aicity-sdk使用说明

版本： v0.1.0

日期： 2021/10/14

编者：AI计算组

目录

---

1. 简介

aicity sdk 是一个用于深度学习模型推理部署的API库；支持多种平台，如cpu， gpu， atlas；支持多种模型格式，如mxnet，acl，trt，onnx；具备低代码开发特性，内置常用前后处理和推理算子，在将通用接口集成调通后，用户只需要根据模型特点编写一个简单的配置文件即可支持新模型推理运行。

2. 版本说明和约束

v0.1.0-20211014:

- 调整aicityInit接口json参数，移除ModelPath
- 调整aicityDetectorCreate接口json参数，ModelConfigFile从文件改为带路径文件件
- 其他约束同v0.0.1

v0.0.1-20210917:

- 支持atlas平台的yolov3和yolov5检测算法
- 暂不支持trt和onnx模型
- gpu版本未严格自测

3. 快速上手

以图片多类别检测为例，关键代码步骤如下：

步骤1. 包含头文件

```
#include "aicity.h"
#include "aicity_detect.h"
```

步骤2. sdk初始化

```
//一个进程里面只需要调用一次aicityInit
//配置参数为json字符串
//授权地址配置可用的授权服务器的ip地址和端口号
const char *acCfg = "{\"LicensePath\":\"licenseServerIp:port\"}";
IF_Result ret = aicityInit(acCfg);
```

步骤3. 创建模型句柄

```
//这里创建一个检测模型的句柄
//配置参数为json字符串，
//其中模型配置文件为预先基于模型文件编写的json文件，要带路径,模型需要在相同路径下
//DeviceId为设备编号，ExeMode为执行模式，还有其他可选参数，详见后文接口说明
const char *acCfg_detect =R"({"ModelConfigFile":"path/city-det-acl-b1-
0.0.0.json",
                              "DeviceId":0,
                              "ExeMode":"ATLAS"})";
IF_Handle pHandle;
IF_Result ret = aicityDetectorCreate(&pHandle, acCfg_detect);
```

步骤4. 执行检测

```
//调用检测接口执行检测，batch接口也类似
//配置参数为json字符串，
//其中模型配置文件为预先基于模型文件编写的json文件,文件需要在aicityInit的配置的模型路径下
//DeviceId为设备编号
auto image = getImage();//此处getImage为伪代码
T_AicityDetectResult *ptResult = nullptr;
IF_Result ret = aicityDetect(pHandle, //pHandle是步骤3中创建的句柄
                             IFACEREC_IMG_DATA, //IFACEREC_IMG_DATA或者
IFACEREC_IMG_FILE，指明image类型
                             (void *) &image, //T_IFImageInfo或者char*图片文件名，对应
image type
                             &ptResult, //检测结果
                             NULL); // cfgPtr可以在单句柄多线程时指定某个线程，NULL
```

> 所有检测结果的内存占用由sdk内部管理，T_AicityDetectResult *ptResult不需要在外部开辟内存空间，使用完后也不需要释放。

步骤5. 销毁模型句柄

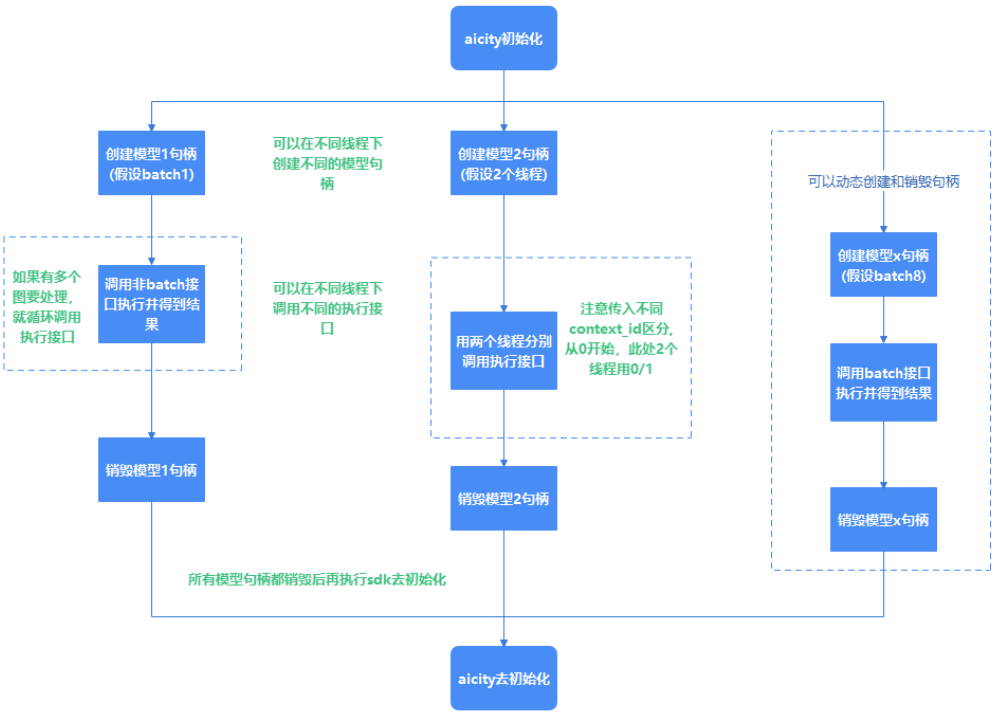
```
//程序退出或不再使用时要销毁句柄，释放资源
IF_Result ret = aicityDetectorDestroy(pHandle);
```

步骤6. sdk去初始化

```
//程序退出时将sdk去初始化，释放资源
IF_Result ret = aicityUnInit();
```

4. 接口调用流程

接口调用流程图



注意事项：

- 创建模型句柄前需要先初始化sdk，
- 单个进程内sdk只能初始化和去初始化一次，
- batch接口单次传入的图像数量需要跟模型实际batch大小一致。
- 模型句柄需要手动销毁
- 所有创建的句柄销毁后再执行sdk去初始化

4. 数据类型

4.1 IF_Result

```
//用于接口返回结果
typedef int IF_Result;
```

成功返回0，即IFACEREC_OK，

失败返回错误码，错误码详情见IFType.h中的E_IFACEREC_ERR中的enum定义和注释

4.2 T_AicityDetectObject

```
//用于描述检测到的单个目标
typedef struct
{
    T_IFRect    tRect; //目标框
    float       fScore; //检测置信度
    int         nType; //目标类别索引，相当于类别数组下标，从0开始
}T_AicityDetectObject;
```

具体的类别名称和数量在模型配置文件中，如：

```
 "class_names": ["person","dog"],"class_num": 2
```

4.3 T_AicityDetectResult

```
//用于描述单个图像的检测结果
typedef struct
{
    int                     nObjNum; //目标数量
    T_AicityDetectObject    *ptObject; //目标数组
}T_AicityDetectResult;
```

4.4 E_IFLogLevel

```
//日志等级
typedef enum
{
    IFLOG_DEBUG = 0,
    IFLOG_INFO,
    IFLOG_WARN,
    IFLOG_ERROR,
    IFLOG_FATAL,
    IFLOG_OFF
}E_IFLogLevel;
```

4.5 IF_Handle

```
//模型句柄，具体对象在sdk内部维护，输出的模型句柄相当于一个索引
typedef int IF_Handle;
```

4.6 E_IFImageType

```
typedef enum
{
    IFACEREC_IMG_DATA       = 0, //图像为数据，当选择此类型时，Image内容为
T_IFImageInfo类型
    IFACEREC_IMG_FILE       = 1, //图像为文件，当选择此类型时，Image内容为文件名字符串
}E_IFImageType;
```

4.7 E_IFImageFormat

```
//图像数据格式
typedef enum
{
    IFACEREC_IMG_GRAY       = 0,
    IFACEREC_IMG_RGB        = 1,
    IFACEREC_IMG_BGR        = 2,
    IFACEREC_IMG_NV12       = 3, //普通摄像头视频流类型
    IFACEREC_IMG_NV21       = 4, //android默认视频流图像类型
    IFACEREC_IMG_YUV420sp   = IFACEREC_IMG_NV12,
    IFACEREC_IMG_RGBA       = 5, //ios默认图像类型
    IFACEREC_IMG_ARGB       = 6, //android默认图像类型
}E_IFImageFormat, E_IFaceRecImageFormat;
```

4.8 T_IFImageInfo

```
//SDK接口接收的数据结构
typedef struct
{
    IF_INT32                nImgId;
    E_IFImageFormat         eFormat;
    int                     nWidth;
    int                     nHeight;
    size_t                  nStep;
    void *                  pData;
    int                     nWidthStride;
    int                     nHeightStride;
}T_IFImageInfo;
```

> eFormat 详情见E_IFImageFormat，目前有如下使用约束：
>
> ATLAS支持IFACEREC_IMG_NV12/IFACEREC_IMG_NV21,
>
> GPU支持IFACEREC_IMG_BGR

5. 接口说明

5.1 aicityGetVersion

获取aicity sdk 版本号

```
IF_Result aicityGetVersion(const char **version);
```

参数名	输入/输出	说明
version	输出	版本号，内存占用由sdk内部管理，不需要在外部开辟内存空间，使用完后也不需要释放

> 该接口不依赖其他接口

5.2 aicityLoggerInit

设置sdk的日志等级和输出方式

```
IF_Result aicityLoggerInit(int nLogFilterLever, aicityLoggerCB pLoggerCB, const char *acCfg = NULL);
```

参数名	输入/输出	说明
nLogFilterLever	输入	日志等级，详情见E_IFLogLevel
pLoggerCB	输入	日志回调函数，aicityLoggerCB，如果不用这种方式请传入NULL 回调函数的形式为 typedef void (*aicityLoggerCB)(int nLogLever, int nMsgLen, const char *acLogMsg);
acCfg	输入	log4cplus的配置文件名，暂未启用，默认为NULL

> 该接口不依赖其他接口

5.3 aicityInit

aicity sdk初始化

```
IF_Result aicityInit(const char *acCfg);
```

参数名	输入/输出	说明
acCfg	输入	初始化sdk的配置参数，json格式的字符串

> sdk初始化在一个进程中只需调用一次

配置参数详情

参数	类型	必须/可选	说明
LicensePath	string	必须	授权地址，授权服务器IP地址:端口号

配置参数示例

```
{
    "LicensePath":"x.x.x.x:xxxx"
}
```

5.4 aicityGetSDKInfo

获取sdk信息

```
IF_Result aicityGetSDKInfo(char **acJson);
```

参数名	输入/输出	说明
acJson	输出	sdk信息，json格式的字符串，内存占用由sdk内部管理，不需要在外部开辟内存空间，使用完后也不需要释放

> 该接口不依赖其他接口

sdk信息详情

参数	类型	必须/可选	说明
SDKVersion	string	可选	sdk版本号
ServiceStatus	string	可选	授权状态

sdk信息示例

```
{
    "SDKVersion":"aicity-V0.0.1",
    "ServiceStatus":"authorization status:(0),authorize success"
}
```

5.5 aicityUnInit

aicity sdk去初始化

```
IF_Result aicityUnInit();
```

> 该接口在一个进程中只需要最后调用一次

5.6 aicityDetectorCreate

创建检测模型句柄

```
IF_Result aicityDetectorCreate(IF_Handle *pHandle, const char *accfg);
```

参数名	输入/输出	说明
pHandle	输出	算法模型句柄，调用一个该接口就会在sdk内部创建一个算法对象，这里返回的句柄IF_Handle相当于对象的一个索引，单个进程内不重复
acCfg	输入	创建检测模型句柄的配置参数，json格式的字符串

> 该接口依赖于aicityInit接口预先完成调用并返回成功

配置参数详情

参数	类型	必须/可选	说明
ModelConfigFile	string	必须	模型配置文件，要带路径，其内配置的模型文件需要在相同路径下
DeviceId	int	必须	设备序号，CPU：-1， GPU或ALTAS设备序号从0开始，一个句柄只能指定一个设备
ExeMode	string	必须	执行模式，根据平台和设备类型从{“CPU”, “GPU”, ”CUDA“, ”TRT“, ”ATLAS“}里选一个
ContextNum	int	可选	上下文数量，即线程数量，单个句柄支持多线程，可以不配置，默认一个句柄一个线程
ConfThresh	float	可选	置信度阈值，用来过滤模型结果，可以不配置，在模型配置文件里面有一个默认值

配置参数示例

```
{
    "ModelConfigFile":"path/xxx.json",
    "DeviceId":x,
    "ExeMode":"xxx",
    "ContextNum":x,
    "ConfThresh":0.xx
}
```

5.7 aicityDetect

执行检测接口

```
IF_Result aicityDetect(IF_Handle hHandle,
                               E_IFImageType eType,
                               void *pImage,
                               T_AicityDetectResult **ptResult,
                               void *cfgPtr = NULL);
```

参数名	输入/输出	说明
hHandle	输入	算法模型句柄，即aicityDetectorCreate创建返回的句柄
eType	输入	E_IFImageType类型，IFACEREC_IMG_DATA表示数据， IFACEREC_IMG_FILE表示文件
pImage	输入	待处理的图像数据 T_IFImageInfo或者文件名，跟eType对应
ptResult	输出	检测结果，T_AicityDetectResult类型，内存占用由sdk内部管理，不需要在外部开辟内存空间， 再次调用该接口时上一次的结果自动释放，如果需要外部继续使用，请及时将结果拷贝保存
cfgPtr	输入	上下文/线程序号，如果aicityDetectorCreate中配置了多线程ContextNum，需要使用该参数指定使用哪个线程， 单线程句柄可以不配置，默认NULL，多线程句柄配置该参数的方法见下面的用法示例。

> 该接口依赖于aicityDetectorCreate接口预先完成调用并得到IF_Handle
>
> aicityDetectorCreate创建的是batch为1的模型句柄，可以用这个接口

cfgPtr参数用法示例

```
int context_id = 0;   //序号从0开始
void *cfgPtr =(void *)(&context_id);
```

5.8 aicityDetectBatch

执行batch检测接口，调用一次处理batch个图像

```
IF_Result aicityDetectBatch(IF_Handle hHandle,
                            int nImgCnt,
                            E_IFImageType eType,
                            void *pImage,
                            T_AicityDetectResult **ptResult,
                            void *cfgPtr = NULL);
```

参数名	输入/输出	说明
hHandle	输入	算法模型句柄，即aicityDetectorCreate创建返回的句柄
nImgCnt	输入	输入图像数量，要求这个数量必须跟模型的batch大小一致， mxnet这种本身不区分batch的模型，图像数量需要跟模型配置文件中配置的batch大小一致。
eType	输入	E_IFImageType类型，batch接口要求必须用IFACEREC_IMG_DATA，不支持文件名形式
pImage	输入	待处理的图像数据 T_IFImageInfo数组地址
ptResult	输出	检测结果，T_AicityDetectResult数组，数组大小跟nImgCnt输入图像数量一致，sdk内部校验保证两者一致， 内存占用由sdk内部管理，不需要在外部开辟内存空间和释放， 再次调用该接口时上一次的结果自动释放，如果需要外部继续使用，请及时将结果拷贝保存
cfgPtr	输入	上下文/线程序号，如果aicityDetectorCreate中配置了多线程ContextNum，需要使用该参数指定使用哪个线程， 单线程句柄可以不配置，默认NULL，多线程句柄配置该参数的方法见下面的用法示例。

> 该接口依赖于aicityDetectorCreate接口预先完成调用并得到[IF_Handle]
>
> aicityDetectorCreate创建的是batch为n的模型句柄，可以用这个接口

5.9 aicityDetectorDestroy

销毁模型句柄

```
IF_Result aicityDetectorDestroy(IF_Handle hHandle);
```

参数名	输入/输出	说明
pHandle	输入	待销毁的算法模型句柄

> 算法模型句柄不再使用时，要手动销毁

————结束