

前向接口参数：

input (IN)：表示输入张量

mode (IN)：表示池化类型

ksize (IN)：表示池化窗口大小。

ksize_len (IN)：表示数组 ksize 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的窗口大小一样。

stride (IN)：表示池化步长。

stride_len (IN)：表示数组 stride 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的跨步一样。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的填充个数一样。

dilation(IN)：表示池化膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以等于 1 或者 2。

ceil_mode (IN)：是否使用 ceil 函数计算输出高度和宽度。

exclusive (IN)：是否忽略填充值。

adaptive (IN)：表示表示是否使用自适应方式，默认为 false。

output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_pool2d_backward(const Tensor grad_out,
                    const Tensor input,
                    const char *mode,
                    const int *ksize,
                    const int ksize_len,
                    const int *stride,
                    const int stride_len,
                    const int *padding,
                    const int padding_len,
                    const int *dilation,
                    const int dilation_len,
                    const bool ceil_mode,
                    const bool exclusive,
                    const bool adaptive,
                    Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input (IN)：表示输入张量。

mode (IN)：表示池化类型

ksize (IN)：表示池化窗口大小。

ksize_len (IN)：表示数组 ksize 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的窗口大小一样。

stride (IN)：表示池化步长。

stride_len (IN)：表示数组 stride 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的跨步一样。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的填充个数一样。

dilation(IN)：表示池化膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的膨胀大小一样。

ceil_mode (IN)：是否使用 ceil 函数计算输出高度和宽度。

adaptive (IN)：表示表示是否使用自适应方式，默认为 false。

exclusive (IN)：是否忽略填充值。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

示例：

```
/* x:  [[1, 1, 1],
        [2, 2, 2],
        [3, 3, 3]] */
/* mode: “max” */
/* ksize: [2, 2] */
/* ksize_len: 2 */
/* stride: [1, 1] */
/* stride_len: 2 */
/* padding: [0, 0] */
/* padding_len: 2 */
/* dilation: [0, 0] */
/* dilation_len: 2 */
/* ceil_mode = false */
/* exclusive = true */
Tensor y;
op_pool2d_forward (x, mode, ksize, ksize_len, stride, stride_len, padding,
padding_len, dilation, dilation_len, ceil_mode, exclusive, &y);
/* y:  [[2, 2],
       [3, 3]] */
```

A.2.5.3 三维池化操作

前向接口 C 语言：

```
Status op_pool3d_forward (const Tensor input,
                          const char *mode,
                          const int *ksize,
                          const int ksize_len,
                          const int *stride,
                          const int stride_len,
                          const int *padding,
                          const int padding_len,
                          const int *dilation,
                          const int dilation_len,
                          const bool ceil_mode,
                          const bool exclusive,
                          const bool adaptive
                          Tensor *output);
```

前向接口参数：

input (IN)：表示输入张量

mode (IN)：表示池化类型

ksize (IN)：表示池化窗口大小。

ksize_len (IN)：表示数组 ksize 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的窗口大小一样。

stride (IN)：表示池化步长。

stride_len (IN)：表示数组 stride 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的跨步一样。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的填充个数一样。

dilation(IN)：表示池化膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以等于 1 或者 2。

ceil_mode (IN)：是否使用 ceil 函数计算输出高度和宽度。

exclusive (IN)：是否忽略填充值。

adaptive (IN)：表示表示是否使用自适应方式，默认为 false。

output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_pool3d_backward(const Tensor grad_out,
                          const Tensor input,
                          const char *mode,
                          const int *ksize,
```

```
                        const int ksize_len,
                        const int *stride,
                        const int stride_len,
                        const int *padding,
                        const int padding_len,
                        const int *dilation,
                        const int dilation_len,
                        const bool ceil_mode,
                        const bool exclusive,
                        const bool adaptive,
                        Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input (IN)：表示输入张量。

mode (IN)：表示池化类型

ksize (IN)：表示池化窗口大小。

ksize_len (IN)：表示数组 ksize 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的窗口大小一样。

stride (IN)：表示池化步长。

stride_len (IN)：表示数组 stride 的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的跨步一样。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的填充个数一样。

dilation(IN)：表示池化膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以等于 1 或者 2。如果为 1，则表示所有维度的膨胀大小一样。

ceil_mode (IN)：是否使用 ceil 函数计算输出高度和宽度。

adaptive (IN)：表示表示是否使用自适应方式，默认为 false。

exclusive (IN)：是否忽略填充值。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

示例：

```
/* input: [[[[1, 2, 3],
             [4, 5, 6],
             [7, 8, 9]],
            [[10, 11, 12],
             [13, 14, 15],
             [16, 17, 18]],
```

```
                [[19, 20, 21],
                 [22, 23, 24],
                 [25, 26, 27]]],
               [[[28, 29, 30],
                 [31, 32, 33],
                 [34, 35, 36]],
                [[37, 38, 39],
                 [40, 41, 42],
                 [43, 44, 45]],
                [[46, 47, 48],
                 [49, 50, 51],
                 [52, 53, 54]]]]] */
/* mode: "max" */
/* ksize: [2, 2, 2] */
/* stride: [1, 1, 1] */
/* padding: [0, 0, 0] */
/* dilation: [1, 1, 1] */
/* ceil_mode: false */
/* exclusive: true */
/* adaptive: false */
Tensor output;
op_pool3d_forward(input, "max", ksize, 3, stride, 3, padding, 3, dilation, 3, false, true,
false, &output);
/* output: [[[[[14, 15],
                [17, 18]],
               [[23, 24],
                [26, 27]]],
              [[[41, 42],
                [44, 45]],
               [[50, 51],
                [53, 54]]]]] */
```

A.2.5.4 自适应一维池化操作

前向接口 C 语言：

```
Status op_adaptive_pool1d_forward(const Tensor input,
                                  const char *mode,
                                  const int *output_size,
                                  Tensor *output);
```

前向接口参数：

input (IN)：表示输入张量。

mode (IN)：表示池化类型。

output_size (IN)：表示输出的长度。

output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_pool1d_backward(const Tensor grad_out,
                          const Tensor input,
                          const char *mode,
                          const int *output_size,
                          Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input (IN)：表示输入张量。

mode (IN)：表示池化类型

output_size (IN)：表示输出的长度。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

示例：

```
/* x:  [[[1, 2, 3, 4, 5, 6],
       [1, 2, 3, 4, 5, 6],
       [1, 2, 3, 4, 5, 6]]] */
/* mode: “max” */
/* output_size: 3 */
Tensor y;
op_adaptive_pool1d_forward (x, mode, output_size, &y);
/* y:  [[[2, 4, 6],
        [2, 4, 6],
        [2, 4, 6]]] */
```

A.2.5.5 自适应二维池化操作

前向接口 C 语言：

```
Status op_adaptive_pool2d_forward(const Tensor input,
                          const char *mode,
                          const int *output_size,
                          Tensor *output);
```

前向接口参数：

input (IN)：表示输入张量。

mode (IN)：表示池化类型。

output_size (IN)：表示输出的长度。

output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_pool2d_backward(const Tensor grad_out,
                          const Tensor input,
                          const char *mode,
                          const int *output_size,
                          Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input (IN)：表示输入张量。

mode (IN)：表示池化类型

output_size (IN)：表示输出的长度。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* x:  [[[[1,2,3,4],
       [1,2,3,4],
       [1, 2, 3, 4]],
       [[1,2,3,4],
       [1,2,3,4],
       [1, 2, 3, 4]],
       [[1,2,3,4],
       [1,2,3,4],
       [1, 2, 3, 4]]]*/
/* mode: “max” */
/* output_size: (2, 2) */
Tensor y;
op_adaptive_pool2d_forward (x, mode, output_size, &y);
/* y:  [[[[2, 4],
        [2, 4]],
        [[2, 4],
        [2, 4]]]*/
```

A.2.5.6 自适应三维池化操作

前向接口 C 语言：

```
Status op_adaptive_pool3d_forward(const Tensor input,
                    const char *mode,
                    const int *output_size,
                    Tensor *output);
```

前向接口参数：

input (IN)：表示输入张量。

mode (IN)：表示池化类型。

output_size (IN)：表示输出的长度。

output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

后向接口 C 语言：

```
Status op_pool3d_backward(const Tensor grad_out,
                    const Tensor input,
                    const char *mode,
                    const int *output_size,
                    Tensor *grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input (IN)：表示输入张量。

mode (IN)：表示池化类型

output_size (IN)：表示输出的长度。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象未初始化。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
    /* x:  [[[[0.18808374, 0.02505297],
               [0.54680401, 0.59056723]],
              [[0.89189357, 0.78992540],
               [0.12694182, 0.90172297]]],
             [[[0.62920904, 0.23778178],
               [0.43813652, 0.13374270]],
              [[0.59860146, 0.12039188],
               [0.46718234, 0.84129763]]],
             [[[0.84327465, 0.81297994],
```

```
                [0.47787419, 0.86686862]],
               [[0.83672047, 0.54137236],
                [0.90560102, 0.99288368]]]]*/
    /* mode: “max” */
    /* output_size: (1, 1) */
    Tensor y;
    op_adaptive_pool2d_forward (x, mode, output_size, &y);
    /* y:  [[[[0.90172297]]],
           [[[0.84129763]]],
           [[[0.99288368]]]]*/
```

A.2.6 卷积函数

A.2.6.1 一维卷积操作

前向 C 接口：

```
Status op_conv1d_forward (const Tensor input,
                          const Tensor filter,
                          const int *stride,
                          const int stride_len,
                          const int *padding,
                          const int padding_len,
                          const int *dilation,
                          const int dilation_len,
                          const int group_count,
                            Tensor *output);
```

前向接口参数：

input (IN)：表示 3-D 输入张量。

filter (IN)：表示卷积核，3-D 张量。

stride (IN)：表示卷积步长。

stride_len (IN)：数组 stride 的长度，可以设置成 1。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以设置成 1。

dilation(IN)：表示卷积膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1。

group_count (IN)：表示进行分组卷积的组数。

output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

STATUS_ALLOC_FAILED：表示输出张量分配空间不足。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_INTERNAL_ERROR：表示内部的调用操作出错。

后向 C 接口：

```
Status op_conv1d_backward_data(const Tensor grad_out,
                               const Tensor filter,
                               const int *stride,
                               const int stride_len,
                               const int *padding,
                               const int padding_len,
                               const int *dilationn,
                               const int dilation_len,
                               const int group_count,
                               Tensor * grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。
filter (IN)：表示卷积核，3-D 张量。
stride (IN)：表示卷积步长。
stride_len (IN)：数组 stride 的长度，可以设置成 1。
padding (IN)：表示填充元素个数。
padding_len (IN)：表示 padding 数组的长度，可以设置成 1。
dilation(IN)：表示卷积膨胀个数。
dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1。
group_count (IN)：表示进行分组卷积的组数。
grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。
STATUS_ALLOC_FAILED：表示输出张量分配空间不足。
STATUS_INVALID_ARGRUMENT：表示参数出错。
STATUS_INTERNAL_ERROR：表示内部的调用操作出错。

后向 C 接口：

```
Status op_conv1d_backward_filter(const Tensor grad_out,
                                 const Tensor input,
                                 const int *stride,
                                 const int stride_len,
                                 const int *padding,
                                 const int padding_len,
                                 const int *dilation,
                                 const int dilation_len,
                                 const int group_count,
                                 Tensor * grad_filter);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。
input (IN)：表示输入张量，4-D 张量。
stride (IN)：表示卷积步长。
stride_len (IN)：数组 stride 的长度，可以设置成 1 或者 2。

padding（IN）：表示填充元素个数。
padding_len（IN）：表示 padding 数组的长度，可以设置成 1 或者 2。
dilation(IN)：表示卷积膨胀个数。
dilation_len（IN）：表示 dilation 数组的长度，可以设置成 1 或者 2。
group_count（IN）：表示进行分组卷积的组数。
grad_filter（OUT）：表示卷积核的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。
STATUS_ALLOC_FAILED：表示输出张量分配空间不足。
STATUS_INVALID_ARGRUMENT：表示参数出错。
STATUS_INTERNAL_ERROR：表示内部的调用操作出错。

示例：

```
/* input: [[[1.0, 2.0, 3.0, 4.0, 5.0]]] */
/* filter: [[[0.2, 0.5, 0.3]]] */
/* stride: [1] */
/* padding: [1] */
/* dilation: [1] */
/* group_count: 1 */
Tensor output;
op_conv1d_forward(input, filter, stride, 1, padding, 1, dilation, 1, group_count,
&output);
/* output: [[[0.7, 1.6, 2.5, 3.4, 3.0]]] */
```

A.2.6.2 二维卷积操作

前向接口 C 语言：

```
Status op_conv2d_forward (const Tensor input,
                          const Tensor filter,
                          const int *stride,
                          const int stride_len,
                          const int *padding,
                          const int padding_len,
                          const int *dilation,
                          const int dilation_len,
                          const int group_count,
                          Tensor *output);
```

前向接口参数：

input（IN）：表示 4-D 输入张量。
filter（IN）：表示卷积核，4-D 张量。
stride（IN）：表示卷积步长。
stride_len（IN）：数组 stride 的长度，可以设置成 1 或者 2。
padding（IN）：表示填充元素个数。
padding_len（IN）：表示 padding 数组的长度，可以设置成 1 或者 2。

dilation(IN)：表示卷积膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1 或者 2。

group_count (IN)：表示进行分组卷积的组数。

output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：类型不匹配

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

后向接口 C 语言：

```
Status op_conv2d_backward_data (const Tensor grad_out,
                        const Tensor filter,
                        const int *stride,
                        const int stride_len,
                        const int *padding,
                        const int padding_len,
                        const int *dilation,
                        const int dilation_len,
                        const int group_count,
                        Tensor * grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

filter (IN)：表示卷积核，4-D 张量。

stride (IN)：表示卷积步长。

stride_len (IN)：数组 stride 的长度，可以设置成 1 或者 2。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以设置成 1 或者 2。

dilation(IN)：表示卷积膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1 或者 2。

group_count (IN)：表示进行分组卷积的组数。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：类型不匹配

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

后向接口 C 语言：

```
Status op_conv2d_backward_filter(const Tensor grad_out,
                        const Tensor input,
                        const int *stride,
                        const int stride_len,
                        const int *padding,
                        const int padding_len,
```

```
                    const int *dilation,
                    const int dilation_len,
                    const int group_count,
                    Tensor * grad_filter);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input (IN)：表示输入张量，4-D 张量。

stride (IN)：表示卷积步长。

stride_len (IN)：数组 stride 的长度，可以设置成 1 或者 2。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以设置成 1 或者 2。

dilation(IN)：表示卷积膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1 或者 2。

group_count (IN)：表示进行分组卷积的组数。

grad_filter (OUT)：表示卷积核的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

STATUS_TYPE_MISMATCH：类型不匹配。

示例：

```
/* input: [[[[1.0, 2.0, 3.0],
             [4.0, 5.0, 6.0],
             [7.0, 8.0, 9.0]]]] */
/* filter: [[[[0.2, 0.5],
              [0.3, 0.7]]]] */
/* stride: [1, 1] */
/* padding: [1, 1] */
/* dilation: [1, 1] */
/* group_count: 1 */
Tensor output;
op_conv2d_forward(input, filter, stride, 2, padding, 2, dilation, 2, group_count,
&output);
/* output: [[[[ 2.2,  3.5,  3.0],
              [ 5.6,  8.3,  6.4],
              [ 6.8,  9.1,  6.1]]]] */
```

A.2.6.3 三维卷积操作

前向接口 C 语言：

```
Status op_conv3d_forward (const Tensor input,
                    const Tensor filter,
                    const int *stride,
                    const int stride_len,
```

```
                        const int *padding,
                        const int padding_len,
                        const int *dilation,
                        const int dilation_len,
                        const int group_count,
                        Tensor *output);
```

前向接口参数：

input (IN)：表示 5-D 输入张量。
filter (IN)：表示卷积核，是一个 5-D 张量。
stride (IN)：表示卷积步长。
stride_len (IN)：表示数组 stride 的长度，可以设置成 1 或者 3。
padding (IN)：表示填充元素个数。
padding_len (IN)：表示 padding 数组的长度，可以设置成 1 或者 3。
dilation(IN)：表示卷积膨胀个数。
dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1 或者 3。
group_count (IN)：表示进行分组卷积的组数。
output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_INVALID_ARGRUMENT: 表示参数出错。
STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。
STATUS_TYPE_MISMATCH：类型不匹配。

后向接口 C 语言：

```
Status op_conv3d_backward_data(const Tensor grad_out,
                        const Tensor filter,
                        const int *stride,
                        const int stride_len,
                        const int *padding,
                        const int padding_len,
                        const int *dilation,
                        const int dilation_len,
                        const int group_count,
                        Tensor * grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。
filter (IN)：表示卷积核，5-D 张量。
stride (IN)：表示卷积步长。
stride_len (IN)：数组 stride 的长度，可以设置成 1 或者 2。
padding (IN)：表示填充元素个数。
padding_len (IN)：表示 padding 数组的长度，可以设置成 1 或者 2。
dilation(IN)：表示卷积膨胀个数。
dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1 或者 2。
group_count (IN)：表示进行分组卷积的组数。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

STATUS_TYPE_MISMATCH：类型不匹配

后向接口 C 语言：

```
Status op_conv3d_backward_filter(const Tensor grad_out,
                    const Tensor input,
                    const int *stride,
                    const int stride_len,
                    const int *padding,
                    const int padding_len,
                    const int *dilation,
                    const int dilation_len,
                    const int group_count,
                    Tensor * grad_filter);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

input (IN)：表示输入张量，5-D 张量。

stride (IN)：表示卷积步长。

stride_len (IN)：数组 stride 的长度，可以设置成 1 或者 2。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以设置成 1 或者 2。

dilation(IN)：表示卷积膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1 或者 2。

group_count (IN)：表示进行分组卷积的组数。

grad_filter (OUT)：表示卷积核的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

STATUS_TYPE_MISMATCH：类型不匹配。

示例：

```
/* input: [[[[[1.0, 2.0, 3.0],
              [4.0, 5.0, 6.0],
              [7.0, 8.0, 9.0]],
             [[10.0, 11.0, 12.0],
              [13.0, 14.0, 15.0],
              [16.0, 17.0, 18.0]],
             [[19.0, 20.0, 21.0],
              [22.0, 23.0, 24.0],
              [25.0, 26.0, 27.0]]]]] */
```

```
    /* filter: [[[[[0.1, 0.2],
                   [0.3, 0.4]],
                  [[0.5, 0.6],
                   [0.7, 0.8]]]]] */
    /* stride: [1, 1, 1] */
    /* padding: [0, 0, 0] */
    /* dilation: [1, 1, 1] */
    /* group_count: 1 */
    Tensor output;
    op_conv3d_forward(input, filter, stride, 3, padding, 3, dilation, 3, group_count,
    &output);
    /* output: [[[[[50.0, 60.0],
                   [80.0, 90.0]],
                  [[140.0, 150.0],
                   [170.0, 180.0]]]]] */
```

A.2.6.4 一维反卷积操作

前向接口 C 语言：

```
    Status op_conv1d_transpose_forward (const Tensor input,
                        const Tensor filter,
                        const int *stride,
                        const int stride_len,
                        const int *padding,
                        const int padding_len,
                        const int *dilation,
                        const int dilation_len,
                        const int group_count,
                        Tensor *output);
```

前向接口参数：

input (IN)：表示 3-D 输入张量。

filter (IN)：表示卷积核，3-D 张量。

stride (IN)：表示卷积步长。

stride_len (IN)：数组 stride 的长度，可以设置成 1。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以设置成 1。

dilation(IN)：表示卷积膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1。

group_count (IN)：表示进行分组卷积的组数。

output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

STATUS_TYPE_MISMATCH：类型不匹配。

后向接口 C 语言：

```
Status op_conv2d_transpose_backward_data (const Tensor grad_out,
                    const Tensor filter,
                    const int *stride,
                    const int stride_len,
                    const int *padding,
                    const int padding_len,
                    const int *dilation,
                    const int dilation_len,
                    const int group_count,
                    Tensor * grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

filter (IN)：表示卷积核，3-D 张量。

stride (IN)：表示卷积步长。

stride_len (IN)：数组 stride 的长度，可以设置成 1。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以设置成 1。

dilation(IN)：表示卷积膨胀个数。dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1。

group_count (IN)：表示进行分组卷积的组数。

grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT: 表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

STATUS_TYPE_MISMATCH：类型不匹配。

示例：

```
/* input: [[[1.0, 2.0, 3.0]]] */
/* filter: [[[0.5, 1.0]]] */
/* stride: [2] */
/* padding: [0] */
/* dilation: [1] */
/* group_count: 1 */
Tensor output;
op_conv1d_transpose_forward(input, filter, stride, 1, padding, 1, dilation, 1,
group_count, &output);
/* output: [[[0.5, 1.0, 2.5, 2.0, 1.5]]] */
```

A.2.6.5 二维反卷积操作

前向接口 C 语言：

```
Status op_conv2d_transpose_forward (const Tensor input,
```

```
                        const Tensor filter,
                        const int *stride,
                        const int stride_len,
                        const int *padding,
                        const int padding_len,
                        const int *dilation,
                        const int dilation_len,
                        const int group_count,
                        Tensor *output);
```

前向接口参数：

input (IN)：表示 4-D 输入张量。

filter (IN)：表示卷积核，4-D 张量。

stride (IN)：表示卷积步长。

stride_len (IN)：数组 stride 的长度，可以设置成 1 或者 2。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以设置成 1 或者 2。

dilation(IN)：表示卷积膨胀个数。

dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1 或者 2。

group_count (IN)：表示进行分组卷积的组数。

output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

STATUS_TYPE_MISMATCH：类型不匹配。

后向接口 C 语言：

```
Status op_conv2d_transpose_backward_data (const Tensor grad_out,
                        const Tensor filter,
                        const int *stride,
                        const int stride_len,
                        const int *padding,
                        const int padding_len,
                        const int *dilation,
                        const int dilation_len,
                        const int group_count,
                        Tensor * grad_in);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。

filter (IN)：表示卷积核，4-D 张量。

stride (IN)：表示卷积步长。

stride_len (IN)：数组 stride 的长度，可以设置成 1 或者 2。

padding (IN)：表示填充元素个数。

padding_len (IN)：表示 padding 数组的长度，可以设置成 1 或者 2。

dilation(IN)：表示卷积膨胀个数。
dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1 或者 2。
group_count (IN)：表示进行分组卷积的组数。
grad_in (OUT)：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_INVALID_ARGRUMENT: 表示参数出错。
STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。
STATUS_TYPE_MISMATCH：类型不匹配。

后向接口 C 语言：

```
Status op_conv2d_transpose_backward_filter(const Tensor grad_out,
                    const Tensor input,
                    const int *stride,
                    const int stride_len,
                    const int *padding,
                    const int padding_len,
                    const int *dilation,
                    const int dilation_len,
                    const int group_count,
                    Tensor * grad_filter);
```

后向接口参数：

grad_out (IN)：表示输出张量的梯度。
input (IN)：表示输入张量，4-D 张量。
stride (IN)：表示卷积步长。
stride_len (IN)：数组 stride 的长度，可以设置成 1 或者 2。
padding (IN)：表示填充元素个数。
padding_len (IN)：表示 padding 数组的长度，可以设置成 1 或者 2。
dilation(IN)：表示卷积膨胀个数。
dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1 或者 2。
group_count (IN)：表示进行分组卷积的组数。
grad_filter (OUT)：表示卷积核的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_INVALID_ARGRUMENT: 表示参数出错。
STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。
STATUS_TYPE_MISMATCH：类型不匹配。

示例：

```
/* input: [[[[1.0, 2.0],
             [3.0, 4.0]]]] */
/* filter: [[[[0.1, 0.2],
              [0.3, 0.4]]]] */
/* stride: [2, 2] */
/* padding: [0, 0] */
```

```
    /* dilation: [1, 1] */
    /* group_count: 1 */
    Tensor output;
    op_conv2d_transpose_forward(input, filter, stride, 2, padding, 2, dilation, 2,
    group_count, &output);
    /* output: [[[[0.1, 0.3, 0.2, 0.4],
                  [0.4, 1.2, 0.6, 1.4],
                  [0.3, 0.9, 0.4, 1.0],
                  [0.6, 1.8, 0.8, 1.6]]]] */
```

A.2.6.6 三维反卷积操作

前向接口 C 语言：

```
    Status op_conv3d_transpose_forward (const Tensor input,
                        const Tensor filter,
                        const int *stride,
                        const int stride_len,
                        const int *padding,
                        const int padding_len,
                        const int *dilation,
                        const int dilation_len,
                        const int group_count,
                        Tensor *output);
```

前向接口参数：

input (IN)：表示 5-D 输入张量。
filter (IN)：表示卷积核，是一个 5-D 张量。
stride (IN)：表示卷积步长。
stride_len (IN)：表示数组 stride 的长度，可以设置成 1 或者 3。
padding (IN)：表示填充元素个数。
padding_len (IN)：表示 padding 数组的长度，可以设置成 1 或者 3。
dilation(IN)：表示卷积膨胀个数。
dilation_len (IN)：表示 dilation 数组的长度，可以设置成 1 或者 3。
group_count (IN)：表示进行分组卷积的组数。
output (OUT)：表示输出张量。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_INVALID_ARGRUMENT：表示参数出错。
STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

后向接口 C 语言：

```
    Status op_conv3d_transpose_backward_data (const Tensor grad_out,
                        const Tensor filter,
                        const int *stride,
                        const int stride_len,
                        const int *padding,
```

```
                    const int padding_len,
                    const int *dilation,
                    const int dilation_len,
                    const int group_count,
                    Tensor * grad_in);
```

后向接口参数：

grad_out（IN）：表示输出张量的梯度。

filter（IN）：表示卷积核，5-D 张量。

stride（IN）：表示卷积步长。

stride_len（IN）：数组 stride 的长度，可以设置成 1 或者 2。

padding（IN）：表示填充元素个数。

padding_len（IN）：表示 padding 数组的长度，可以设置成 1 或者 2。

dilation(IN)：表示卷积膨胀个数。

dilation_len（IN）：表示 dilation 数组的长度，可以设置成 1 或者 2。

group_count（IN）：表示进行分组卷积的组数。

grad_in（OUT）：表示输入张量的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

STATUS_TYPE_MISMATCH：类型不匹配。

后向接口 C 语言：

```
Status op_conv3d_transpose_backward_filter(const Tensor grad_out,
                    const Tensor input,
                    const int *stride,
                    const int stride_len,
                    const int *padding,
                    const int padding_len,
                    const int *dilation,
                    const int dilation_len,
                    const int group_count,
                    Tensor * grad_filter);
```

后向接口参数：

grad_out（IN）：表示输出张量的梯度。

input（IN）：表示输入张量，5-D 张量。

stride（IN）：表示卷积步长。

stride_len（IN）：数组 stride 的长度，可以设置成 1 或者 2。

padding（IN）：表示填充元素个数。

padding_len（IN）：表示 padding 数组的长度，可以设置成 1 或者 2。

dilation(IN)：表示卷积膨胀个数。

dilation_len（IN）：表示 dilation 数组的长度，可以设置成 1 或者 2。

group_count（IN）：表示进行分组卷积的组数。

grad_filter（OUT）：表示卷积核的梯度。

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_INVALID_ARGRUMENT：表示参数出错。

STATUS_UNINITIALIZED_OBJECT：表示输入张量对象不合法。

STATUS_TYPE_MISMATCH：类型不匹配。

示例：

```
/* input: [[[[[1.0, 2.0],
              [3.0, 4.0]],
             [[5.0, 6.0],
              [7.0, 8.0]]]]] */
/* filter: [[[[[0.1, 0.2],
               [0.3, 0.4]],
              [[0.5, 0.6],
               [0.7, 0.8]]]]] */
/* stride: [2, 2, 2] */
/* padding: [0, 0, 0] */
/* dilation: [1, 1, 1] */
/* group_count: 1 */
Tensor output;
op_conv3d_transpose_forward(input, filter, stride, 3, padding, 3, dilation, 3,
group_count, &output);
/* output: [[[[[0.1, 0.5, 0.6, 0.2],
              [0.5, 2.1, 2.4, 0.8],
              [0.7, 3.1, 3.4, 1.2],
              [0.2, 0.8, 1.2, 0.4]],
             [[0.3, 0.9, 1.2, 0.4],
              [1.1, 3.7, 4.2, 1.6],
              [1.4, 4.8, 5.2, 2.0],
              [0.5, 1.8, 2.4, 0.8]]]]] */
```

A.2.7 评估函数

A.2.7.1 准确率函数

C 语言：

```
Status op_accuracy(const Tensor input,
                   const Tensor label,
                   const int k,
                   Tensor *correct,
                   Tensor *total,
                   Tensor *output);
```

参数：

input (IN)：表示输入张量

label (IN)：表示期望的标签值张量。

k (IN)：取每个类别中前 k 个预测值用于计算。

correct (OUT): 表示正确预测值的个数的张量。
total (OUT): 表示总共预测值的个数的张量。
output (OUT)：表示正确率的张量。

返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* input: [[1, 2, 3], [2, 1, 3], [3, 1, 2]] */
/* label: [0, 1, 2] */
/* k: 1 */
Tensor correct;
Tensor total;
Tensor output;
op_accuracy(input, label, k, &correct, &total, &output);
/* correct: [2] */
/* total: [3] */
/* output: [0.6667] */
```

A.2.7.2 AUC函数

C 语言：

```
Status op_auc(const Tensor input,
                  const Tensor label,
                  const char *curve,
                  const int num_thresholds,
                  const int slide_steps,
                  Tensor *output);
```

参数：

input (IN)：表示输入张量。
label (IN)：表示期望的标签值。
curve (IN): 表示曲线类型。
num_thresholds (IN): 表示将 ROC 曲线离散化时使用的临界数值。
slide_steps (IN):表示滑步值。
output (OUT)：表示 AUC 结果的张量。

返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* input: [0.9, 0.7, 0.6, 0.4, 0.2] */
/* label: [1, 1, 0, 0, 0] */
/* curve: "ROC" */
/* num_thresholds: 10 */
/* slide_steps: 1 */
Tensor output;
```

```
op_auc(input, label, curve, num_thresholds, slide_steps, &output);
/* output: [0.75] */
```

A.2.8 循环网络函数

A.2.8.1 简单循环网络基本单元

前向接口 C 语言：

```
Status op_rnn_cell_forward(const Tensor input,
                           const Tensor hidden,
                           const Tensor weight_ih,
                           const Tensor bias_ih,
                           const Tensor weight_hh,
                           const Tensor bias_hh,
                           const char *activation,
                           const _Bool is_train,
                              Tensor *next_hidden);
```

前向接口参数：

input (IN)：表示输入张量。
hidden (IN)：表示隐藏状态。
weight_ih (IN)：表示输入权重。
bias_ih (IN)：表示输入偏置系数。
weight_hh (IN)：表示隐藏状态权重。
bias_hh (IN)：隐藏状态偏置系数。
activation (IN)：表示激活函数。
Is_train (IN)：表示是否训练
next_hidden (INOUT)：下个隐藏状态。

前向接口返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_UNINITIALIZED_OBJECT：对象未初始化。
STATUS_INVALID_ARGUMENT：非法参数。
STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。
STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

后向接口 C 语言：

```
Status op_rnn_cell_backward(const Tensor grad_in,
                            const Tensor hidden,
                            const Tensor weight_ih,
                            const Tensor bias_ih,
                            const Tensor weight_hh,
                            const Tensor bias_hh,
                            const char *activation,
                            Tensor* grad_hidden,
                            Tensor* grad_weight_ih,
                            Tensor* grad_bias_ih,
                            Tensor* grad_weight_hh,
```

```
                    Tensor* grad_bias_hh,
                    Tensor *grad_out);
```

后向接口参数：

grad_in (IN)：表示输入梯度。

hidden (IN)：表示隐藏状态。

weight_ih (IN)：表示输入权重。

bias_ih (IN)：表示输入偏置系数。

weight_hh (IN)：表示隐藏状态权重。

bias_hh (IN)：隐藏状态偏置系数。

activation (IN)：表示激活函数。

grad_hidden (OUT)：表示隐藏状态梯度

grad_weight_ih (OUT)：表示输入权重梯度

grad_bias_ih (OUT)：表示输入偏置梯度

grad_weight_hh (OUT)：表示隐藏状态权重梯度

grad_bias_hh (OUT)：表示隐藏状态偏置梯度

grad_out (INOUT)：表示输入对应梯度

后向接口返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

示例：

```
/* input: [0.5, 0.3] */
/* hidden: [0.1, 0.2] */
/* weight_ih: [[0.2, 0.4], [0.5, 0.6]] */
/* bias_ih: [0.1, 0.1] */
/* weight_hh: [[0.7, 0.8], [0.9, 1.0]] */
/* bias_hh: [0.2, 0.2] */
/* activation: "tanh" */
/* is_train: true */
Tensor next_hidden;
op_rnn_cell_forward(input, hidden, weight_ih, bias_ih, weight_hh, bias_hh, activation,
is_train, &next_hidden);
/* next_hidden: [0.7275, 0.6899] */
```

A.2.8.2 简单循环网络

前向 C 语言：

```
Status op_rnn_forward(const Tensor input,
                      const Tensor *hidden0,
                      const Tensor *weight_ih,
                      const Tensor *bias_ih,
                      const Tensor *weight_hh,
                      const Tensor *bias_hh,
                      const int num_layers,
```

```
                    const _Bool bidirectional,
                    const float dropout_rate,
                    const char *activation,
                    const _Bool is_train,
                    Tensor* mask,
                    Tensor *output,
                    Tensor **out_hiddens);
```

前向参数：

input (IN)：表示输入张量。
hidden0 (IN)：表示隐藏状态张量数组。
weight_ih (IN)：表示输入权重张量数组。
bias_ih (IN)：表示输入偏置系数张量数组。
weight_hh (IN)：表示隐藏状态权重张量数组。
bias_hh (IN)：表示隐藏状态偏置系数张量数组。
num_layers (IN)：表示循环网络的层数。
bidirectional (IN)：表示是否双向。
dropout_rate (IN)：表示随机失活率。
activation (IN)：表示激活函数。
is_train (IN)：表示是否是训练
mask (OUT)：表示随机失活的掩码
output (INOUT)：表示最后一层的隐藏状态。
out_hiddens(INOUT):表示下个隐藏状态张量数组。

前向返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。
STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

后向 C 语言：

```
Status op_rnn_backward(const Tensor grad_in,
                    const Tensor *hidden0,
                    const Tensor *weight_ih,
                    const Tensor *bias_ih,
                    const Tensor *weight_hh,
                    const Tensor *bias_hh,
                    const Tensor mask,
                    const int num_layers,
                    const _Bool bidirectional,
                    const char *activation,
                    Tensor *output_hidden,
                    Tensor** grad_hidden0,
                    Tensor** grad_weight_ih,
                    Tensor** grad_bias_ih,
                    Tensor** grad_weight_hh,
                    Tensor** grad_bias_hh,
```

```
                    Tensor *grad_out);
```

后向参数：

grad_in (IN)：表示输出对应梯度。

hidden0 (IN)：表示隐藏状态张量数组。

weight_ih (IN)：表示输入权重张量数组。

bias_ih (IN)：表示输入偏置系数张量数组。

weight_hh (IN)：表示隐藏状态权重张量数组。

bias_hh (IN)：表示隐藏状态偏置系数张量数组。

num_layers (IN)：表示循环网络的层数。

mask (IN)：随机失活的掩码

bidirectional (IN)：表示是否双向。

activation (IN)：表示激活函数。

output_hidden (IN)：表示下个隐藏状态张量数组。

grad_hidden0 (OUT)：表示隐藏状态梯度

grad_weight_ih (OUT)：表示权重梯度

grad_bias_ih (OUT)：表示偏置系数梯度

grad_weight_hh (OUT)：表示隐藏状态权重梯度

grad_bias_hh (OUT)：表示输入偏置系数梯度

grad_out (OUT)：表示输入对应梯度

后向返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

示例：

```
/* input: [[[0.5, 0.1], [0.4, 0.2]], [[0.6, 0.2], [0.5, 0.3]], [[0.7, 0.3], [0.6, 0.4]]] */
/* hidden0: [[0.1, 0.2], [0.3, 0.4]] */
/* weight_ih: [[[0.2, 0.4], [0.5, 0.6]], [[0.3, 0.7], [0.8, 0.9]]] */
/* bias_ih: [[0.1, 0.1], [0.2, 0.2]] */
/* weight_hh: [[[0.6, 0.8], [0.7, 0.9]], [[0.5, 0.6], [0.4, 0.5]]] */
/* bias_hh: [[0.1, 0.1], [0.2, 0.2]] */
/* num_layers: 1 */
/* bidirectional: false */
/* dropout_rate: 0.0 */
/* activation: "tanh" */
/* is_train: true */
/* mask: null */
Tensor output;
Tensor *out_hiddens;
op_rnn_forward(input, hidden0, weight_ih, bias_ih, weight_hh, bias_hh, num_layers, bidirectional, dropout_rate, activation, is_train, mask, &output, &out_hiddens);
/* output: [[[0.44, 0.34], [0.45, 0.35]], [[0.48, 0.37], [0.49, 0.38]], [[0.51, 0.39], [0.52, 0.40]]] */
```

```
/* out_hiddens: [[0.51, 0.39], [0.52, 0.40]] */
```

A.2.8.3 长短期记忆单元

前向 C 语言：

```
Status op_lstm_cell_forward(const Tensor input,
                            const Tensor hidden,
                            const Tensor weight_ih,
                            const Tensor bias_ih,
                            const Tensor weight_hh,
                            const Tensor bias_hh,
                            const char *activation,
                            const char *recurrent_activation,
                            Tensor **grad_hidden0,
                            Tensor **grad_weight_ih,
                            Tensor** grad_bias_ih,
                            Tensor** grad_weight_hh,
                            Tenosr** grad_bias_hh,
                            const _Bool is_train,
                            Tensor *next_hidden);
```

前向参数：

input (IN)：表示输入张量。
hidden (IN)：表示当前状态。
weight_ih (IN)：表示输入权重。
bias_ih (IN)：表示输入偏置系数。
weight_hh (IN)：表示状态权重。
bias_hh (IN)：表示状态偏置系数。
activation (IN)：表示激活函数。
recurrent activation (IN)：表示循环激活函数。
is_train (IN)：表示是否训练
next_hidden (INOUT)：表示下组状态。

前向返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。
STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

后向 C 语言：

```
Status op_lstm_cell_backward(const Tensor grad_in,
                            const Tensor hidden,
                            const Tensor weight_ih,
                            const Tensor bias_ih,
                            const Tensor weight_hh,
                            const Tensor bias_hh,
                            const char *activation,
                            const char *recurrent_activation,
```

```
                                Tensor *grad_out);
```

后向参数：

grad_input (IN)：表示输出对应梯度。
hidden (IN)：表示当前状态。
weight_ih (IN)：表示输入权重。
bias_ih (IN)：表示输入偏置系数。
weight_hh (IN)：表示状态权重。
bias_hh (IN)：表示状态偏置系数。
activation (IN)：表示激活函数。
recurrent activation (IN)：表示循环激活函数。
grad_hidden (INOUT)：当前状态梯度
grad_weight_ih (INOUT)：权重梯度
grad_bias_ih (INOUT)：偏置梯度
grad_weight_hh (INOUT)：状态权重梯度
grad_bias_hh (INOUT)：状态偏置梯度
grad_out(INOUT)：表示输入对应梯度。

后向返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。
STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

示例：

```
/* input: [0.5, 0.7] */
/* hidden: [[0.1, 0.2], [0.3, 0.4]] */
/* weight_ih: [[0.2, 0.3, 0.4, 0.5], [0.6, 0.7, 0.8, 0.9]] */
/* bias_ih: [0.1, 0.1, 0.1, 0.1] */
/* weight_hh: [[0.7, 0.8, 0.9, 1.0], [1.1, 1.2, 1.3, 1.4]] */
/* bias_hh: [0.2, 0.2, 0.2, 0.2] */
/* activation: "tanh" */
/* recurrent_activation: "sigmoid" */
/* grad_hidden0: NULL */
/* grad_weight_ih: NULL */
/* grad_bias_ih: NULL */
/* grad_weight_hh: NULL */
/* grad_bias_hh: NULL */
/* is_train: true */
Tensor next_hidden;
op_lstm_cell_forward(input, hidden, weight_ih, bias_ih, weight_hh, bias_hh, activation,
recurrent_activation, NULL, NULL, NULL, NULL, NULL, is_train, &next_hidden);
/* next_hidden: [[0.6245, 0.7358], [0.7823, 0.8725]] */
```

A.2.8.4 长短期记忆网络

前向 C 语言：

```
Status op_rnn_forward(const Tensor input,
```

```
                    const Tensor *hidden0,
                    const Tensor *weight_ih,
                    const Tensor *bias_ih,
                    const Tensor *weight_hh,
                    const Tensor *bias_hh,
                    const int num_layers,
                    const _Bool bidirectional,
                    const float dropout,
                    dropout(IN)：表示随机失活率。
                    const char *activation,
                    const char *recurrent_activation,
                    Tensor *mask,
                    const _Bool is_train,
                    Tensor *output,
                    Tensor **out_hiddens);
```

前向参数：

input (IN)：表示某个时间步的输入张量。
hidden0 (IN)：表示初始的状态 Tensor 数组。
weight_ih (IN)：表示从 input 到 next_hidden 的权重 Tensor 数组。
bias_ih (IN)：表示从 input 到 next_hidden 的偏置值 Tensor 数组。
weight_hh (IN)：表示从 hidden 到 next_hidden 的权重 Tensor 数组。
bias_hh (IN)：表示从 input 到 next_hidden 的偏置值 Tensor 数组。
num_layers (IN)：表示循环的层数。
bidirectional (IN)：表示是否双向。
activation (IN)：表示激活函数。
recurrent_activation (IN)：表示用于循环的激活函数。
mask (IN)：表示随机失活的掩码
is_train (IN)：表示是否是训练
output (INOUT)：表示网络最后一层所有时间步的隐藏状态。
out_hiddens(INOUT):表示网络每层最后时间步的隐藏状态 Tensor 指针数组。

前向返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。
STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

后向 C 语言：

```
    Status op_rnn_backward(const Tensor grad_in,
                    const Tensor *hidden0,
                    const Tensor *weight_ih,
                    const Tensor *bias_ih,
                    const Tensor *weight_hh,
                    const Tensor *bias_hh,
                    const Tensor mask,
                    const int num_layers,
```

```
                const _Bool bidirectional,
                const char *activation,
                const char *recurrent_activation,
                Tensor *output_hiddens,
                Tensor **grad_hidden0,
                Tensor **grad_weight_ih,
                Tensor** grad_bias_ih,
                Tensor** grad_weight_hh,
                Tenosr** grad_bias_hh,
                Tensor *grad_out);
```

后向参数：

grad_in (IN)：表示输出对应梯度。
hidden0 (IN)：表示初始的状态 Tensor 数组。
weight_ih (IN)：表示从 input 到 next_hidden 的权重 Tensor 数组。
bias_ih (IN)：表示从 input 到 next_hidden 的偏置值 Tensor 数组。
weight_hh (IN)：表示从 hidden 到 next_hidden 的权重 Tensor 数组。
bias_hh (IN)：表示从 input 到 next_hidden 的偏置值 Tensor 数组。
mask (IN)：表示随机失活掩码。
num_layers (IN)：表示循环的层数。
bidirectional (IN)：表示是否双向。
activation (IN)：表示激活函数。
recurrent_activation (IN)：表示用于循环的激活函数。
out_hiddens(IN):表示网络每层最后时间步的隐藏状态 Tensor 指针数组。
grad_hidden0 (OUT)：初始状态梯度
grad_weight_ih(OUT)：权重梯度
grad_bias_ih (OUT)：偏置梯度
grad_weight_hh (OUT)：权重梯度
grad_bias_hh (OUT)：偏置梯度
grad_out (OUT)：表示输入对应梯度

后向返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。
STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

示例：

```
/* input: [[[0.6, 0.2], [0.3, 0.4]], [[0.7, 0.1], [0.2, 0.5]], [[0.4, 0.6], [0.1, 0.3]]] */
/* hidden0: [[0.1, 0.2], [0.3, 0.4]] */
/* weight_ih: [[[0.1, 0.2], [0.3, 0.4]], [[0.5, 0.6], [0.7, 0.8]]] */
/* bias_ih: [[0.1, 0.1], [0.2, 0.2]] */
/* weight_hh: [[[0.9, 1.0], [1.1, 1.2]], [[1.3, 1.4], [1.5, 1.6]]] */
/* bias_hh: [[0.3, 0.3], [0.4, 0.4]] */
/* num_layers: 2 */
/* bidirectional: true */
```

```
/* dropout: 0.5 */
/* activation: "tanh" */
/* recurrent_activation: "sigmoid" */
/* mask: NULL */
/* is_train: true */
Tensor output;
Tensor *out_hiddens[2];
op_rnn_forward(input, hidden0, weight_ih, bias_ih, weight_hh, bias_hh, num_layers,
bidirectional, dropout, activation, recurrent_activation, NULL, is_train, &output,
out_hiddens);
/* output: [[[0.625, 0.688], [0.577, 0.633]], [[0.635, 0.695], [0.584, 0.641]], [[0.644,
0.700], [0.590, 0.648]]] */
/* out_hiddens[0]: [[0.641, 0.692], [0.586, 0.645]] */
/* out_hiddens[1]: [[0.654, 0.704], [0.595, 0.652]] */
```

A.2.8.5 门控循环单元网络基本单元

前向 C 语言：

```
Status op_gru_cell_forward(const Tensor input,
                           const Tensor hidden,
                           const Tensor weight_ih,
                           const Tensor bias_ih,
                           const Tensor weight_hh,
                           const Tensor bias_hh,
                           const char *activation,
                           const char *recurrent_activation,
                           const _Bool is_train,
                           Tensor *next_hidden);
```

前向参数：

input (IN)：表示某个时间步的输入张量。

hidden (IN)：表示上一个隐藏状态h_{t-1}。

weight_ih (IN)：表示各个门中从 input 到 next_hidden 的权重 Tensor 数组。

bias_ih (IN)：表示各个门中从 input 到 next_hidden 的偏置值 Tensor 数组。

weight_hh (IN)：表示各个门中从 hidden 到 next_hidden 的权重 Tensor 数组。

bias_hh (IN)：表示各个门中从 hidden 到 next_hidden 的偏置值 Tensor 数组。

activation (IN)：表示激活函数。

recurrent_activation (IN)：表示用于循环的激活函数。

is_train (IN)：表示是否训练

output (INOUT)：表示输出张量，即隐藏状态h_t。

前向返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

后向 C 语言：

```
Status op_gru_cell_backward(const Tensor grad_in,
                            const Tensor hidden,
                            const Tensor weight_ih,
                            const Tensor bias_ih,
                            const Tensor weight_hh,
                            const Tensor bias_hh,
                            const char *activation,
                            const char *recurrent_activation,
                            Tensor* grad_hidden,
                            Tensor* grad_weight_ih,
                            Tensor* grad_bias_ih,
                            Tensor* grad_weight_hh,
                            Tensor* grad_bias_hh,
                            Tensor *grad_out);
```

后向参数：

grad_in(IN)：表示输出对应梯度。

hidden (IN)：表示上一个隐藏状态h_{t-1}。

weight_ih (IN)：表示各个门中从 input 到 next_hidden 的权重 Tensor 数组。

bias_ih (IN)：表示各个门中从 input 到 next_hidden 的偏置值 Tensor 数组。

weight_hh (IN)：表示各个门中从 hidden 到 next_hidden 的权重 Tensor 数组。

bias_hh (IN)：表示各个门中从 hidden 到 next_hidden 的偏置值 Tensor 数组。

activation (IN)：表示激活函数。

recurrent_activation (IN)：表示用于循环的激活函数。

grad_hidden (INOUT)：隐藏状态梯度

grad_weight_ih (INOUT)：权重梯度

grad_bias_ih (INOUT)：偏置梯度

grad_weight_hh (INOUT)：权重梯度

grad_bias_hh (INOUT)：偏置梯度

grad_out (INOUT)：表示输入对应梯度。

后向返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

示例：

```
/* input: [0.5, 0.3] */
/* hidden: [0.2, 0.4] */
/* weight_ih: [[0.1, 0.2, 0.3, 0.4], [0.5, 0.6, 0.7, 0.8]] */
/* bias_ih: [0.1, 0.1, 0.1, 0.1] */
/* weight_hh: [[0.9, 1.0, 1.1, 1.2], [1.3, 1.4, 1.5, 1.6]] */
/* bias_hh: [0.2, 0.2, 0.2, 0.2] */
/* activation: "tanh" */
/* recurrent_activation: "sigmoid" */
/* is_train: true */
```

```
Tensor next_hidden;
op_gru_cell_forward(input, hidden, weight_ih, bias_ih, weight_hh, bias_hh, activation,
recurrent_activation, is_train, &next_hidden);
/* next_hidden: [0.623, 0.743] */
```

A.2.8.6 门控循环单元网络

前向 C 语言：

```
Status op_gru_forward(const Tensor input,
                      const Tensor *hidden0,
                      const Tensor *weight_ih,
                      const Tensor *bias_ih,
                      const Tensor *weight_hh,
                      const Tensor *bias_hh,
                      const int num_layers,
                      const _Bool bidirectional,
                      const float dropout,
                      const char *activation,
                      const char *recurrent_activation,
                      const _Bool is_train,
                      Tensor* mask,
                      Tensor *output,
                      Tensor **out_hiddens);
```

前向参数：

input (IN)：表示某个时间步的输入张量。
hidden0 (IN)：表示初始的隐藏状态数组。
weight_ih (IN)：表示从 input 到 next_hidden 的权重 Tensor 数组。
bias_ih (IN)：表示从 input 到 next_hidden 的偏置值 Tensor 数组
weight_hh (IN)：表示从 hidden 到 next_hidden 的权重 Tensor 数组
weight_hh (IN)：表示从 hidden 到 next_hidden 的偏置值 Tensor 数组。
num_layers (IN)：表示循环的层数。
bidirectional (IN)：表示是否双向。
dropout (IN)：表示随机失活率。
activation (IN)：表示激活函数。
recurrent_activation (IN)：表示用于循环的激活函数。
is_train (IN)：表示是否是训练
mask (OUT)：随机失活掩码
output (INOUT)：表示网络最后一层所有时间步的隐藏状态。
out_hiddens(INOUT):表示网络每层的最后时间步的隐藏状态 Tensor 指针数组。

前向返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。
STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

后向 C 语言：

```
Status op_gru_backward (const Tensor grad_in,
                        const Tensor *hidden0,
                        const Tensor *weight_ih,
                        const Tensor *bias_ih,
                        const Tensor *weight_hh,
                        const Tensor *bias_hh,
                        const Tensor mask,
                        const int num_layers,
                        const _Bool bidirectional,
                        const char *activation,
                        const char *recurrent_activation,
                        Tensor *output_hiddens,
                        Tensor **grad_hidden0,
                        Tensor** grad_weight_ih,
                        Tensor** grad_bias_ih,
                        Tensor** grad_weight_hh,
                        Tensor** grad_bias_hh,
                        Tensor *grad_out);
```

后向参数：

grad_in (IN)：表示输出对应梯度。

hidden0 (IN)：表示初始的隐藏状态数组。

weight_ih (IN)：表示从 input 到 next_hidden 的权重 Tensor 数组。

bias_ih (IN)：表示从 input 到 next_hidden 的偏置值 Tensor 数组

weight_hh (IN)：表示从 hidden 到 next_hidden 的权重 Tensor 数组

bias_hh (IN)：表示从 hidden 到 next_hidden 的偏置值 Tensor 数组。

mask (IN)：表示随机失活掩码。

num_layers (IN)：表示循环的层数。

bidirectional (IN)：表示是否双向。

activation (IN)：表示激活函数。

recurrent_activation (IN)：表示用于循环的激活函数。

output_hiddens (IN)：表示网络每层的最后时间步的隐藏状态 Tensor 数组。

grad_hidden0 (INOUT)：表示初始隐藏状态梯度。

grad_weight_ih (INOUT)：表示权重梯度。

grad_bias_ih (INOUT)：表示偏置梯度。

grad_weight_hh (INOUT)：表示权重梯度

grad_bias_hh (INOUT)：表示偏置梯度

grad_out (INOUT):表示输入对应梯度。

后向返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

示例：

```
/* input: [[[0.2,  0.5], [0.4,  0.6]], [[0.7,  0.8], [0.5,  0.9]], [[0.1,  0.4], [0.6,
```

```
    0.7]]] */
    /* hidden0: [[0.1, 0.2], [0.3, 0.4]] */
    /* weight_ih: [[[0.1, 0.2, 0.3, 0.4], [0.5, 0.6, 0.7, 0.8]], [[0.9, 1.0, 1.1, 1.2],
    [1.3, 1.4, 1.5, 1.6]]] */
    /* bias_ih: [[0.1, 0.1, 0.1, 0.1], [0.2, 0.2, 0.2, 0.2]] */
    /* weight_hh: [[[1.0, 1.1, 1.2, 1.3], [1.4, 1.5, 1.6, 1.7]], [[1.8, 1.9, 2.0, 2.1],
    [2.2, 2.3, 2.4, 2.5]]] */
    /* bias_hh: [[0.2, 0.2, 0.2, 0.2], [0.3, 0.3, 0.3, 0.3]] */
    /* num_layers: 2 */
    /* bidirectional: true */
    /* dropout: 0.3 */
    /* activation: "tanh" */
    /* recurrent_activation: "sigmoid" */
    /* is_train: true */
    /* mask: NULL */

    Tensor output;
    Tensor *out_hiddens[4]; // 2 layers x 2 directions

    op_gru_forward(input, hidden0, weight_ih, bias_ih, weight_hh, bias_hh, num_layers,
    bidirectional, dropout, activation, recurrent_activation, NULL, &output, out_hiddens);

    /* output: [[[0.623, 0.738], [0.641, 0.752]], [[0.645, 0.759], [0.655, 0.764]], [[0.664,
    0.775], [0.668, 0.778]]] */
    /* out_hiddens[0]: [[0.632, 0.742], [0.652, 0.753]] */
    /* out_hiddens[1]: [[0.645, 0.757], [0.663, 0.764]] */
    /* out_hiddens[2]: [[0.648, 0.766], [0.667, 0.773]] */
    /* out_hiddens[3]: [[0.660, 0.774], [0.675, 0.782]] */
```

A.2.9 编码操作

A.2.9.1 词嵌入操作

前向 C 语言：

```
    Status op_embedding_forward(const Tensor input,
                                Tensor *weight,
                                const float max_norm,
                                const float norm_type,
                                const _Bool is_train,
                                Tensor *output);
```

前向参数：

input (IN)：表示输入的索引张量。
weight (INOUT)：表示输入的权重张量
max_norm (IN)：表示范数上界
norm_type (IN)：表示范数类型

output (OUT): 表示输出张量

前向返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

后向 C 语言：

```
Status op_embedding_forward(const Tensor grad_in,
                        Const Tensor *weight,
                        Tensor* grad_weight,
                        Tensor *grad_out);
```

后向参数：

grad_in (IN)：表示输入梯度。

weight (IN): 表示输入的权重张量

grad_weight(OUT): 表示输出权重梯度

grad_out (OUT): 表示输出梯度

后向返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_DIMENSIONS_MISMATCH：表示维度不匹配。

示例：

```
/* input: [[1, 2, 4, 5],
           [4, 3, 2, 9]]   */
/* weight: rand(10, 3)  */
Tensor output;
op_embedding_forward (input, &weight, 0.1, 2, &output);
/* output:  [[[0.0394, 0.0749, 0.0533],
          [0.0049, 0.0449, 0.0892],
          [0.0772, 0.0455, 0.0444],
          [0.0709, 0.0495, 0.0502]],
         [[0.0772, 0.0455, 0.0444],
          [0.0697, 0.0289, 0.0656],
          [0.0049, 0.0449, 0.0892],
          [0.0706, 0.0387, 0.0593]]] */
```

A.2.9.2 独热编码

C 语言：

```
Status op_one_hot( const Tensor indices,
                        const int num_classes,
                        Tensor *y);
```

参数：

indices (IN)：表示输入张量。

num_classes (IN): 表示独热编码最后一个维度的大小。

y (OUT)：表示输出张量。

返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_INVALID_ARGUMENT：表示类别数参数不满足要求或者输入张量有负值。

示例：

```
/* indices: [0,1,2]  */
/* num_classes: 3  */
Tensor y;
op_one_hot(indices, num_classes, &y);
/* y:  [[1, 0, 0],
[0, 1, 0],
[0, 0, 1]] */
```

A.2.10 距离函数

A.2.10.1 余弦相似度

C 语言：

```
Status op_cosine_similarity(const Tensor x1,
                            const Tensor x2,
                            const int axis,
                            const double epsilon,
                            Tensor *y);
```

参数：

x1 (IN)：表示第一个输入张量，元素数据类型类型可以为 FLOAT32、FLOAT64，维度为$[*_1,C,*_2]$。

x2 (IN): 表示第二个输入张量，形状、数据类型与第一个输入张量相同。

axis (IN): 指定维度。

epsilon (IN)：表示防止除数为 0 的小数字。

y (OUT): 表示输出张量，数据类型与输入张量相同，维度为$[*_1,*_2]$。

返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

STATUS_INVALID_ARGRUMENT: 表示计算轴参数不满足要求。

示例：

```
/* x1: [[0.5488, 0.7152, 0.6028], [0.5449, 0.4237, 0.6459]]      */
/* x2: [[0.4376, 0.8918, 0.9637], [0.3834, 0.7917, 0.5289]]  */
Tensor y;
op_cosine_similarity (x1, x2, 0, 1e-8, &y);
/* y:  [0.9981, 0.9818, 0.9499]      */
```

A.2.11 视觉函数

A.2.11.1 网络插值采样

C 语言：

```
Status op_grid_sample(const Tensor x,
                      const Tensor grid,
                      const char *mode,
                      Tensor *y);
```

参数：

x (IN)：表示输入张量，形状为[N,C,H,W]，元素数据类型可以为 FLOAT32、FLOAT64。

grid (IN)：表示输入网格数据张量，形状为[N,H,W,2]，元素数据类型可以为 FLOAT32、FLOAT64。

mode (IN)：表示插值方法，可为双线性插值(“bilinear”)、近邻插值(“nearest”)等。

y (OUT)：表示输入张量基于输入网格的插值计算结果，维度为[N,C,H,W] 的 4-D 张量。

返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_INVALID_ARGRUMENT：表示参数出错。

示例：

```
/* x.shape: [1, 2, 1, 1] */
/* x.data: [[[[0.5218]],
[[0.4147]]]] */
/* grid.shape: [1, 1, 1, 2] */
/* grid.data:  [[[[0.2646, 0.7742]]]]  */
Tensor y;
op_grid_sample (x, grid, “bilinear”, &y);
/* y.shape: [1, 2, 1, 1]    */
/* y.data: [[[[0.5218]],
[[0.4147]]]] */
```

A.2.11.2 仿射函数

C 语言：

```
Status op_affine_grid(const Tensor theta,
                      const Shape output_shape,
                      Tensor *output);
```

参数：

theta (IN)：表示用于放射变换的变换矩阵张量，形状为[N, 2, 3]，表示 N 个 2×3 的变换矩阵。元素数据类型可以为 FLOAT32、FLOAT64。

output_shape (IN)：表示目标图像的形状，取值为[N, C, H, W]。

output (OUT)：表示形状为[N, H, W, 2]的 4-D 输出张量，表示仿射变换前后的坐标的映射关系。

返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

示例：

```
/* theta.shape: [1, 2, 3]  */
/* input.data: [[[0.4615, 0.7805, 0.1183],
```

```
                [0.6399, 0.1434, 0.9447]]] */
    /* output_shape: [1, 3, 2, 2] */
    Tensor output;
    op_affine_grid(input, output_shape, &output);
    /* output.shape: [1, 2, 2, 2] */
    /* output.data: [[[[-1.1237, 0.1614],
                    [-0.2008, 1.4412]],
                   [[0.4373, 0.4481],
                    [1.3603, 1.7279]]]] */
```

A.2.11.3 像素重排

C 语言：

```
    Status op_pixel_shuffle(const Tensor input,
                            const float upscale_factor,
                            Tensor *output);
```

参数：

input (IN)：表示输入张量，形状为$[N_1, N_2, \ldots, N_k, \mathrm{D}]$，其中最后一维 D 是类别数目。元素数据类型可以为 FLOAT32、FLOAT64。

upscale_factor (IN)：表示增大空间分辨率的增大因子。

output (OUT)：表示根据新的维度信息进行重组后的输出张量。

返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

STATUS_INVALID_ARGUMENT：表示增大因子的平方不能整除输入的通道大小。

示例：

```
    /* input.shape: [1, 4, 1, 1] */
    /* input.data: [[[[0.7782]],
                    [[0.8711]],
                    [[0.9786]],
                    [[0.7992]]]] */
    Tensor output;
    op_pixel_shuffle(input, 2.0, &output);
    /* y:  [[[[0.7782, 0.8700],
        [0.9786, 0.7992]]]] */
```

A.2.12 优化器

A.2.12.1 SGD优化器

C 接口：

```
Status op_sgd (const Tensor param,
               const Tensor grad,
               const double learning_rate,
               Tensor *param_out);
```

参数：

param (IN)：表示待更新参数。
grad (IN)：表示梯度张量。
learning_rate (IN)：表示学习率。
param_out (OUT)：更新后的参数。

返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* param: [0.5, 0.2, -0.3] */
/* grad: [0.1, -0.2, 0.3] */
/* learning_rate: 0.01 */

Tensor param_out;
op_sgd(param, grad, 0.01, &param_out);

/* param_out: [0.499, 0.202, -0.303] */
```

A.2.12.2 Momentum优化器

C 接口：

```
Status op_momentum (const Tensor param,
                    const Tensor grad,
                    const Tensor momentum,
                    const double momentum_factor,
                    const double learning_rate,
                    cont bool use_nestrov,
                    Tensor *param_out,
                    Tensor *momentum_out);
```

参数：

param (IN)：表示待更新参数。
grad (IN)：表示梯度张量。
momentum (IN)：表示动量。
momentum_factor (IN)：表示动量因子。
learning_rate (IN)：表示学习率。
use_nestrov (IN)：表示是否在参数更新过程中使用赋能牛顿动量。
param_out (OUT)：表示更新后的参数。
momentum_out (OUT)：表示更新后的动量因子。

返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* param: [0.5, -0.3, 0.2] */
/* grad: [0.1, -0.2, 0.05] */
/* momentum: [0.05, -0.05, 0.02] */
/* momentum_factor: 0.9 */
/* learning_rate: 0.01 */
/* use_nesterov: false */

Tensor param_out;
Tensor momentum_out;

op_momentum(param, grad, momentum, 0.9, 0.01, false, &param_out, &momentum_out);

/* param_out: [0.4995, -0.298, 0.1995] */
/* momentum_out: [0.095, -0.095, 0.0475] */
```

A.2.12.3 AdaGrad优化器

C 接口：

```
Status op_adagrad (const Tensor param,
                   const Tensor grad,
                   const Tensor accum,
                   const double learning_rate,
                   Tensor *param_out,
                   Tensor *accum_out);
```

参数：

param (IN)：表示待更新参数。

grad (IN)：表示梯度张量。

accum (IN)：表示梯度平方和。

learning_rate (IN)：表示学习率。

param_out (OUT)：更新后的参数。

mean_square_out (OUT)：表示更新后的梯度平方和。

返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* param: [0.5, -0.2, 0.3] */
/* grad: [0.1, -0.2, 0.05] */
/* accum: [0.01, 0.04, 0.0025] */
/* learning_rate: 0.01 */

Tensor param_out;
```

```
Tensor accum_out;

op_adagrad(param, grad, accum, 0.01, &param_out, &accum_out);

/* param_out: [0.4975, -0.195, 0.2995] */
/* accum_out: [0.0101, 0.044, 0.00255] */
```

A.2.12.4 AdaDelta优化器

C接口：

```
Status op_adadelta (const Tensor param,
                    const Tensor grad,
                    const Tensor mean_square,
                    const double alpha,
                    const Tensor accum_update,
                    const double learning_rate,
                    const double epsilon,
                    Tensor *param_out,
                    Tensor *mean_square_out,
                    Tensor *accum_update_out);
```

参数：

param (IN)：表示待更新参数。
grad (IN)：表示梯度张量。
mean_square (IN)：表示梯度均方。
alpha (IN)：表示衰减率。
accum_update(IN)：表示参数更新量均方。
learning_rate (IN)：表示学习率。
epsilon (IN)：表示平滑项。
param_out (OUT)：更新后的参数。
mean_square_out (OUT)：表示更新后的梯度均方。
accum_update_out (OUT)：表示更新后的参数更新量均方。

返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH：表示参数的数据类型不一致。

示例：

```
/* param: [0.5, -0.3, 0.2] */
/* grad: [0.1, -0.2, 0.05] */
/* mean_square: [0.01, 0.04, 0.0025] */
/* alpha: 0.9 */
/* accum_update: [0.005, 0.01, 0.0005] */
/* learning_rate: 0.01 */
/* epsilon: 1e-8 */
```

```
Tensor param_out;
Tensor mean_square_out;
Tensor accum_update_out;

op_adadelta(param, grad, mean_square, 0.9, accum_update, 0.01, 1e-8, &param_out,
&mean_square_out, &accum_update_out);

/* param_out: [0.4955, -0.294, 0.19875] */
/* mean_square_out: [0.0109, 0.045, 0.00255] */
/* accum_update_out: [0.005045, 0.0101, 0.000505] */
```

A.2.12.5 RMSProp优化器

C接口：

```
Status op_rmsprop (const Tensor param,
                   const Tensor grad,
                   const Tensor mean_square,
                   const Tensor momentum,
                   const double momentum_factor,
                   const double alpha,
                   const double learning_rate,
                   const double epsilon,
                   Tensor *param_out,
                   Tensor *mean_square_out,
                   Tensor *momentum_out);
```

参数：

param (IN)：表示待更新参数。
grad (IN)：表示梯度张量。
mean_square (IN)：表示梯度均方。
momentum (IN)：表示动量。
momentum_factor (IN)：表示动量因子。
alpha (IN)：表示衰减率。
learning_rate (IN)：表示学习率。
epsilon (IN)：表示平滑项。
param_out (OUT)：更新后的参数。
mean_square_out (OUT)：表示更新后的梯度均方。
momentum_out (OUT)：表示更新后的动量因子。

返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* param: [0.5, -0.3, 0.2] */
/* grad: [0.1, -0.2, 0.05] */
/* mean_square: [0.01, 0.04, 0.0025] */
```

```
/* momentum: [0.02, -0.01, 0.005] */
/* momentum_factor: 0.9 */
/* alpha: 0.99 */
/* learning_rate: 0.01 */
/* epsilon: 1e-8 */

Tensor param_out;
Tensor mean_square_out;
Tensor momentum_out;

op_rmsprop(param, grad, mean_square, momentum, 0.9, 0.99, 0.01, 1e-8, &param_out,
&mean_square_out, &momentum_out);

/* param_out: [0.496, -0.291, 0.198] */
/* mean_square_out: [0.0101, 0.0396, 0.002525] */
/* momentum_out: [0.019, -0.009, 0.00505] */
```

A.2.12.6 CenteredRMSProp优化器

C接口：

```
Status op_centered_rmsprop (const Tensor param,
                            const Tensor grad,
                            const Tensor mean_grad,
                            const Tensor mean_square,
                            const Tensor momentum,
                            const double momentum_factor,
                            const double alpha,
                          const double learning_rate,
                          const double epsilon,
                          Tensor *param_out,
                          Tensor *mean_grad_out,
                          Tensor *mean_square_out,
                          Tensor *momentum_out);
```

参数：

param (IN)：表示待更新参数。
grad (IN)：表示梯度张量。
mean_grad (IN)：表示平均梯度。
mean_square (IN)：表示梯度均方。
momentum (IN)：表示动量。
momentum_factor (IN)：表示动量因子。
alpha (IN)：表示衰减率。
learning_rate (IN)：表示学习率。
epsilon (IN)：表示平滑项。

param_out (OUT)：更新后的参数。

mean_grad_out (OUT)：表示更新后的平均梯度。

mean_square_out (OUT)：表示更新后的梯度均方。

momentum_out (OUT)：表示更新后的动量。

返回值：

STATUS_SUCCESS：表示操作成功。

STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* param: [0.5, -0.3, 0.2] */
/* grad: [0.1, -0.2, 0.05] */
/* mean_grad: [0.01, -0.01, 0.005] */
/* mean_square: [0.02, 0.04, 0.01] */
/* momentum: [0.02, -0.01, 0.005] */
/* momentum_factor: 0.9 */
/* alpha: 0.99 */
/* learning_rate: 0.01 */
/* epsilon: 1e-8 */

Tensor param_out;
Tensor mean_grad_out;
Tensor mean_square_out;
Tensor momentum_out;

op_centered_rmsprop(param, grad, mean_grad, mean_square, momentum, 0.9, 0.99, 0.01,
1e-8, &param_out, &mean_grad_out, &mean_square_out, &momentum_out);

/* param_out: [0.4962, -0.2906, 0.1988] */
/* mean_grad_out: [0.0099, -0.0098, 0.00495] */
/* mean_square_out: [0.0182, 0.0396, 0.00955] */
/* momentum_out: [0.019, -0.009, 0.00505] */
```

A.2.12.7 Adam优化器

C 接口：

```
Status op_adam (const Tensor param,
                const Tensor grad,
                const Tensor m,
                const Tensor v,
                const double beta1,
                const double beta2,
                const double beta1_power,
                const double beta2_power,
                const double learning_rate,
                const double epsilon,
```

```
                    Tensor *param_out,
                    Tensor *m_out,
                    Tensor *v_out,
                    const double *beta1_power_out,
                    const double *beta2_power_out);
```

参数：

param (IN)：表示待更新参数。
grad (IN)：表示梯度张量。
m (IN)：表示训练过程中累加的动量一。
v (IN)：表示训练过程中累加的动量二。
beta1 (IN)：表示计算动量一时的衰减率。
beta2 (IN)：表示计算动量二时的衰减率。
beta1_power (IN)：表示衰减率一的幂。
beta2_power (IN)：表示衰减率二的幂。
learning_rate (IN)：表示学习率。
epsilon (IN)：表示平滑项。
param_out (OUT)：更新后的参数。
m_out (OUT)：表示更新后的动量一。
v_out (OUT)：表示更新后的动量二。
beta1_power_out (OUT)：表示更新后的衰减率一的幂。
beta2_power_out (OUT)：表示更新后的衰减率二的幂。

返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* param: [0.5, -0.3, 0.2] */
/* grad: [0.1, -0.2, 0.05] */
/* m: [0.01, -0.01, 0.005] */
/* v: [0.02, 0.04, 0.01] */
/* beta1: 0.9 */
/* beta2: 0.999 */
/* beta1_power: 0.85 */
/* beta2_power: 0.95 */
/* learning_rate: 0.001 */
/* epsilon: 1e-8 */

Tensor param_out;
Tensor m_out;
Tensor v_out;
double beta1_power_out;
double beta2_power_out;
```

```
op_adam(param, grad, m, v, 0.9, 0.999, 0.85, 0.95, 0.001, 1e-8, &param_out, &m_out,
&v_out, &beta1_power_out, &beta2_power_out);

/* param_out: [0.499, -0.299, 0.199] */
/* m_out: [0.009, -0.018, 0.004] */
/* v_out: [0.019, 0.038, 0.0095] */
/* beta1_power_out: 0.9 */
/* beta2_power_out: 0.999 */
```

A.2.12.8 AdaMax优化器

C接口：

```
Status op_adamax (const Tensor param,
                  const Tensor grad,
                  const Tensor mean_grad,
                  const double alpha,
                  const double alpha_power,
                  const double beta,
                  const Tensor grad_max,
                  const double learning_rate,
                  const double epsilon,
                  Tensor *param_out,
                  Tensor *mean_grad_out,
                  double *alpha_power_out,
                  Tensor *grad_max_out);
```

参数：

param (IN)：表示待更新参数。
grad (IN)：表示梯度张量。
mean_grad (IN)：表示平均梯度。
alpha (IN)：表示计算平均梯度时使用的衰减率。
alpha_power (IN)：表示计算平均梯度时使用的衰减率的幂。
beta (IN)：表示计算梯度最大值时使用的衰减率。
grad_max (IN)：表示梯度最大值。
learning_rate (IN)：表示学习率。
epsilon (IN)：表示平滑项。
param_out (OUT)：更新后的参数。
mean_grad_out (OUT)：表示更新后的平均梯度。
alpha_power_out (OUT)：表示更新后的平均梯度衰减率的幂。
grad_max_out (OUT)：表示更新后的梯度最大值。

返回值：

STATUS_SUCCESS：表示操作成功。
STATUS_TYPE_MISMATCH: 表示参数的数据类型不一致。

示例：

```
/* param: [0.5, -0.3, 0.2] */
```

```
/* grad: [0.1, -0.2, 0.05] */
/* mean_grad: [0.01, -0.01, 0.005] */
/* alpha: 0.9 */
/* alpha_power: 0.85 */
/* beta: 0.999 */
/* grad_max: [0.02, 0.04, 0.01] */
/* learning_rate: 0.001 */
/* epsilon: 1e-8 */

Tensor param_out;
Tensor mean_grad_out;
double alpha_power_out;
Tensor grad_max_out;

op_adamax(param, grad, mean_grad, 0.9, 0.85, 0.999, grad_max, 0.001, 1e-8, &param_out,
&mean_grad_out, &alpha_power_out, &grad_max_out);

/* param_out: [0.4995, -0.2985, 0.19925] */
/* mean_grad_out: [0.009, -0.018, 0.0045] */
/* alpha_power_out: 0.85 */
/* grad_max_out: [0.02, 0.04, 0.01] */
```


参 考 文 献

[1] IEEE 2941.1-2022 IEEE Standard for Operator Interfaces of Artificial Intelligence

[2] 杨超.“人工智能算子接口标准化研究.”知网，2020，https://mall.cnki.net/magazine/Article/DKJS202003002.htm