

UNISOC AISDK 用户手册

文档版本

V1.0

发布日期

2020-02-18

Unisoc Confidential For hiar

版权所有 © 紫光展锐科技有限公司。保留一切权利。

本文件所含数据和信息都属于紫光展锐所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负责任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。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。

Unisoc Confidential For hiar

紫光展锐科技有限公司

版本历史

Version	Date	Owner	Notes
1.0	2020-02-18	UNISOC	输出 1.0 版本

Unisoc Confidential For hiar

Unisoc Confidential For hiar

前言

概述

本文档主要说明 UNISOC AISDK（以下简称 AISDK）的 API 接口定义及设计框架，阐述 AISDK 对应用输入的神经网络模型和数据的处理流程，旨在指导用户如何使用 AISDK；将就以下几个方面展开描述：AISDK 的基本结构、AISDK 的编程示例、AISDK 的特殊数据类型、AISDK 的接口描述。

读者对象

本文档主要适用于 AISDK 开发人员&AISDK API 使用人员，需要相关人员具备以下基础：


- 熟悉 C/C++接口的调用规则。
- 知悉神经网络模型输入输出数据的类型和大小。

缩略语

缩略语	英文全名	中文解释
AI	Artificial Intelligence	人工智能
NPU	Neural Processing Unit	编号

符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
 说明	用于突出重要/关键信息、补充信息和小窍门等。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害。

目录

1 简介	1
1.1 概述	1
1.2 AISDK 软件栈	1
1.3 AISDK 基本功能	2
1.4 AISDK workflow	2
2 编程模型及示例	3
2.1 编程模型	3
2.2 AISDK 部署示例	4
2.2.1 AISDK 在 Android 上部署	4
2.2.2 AISDK 在 Ubuntu 上部署	4
3 数据类型	5
3.1 AIRet	5
3.2 DataType	8
3.3 DataFormat	8
3.4 NpuPref	9
3.5 Mix model configfile	10
4 AISDK API 描述	12
4.1 CreateModelManager - 模型管理器创建	12
4.2 InitDataFormat - Dataformat 初始化	12
4.3 LoadModel - 模型加载	13
4.3.1 LoadModel	13
4.3.2 LoadMixModel	13
4.4 RunModel - 模型运行	14
4.4.1 RunModel - with file	14
4.4.2 RunModel - with buffer	14
4.4.3 RunMixModel	15
4.5 DestroyModelManager - 释放模型管理器	15
4.6 SetLogLevel - 设置 Log 等级	16
4.7 GetSDKVersion - 获取 AISDK 版本号	16
4.8 GetSDKVersionLength - 获取 AISDK 版本号长度	17

1 简介

1.1 概述

由于离线模型（即使用 mapping 工具对 caffe/TensorFlow 等模型做转换，生成的适配 NPU 的模型）处理的接口和 TensorFlow 的接口皆有不同的实现，对于应用来说，没有统一的接口将很难适配。AISDK 通过抽象 TensorFlow 和硬件加速器的接口，提供统一的 API 给应用使用，使得应用可以不再考虑底层实现，通过调用 AISDK 的 API，直接做推理，省略了中间调用过程。让开发人员更便捷更高效的编写人工智能应用程序。

1.2 AISDK 软件栈

AISDK 的软件栈如图 1-1 所示。

对于 Native 的应用，可以直接调用 AISDK 的 API 来进行软件编程。

对于 Java 应用，可以通过 JNI（需用户自行开发）来实现调用 AISDK 的 API 来进行软件编程。

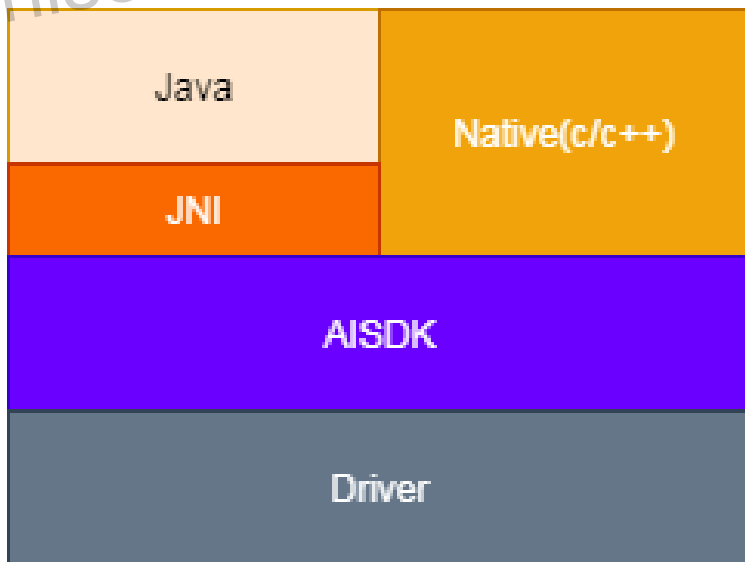


图 1-1 AISDK 的软件栈

1.3 AISDK 基本功能

AISDK 主要包含以下功能：

- 流程控制：管理内部各模块交互，实现整个推理过程的控制；
- 模型检测/加载：PB/离线网络模型识别并加载；
- Fallback 处理：当有 NPU 不支持的算子时可 fallback 到 CPU 处理。
- 设备管理：获取 NPU 设备来执行模型；
- Memory 管理：负责为模型及 input 分配/释放内存；
- 提供 debug 接口：支持多种 log 等级设定及对应等级的 log 函数；

1.4 AISDK workflow

AISDK 的 workflow 大致可以分为四个阶段，具体每个阶段需要调用哪些 API，可以参考 2.2 的示例程序。

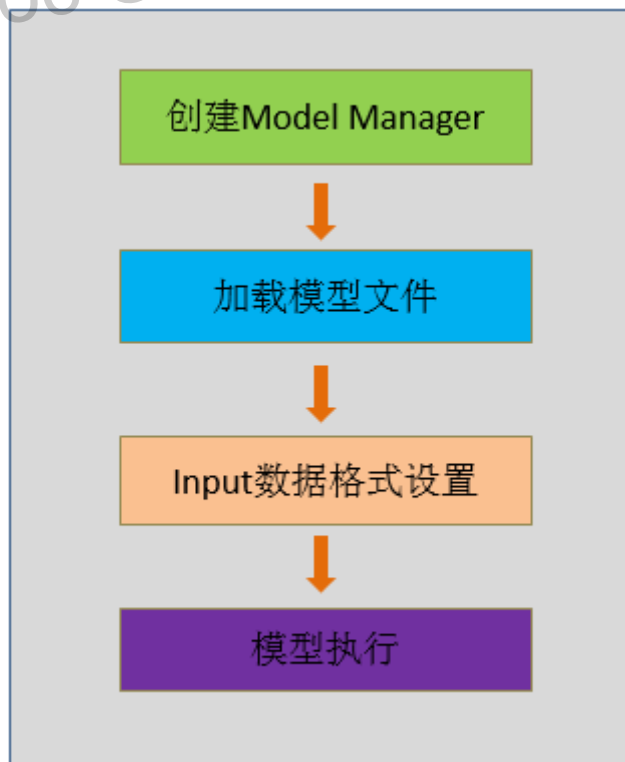


图 1.2 AISDK 的 workflow

2 编程模型及示例

2.1 编程模型

AISDK 当前支持的是单核编程，在模型运行阶段，系统会启动 NPU/CPU 来进行计算，对于单个用户来说处理过程是阻塞的。

AISDK 可以支持多用户，用户本身不用关注 AISDK 是如何处理其它应用的请求的。在同样的优先级设定下，它们之间遵循队列的“先进先出”的规则。

图 2-1 展示了多个应用在通过 AISDK 获取 NPU driver 的过程，不同颜色的箭头代表不同应用的数据发送/接收，它们在时间线上遵循先后顺序。

图 2-1 中 AISDK 是一个整体，实际在运行过程中，AISDK 运行于应用自身进程空间内。

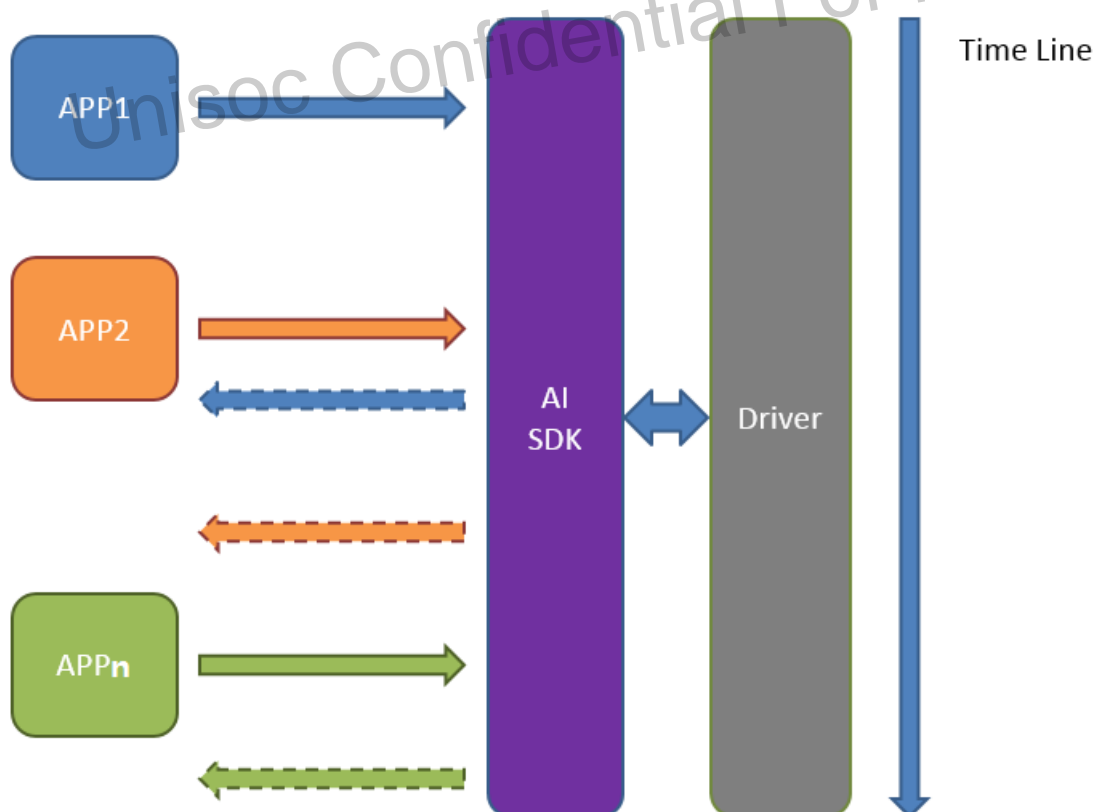


图 2-1 多个应用在通过 AISDK 获取 NPU driver 的过程

2.2 AISDK 部署示例

2.2.1 AISDK 在 Android 上部署

应用在 Android 平台上使用 AISDK 时，可以按照下面的方式来写 Android.mk：

```
1. include $(CLEAR_VARS)
2. LOCAL_MODULE := aisdk_demo
3. LOCAL_MODULE_TAGS := optional
4. LOCAL_MODULE_CLASS := EXECUTABLES
5. LOCAL_MODULE_PATH := $(TARGET_OUT_VENDOR)/bin/hw
6.
7. LOCAL_SRC_FILES := sample/main.cpp
8.
9. LOCAL_CPPFLAGS := -std=c++11 -fexceptions
10. LOCAL_MULTILIB := 64 #必须加上这一条
11.
12. LOCAL_C_INCLUDES := $(LOCAL_PATH)/sample #包含 libaisdk 头文件
13. LOCAL_SHARED_LIBRARIES := libaisdk2.0 liblog #添加 libaisdk
14. LOCAL_PROPRIETARY_MODULE := true
15. include $(BUILD_EXECUTABLE)
```

需要注意的是，需要将 aisdk 的头文件加入到\$(LOCAL_PATH)/sample 中，路径可自己指定。此外 aisdk 库只有 64bit，所以 demo 也只能是支持 64bit 的

如果应用使用 Android.bp 来编译自己的模块，则可以参考 AISDK 中 sample 代码中包含的 Android.bp，这里就不展开了。

2.2.2 AISDK 在 Ubuntu 上部署

aisdk 同样支持在 ubuntu 运行（需要使用专门为 ubuntu 适配的 aisdk 库，库名为 libUNIAI.so），建议使用 cmake 来构建编译，具体示例如下：

```
1. #1.cmake version
2. cmake_minimum_required(VERSION 3.4.8)
3.
4. #2.project name
5. project(Label_image C CXX)
6.
7. include_directories(
8.     include/
9.     include/tclap/
10. )
11.
12. # set output dir
13. set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BINARY_DIR}/bin)
14. message(${CMAKE_CURRENT_SOURCE_DIR})
15.
16. set(LINK_DIR ${CMAKE_CURRENT_SOURCE_DIR}/lib/arm64/)
17. link_directories(${LINK_DIR})
18. find_package(OpenCV 3.2.0 REQUIRED)
19. add_executable(Label_image label_image.cpp)
```

```

20. SET(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -nostdlib")
21. SET(CMAKE_SHARED_LINKER_FLAGS "-Wl,--allow-shlib-undefined")
22. set_target_properties(${TARGET}
23.     PROPERTIES
24.     COMPILE_FLAGS "-Wall")
25. set_target_properties(${TARGET}
26.     PROPERTIES
27.     C_STANDARD 99 CXX_STANDARD 11)
28. add_definitions(-std=c++11)
29.
30. add_definitions(-D__LINUX__)
31. TARGET_LINK_LIBRARIES(Label_image UNIAI)
32.
33. TARGET_LINK_LIBRARIES(${PROJECT_NAME} dl)
34. TARGET_LINK_LIBRARIES(${PROJECT_NAME} pthread)

```

3 数据类型

3.1 AIRet

枚举类型，用于返回 Function 状态，用户可以根据返回值判断当前调用出现了那种类型的异常。

Enum value	Enum word	Enum Decription
-1	AI_FAILED	return fail when error
0	AI_SUCCESS	All operations completed successfully
1	AI_INVALID_PARAMS	Invalid parameters were provided
2	AI_TIMEOUT	A timeout occurred while executing the network
3	AI_OUT_OF_MEMORY	Out of memory occurred while allocating the memory or creating the session
4	AI_ACCES_ERROR	Read model or input file permission denied
5	AI_INIT_ERROR	Uninitialized

6	AI_NO_DEVICE	No devices were found when creating the session
7	AI_BUSY_ERROR	Device or resource busy
8	AI_SESSION_ERROR	Failure on create session
9	AI_INIT_DATAFORMAT_ERROR	Init DataFormat Error
10	AI_INIT_DATAFORMAT_SUCESS	Init DataFormat sucess
11	AI_INVALID_MODEL	Invalid model which NPU couldn't
12	AI_READ_MODEL_ERROR	A Read error while reading the model
13	AI_MODEL_PARSE_ERROR	A generic error occurred while parsing the model file
14	AI_HOST_OPS_ERROR	An operation on the host layer returned an error
15	AI_UNSUPPORTED_OP_ERROR	An unsupported operation was detected when parsing the model
16	AI_INVALID_CONFIG_FILE	Invalid config file for mix model
17	AI_LOAD_MODEL_ERROR	load model error
18	AI_LOAD_MIX_MODEL_ERROR	load mix model error
19	AI_NO_EXIST_ERROR	Resource or file doesn't exist
20	AI_INVALID_SIZE_ERROR	Input buffer or file size was invalid
21	AI_INBUF_POINTER_ERROR	Input buffer was invalid
22	AI_INVALID_TYPE_ERROR	Input an invalid type

23	AI_STREAM_ERROR	Failure on Stream
24	AI_OUTPUT_ERROR	Couldn't get output data
25	AI_UNEXPECTED_STATE_ERROR	Internal runtime logic detected an unexpected state
26	AI_BROKEN_ERROR	Data transmission is broken
27	AI_FUNC_CALL_ERROR	Failure to call runtime functions
28	AI_UNHANDLED_ERROR	Unhandled error
29	AI_EVENT_ERROR	Failure on event operation
30	AI_RESHAPE_ERROR	Failure on data reshape
31	AI_INVALID_DATADESC_ERROR	Invalid data descriptor
32	AI_RUN_MODEL_ERROR	Failure on run model
33	AI_RUN_MIX_MODEL_ERROR	Failure on run mix model
34	AI_INIT_XMLPARSER_ERROR	new xmlparser error
35	AI_INIT_TENSORFLOWIMPL_ERROR	new tensorflow impl error
36	AI_SET_CACHE_ERROR	set cache error
37	AI_SET_QUEUE_SIZE_ERROR	set queue size error
38	AI_NULLPTR_ERROR	nullptr error
39	AI_BAD_DIM_SIZE	bad dim size
40	AI_BAD_NPU_VERSION	bad NPU version

41	AI_MALLOC_FAILED	Malloc failed
42	AI_UNSUPPORTED_FUNCTION	unsupport function
43	AI_UNKNOWN_ERROR	Unknown error
44	AI_RET_MAX	The last one

3.2 DataType

定义数据类型的枚举值。

Enum value	Enum word	Enum Decription
0	AISDK_NONE	No specific meaning, just the starting value of the data type definition
1	AISDK_Q8A	Input data type is quant uint8
2	AISDK_FLOAT32	Input data type is float32
3	AISDK_ERROR	Error input data type

3.3 DataFormat

input/output 的数据信息，包括数据类型，是否是 unpacked 类型的数据（所谓 unpacked 指的是 CPU 使用的数据存储格式，而 packed 指的是 NPU 使用的数据存储格式，通常使用 unpacked 就可以了），input/output 的 shape 等。同时这个数据结构提供了两个构造函数，一个只需要指定 input/output 数据类型即可；另一个还需要指定是否是 unpacked input/output。对于 input/output 的 shape，在初始化了 DataFormat 的对象之后，还需要给它们赋值，这要求用户必须清楚模型的相关信息，若是 Pb 模型，则必须初始化模型的输入/输出节点名字，但赋值前注意需要先给 input_nodes/output_nodes 分配空间，具体如何操作可以查看 sample 路径下的示例代码。

PS:若不清楚节点信息时，可通过模型解析工具进行查看，如 Netron 等。

```
1. typedef struct {
2.     char *node_name;
3.     unsigned int node_dim_size;
4.     unsigned int node_shape[4];
5. } NodeShape;
6.
7. typedef struct {
8.     // is_nchw:
9.     //     input shape is NCHW set true;
10.    //     input shape is NHWC set false;
11.    bool is_nchw;
12.    // need set both of input and output type, model input type maybe
13.    // different with output type, like SSD mobilenet v1/v2.
14.    DataType input_type;
15.    DataType output_type;
16.    // input data has packed for NNA or not, true is unpacked
17.    bool unpacked_in;
18.    // output data has packed by NNA or not, true is unpacked
19.    bool unpacked_out;
20.    // how many input nodes of the model
21.    unsigned int input_node_count;
22.    // the input nodes of the model
23.    NodeShape **input_nodes;
24.    // how many output nodes of the model
25.    unsigned int output_node_count;
26.    // the output nodes of the model
27.    NodeShape **output_nodes;
28. } DataFormat;
```

Unisoc Confidential For hiar

3.4 NpuPref

NPU 工作模式的枚举值，共设定了三种模式，分别是高优先级模式/常规优先级模式/默认优先级模式。如果不知道选用什么模式，直接选用 DEFAULT_PERF 即可。

Enum value	Enum word	Enum Decription
0	HIGH_PERF	High priority mode.
1	NORMAL_PERF	Normal priority mode.
2	DEFAULT_PERF	Default priority mode.

3.5 Mix model configfile

当运行混合模型时，AISDK 需要接收混合模型对应的 xml 配置文件即 mix model configfile，参考文件如下：

<Root>

<segments>

<segment>

<modelpath>./actual_mix.mbs</modelpath>

<isoffline>>true</isoffline>

<inputtype>float32</inputtype>

<outputtype>float32</outputtype>

<innodes>

<innode>

<nodename>input</nodename>

<inputdim>4</inputdim>

<inshapes>1,224,224,3</inshapes>

</innode>

</innodes>

<outnodes>

<outnode>

<nodename>mnasnet_1/lead_cell_17/op_0/Relu</nodename>

<outputdim>4</outputdim>

<outshapes>1,7,7,1280</outshapes>

</outnode>

</outnodes>

</segment>

<segment>

<modelpath>./aisdk_st_res/mix_models/actual_mix.pb</modelpath>

<isoffline>>false</isoffline>

<inputtype>float32</inputtype>

<outputtype>float32</outputtype>

<innodes>

<innode>

<nodename>mnasnet_1/lead_cell_17/op_0/Relu</nodename>


```

        <inputdim>4</inputdim>
        <inshapes>1,7,7,1280</inshapes>
    </innode>
</innodes>
<outnodes>
    <outnode>
        <nodename>final_result</nodename>
        <outputdim>2</outputdim>
        <outshapes>1,54</outshapes>
    </outnode>
</outnodes>
</segment>
</segments>
</Root>

```

以下是其节点说明:

XML Node	Xml node Decription
modelpath	The path of the model run
isoffline	Is the model is offline, true or false
inputtype	The model's input type, float32 or q8a
outputtype	The model's input type, float32 or q8a
innodes	The input nodes of model
innode	The input node of model
nodename	The input/output node name
inputdim	The input dims of model
inshapes	The input shapes of model
outnodes	The output nodes of model

outnode	The output node of model
outputdim	The output dims model
outshapes	The output shapes of model

4 AISDK API 描述

4.1 CreateModelManager – 模型管理器创建

创建模型管理对象实例后才可以使用模型接口，当前支持一个进程创建一个模型管理实例：

功能描述	创建模型管理实例
接口原型	ModelMgr *CreateModelManager();
参数说明	NA
返回值	Return ModelMgr pointer

4.2 InitDataFormat – Dataformat 初始化

功能描述	初始化Dataformat数据结构
接口原型	int InitDataFormat(DataFormat *dataformat);
参数说明	dataformat: DataFormat variable. @DataFormat define
返回值	run success will return 0, run fail will return error code

4.3 LoadModel – 模型加载

模型加载接口分为加载 PB/离线模型和分段模型两种，根据给定路径下的模型文件调用模型加载的接口实现模型加载。

4.3.1 LoadModel

功能描述	从路径中加载PB/离线模型
接口原型	<code>int LoadModel(ModelMgr *modelManager, const char *modelfile, NpuPref perf);</code>
参数说明	modelManager: pointer to ModelManager. modelfile: pointer to model file path. perf: set the mode of running model.
返回值	load success will return 0, load fail will return error code

4.3.2 LoadMixModel

功能描述	从路径中加载分段模型
接口原型	<code>int LoadMixModel(ModelMgr *modelManager, const char *configfile, NpuPref perf);</code>
参数说明	modelManager: pointer to ModelManager. modelfile: pointer to mix model file path. @3.5 Mix model configfile perf: set the mode of running model.
返回值	load success will return 0, load fail will return error code

4.4 RunModel - 模型运行

模型运行时，需要通过参数提供 input 数据（buffer 或文件）/数据的特征信息/结果输出接口（buffer 或文件）等信息。

4.4.1 RunModel - with file

功能描述	模型运行接口（通过文件导入input,目前仅imgtect模式下可用）
接口原型	<pre>int RunModel(ModelMgr *modelManager, const char *infiles[], int filecount, DataFormat *dataformat, const char *outputpath, const int timeout);</pre>
参数说明	<p>modelManager: pointer to ModelManager.</p> <p>infiles: input files.</p> <p>filecount: infiles size</p> <p>dataformat: input/output format, see @DataFormat define</p> <p>outputpath: the output file save path.</p> <p>timeout: set time out for inference.</p>
返回值	run success will return 0, run fail will return error code

4.4.2 RunModel - with buffer

功能描述	模型运行接口（通过buffer导入input）
接口原型	<pre>int RunModel(ModelMgr *modelManager, void *inputbufs[], int inputcount, DataFormat *dataformat, void *outputbufs[], int ouputcount, const int timeout);</pre>
参数说明	<p>modelManager: pointer to ModelManager.</p> <p>inputbufs: input buffer.</p> <p>inputcount: inputbufs size</p>

	<p>dataformat: input/output format, see @DataFormat define</p> <p>outputbufs: the output buffer.</p> <p>ouputcount: outputbufs size</p> <p>timeout: set time out for inference.</p>
返回值	run success will return 0, run fail will return error code

4.4.3 RunMixModel

功能描述	分段模型运行接口（通过buffer导入input）
接口原型	<pre>int RunMixModel(ModelMgr *modelManager, void *inputbufs[], int inputcount, void *outputbufs[], int ouputcount, const int timeout);</pre>
参数说明	<p>modelManager: pointer to ModelManager.</p> <p>inputbufs: input buffer.</p> <p>inputcount: inputbufs size</p> <p>outputbufs: the output buffer.</p> <p>ouputcount: outputbufs size</p> <p>timeout: set time out for inference.</p>
返回值	run success will return 0, run fail will return error code

4.5 DestroyModelManager – 释放模型管理器

功能描述	释放模型管理器实例
接口原型	<pre>void DestroyModelManager(ModelMgr *modelManager);</pre>

参数说明	modelManager: pointer to ModelManager.
返回值	NA

4.6 SetLogLevel – 设置 Log 等级

功能描述	设置Log等级
接口原型	void SetLogLevel(LogLevel loglevel);
参数说明	loglevel: the level of the log which will be set .
返回值	NA

4.7 GetSDKVersion – 获取 AISDK 版本号

功能描述	获取AISDK版本号
接口原型	void GetSDKVersion(char *sdkver);
参数说明	<p>sdkver: pointer to sdkver.</p> <p>note: the sdkver must have been allocated,and the length is longer than @SDKVER_LEN</p>
返回值	NA

4.8 GetSDKVersionLength – 获取 AISDK 版本号长度

功能描述	获取AISDK版本号长度
接口原型	unsigned int GetSDKVersionLength();
参数说明	NA
返回值	Return the sdk version length

Unisoc Confidential For hiar