ICS 35.040

CCS L 71

团体标准

T/AI XXX.XX—XXXX

|  |
| --- |
|  |

人工智能 终端硬件推理平台接口标准

Artificial Intelligence Terminal Hardware Inference Interface

|  |
| --- |
| （征求意见稿）  （在提交反馈意见时，请将您知道的相关专利连同支持性文件一并附上） |
|  |

XXXX - XX - XX实施

中关村视听产业技术创新联盟 发布

XXXX - XX - XX发布

目  次

[前言 II](#_Toc185963486)

[1 范围 1](#_Toc185963488)

[2 规范性引用文件 1](#_Toc185963489)

[3 术语和定义 1](#_Toc185963490)

[4 缩略语 1](#_Toc185963491)

[5 整体架构 2](#_Toc185963492)

[5.1 使用场景 2](#_Toc185963493)

[5.2 接口分类 2](#_Toc185963494)

[6 算子 3](#_Toc185963495)

[7 模型构建接口 3](#_Toc185963496)

[7.1 算子注册类接口 3](#_Toc185963497)

[7.2 Graph接口 5](#_Toc185963498)

[7.3 Model接口 7](#_Toc185963499)

[8 模型编译接口 9](#_Toc185963500)

[8.1 BuildIRModel接口 9](#_Toc185963501)

[9 模型执行接口 9](#_Toc185963502)

[9.1 Init接口 9](#_Toc185963503)

[9.2 Load接口 9](#_Toc185963504)

[9.3 Process接口 10](#_Toc185963505)

[9.4 UnLoad 接口 10](#_Toc185963506)

[9.5 Cancel接口 10](#_Toc185963507)

[9.6 DeInit接口 10](#_Toc185963508)

[附　录　A （资料性附录） 接口示例基于C++语言描述 12](#_Toc185963509)

[附　录　B （资料性附录） 使用步骤描述 19](#_Toc185963510)

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。

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：华为技术有限公司、旷世科技有限公司、科大讯飞有限公司、字节跳动有限公司。

本文件主要起草人：文长春、马伟刚、秦巍、纪东良、麦睿楷、李小娟、关贺、刘勇。

人工智能 终端硬件推理平台接口标准

1. 范围

本文件规定了终端硬件推理平台的算子、模型构建、模型编译以及模型执行等不同类型接口的定义。

本文件适用于终端硬件推理平台API的设计和开发。

1. 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；不注明日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

T/AI XXX.1-202X 人工智能 算子接口 第1部分：基础数学类算子

T/AI XXX.2-202X 人工智能 算子接口 第2部分：神经网络类算子

T/AI XXX.3-202X 人工智能 算子接口 第3部分：机器学习类算子

1. 术语和定义

下列术语和定义适用于本文件。

AI模型 artificial intelligence model

使用数据和专家知识进行“训练”，并模拟人的决策过程以实现自动化和理解的数学算法。



推理平台接口 inference platform interface

推理平台对AI模型进行构建、编译、执行等操作的功能接口。



IR模型 intermediate representation model

与硬件无关的、神经网络计算的图表达中间层。

1. 缩略语

下列缩略语适用于本部分。

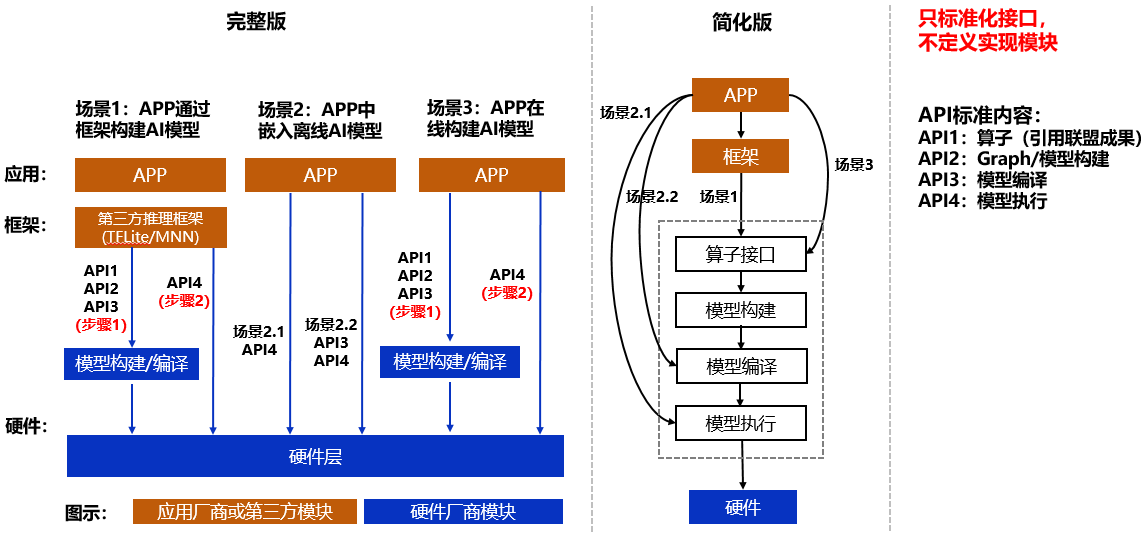
AI：人工智能（Artificial Intelligence）

API：应用程序编程接口（Application Programming Interface）

APP：应用程序（Application）

IR：中间表示（Intermediate Representation）

1. 整体架构
   1. 使用场景



1. 整体架构

应根据使用场景的需求调用相应API，实现AI模型硬件加速。API编程语言宜使用C++实现。本标准的应用场景包含但不限于下列三种典型场景。

场景1：APP通过第三方框架构建AI模型。该场景适用于使用第三方框架训练好的AI模型。三方框架通过调用算子类接口完成算子级联；通过Graph/模型构建类接口完成Graph/模型构建；然后通过调用模型编译类接口将训练好的AI模型编译成适合在硬件上执行的AI模型文件；最后通过调用模型执行类接口实现模型加载，模型推理，模型销毁等功能。

场景2：APP中嵌入离线AI模型。

场景2.1：该场景适用于将已经编译好的可执行的AI模型文件直接导入APP中，APP通过调用模型执行接口实现模型加载，模型推理，模型销毁等功能，模型不支持跨平台运行。

场景2.2：该场景适用于将已经编译好的AI IR模型文件直接导入APP中，APP先通过调用模型编译类接口将训练好的AI模型编译成适合在硬件上执行的AI模型文件，再通过调用模型行接口实现模型加载，模型推理，模型销毁等功能。IR模型支持跨平台运行。

场景3：APP在线构建AI模型。该场景适用于APP在线构建AI模型的情况。APP通过调用算子类接口完成算子级联；通过Graph/模型构建类接口完成Graph/模型构建；然后通过模型编译类接口将构建好的模型编译成适合在硬件执行的AI模型文件；最后通过调用模型执行类接口实现模型加载，模型推理，模型销毁等功能。

* 1. 接口分类

本标准涉及的接口作如下分类：

1) 算子：定义实现某种具体运算的单算子，引用AITISA《人工智能 算子接口 第1部分：基础数学类算子》、《人工智能 算子接口 第2部分：神经网络类算子》和《人工智能 算子接口 第3部分：机器学习类算子》；

2) 模型构建接口：定义与模型/Graph构建相关的接口，包括注册类接口、Graph类接口、Model类接口等；

3) 模型编译接口：定义与模型编译相关的接口，包括模型编译接口等；

4) 模型执行接口：定义与模型在硬件平台执行推理功能相关的接口，包括模型加载接口、模型推理接口、模型卸载接口等。

1. 算子

所使用的单算子包括张量计算类算子、常数定义算子、维度置换算子、数学计算算子、神经网络类算子、随机数生成算子和图像处理类算子。具体单算子引用AITISA《人工智能 算子接口 第1部分：基础数学类算子》、《人工智能 算子接口 第2部分：神经网络类算子》和《人工智能 算子接口 第3部分：机器学习类算子》。

该算子部分为模型构建接口中算子相关内容的输入。

1. 模型构建接口
   1. 算子注册类接口
      1. 接口说明

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，如：

REG\_OP(FullyConnection)

.INPUT(x, TensorType::ALL()))

.INPUT(w, TensorType::ALL()))

.OPTIONAL\_INPUT(b, TensorType::ALL())

.OPTIONAL\_INPUT(offset\_w, TensorType::ALL())

.OUTPUT(y, TTensorType::ALL())

.REQUIRED\_ATTR(num\_output, AttrValue::INT)

.ATTR(transpose, AttrValue::BOOL { false })

.ATTR(axis, AttrValue::INT { 1 })

.ATTR(offset\_x, AttrValue::INT { 0 })

.OP\_END()

使用的接口如下：

1. REG\_OP(x)：注册算子类型；
2. ATTR(x, type)：注册算子可选属性，不设置即为注册时的默认值；
3. REQUIRED\_ATTR(x, type)：注册算子必选属性；
4. INPUT (x, t) ：注册算子输入信息；
5. OPTIONAL\_INPUT (x, t) ：注册算子可选输入信息
6. OUTPUT (x, t)：注册算子输出信息；
7. OP\_END ()：结束算子注册。
   * 1. REG\_OP接口

注册算子类型，同时自动生成算子类型的两个构造函数，见表1。

表1 REG\_OP接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子类型名称 | 输入参数 | 算子的类型名称 |

返回值：无。

* + 1. ATTR接口

注册算子可选属性，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，见表2。

表2 ATTR接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子属性名称 | 输入参数 | 注册算子属性的名称 |
| 算子属性类型 | 输入参数 | 算子属性类型需指定默认值。该参数可以由用户设置，也可以在用户不设置时使用默认值。 |

返回值：无。

* + 1. REQUIRED\_ATTR 接口

注册算子必选属性，用户必须配置对应属性，否则算子无法计算，见表3。

表3 REQUIRED\_ATTR接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子属性名称 | 输入参数 | 注册算子属性的名称 |
| 算子属性类型 | 输入参数 | 算子属性类型。该参数可以由用户设置 |

返回值：无

* + 1. INPUT接口

注册算子输入信息，见表4。

表4 INPUT接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子输入名称 | 输入参数 | 算子输入的名称 |
| 算子输入接收的数据类型 | 输入参数 | 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 |

返回值：无。

* + 1. OPTIONAL\_INPUT接口

注册算子可选输入信息，算子计算时输入可选，见表5

表5 OPTIONAL\_INPUT接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子输入名称 | 输入参数 | 算子输入的名称 |
| 算子输入接收的数据类型 | 输入参数 | 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 |

返回值：无

* + 1. DYNAMIC\_INPUT接口

注册动态算子输入信息，见表6。

表6 DYNAMIC\_INPUT接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子输入名称 | 输入参数 | 算子输入的名称 |
| 算子接收的数据类型 | 输入参数 | 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。 |

返回值：无。

* + 1. OUTPUT接口

注册算子输出信息，见表7。

表7 OUTPUT接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子输出名称 | 输入参数 | 算子输入的名称 |
| 算子输出的数据类型 | 输入参数 | 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 |

返回值：无。

* + 1. DYNAMIC\_OUTPUT接口

注册动态算子输出信息，见表8。

表8 DYNAMIC\_OUTPUT接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子输出名称 | 输入参数 | 算子输入的名称 |
| 算子输出的数据类型 | 输入参数 | 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 |

返回值：无。

* + 1. OP\_END接口

结束算子注册，见表9。

表9 OP\_END接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| - | - | - |

返回值：无。

* 1. Graph接口

在完成网络单算子构建之后，要利用Graph类接口进行Graph构建，并设置Graph的输入和输出。使用的接口如下：

1. SetInputs接口：设置Graph内的输入算子；
2. SetOutputs接口：设置Graph关联的输出算子；
3. IsValid接口：判断Graph对象是否有效；
4. AddOp接口：在Graph中增加算子；
5. FindOpByName接口：根据算子的名称，返回Graph中的算子实例；
6. CheckOpByName接口：在Graph中检查指定名称的算子是否存在；
7. GetAllOpName接口：返回Graph中所有算子的名称。
   * 1. SetInputs接口

用于设置Graph内的输入算子的接口，见表10。

表10 SetInputs接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 输入算子集合 | 输入参数 | Graph内的输入算子集合 |

返回值：返回调用者本身。

* + 1. SetOutputs接口

用于设置Graph关联的输出算子的接口，见表11。

表11 SetOutputs接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 输出算子集合 | 输出参数 | 与Graph关联的输出算子 |

返回值：返回调用者本身。

* + 1. IsValid接口

判断Graph对象是否有效，见表12。

表12 IsValid 接口参数列表

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 | 接口示例 |
| - | - | - | bool IsValid() const |

返回值：true，构建Graph对象有效，非空；false，构建Graph对象无效，空指针。

* + 1. AddOp接口

在Graph中增加算子，见表13。

表13 AddOp 接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子 | 输入参数 | 输入需增加的算子 |

返回值：GRAPH\_SUCCESS：操作成功；GRAPH\_FAILED： 操作失败。

* + 1. FindOpByName接口

根据算子的名称，返回Graph中的算子实例，见表14。

表14 FindOpByName 接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子名称 | 输入参数 | 输入需指定的算子名称 |

返回值：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。

* + 1. CheckOpByName接口

根据算子的名称，返回Graph中的算子实例，见表15。

表15 CheckOpByName 接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子名称 | 输入参数 | 输入需指定的算子名称 |

返回值：GRAPH\_SUCCESS： 指定名称的算子存在；GRAPH\_FAILED：指定名称的算子不存在。

* + 1. GetAllOpName 接口

返回Graph中所有算子的名称，见表16。

表16 GetAllOpName接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 算子名称集合 | 输出参数 | Graph中的算子名称集合 |

返回值：GRAPH\_SUCCESS：操作成功；GRAPH\_FAILED： 操作失败。

* 1. Model接口

在完成Graph构建之后，利用Model类接口，完成Model构建，将Graph对象添加到模型中。使用的接口如下：

1. SetName接口：设置模型的名称；
2. GetName接口：获取模型的名称；
3. SetVersion接口：设置模型的版本号；
4. GetVersion接口：获取模型的版本号；
5. GetGraph接口：获取模型中的Graph对象；
6. SetGraph接口：设置模型的Graph对象；
7. Save接口：将模型对象序列化；
8. Load接口：加载序列化数据，反序列化构建模型对象；
9. IsValid接口：判断模型中Graph对象是否有效；
   * 1. SetName接口

设置模型的名称，见表17。

表17 SetName 接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 模型名称 | 输入参数 | 模型名称 |

返回值：无。

* + 1. GetName接口

获取模型的名称，见表18。

表18 GetName 接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| - | - | - |

返回值：无。

* + 1. SetVersion接口

设置模型的版本号，见表19。

表19 SetVersion接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 模型版本号 | 输入参数 | 用于设置模型的版本号 |

返回值：无。

* + 1. GetVersion接口

获取模型的版本号，见表20。

表20 GetVersion接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| - | - | - |

返回值：无。

* + 1. GetGraph接口

获取模型中的Graph对象，见表21。

表21 GetGraph接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| - | - | - |

返回值：无。

* + 1. SetGraph接口

设置模型中的Graph对象，见表22。

表22 SetGraph接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| Graph对象 | 输入参数 | 用于设置Graph对象 |

返回值：无。

* + 1. Save接口

将模型对象序列化，见表23。

表23 Save接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 模型对象数据的地址 | 输出参数 | 用于保存序列化后的模型对象数据 |
| 模型对象数据的大小 | 输出参数 | 用于保存序列化后的模型对象大小 |

返回值：序列化成功，返回GRAPH\_SUCCESS，否则，返回GRAPH\_FAILED。

* + 1. Load接口

加载序列化数据，反序列化构建模型对象，见表24。

表24 Load接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 模型对象数据的地址 | 输入参数 | 输入序列化后模型对象数据 |
| 模型对象数据的大小 | 输入参数 | 输入序列化后模型对象数据长度 |
| 模型对象 | 输出参数 | 输出反序列化后的模型对象 |

返回值：序列化成功，返回GRAPH\_SUCCESS，否则，返回GRAPH\_FAILED。

* + 1. IsValid接口

判断模型中Graph对象是否有效，如果Graph对象为空或者构建Graph失败，则返回无效，见表25。

表25 IsValid接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| - | - | - |

返回值：true表示有效，false表示无效。

1. 模型编译接口
   1. BuildIRModel接口

编译IR模型，输入IR模型对象，输出编译后模型的buffer，见表26。

表26 BuildIRModel接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| IR模型 | 输入参数 | 模型对象 |
| 编译后模型：  Output | 输出参数 | 编译后模型的buffer，可用于文件保存或加载模型 |
| 编译选项 | 输入参数 | 模型编译时的配置项，如精度模式等 |

返回值：true: 模型构建成功；false:模型构建失败。

1. 模型执行接口
   1. Init接口

初始化，创建模型管家，见表27。

表27 Init接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 模型执行监听函数 | 输入参数 | 用于异步推理场景的回调；  函数为空时，则为同步执行。 |

返回值：AIStatus::AI\_SUCCESS：成功，Others：失败。

* 1. Load接口

加载模型，见表28。

表28 Init接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 编译模型描述 | 输入参数 | 编译模型的描述，可以指定从文件加载或从内存加载。 |

返回值：AIStatus::AI\_SUCCESS：成功，Others：失败。

* 1. Process接口

模型推理执行，见表29。

表29 Run接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 上下文参数 | 输入参数 | 模型运行上下文, 须带model\_name字段 |
| 模型输入节点tensor信息集合 | 输入参数 | 模型输入节点tensor信息集合 |
| 模型输出节点tensor信息集合 | 输入参数 | 模型输出节点tensor信息集合 |
| 推理超时时间 | 输入参数 | 推理超时时间(单位：毫秒) |
| 推理时间戳 | 输入参数 | 异步返回标识，基于该标识和模型名称做回调索引 |

返回值：AIStatus::AI\_SUCCESS：成功，Others：失败。

* 1. UnLoad 接口

卸载已加载的模型，见表30。

表30 UnLoad 接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| - | - | - |

返回值：AIStatus::AI\_SUCCESS：成功，Others：失败。

* 1. Cancel接口

取消异步推理任务，见表31。

表31 Cancel 接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| 模型名称 | 输入参数 | 要取消推理的模型名称 |

返回值：无。

* 1. DeInit接口

去初始化，创建模型管家资源，见表32。

表32 DeInit接口参数列表

|  |  |  |
| --- | --- | --- |
| 参数 | 输入/输出类型 | 参数说明 |
| - | - | - |

返回值：无。

1. （资料性附录）  
   接口示例基于C++语言描述
   1. 模型构建接口示例,

A.1.1 算子注册接口示例

注册接口示例，见表A.1。

表A.1 注册接口示例

|  |  |  |
| --- | --- | --- |
| 注册接口 | 接口示例 | 参数说明 |
| REG\_OP接口 | REG\_OP(x) | 输入：x为算子类型名称  输出：无  返回值：无 |
| ATTR接口 | ATTR(x, type) | 输入：  x为算子可选属性名称；  type为算子属性类型。  输出：无  返回值：无 |
| REQUIRED\_ATTR接口 | ATTR(x, type) | 输入：  x为算子必选属性名称；  type为算子属性类型。  输出：无  返回值：无 |
| INPUT接口 | INPUT(x, t) | 输入：  x为算子输入名称；  t为算子输入接收的TensorType。  输出：无  返回值：无 |

表A.1 注册接口示例（续）

|  |  |  |
| --- | --- | --- |
| 注册接口 | 接口示例 | 参数说明 |

|  |  |  |
| --- | --- | --- |
| OPTIONAL\_INPUT接口 | INPUT(x, t) | 输入：  x为算子可选输入名称；  t为算子输入接收的TensorType。  输出：无  返回值：无 |
| DYNAMIC\_INPUT接口 | DYNAMIC\_INPUT(x, t) | 输入：  x为算子动态输入名称；  t为算子输入接收的TensorType。  输出：无  返回值：无 |
| OUTPUT接口 | OUTPUT(x, t) | 输入：  x为算子输出名称；  t为算子输入接收的TensorType。  输出：无  返回值：无 |
| DYNAMIC\_OUTPUT接口 | DYNAMIC\_OUTPUT(x, t) | 输入：  x为算子动态输出名称；  t为算子输入接收的TensorType。  输出：无  返回值：无 |

A.1.2 Graph接口示例

Graph接口示例，见表A.2。

表A.2 Graph接口示例

|  |  |  |
| --- | --- | --- |
| Graph接口 | 接口示例 | 参数说明 |
| SetInputs接口 | Graph& SetInputs(const  std::vector<Operator>& inputs) | 输入：  inputs为输入算子集合。  输出：无  返回值： 调用者本身 |
| SetOutputs接口 | Graph& SetOutputs(const std::vector<Operator>& outputs) | 输入：无  输出：  outputs为输出算子集合。  返回值：调用者本身 |
| IsValid接口 | bool IsValid() const | 输入：无  输出：无  返回值：true，构建Graph对象有效，非空；false，构建Graph对象无效，空指针。 |
| AddOp接口 | graphStatus AddOp(Operator& op) | 输入：op为算子  输出：无  返回值：GRAPH\_SUCCESS：操作成功；GRAPH\_FAILED： 操作失败。 |
| FindOpByName接口 | Operator FindOpByName(const string& name) const | 输入：name为算子名称  输出：无  返回值：  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。 |

表A.2 Graph接口示例（续）

|  |  |  |
| --- | --- | --- |
| Graph接口 | 接口示例 | 参数说明 |
| CheckOpByName接口 | graphStatus CheckOpByName(const string& name) const | 输入：name为算子名称  输出：无  返回值：  返回值：GRAPH\_SUCCESS： 指定名称的算子存在；GRAPH\_FAILED：指定名称的算子不存在。 |
| GetAllOpName接口 | graphStatus GetAllOpName(std::vector<string>& opNames) const | 输入：无  输出：opNames为算子名称集合；  返回值：GRAPH\_SUCCESS：操作成功；GRAPH\_FAILED： 操作失败。 |

A.1.3 Model接口示例

Model接口示例，见表A.3。

**表A.3 Model接口示例**

|  |  |  |
| --- | --- | --- |
| Model接口 | 接口示例 | 参数说明 |
| SetName接口 | void SetName(const string& name) | 输入：name为模型名称  输出：无  返回值：无 |
| GetName接口 | string GetName() const | 输入：无  输出：无  返回值：模型名称 |
| SetVersion接口 | void SetVersion(uint32\_t version) | 输入：version为模型版本号。  输出：无  返回值：无 |

表A.3 Model接口示例（续）

|  |  |  |
| --- | --- | --- |
| Model接口 | 接口示例 | 参数说明 |
| GetVersion接口 | uint32\_t GetVersion() const; | 输入：无  输出：无  返回值：模型版本号 |
| GetGraph接口 | Graph GetGraph() const | 输入：无  输出：无  返回值：返回模型Graph对象 |
| SetGraph接口 | void SetGraph(const Graph& graph) | 输入：graph为Graph对象  输出：无  返回值：无 |
| Save接口 | graphStatus Save(uint8\_t\*\* data, size\_t& len) const | 输入：无  输出：data为序列化后模型对象数据的地址；  len为序列化后模型对象数据的大小  返回值：序列化成功，返回GRAPH\_SUCCESS，否则，返回GRAPH\_FAILED。 |
| Load接口 | static graphStatus Load(const uint8\_t\* data, size\_t len, Model& model) | 输入：  data为序列化后模型对象数据的地址；  len为序列化后模型对象数据的大小。  输出：  model为模型对象。  返回值：反序列化成功，返回GRAPH\_SUCCESS，否则，返回GRAPH\_FAILED。 |

表A.3 Model接口示例（续）

|  |  |  |
| --- | --- | --- |
| Model接口 | 接口示例 | 参数说明 |
| IsValid接口 | bool IsValid() const | 输入：无  输出：无  返回值：true表示有效，false表示无效。 |

* 1. 模型编译接口示例

模型编译接口实例，见表A.4。

**表A.4 模型编译接口示例**

|  |  |  |
| --- | --- | --- |
| 模型编译接口 | 接口示例 | 参数说明 |
| BuildIRModel接口 | bool  BuildIRModel(ge::Model& irModel,  ModelBufferData& output,const BuildOptions& options); | 输入：  irModel为IR模型。  options为编译选项，可扩展，如模型精度等  输出：  output为编译后模型的buffer。  返回值：true: 模型构建成功；false:模型构建失败。 |

* 1. 模型执行接口示例

模型执行接口示例，见表A.5

**表A.5 模型执行接口示例**

|  |  |  |
| --- | --- | --- |
| 模型执行接口 | 接口示例 | 参数说明 |
| Init接口 | AIStatus Init(std::shared\_ptr<AiModelManagerClientListener> listener); | 通知回调接口。  nullptr：同步模型管家。  非nullptr：异步模型管家。 |
| DeInit接口 | void DeInit() | 去初始化，释放资源。 |
| Load接口 | AIStatus Load(std::vector<std::shared\_ptr<AiModelDescription>>& pmodelDesc); | 模型描述信息数组，可输入多个模型，模型名称不能重复。 |
| Process接口 | AIStatus Process(AiContext& context, std::vector<std::shared\_ptr<AiTensor>>& pinputTensor, std::vector<std::shared\_ptr<AiTensor>>& poutputTensor, uint32\_t timeout, int32\_t& piStamp); | context: 模型运行上下文，必须带model\_name字段  pinputTensor: 模型输入节点Tensor信息  poutputTensor:  timeout: 推理超时时间（单位：毫秒）  piStamp: 异步返回标识，基于该标识和模型名称做回调索引 |
| Cancel接口 | void Cancel(const std::string& modelName); | 要取消推理的模型名称 |
| UnLoad接口 | AIStatus UnLoad(); | 卸载模型 |

1. （资料性附录）  
   使用步骤描述
   1. 构造IR模型

a)构造网络单算子：调用Graph类接口，构建单算子，并根据原生模型中算子之间的关系，进行算子级联；

b)构造Graph，设置Graph的输入、输出；

c)构造Model，将Graph对象添加到模型中。

* 1. 编译模型

调用“BuildIRModel”接口，输入步骤1构造的Model对象，输出编译后的模型buffer。

* 1. 加载模型

使用Init初始化，如果进行异步推理，Init需要传入监听函数

创建模型执行器对象，调用“Load”接口，传入步骤2得到的模型buffer或文件及加载选项，加载模型。

* 1. 执行推理

创建输入输出TensorBuffer对象，拷贝用户输入数据到此TensorBuffer对象中；

调用模型执行器的“Process”接口，传入创建的输入输出TensorBuffer对象，执行模型推理。推理执行完成后，推理输出数据保存到了输出TensorBuffer对象中。

* 1. 卸载模型

调用模型执行器对象的“UnLoad”接口，卸载模型。

调用DeInit接口去初始化，卸载模型管家资源。