GE API 参考

文档版本 01

发布日期 2020-05-09

华为技术有限公司

版权所有 © 华为技术有限公司 2020。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明

HUAWEI和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 简介.....1

2 属性类接口.....2

2.1 SetAttr.....2
2.2 GetAttr.....3
2.3 HasAttr.....4
2.4 DelAttr.....5
2.5 SetName.....5
2.6 GetName.....6
2.7 GetItem.....7
2.8 GetValueType.....7
2.9 IsEmpty.....8
2.10 Copy.....9
2.11 MutableTensor.....9
2.12 MutableListTensor.....10
2.13 InitDefault.....11

3 缓存类接口.....12

3.1 ClearBuffer.....12
3.2 GetData.....13
3.3 GetSize.....13
3.4 CopyFrom.....14

4 Graph 类接口.....16

4.1 SetInputs.....16
4.2 SetOutputs.....17
4.3 IsValid.....18
4.4 AddOp.....18
4.5 FindOpByName.....19
4.6 CheckOpByName.....20
4.7 GetAllOpName.....21

5 Model 类接口.....22

5.1 SetName.....22
5.2 GetName.....23
5.3 SetVersion.....23

5.4 GetVersion....24
5.5 SetPlatformVersion....25
5.6 GetPlatformVersion....25
5.7 GetGraph....26
5.8 SetGraph....26
5.9 Save....27
5.10 Load....28
5.11 IsValid....29

6 Operator 类接口....30

6.1 Operator 构造函数....30
6.2 GetName....31
6.3 SetInput（srcOprt 只存在一个输出）....32
6.4 SetInput（srcOprt 存在多个输出）....32
6.5 GetInputDesc....33
6.6 TryGetInputDesc....34
6.7 UpdateInputDesc....35
6.8 GetOutputDesc....36
6.9 UpdateOutputDesc....37
6.10 GetDynamicInputDesc....38
6.11 UpdateDynamicInputDesc....39
6.12 GetDynamicOutputDesc....40
6.13 UpdateDynamicOutputDesc....41
6.14 SetAttr....41
6.15 GetAttr....43

7 Shape 类接口....45

7.1 GetDimNum....45
7.2 GetDim....46
7.3 GetDims....46
7.4 SetDim....47
7.5 GetShapeSize....48

8 Tensor 类接口....49

8.1 GetTensorDesc....49
8.2 MutableTensorDesc....50
8.3 SetTensorDesc....50
8.4 GetData....51
8.5 MutableData....52
8.6 SetData....52
8.7 Clone....53

9 TensorDesc 类接口....55

9.1 Update....55
9.2 GetShape....57

9.3 MutableShape....57
9.4 SetShape....58
9.5 GetFormat....59
9.6 SetFormat....59
9.7 GetDataType....60
9.8 SetDataType....61
9.9 Clone....61
9.10 IsValid....62
10 Operator 注册类....63
10.1 REG_OP....64
10.2 ATTR....64
10.3 REQUIRED_ATTR....66
10.4 INPUT....68
10.5 OPTIONAL_INPUT....69
10.6 DYNAMIC_INPUT....70
10.7 OUTPUT....71
10.8 DYNAMIC_OUTPUT....72
10.9 INFER_SHAPE_AND_TYPE....73
10.10 ATTR_ALL_VERIFY....75
10.11 OP_END....76
10.12 内置的算子类型列表....76
11 模型构建类接口....78
11.1 CreateModelBuff....78
11.2 BuildIRModel....79
11.3 ReleaseModelBuff....80
12 附录....81
12.1 修订记录....81

1 简介

Graph Engine（GE）提供了安全易用的构图接口集合，用户可以调用这些接口构建网络模型，设置模型所包含的图、图内的算子、以及模型和算子的属性。该集合包含如下几个模块：

- 接口类型
  - Operator

    Operator表达了基础的算子类型，提供了算子的属性存取接口、输入/输出的设置接口。
  - Graph

    Graph表达了由多个Operator通过连接关系构建起来图。该模块提供了整图Input/Output的设置以及图的有效性校验接口。
  - Model

    Model表达了由一个Graph构成的模型。该模块提供了模型版本号、模型属性的存取接口，以及模型内部的Graph的存取接口。
- 辅助（数据）类型
  - AttrValue
    - NamedAttr
      - 属性的实例，按照key-value的格式存储。
    - AttrHolder
      - 用于存放Operator的属性值。
  - Tensor
    - TensorDesc：用于描述Tensor的一些信息，包括：
      - Shape：Tensor的形状的描述。
      - DataType：Tensor所存储的数据的类型。
      - Format：Tensor的format信息。

您可以在DDK包的安装目录下的“ddk/include/inc/graph”目录下查看接口的定义文件。如果通过引导安装的方式同时安装Mind Studio和DDK，您可以使用Mind Studio安装用户登录Mind Studio服务器，在“~/tools/che/ddk/ddk/include/inc/graph”路径下查看接口定义文件。每个接口对应的定义文件请参见具体的接口描述。

2 属性类接口

该类接口在attr_value.h、detail/attributes_holder.h中定义。

2.1 SetAttr
2.2 GetAttr
2.3 HasAttr
2.4 DelAttr
2.5 SetName
2.6 GetName
2.7 GetItem
2.8 GetValueType
2.9 IsEmpty
2.10 Copy
2.11 MutableTensor
2.12 MutableListTensor
2.13 InitDefault

2.1 SetAttr

函数原型

graphStatus SetAttr(const string& name, const AttrValue& value);

功能说明

设置属性值。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	属性名称
value	输入	const AttrValue&	属性值

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	成功返回GRAPH_SUCCESS， 否则，返回GRAPH_FAILED

异常处理

无。

约束说明

无。

2.2 GetAttr

函数原型

graphStatus GetAttr(const string& name, AttrValue& value) const;

功能说明

获取属性值。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	属性名称
value	输出	AttrValue&	属性值

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	成功返回GRAPH_SUCCESS， 否则，返回GRAPH_FAILED

异常处理

无。

约束说明

无。

2.3 HasAttr

函数原型

bool HasAttr(const string& name) const;

功能说明

判断是否有对应名称的属性

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	属性名称

返回值

参数名	类型	描述（参数说明、取值范围等）
-	bool	true：有名称为name的属性 false：无名称为name的属性

异常处理

无。

约束说明

无。

2.4 DelAttr

函数原型

graphStatus DelAttr(const string& name);

功能说明

删除属性。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	属性名称

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	成功返回GRAPH_SUCCESS， 否则，返回GRAPH_FAILED

异常处理

无。

约束说明

无。

2.5 SetName

函数原型

void SetName(const std::string& name);

功能说明

设置属性的名称。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	属性名称

返回值

无。

异常处理

无。

约束说明

无。

2.6 GetName

函数原型

string GetName() const;

功能说明

获取属性的名称。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	string	属性名称

异常处理

无。

约束说明

无。

2.7 GetItem

函数原型

AttrValue GetItem(const string& key) const;

功能说明

获取对应名称的属性值。

参数说明

参数名	输入/输出	类型	描述
key	输入	const string&	属性名称

返回值

参数名	类型	描述（参数说明、取值范围等）
-	AttrValue	获取到的属性值

异常处理

无。

约束说明

无。

2.8 GetValueType

函数原型

ValueType GetValueType() const;

功能说明

获取当前属性值的类型；

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	ValueType	返回对应的属性类型

异常处理

无。

约束说明

无。

2.9 IsEmpty

函数原型

bool IsEmpty() const;

功能说明

判断属性值是否设置。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	bool	false：未设置值 true：已设置属性值

异常处理

无。

约束说明

无。

2.10 Copy

函数原型

AttrValue Copy() const;

功能说明

复制属性值，返回一个属性对象。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	AttrValue	属性对象

异常处理

无。

约束说明

无。

2.11 MutableTensor

函数原型

graphStatus MutableTensor(TensorPtr& tensor);

功能说明

获取可以修改的tensor引用。

参数说明

参数名	输入/输出	类型	描述
tensor	输出	TensorPtr&	tensor引用。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	成功返回GRAPH_SUCCESS， 否则，返回GRAPH_FAILED

异常处理

无。

约束说明

无。

2.12 MutableListTensor

函数原型

graphStatus MutableListTensor(vector<TensorPtr>& list_tensor);

功能说明

获取可以修改的Tensor列表的引用。

参数说明

参数名	输入/输出	类型	描述
list_tensor	输出	vector<TensorPtr>&	Tensor列表的引用

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	成功返回GRAPH_SUCCESS， 否则，返回GRAPH_FAILED

异常处理

无。

约束说明

无。

2.13 InitDefault

函数原型

void InitDefault();

功能说明

初始化默认值。

参数说明

无。

返回值

无。

异常处理

无。

约束说明

无

3 缓存类接口

该类接口在buffer.h中定义。

3.1 ClearBuffer

3.2 GetData

3.3 GetSize

3.4 CopyFrom

3.1 ClearBuffer

函数原型

```
void ClearBuffer();
```

功能说明

清空Buffer, 将保存数据的Buffer指针置为null。

参数说明

无。

返回值

无。

异常处理

无。

约束说明

无

3.2 GetData

函数原型

```
std::uint8_t* GetData();
const std::uint8_t* GetData() const;
```

功能说明

获取Buffer中存放的数据的指针。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	uint8_t*	返回Buffer中存放的数据的指针。

异常处理

异常场景	说明
Buffer为空	返回空指针

约束说明

无。

3.3 GetSize

函数原型

```
std::size_t GetSize() const;
```

功能说明

获取Buffer中存放的数据的大小。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	size_t	返回Buffer中存放的数据的大小。

异常处理

异常场景	说明
Buffer为空	返回数据大小为0。

约束说明

无。

3.4 CopyFrom

函数原型

static Buffer CopyFrom(std::uint8_t* data, std::size_t bufferSize);

功能说明

根据data中指针所指向的地址，拷贝bufferSize大小的数据，并将数据地址存放到Buffer中。

参数说明

参数名	输入/输出	类型	描述
data	输入	uint8_t*	存放要拷贝的数据指针。
bufferSize	输入	size_t	拷贝数据的大小。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Buffer	用于存放拷贝数据的Buffer对象。

异常处理

无。

约束说明

无。

4 Graph 类接口

该类接口在graph.h中定义。

4.1 SetInputs
4.2 SetOutputs
4.3 IsValid
4.4 AddOp
4.5 FindOpByName
4.6 CheckOpByName
4.7 GetAllOpName

4.1 SetInputs

函数原型

Graph& SetInputs(std::vector<Operator>& inputs);

功能说明

设置Graph内的输入算子。

参数说明

参数名	输入/输出	类型	描述
inputs	输入	std::vector<Operator>&	Graph内的输入算子。

返回值

参数名	类型	描述
-	Graph&	返回调用者本身

异常处理

无。

约束说明

无。

4.2 SetOutputs

函数原型

Graph& SetOutputs(std::vector<Operator>& outputs);

功能说明

设置Graph关联的输出算子。

参数说明

参数名	输入/输出	类型	描述
outputs	输入	std::vector<Operator>&	与Graph关联的输出算子。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Graph&	返回调用者本身

异常处理

无。

约束说明

无。

4.3 IsValid

函数原型

bool IsValid() const;

功能说明

判断Graph对象是否有效。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	bool	● true，构建Graph对象有效，非空。 ● false，构建Graph对象无效，空指针。

异常处理

无。

约束说明

无。

4.4 AddOp

函数原型

graphStatus AddOp(ge::Operator& op);

功能说明

在Graph中增加算子。

参数说明

参数名	输入/输出	类型	描述
op	输入	ge::Operator&	需增加的算子

返回值

参数名	类型	描述
-	graphStatus	GRAPH_SUCCESS：操作成功 GRAPH_FAILED： 操作失败

异常处理

无。

约束说明

无。

4.5 FindOpByName

函数原型

ge::Operator FindOpByName(const string& name) const;

功能说明

根据算子的名称，返回Graph中的算子实例。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	需指定的算子名称。

返回值

参数名	类型	描述
-	ge::Operator	如果Graph中包含对应名称算子，返回Graph中的这个算子；否则返回type为“NULL”的算子。

异常处理

无。

约束说明

无。

4.6 CheckOpByName

函数原型

graphStatus CheckOpByName(const string& name) const;

功能说明

在Graph中检查指定名称的算子是否存在。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	需指定的算子名称。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	GRAPH_SUCCESS： 指定名称的算子存在 GRAPH_FAILED：指定名称的算子不存在

异常处理

无。

约束说明

无。

4.7 GetAllOpName

函数原型

graphStatus GetAllOpName(std::vector<string>& opName) const ;

功能说明

返回Graph中所有算子的名称。

参数说明

参数名	输入/输出	类型	描述
opName	输出	std::vector<string>&	返回Graph中的所有算子的名称。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	GRAPH_SUCCESS：操作成功 GRAPH_FAILED： 操作失败

异常处理

无。

约束说明

无。

5 Model 类接口

5.1 SetName
5.2 GetName
5.3 SetVersion
5.4 GetVersion
5.5 SetPlatformVersion
5.6 GetPlatformVersion
5.7 GetGraph
5.8 SetGraph
5.9 Save
5.10 Load
5.11 IsValid

5.1 SetName

函数原型

void SetName(const string& name);

功能说明

设置模型的名称。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	模型名称。

返回值

无。

异常处理

无。

约束说明

无。

5.2 GetName

函数原型

string GetName() const;

功能说明

获取模型的名称。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	string	返回模型的名称。

异常处理

无。

约束说明

无。

5.3 SetVersion

函数原型

void SetVersion(uint32_t version)

功能说明

设置模型的版本号。

参数说明

参数名	类型	描述（参数说明、取值范围等）
version	uint32_t	模型的版本号。

返回值

无。

异常处理

无。

约束说明

无。

5.4 GetVersion

函数原型

uint32_t GetVersion() const;

功能说明

获取模型的版本号。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	uint32_t	返回模型的版本号。

异常处理

无。

约束说明

无。

5.5 SetPlatformVersion

函数原型

void SetPlatformVersion(string version)

功能说明

设置用户自定义模型的版本号。

参数说明

参数名	类型	描述（参数说明、取值范围等）
version	string	用户自定义模型的版本号。

返回值

无。

异常处理

无。

约束说明

无。

5.6 GetPlatformVersion

函数原型

std::string GetPlatformVersion() const;

功能说明

获取用户自定义模型的版本号，版本号是常量。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	string	模型版本号。

异常处理

无。

约束说明

无。

5.7 GetGraph

函数原型

Graph GetGraph() const;

功能说明

获取模型中的Graph对象。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Graph	模型中的Graph对象。

异常处理

无。

约束说明

无。

5.8 SetGraph

函数原型

void SetGraph(const Graph& graph);

功能说明

设置模型的Graph对象。

参数说明

参数名	输入/输出	类型	描述
graph	输入	const Graph&	graph 对象。

返回值

无。

异常处理

无。

约束说明

无。

5.9 Save

函数原型

graphStatus Save(Buffer& buffer) const;

功能说明

将模型对象序列化。

参数说明

参数名	输入/输出	类型	描述
buffer	输入	Buffer&	序列化输出的对象的引用。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	序列化成功，返回GRAPH_SUCCESS，否则，返回GRAPH_FAILED。

异常处理

无。

约束说明

无。

5.10 Load

函数原型

static graphStatus Load(const uint8_t* data, size_t len, Model& model);

功能说明

加载序列化数据，反序列化构建模型对象。

参数说明

参数名	输入/输出	类型	描述
data	输入	const uint8_t *	序列化数据指针。
len	输入	size_t	序列化数据长度。
model	输出	Model &	承载反序列化后的模型对象。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	执行成功，返回GRAPH_SUCCESS，否则，返回GRAPH_FAILED。

异常处理

无。

约束说明

无。

5.11 IsValid

函数原型

bool IsValid() const;

功能说明

判断模型中Graph对象是否有效，如果Graph对象为空指针，则无效。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	bool	Model中Graph对象是否有效，true表示有效，false表示无效。

异常处理

无。

约束说明

无。

6 Operator 类接口

该类接口在operator.h中定义。

6.1 Operator构造函数
6.2 GetName
6.3 SetInput（srcOprt只存在一个输出）
6.4 SetInput（srcOprt存在多个输出）
6.5 GetInputDesc
6.6 TryGetInputDesc
6.7 UpdateInputDesc
6.8 GetOutputDesc
6.9 UpdateOutputDesc
6.10 GetDynamicInputDesc
6.11 UpdateDynamicInputDesc
6.12 GetDynamicOutputDesc
6.13 UpdateDynamicOutputDesc
6.14 SetAttr
6.15 GetAttr

6.1 Operator 构造函数

函数原型

Operator();

功能说明

Operator构造函数。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Operator	算子对象。

异常处理

无。

约束说明

无。

6.2 GetName

函数原型

string GetName() const;

功能说明

获取算子名称。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	string	算子名称。

异常处理

无。

约束说明

无。

6.3 SetInput（srcOprt 只存在一个输出）

函数原型

Operator& SetInput(const string& dstName, const Operator& srcOprt);

功能说明

设置当前算子的Input。

参数说明

参数名	输入/输出	类型	描述
dstName	输入	const string&	当前算子的Input名称。
srcOprt	输入	const Operator&	Input名称为dstName的输入算子对象。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Operator&	当前调度者本身。

异常处理

无。

约束说明

要求Operator& srcOprt只存在一个输出。

6.4 SetInput（srcOprt 存在多个输出）

函数原型

Operator& SetInput(const string& dstName, const Operator& srcOprt, const string &name);

功能说明

如果srcOprt存在多个输出，则根据srcOprt的Output名称设置当前算子的Input。

参数说明

参数名	输入/输出	类型	描述
dstName	输入	const string&	当前算子的Input名称。
srcOprt	输入	const Operator&	Input名称为dstName的输入算子对象。
name	输入	const string&	srcOprt的Ouput名称。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Operator&	当前调度者本身。

异常处理

无。

约束说明

无。

6.5 GetInputDesc

函数原型

```
TensorDesc GetInputDesc(const string& name) const;
TensorDesc GetInputDesc(uint32_t index) const;
```

功能说明

根据算子Input名称或Input索引获取Input的TensorDesc。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	算子Input名称。 无算子Input名称，则返回TensorDesc默认构造的对象，其中，主要设置DataType为DT_FLOAT（表示float类型），Format为FORMAT_NCHW（表示NCHW）。
index	输入	uint32_t	算子Input索引。 无算子Input索引，则返回TensorDesc默认构造的对象，其中，主要设置DataType为DT_FLOAT（表示float类型），Format为FORMAT_NCHW（表示NCHW）。

返回值

参数名	类型	描述（参数说明、取值范围等）
TensorDesc	TensorDesc	算子Input的TensorDesc。

异常处理

无。

约束说明

无。

6.6 TryGetInputDesc

函数原型

```
bool TryGetInputDesc(const string& name, TensorDesc& tensorDesc) const;
```

功能说明

根据算子Input名称获取Input的TensorDesc。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	Input名称
tensorDesc	输出	TensorDesc&	TensorDesc对象

返回值

参数名	类型	描述（参数说明、取值范围等）
-	bool	获取TensorDesc成功， 返回true，否则，返回false。

异常处理

无。

约束说明

无。

6.7 UpdateInputDesc

函数原型

graphStatus UpdateInputDesc(const string& name, const TensorDesc& tensorDesc);

功能说明

根据算子Input名称更新Input的TensorDesc。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	算子Input名称
tensorDesc	输入	const TensorDesc&	TensorDesc对象

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	更新TensorDesc成功，返回GRAPH_SUCCESS， 否则，返回GRAPH_FAILED。

异常处理

无。

约束说明

无。

6.8 GetOutputDesc

函数原型

TensorDesc GetOutputDesc(const string& name) const;

TensorDesc GetOutputDesc(uint32_t index) const;

功能说明

根据算子Output名称或Output索引获取Output的TensorDesc。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	算子Output名称。
index	输入	uint32_t	算子Output索引。 无算子Output索引，则返回TensorDesc默认构造的对象，其中，主要设置DataType为DT_FLOAT（表示float类型），Format为FORMAT_NCHW（表示NCHW）。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	TensorDesc	获取TensorDesc成功，返回对应TensorDesc， 否则，返回TensorDesc默认构造的对象，其中，主要设置DataType为DT_FLOAT（表示float类型），Format为FORMAT_NCHW（表示NCHW）。

异常处理

无。

约束说明

无。

6.9 UpdateOutputDesc

函数原型

graphStatus UpdateOutputDesc (const string& name, const TensorDesc& tensorDesc);

功能说明

根据算子Output名称更新Output的TensorDesc。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	算子Output名称。
tensorDesc	输入	const TensorDesc&	TensorDesc对象

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	更新TensorDesc成功，返回GRAPH_SUCCESS， 否则，返回GRAPH_FAILED。

异常处理

无。

约束说明

无。

6.10 GetDynamicInputDesc

函数原型

TensorDesc GetDynamicInputDesc(const string& name, const unsigned int index) const;

功能说明

根据name和index的组合获取算子动态Input的TensorDesc。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	算子动态Input的名称
index	输入	const unsigned int	算子动态Input编号，编号从1开始

返回值

参数名	类型	描述（参数说明、取值范围等）
TensorDesc	TensorDesc	获取TensorDesc成功，则返回算子动态Input的TensorDesc；获取失败，则返回TensorDesc默认构造的对象，其中，主要设置DataType为DT_FLOAT（表示float类型），Format为FORMAT_NCHW（表示NCHW）。

异常处理

无。

约束说明

无。

6.11 UpdateDynamicInputDesc

函数原型

graphStatus UpdateDynamicInputDesc(const string& name, const unsigned int index, const TensorDesc& tensorDesc);

功能说明

根据name和index的组合更新动态Input的TensorDesc。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	算子动态Input的名称
index	输入	const unsigned int	算子动态Input编号，编号从1开始
tensorDesc	输入	const TensorDesc&	TensorDesc对象

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	更新动态Input成功，返回GRAPH_SUCCESS， 否则，返回GRAPH_FAILED。

异常处理

无。

约束说明

无。

6.12 GetDynamicOutputDesc

函数原型

TensorDesc GetDynamicOutputDesc (const string& name, const unsigned int index) const;

功能说明

根据name和index的组合获取算子动态Output的TensorDesc；

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	算子动态Output的名称
index	输入	const unsigned int	算子动态Output编号，编号从1开始

返回值

参数名	类型	描述（参数说明、取值范围等）
TensorDesc	TensorDesc	获取TensorDesc成功，则返回算子动态Output的TensorDesc；获取失败，则返回TensorDesc默认构造的对象，其中，主要设置DataType为DT_FLOAT（表示float类型），Format为FORMAT_NCHW（表示NCHW）。

异常处理

无。

约束说明

无。

6.13 UpdateDynamicOutputDesc

函数原型

graphStatus UpdateDynamicOutputDesc (const string& name, const unsigned int index, const TensorDesc& tensorDesc);

功能说明

根据name和index的组合更新算子动态Output的TensorDesc。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	算子动态Output的名称
index	输入	const unsigned int	算子动态Output编号
tensorDesc	输入	const TensorDesc&	TensorDesc对象

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	更新动态Output成功，返回GRAPH_SUCCESS， 否则，返回GRAPH_FAILED。

异常处理

无。

约束说明

无。

6.14 SetAttr

函数原型

Operator& SetAttr(const string& name, AttrValue&& attrValue);

Operator& SetAttr(const string& name, const UsrQuantizeFactorParams& attrValue);

功能说明

设置算子属性的属性值。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	属性名称。
attrValue	输入	AttrValue&&	需设置的属性值。
attrValue	输入	const UsrQuantizeFactorParams&	量化因子参数。 关于UsrQuantizeFactorParams，请参见**数据类型说明**。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Operator&	对象本身

异常处理

无。

约束说明

无。

数据类型说明

- UsrQuantizeFactorParams结构体：

```
struct UsrQuantizeFactorParams
{
    uint32_t quantize_algo{0};
    uint32_t scale_type{0};
    UsrQuantizeFactor quantize_param;
    UsrQuantizeFactor dequantize_param;
    UsrQuantizeFactor requantize_param;
    UsrQuantizeCalcFactor quantizecalc_param;
};
```

- UsrQuantizeFactor结构体：

```
struct UsrQuantizeFactor
{
    //QuantizeScaleMode scale_mode;
    uint32_t scale_mode{0};
    std::vector<uint8_t> scale_value;
```

```
    int64_t scale_offset{0};
    std::vector<uint8_t> offset_data_value;
    int64_t offset_data_offset{0};
    std::vector<uint8_t> offset_weight_value;
    int64_t offset_weight_offset{0};
    std::vector<uint8_t> offset_pad_value;
    int64_t offset_pad_offset{0};
};
```

- UsrQuantizeCalcFactor结构体：

```
struct UsrQuantizeCalcFactor
{
    std::vector<uint8_t> offsetw;
    int64_t offsetw_offset{0};
    std::vector<uint8_t> offsetd;
    int64_t offsetd_offset{0};
    std::vector<uint8_t> scalereq;
    int64_t scaledreq_offset{0};
    std::vector<uint8_t> offsetdnext;
    int64_t offsetdnext_offset{0};
};
```

6.15 GetAttr

函数原型

graphStatus GetAttr(const string& name, AttrValue& attrValue) const;

graphStatus GetAttr(const string& name, UsrQuantizeFactorParams& attrValue) const;

功能说明

根据属性名称获取其属性值。

参数说明

参数名	输入/输出	类型	描述
name	输入	const string&	属性名称。
attrValue	输入	AttrValue&	属性值。
attrValue	输入	UsrQuantizeFactorParams&	量化因子参数。 关于UsrQuantizeFactorParams，请参见数据类型说明。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	获取属性值成功，返回GRAPH_SUCCESS，否则返回GRAPH_FAILED

异常处理

无。

约束说明

无。

7 Shape 类接口

该类接口在tensor.h中定义。

7.1 GetDimNum

7.2 GetDim

7.3 GetDims

7.4 SetDim

7.5 GetShapeSize

7.1 GetDimNum

函数原型

size_t GetDimNum() const;

功能说明

获取Shape的维度个数。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	size_t	Tensor Shape的维度个数。

异常处理

无。

约束说明

无。

7.2 GetDim

函数原型

int64_t GetDim(size_t idx) const;

功能说明

获取Shape第idx维的长度。

参数说明

参数名	输入/输出	类型	描述
idx	输入	size_t	维度索引，索引从0开始。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	int64_t	第idx维的长度

异常处理

无

约束说明

无。

7.3 GetDims

函数原型

std::vector<int64_t> GetDims() const;

功能说明

获取Shape所有维度组成的向量。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	std::vector<int64_t>	Shape的所有维度组成的向量。

异常处理

无。

约束说明

无。

7.4 SetDim

函数原型

graphStatus SetDim(size_t idx, int64_t value);

功能说明

将Shape中第idx维度的值设置为value。

参数说明

参数名	输入/输出	类型	描述
idx	输入	size_t	Shape维度的索引，索引从0开始。
value	输入	int64_t	需设置的值。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	设置成功返回GRAPH_SUCCESS，否则，返回GRAPH_FAILED。

异常处理

无。

约束说明

无。

7.5 GetShapeSize

函数原型

int64_t GetShapeSize() const;

功能说明

获取Shape中所有dim的累乘结果。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	int64_t	返回所有dim的累乘结果。

异常处理

无。

约束说明

无。

8 Tensor 类接口

该类接口在tensor.h中定义。

8.1 GetTensorDesc
8.2 MutableTensorDesc
8.3 SetTensorDesc
8.4 GetData
8.5 MutableData
8.6 SetData
8.7 Clone

8.1 GetTensorDesc

函数原型

TensorDesc GetTensorDesc() const;

功能说明

获取Tensor的描述符（TensorDesc）。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	TensorDesc	返回当前Tensor的描述符。

异常处理

无。

约束说明

无。

8.2 MutableTensorDesc

函数原型

TensorDesc& MutableTensorDesc();

功能说明

获取当前Tensor的描述符，可修改。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	TensorDesc&	当前Tensor的描述符，可修改。

异常处理

无。

约束说明

无。

8.3 SetTensorDesc

函数原型

graphStatus SetTensorDesc(const TensorDesc &tensorDesc);

功能说明

设置Tensor的描述符。

参数说明

参数名	输入/输出	类型	描述
tensorDesc	输入	const TensorDesc &	需设置的Tensor描述符。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	设置成功返回GRAPH_SUCCESS，否则，返回GRAPH_FAILED。

异常处理

无。

约束说明

无。

8.4 GetData

函数原型

const Buffer GetData() const;

功能说明

获取Tensor中的数据。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	const Buffer	Tensor中所存放的数据。

异常处理

无。

约束说明

无。

8.5 MutableData

函数原型

Buffer MutableData();

功能说明

获取Tensor中存放的数据。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Buffer	Tensor中存放的数据。

异常处理

无。

约束说明

无。

8.6 SetData

函数原型

graphStatus SetData(std::vector<uint8_t> &&data);

graphStatus SetData(const std::vector<uint8_t> &data);

graphStatus SetData(const Buffer &data);

graphStatus SetData(const uint8_t *data, size_t size);

功能说明

向Tensor中设置数据。

参数说明

参数名	输入/输出	类型	描述
data	输入	std::vector<uint8_t> && 或const std::vector<uint8_t> & 或const Buffer & 或const uint8_t *	需设置的数据。
size	输入	size_t	数据的长度，单位为字节。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	设置成功返回GRAPH_SUCCESS，否则，返回GRAPH_FAILED。

异常处理

无。

约束说明

无。

8.7 Clone

函数原型

Tensor Clone() const;

功能说明

拷贝Tensor。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Tensor	返回拷贝的Tensor对象。

异常处理

无。

约束说明

无。

9 TensorDesc 类接口

该类接口在tensor.h中定义。

9.1 Update
9.2 GetShape
9.3 MutableShape
9.4 SetShape
9.5 GetFormat
9.6 SetFormat
9.7 GetDataType
9.8 SetDataType
9.9 Clone
9.10 IsValid

9.1 Update

函数原型

void Update(Shape shape, Format format = FORMAT_NCHW, DataType dt = DT_FLOAT);

功能说明

更新TensorDesc对象的shape、format、datatype属性。

参数说明

参数名	输入/输出	类型	描述
shape	输入	Shape	需刷新的shape对象。

参数名	输入/输出	类型	描述
format	输入	Format	需刷新的format对象，默认取值FORMAT_NCHW。 关于Format数据类型的定义，请参见•**Format**。
dt	输入	DataType	需刷新的datatype对象，默认取值DT_FLOAT。 关于DataType数据类型的定义，请参见•**DataType**。

返回值

无。

异常处理

无。

约束说明

无。

数据类型说明

- Format

```
enum Format {
    FORMAT_NCHW = 0,          /**< NCHW */
    FORMAT_NHWC,              /**< NHWC */
    FORMAT_ND,                /**< Nd Tensor */
    FORMAT_NC1HWC0,           /**< NC1HWC0 */
    FORMAT_FRACTAL_Z,         /**< FRACTAL_Z */
    FORMAT_NC1C0HWPAD,
    FORMAT_NHWC1C0,
    FORMAT_FSR_NCHW,
    FORMAT_FRACTAL_DECONV,
    FORMAT_C1HWNC0,
    FORMAT_FRACTAL_DECONV_TRANSPOSE,
    FORMAT_FRACTAL_DECONV_SP_STRIDE_TRANS,
    FORMAT_NC1HWC0_C04,  /**< NC1HWC0, C0 =4*/
    FORMAT_FRACTAL_Z_C04,/**< FRACZ格式，C0 =4 */
    FORMAT_CHWN,
    FORMAT_FRACTAL_DECONV_SP_STRIDE8_TRANS,
    FORMAT_HWCN,
    FORMAT_NC1KHKWHWC0, /** < KH,KW kernel h& kernel w maxpooling max output format*/
    FORMAT_BN_WEIGHT,
    FORMAT_FILTER_HWCK,     /* filter input tensor format */
    FORMAT_HASHTABLE_LOOKUP_LOOKUPS=20,
    FORMAT_HASHTABLE_LOOKUP_KEYS,
    FORMAT_HASHTABLE_LOOKUP_VALUE,
    FORMAT_HASHTABLE_LOOKUP_OUTPUT,
    FORMAT_HASHTABLE_LOOKUP_HITS=24,
    FORMAT_RESERVED

};
```

- DataType

```
enum DataType {
    DT_UNDEFINED = 16,   // Used to indicate a DataType field has not been set.
    DT_FLOAT = 0,          // float type
    DT_FLOAT16 = 1,        // fp16 type
    DT_INT8 = 2,           // int8 type
    DT_INT16 = 6,     // int16 type
    DT_UINT16 = 7,     // uint16 type
    DT_UINT8 = 4,          // uint8 type
    DT_INT32 = 3,          //
    DT_INT64 = 9,          // int64 type
    DT_UINT32 = 8,         // unsigned int32
    DT_UINT64 = 10,        // unsigned int64
    DT_BOOL = 12,          // bool type
    DT_DOUBLE = 11,        // double type
    DT_DUAL = 13,          /**< dual output type */
    DT_DUAL_SUB_INT8 = 14,    /**< dual output int8 type */
    DT_DUAL_SUB_UINT8 = 15,    /**< dual output uint8 type */
};
```

9.2 GetShape

函数原型

Shape GetShape() const;

功能说明

获取TensorDesc描述的shape。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Shape	TensorDesc描述的shape。

异常处理

无。

约束说明

返回的shape为const，不可修改。

9.3 MutableShape

函数原型

Shape& MutableShape();

功能说明

获取TensorDesc中可修改的shape引用。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Shape &	返回TensorDesc中的shape引用，可修改。

异常处理

无。

约束说明

无。

9.4 SetShape

函数原型

void SetShape(Shape shape);

功能说明

设置TensorDesc的shape。

参数说明

参数名	输入/输出	类型	描述
shape	输入	Shape	需向TensorDesc设置的shape对象

返回值

无。

异常处理

无。

约束说明

无。

9.5 GetFormat

函数原型

Format GetFormat() const;

功能说明

获取TensorDesc所描述的Tensor的format信息。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	Format	TensorDesc所描述的Tensor的format信息。

异常处理

无。

约束说明

无。

9.6 SetFormat

函数原型

void SetFormat(Format format);

功能说明

设置TensorDesc所描述的Tensor的format信息。

参数说明

参数名	输入/输出	类型	描述
format	输入	Format	需设置的format信息。

返回值

无。

异常处理

无。

约束说明

无。

9.7 GetDataType

函数原型

DataType GetDataType() const;

功能说明

获取TensorDesc所描述的Tensor的数据类型。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	DataType	TensorDesc所描述的Tensor的数据类型。

异常处理

无。

约束说明

无。

9.8 SetDataType

函数原型

void SetDataType(DataType dt);

功能说明

设置TensorDesc所描述的Tensor的数据类型。

参数说明

参数名	输入/输出	类型	描述
dt	输入	DataType	需设置的dt信息。 关于DataType，请参见**•DataType**。

返回值

无。

异常处理

无。

约束说明

无。

9.9 Clone

函数原型

TensorDesc Clone() const;

功能说明

拷贝TensorDesc。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	TensorDesc	拷贝的TensorDesc对象。

异常处理

无。

约束说明

无。

9.10 IsValid

函数原型

graphStatus IsValid();

功能说明

判断tensor对象是否有效。

参数说明

无。

返回值

参数名	类型	描述（参数说明、取值范围等）
-	graphStatus	tensor对象是否有效，有效返回GRAPH_SUCCESS，否则返回GRAPH_FAILED

异常处理

无。

约束说明

无。

10 Operator 注册类

注册算子类型以REG_OP为起始，以“.”链接INPUT、OUTPUT、ATTR等接口注册算子的输入、输出和属性信息，最终以OP_END接口结束。注册算子类型成功后，自动生成以算子类型名称命名的类。

例如：

```
REG_OP(FullConnection)
.INPUT(x, TensorType::ALL())
.INPUT(w, TensorType::ALL())
.INPUT(b, TensorType::ALL())
.OUTPUT(y, TensorType::ALL())
.ATTR(num_output, AttrValue::INT{0})
.INFER_SHAPE_AND_TYPE(FullConnectionInfer)
.ATTR_ALL_VERIFY(FullConnectionVerify)
.OP_END()
```

Operator注册类接口在operator_reg.h中定义。已注册的算子及对应的头文件，请参见**10.12 内置的算子类型列表**。

10.1 REG_OP

10.2 ATTR

10.3 REQUIRED_ATTR

10.4 INPUT

10.5 OPTIONAL_INPUT

10.6 DYNAMIC_INPUT

10.7 OUTPUT

10.8 DYNAMIC_OUTPUT

10.9 INFER_SHAPE_AND_TYPE

10.10 ATTR_ALL_VERIFY

10.11 OP_END

10.12 内置的算子类型列表

10.1 REG_OP

函数原型

REG_OP(x)

功能说明

注册算子类型，同时自动生成算子类型的两个构造函数。

例如，注册算子的类型名称Conv2D，可调用REG_OP(Conv2D)接口，调用该接口后，定义了算子的类型名称Conv2D，同时产生Conv2D的两个构造函数，其中，**Conv2D(const string& name)**需指定算子名称，Conv2D()使用默认算子名称，例如“Conv2D*唯一编号*”。

```
class Conv2D : public Operator {
    typedef Conv2D _THIS_TYPE;
public:
    explicit Conv2D(const string& name);
    explicit Conv2D();
}
```

参数说明

参数名	输入/输出	类型	描述
x	输入	-	宏参数，被注册算子的类型名称

返回值

无。

异常处理

无。

约束说明

注册的算子类型名称需保持唯一，不能重复。

10.2 ATTR

函数原型

ATTR(x, default_value)

功能说明

注册算子属性，必须指定默认值，用户不设置算子对象的属性值时使用默认值。

注册算子属性成功后，自动生成算子属性的3个对外接口，用于获取属性的名称、获取属性的值、设置属性的值。

现以注册类型为int64_t的属性、类型为int64_t列表两种场景为例，说明所生成的算子属性接口：

- 调用ATTR(mode, AttrValue::INT{1})接口，注册属性mode，属性类型为int64_t，默认值为1。

  注册属性成功后，自动生成以下接口：

  ```
  static const string name_attr_mode(); // 返回属性的名称，即“mode”
  int64_t get_attr_mode() const; // 返回mode属性的值
  _THIS_TYPE& set_attr_mode(int64_t v); // 设置mode属性的值，返回算子对象本身
  ```

- 调用ATTR(pad, AttrValue::LIST_INT{0, 0, 0, 0})接口，注册属性pad，属性类型为int64_t列表，默认值为{0,0,0,0}。

  注册属性成功后，自动生成以下接口：

  ```
  static const string name_attr_pad(); // 返回属性的名称，即“pad”
  vector<int64_t> get_attr_pad() const; ; // 返回属性pad的值
  _THIS_TYPE& set_attr_pad(vector<int64_t> v); // 设置属性pad的值，返回算子对象本身
  ```

参数说明

参数名	输入/输出	类型	描述
x	输入	-	宏参数，算子属性的名称。

参数名	输入/输出	类型	描述
default_value	输入	-	算子属性的值，根据不同类型指定默认值，支持的属性类型包括： ● AttrValue::INT，属性类型为int64_t ● AttrValue::FLOAT, 属性类型为float ● AttrValue::STR，属性类型为string ● AttrValue::BOOL，属性类型为bool ● AttrValue::TENSOR，属性类型为Tensor ● AttrValue::LIST_INT，属性类型为vector<int64_t>, int64_t列表 ● AttrValue::LIST_FLOAT, 属性类型为vector<float>, float列表 ● AttrValue::LIST_STR，属性类型为vector<string>，string列表 ● AttrValue::LIST_BOOL，属性类型为vector<bool>，bool列表 ● AttrValue::LIST_TENSOR，属性类型为vector<Tensor>，Tensor列表

返回值

无。

异常处理

无。

约束说明

对于同一个算子，注册的算子属性名称需保持唯一，不能重复。

10.3 REQUIRED_ATTR

函数原型

REQUIRED_ATTR (x, type)

功能说明

注册算子属性，没有默认值，用户必须设置算子对象的属性值。

注册算子属性成功后，自动生成算子属性的3个对外接口，用于获取属性的名称、获取属性的值、设置属性的值。

例如，注册类型为int64_t的属性mode，可调用**REQUIRED_ATTR (mode, Int)**接口，注册算子属性成功后，会自动生成如下接口：

```
static const string name_attr_mode(); // 返回属性的名称，即“mode”
OpInt get_attr_mode() const; // 返回mode属性的值，OpInt即int64_t
_THIS_TYPE& set_attr_mode(const OpInt& v); // 设置mode属性的值，返回this对象
```

参数说明

参数名	输入/输出	类型	描述
x	输入	-	宏参数，算子属性的名称。
type	输入	-	算子属性的类型，包括： ● AttrValue::INT，属性类型为int64_t ● AttrValue::FLOAT, 属性类型为float ● AttrValue::STR，属性类型为string ● AttrValue::BOOL，属性类型为bool ● AttrValue::TENSOR，属性类型为Tensor ● AttrValue::LIST_INT，属性类型为vector<int64_t>, int64_t列表 ● AttrValue::LIST_FLOAT, 属性类型为vector<float>, float列表 ● AttrValue::LIST_STR，属性类型为vector<string>，string列表 ● AttrValue::LIST_BOOL，属性类型为vector<bool>，bool列表 ● AttrValue::LIST_TENSOR，属性类型为vector<Tensor>，Tensor列表

返回值

无。

异常处理

无。

约束说明

对于同一个算子，注册的算子属性名称需保持唯一，不能重复。

10.4 INPUT

函数原型

INPUT (x, t)

功能说明

注册算子输入信息。

注册算子输入信息成功后，自动生成算子输入的相关接口，用于获取算子输入的名称、设置算子输入的对应描述等。

例如，注册算子输入x，算子输入接收的数据类型为TensorType{DT_FLOAT}，可调用**INPUT(x, TensorType{DT_FLOAT})**接口，注册算子输入成功后，自动生成以下相关接口：

```
static const string name_in_x(); // 返回输入的名称，即“x”
_THIS_TYPE& set_input_x(Operator& v, const string& srcName);// 指定输入x与算子对象v的输出srcName存在连接关系，返回算子对象本身
_THIS_TYPE& set_input_x(Operator& v); // 指定输入x与算子对象v的索引0的输出存在连接关系，返回算子对象本身
TensorDesc get_input_desc_x(); // 返回输入x对应的描述
graphStatus update_input_desc_x(const TensorDesc& tensorDesc);// 设置输入x对应的描述，包括Shape、DataType、Format等信息，graphStatus即uint32_t类型，返回非0表示出错
```

参数说明

参数名	输入/输出	类型	描述
x	输入	-	宏参数，算子输入的名称
t	输入	-	算子输入接收的数据类型，可以是TensorType定义的一个或多个，如果多个，通过“,”隔离，例如： TensorType{DT_FLOAT} TensorType({DT_FLOAT, DT_INT8} 关于TensorType类，请参见**TensorType类说明**。

返回值

无。

异常处理

无。

约束说明

对于同一个算子，注册的算子输入名称需保持唯一，不能重复。

TensorType 类说明

TensorType类用以定义输入或者输出支持的数据类型，TensorType提供以下接口指定支持的数据类型：

- TensorType(DataType dt)：指定仅支持一个数据类型；
- TensorType(std::initializer_list<DataType> types)：指定支持多个数据类型；
- static TensorType ALL()：指定支持所有数据类型；
- static TensorType FLOAT()：指定支持DT_FLOAT和DT_FLOAT16数据类型。

10.5 OPTIONAL_INPUT

函数原型

OPTIONAL_INPUT(x, t)

功能说明

注册可选算子输入信息。

注册可选算子输入信息成功后，自动生成算子输入的相关接口，用于获取算子输入的名称、设置算子输入的对应描述等。

例如，注册算子输入b，算子输入接收的数据类型为TensorType{DT_FLOAT}，可调用**OPTIONAL_INPUT(b, TensorType{DT_FLOAT})**接口，注册算子输入成功后，自动生成以下相关接口：

```
static const string name_in_b(); // 返回输入的名称，即“b”
_THIS_TYPE& set_input_b(Operator& v, const string& srcName);// 指定输入b与算子对象v的输出srcName存在连接关系，返回算子对象本身
_THIS_TYPE& set_input_b(Operator& v); // 指定输入b与算子对象v的索引0的输出存在连接关系，返回算子对象本身
TensorDesc get_input_desc_b(); // 返回输入b对应的描述
graphStatus update_input_desc_b(const TensorDesc& tensorDesc);// 设置输入b对应的描述，包括Shape、DataType、Format等信息
```

参数说明

参数名	输入/输出	类型	描述
x	输入	-	宏参数，算子输入的名称。

参数名	输入/输出	类型	描述
t	输入	-	算子输入接收的数据类型，可以是TensorType定义的一个或多个，如果多个，通过“,”隔离，例如： TensorType{DT_FLOAT} TensorType({DT_FLOAT, DT_INT8}

返回值

无。

异常处理

无。

约束说明

对于同一个算子，注册的算子输入名称需保持唯一，不能重复。

10.6 DYNAMIC_INPUT

函数原型

DYNAMIC_INPUT (x, t)

功能说明

注册动态算子输入信息。

注册动态算子输入信息成功后，自动生成算子输入的相关接口，用于创建动态输入、设置算子输入的对应描述等。

例如，注册动态算子输入d，算子输入接收的数据类型为TensorType{DT_FLOAT}，可调用**DYNAMIC_INPUT(d, TensorType{DT_FLOAT})**接口，注册动态算子输入成功后，自动生成以下相关接口：

```
_THIS_TYPE& create_dynamic_input_d(unsigned int num); // 创建动态输入d，包括num个输入
TensorDesc get_dynamic_input_desc_d(unsigned int index);// 返回动态输入d第index个描述，包括Shape、DataType、Format等信息
graphStatus update_dynamic_input_desc_d(unsigned int index, const TensorDesc& tensorDesc);// 更新动态输入d的第index个描述
_THIS_TYPE& set_dynamic_input_d(unsigned int dstIndex, Operator &v); // 指定输入d的第dstIndex个输入与算子对象v的索引0的输出存在连接关系，返回算子对象本身
_THIS_TYPE& set_dynamic_input_d(unsigned int dstIndex, Operator &v, const string &srcName); / /指定动态输入d的第dstIndex个输入与算子对象v的输出srcName存在连接关系，返回算子对象本身
```

参数说明

参数名	输入/输出	类型	描述
x	输入	-	宏参数，算子输入的名称
t	输入	-	算子输入接收的数据类型，可以是TensorType定义的一个或多个，如果多个，通过“,”隔离，例如： TensorType{DT_FLOAT} TensorType({DT_FLOAT, DT_INT8}

返回值

无。

异常处理

无。

约束说明

对于同一个算子，注册的算子输入名称需保持唯一，不能重复。

10.7 OUTPUT

函数原型

OUTPUT (x, t)

功能说明

注册算子输出信息。

注册算子输出信息成功后，自动生成算子输出的相关接口，用户获取算子输出的名称、获取算子输出的描述、设置算子输出的描述。

例如，注册算子输出y，算子输出接收的数据类型为TensorType{DT_FLOAT}，可调用**OUTPUT(y, TensorType{DT_FLOAT})**接口，注册算子输出成功后，自动生成以下相关接口

```
static const string name_out_y();// 返回输出的名称，即“y”
TensorDesc get_output_desc_y();// 返回输出y对应的描述
graphStatus update_output_desc_y(const TensorDesc& tensorDesc); );// 设置输出y对应的描述，包括Shape、DataType、Format等信息
```

参数说明

参数名	输入/输出	类型	描述
x	输入	-	宏参数，算子输出的名称。
t	输入	-	算子输出接收的数据类型，可以是TensorType定义的一个或多个，如果多个，通过“,”隔离，例如： TensorType{DT_FLOAT} TensorType({DT_FLOAT, DT_INT8}

返回值

无。

异常处理

无。

约束说明

对于同一个算子，注册的算子输出名称需保持唯一，不能重复。

10.8 DYNAMIC_OUTPUT

函数原型

DYNAMIC_OUTPUT (x, t)

功能说明

注册动态算子输出信息。

注册动态算子输出信息成功后，自动生成动态算子输出的相关接口，包括用于创建动态输出、设置算子输出的对应描述等

例如，注册动态算子输出d，算子输出接收的数据类型为TensorType{DT_FLOAT}，可调用**DYNAMIC_OUTPUT (d, TensorType{DT_FLOAT})**接口，注册动态算子输出成功后，自动生成以下相关接口

```
_THIS_TYPE& create_dynamic_output_d(unsigned int num); // 创建动态输出d，包括num个输出
TensorDesc get_dynamic_output_desc_d(unsigned int index);// 返回动态输出d第index个描述，包括Shape、DataType、Format等信息
graphStatus update_dynamic_output_desc_d(unsigned int index, const TensorDesc& tensorDesc);// 更新动态输出d的第index个描述
```

参数说明

参数名	输入/输出	类型	描述
x	输入	-	宏参数，算子输出的名称。
t	输入	-	算子输出接收的数据类型，可以是TensorType定义的一个或多个，如果多个，通过“,”隔离，例如： TensorType{DT_FLOAT} TensorType({DT_FLOAT, DT_INT8}

返回值

无。

异常处理

无。

约束说明

对于同一个算子，注册的算子输出名称需保持唯一，不能重复。

10.9 INFER_SHAPE_AND_TYPE

函数原型

INFER_SHAPE_AND_TYPE (x)

功能说明

注册用于推理算子的Shape和DataType的函数。

参数说明

参数名	输入/输出	类型	描述
x	输入	-	宏参数，算子Shape和DataType推理函数。 例如，**INFER_SHAPE_AND_TYPE(FullConnectionInfer)**用于注册**FullConnectionInfer**函数，用于推理算子的Shape和DataType。 FullConnectionInfer通过DECLARE_INFERFUNC声明，通过IMPLEMT_INFERFUNC定义，详情请参见**DECLARE_INFERFUNC和IMPLEMT_INFERFUNC宏说明**。

返回值

无。

异常处理

无。

约束说明

无。

DECLARE_INFERFUNC 和 IMPLEMT_INFERFUNC 宏说明

在注册用于推理算子的Shape和DataType的函数前，需要先用DECLARE_INFERFUNC宏申明函数、用IMPLEMT_INFERFUNC宏定义函数。

- 声明函数

```
DECLARE_INFERFUNC(FullConnection, FullConnectionInfer)
```

DECLARE_INFERFUNC宏展开后的实现为：

```
namespace op {
class FullConnection;
}
static graphStatus FullConnectionInfer(op::FullConnection& op);
```

- 定义函数

```
IMPLEMT_INFERFUNC(FullConnection, FullConnectionInfer) {
// 实现细节
}
```

IMPLEMT_INFERFUNC宏展开后的实现为：

```
static graphStatus FullConnectionInfer(op::FullConnection& op){
// 实现细节
}
```

10.10 ATTR_ALL_VERIFY

函数原型

ATTR_ALL_VERIFY (x)

功能说明

注册算子校验函数。

参数说明

参数名	输入/输出	类型	描述
x	输入	-	宏参数，算子校验函数。 例如，**ATTR_ALL_VERIFY(FullConnectionVerify)**用于注册算子校验函数为**FullConnectionVerify**。FullConnectionVerify通过DECLARE_VERIFIER声明，通过IMPLEMT_VERIFIER定义，详情请参见**DECLARE_VERIFIER和IMPLEMT_VERIFIER宏说明**。

返回值

无。

异常处理

无。

约束说明

无。

DECLARE_VERIFIER 和 IMPLEMT_VERIFIER 宏说明

在注册算子校验函数前，需要先用DECLARE_VERIFIER宏申明函数、用IMPLEMT_VERIFIER宏定义函数。

- 声明函数

  ```
  DECLARE_VERIFIER(FullConnection, FullConnectionVerify)
  ```

  DECLARE_VERIFIER宏展开后的实现为：

  ```
  namespace op {
  class FullConnection;
  }
  static graphStatus FullConnectionVerify(op::FullConnection op);
  ```

- 定义函数

```
IMPLEMT_VERIFIER(FullConnection, FullConnectionVerify) {
// 实现细节
}
```

IMPLEMT_VERIFIER宏展开后的实现为：

```
static graphStatus FullConnectionVerify(op::FullConnection op){
// 实现细节
}
```

10.11 OP_END

函数原型

OP_END ()

功能说明

结束算子注册。

参数说明

无。

返回值

无。

异常处理

无。

约束说明

无。

10.12 内置的算子类型列表

您可以在注册算子类型的头文件中查看内置的算子类型，注册各算子类型时设置的算子输入、输出、属性等信息。

表 10-1 算子列表

算子名称	所在头文件
Data	array_defs.h
Concat	
Flatten	
Reshape	

算子名称	所在头文件
Split	
Const	const_defs.h
Permute	detection_defs.h
Add	math_defs.h
Mul	
Activation	nn_defs.h
BatchNorm	
Convolution	
Eltwise	
LRN	
ConvolutionDepthwise	
FullConnection	
Pooling	
Scale	
ShuffleChannel	
Softmax	

11 模型构建类接口

模型构建时的接口调用顺序如下：

CreateModelBuff-->BuildIRModel-->ReleaseModelBuff

11.1 CreateModelBuff

11.2 BuildIRModel

11.3 ReleaseModelBuff

11.1 CreateModelBuff

函数原型

bool CreateModelBuff(ge::Model& irModel,ModelBufferData& output);

功能说明

创建模型Buffer。

参数说明

参数名	输入/输出	类型	描述
irModel	输入	ge::Model&	模型对象。

output	输出	ModelBufferData&	离线模型结构体对象。
			struct ModelBufferData
			{
			void* data;
			uint32_t length;
			};

返回值

参数名	类型	描述（参数说明、取值范围等）
-	bool	● true，创建模型Buffer成功。 ● false，创建模型Buffer失败。

异常处理

无。

约束说明

无。

11.2 BuildIRModel

函数原型

bool BuildIRModel(ge::Model& irModel,ModelBufferData& output);

功能说明

离线模型构建接口，输入模型对象，输出离线模型。

参数说明

参数名	输入/输出	类型	描述
irModel	输入	ge::Model&	模型对象。

output	输出	ModelBufferData&	离线模型结构体对象。
			struct ModelBufferData
			{
			void* data;
			uint32_t length;
			};

返回值

参数名	类型	描述（参数说明、取值范围等）
-	bool	● true，构建模型成功。 ● false，构建模型失败。

异常处理

无。

约束说明

无。

11.3 ReleaseModelBuff

函数原型

void ReleaseModelBuff(ModelBufferData& output);

功能说明

释放模型Buffer。

参数说明

参数名	输入/输出	类型	描述

output	输出	ModelBufferData&	离线模型结构体对象。
			struct ModelBufferData
			{
			void* data;
			uint32_t length;
			};

返回值

无。

异常处理

无。

约束说明

无。

12 附录

12.1 修订记录

12.1 修订记录

文档版本	发布日期	修改说明
01	2020-05-09	第一次正式发布。