AISDK demo 流程及使用说明

Release Date	2019-7-31
Version	V1.0
Document Type	User Guide
Platform	UDS710
OS Version	Android9.0

声明 Statement

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。

All data and information contained in or disclosed by this document is confidential and proprietary information of UNISOC and all rights therein are expressly reserved. This document is provided for reference purpose, no license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, and no express and implied warranties, including butwithout limitation, the implied warranties of fitness for any particular purpose, and non-infringement, as well as any performance. By accepting this material, the recipient agrees that the material and the information contained therein is to be held in confidence and in trust and will not be used, copied, reproduced in whole or in part, nor its contents revealed in any manner to others without the express written permission of UNISOC. UNISOC may make any changes at any time without prior notice. Although every reasonable effort is made to present current and accurate information, UNISOC makes no guarantees of any kind with respect to the matters addressed in this document. In no event shall UNISOC be responsible or liable, directly or indirectly, for any damage or loss caused or alleged to be caused by or in connection with the use of or reliance on any such content.

关键字 Keywords

AISDK 分段 demo

版本历史 Revision history

版本 Version	日期 Date	作者 Author	描述 Description
1.0	2019-7-31	UNISOC	根据内部文档修改，适用于外发要求。

前　言 Foreword

一 范围 Scope

本文档主要用来记录 UNISOC AI SDK 设计框架及 API 说明，阐述 SDK 对应用输入的神经网络模型和数据的处理流程。本文档的预期读者为系统设计人员，软件开发人员、客户支持人员。

二 内容定义 Details Definitions

N/A。

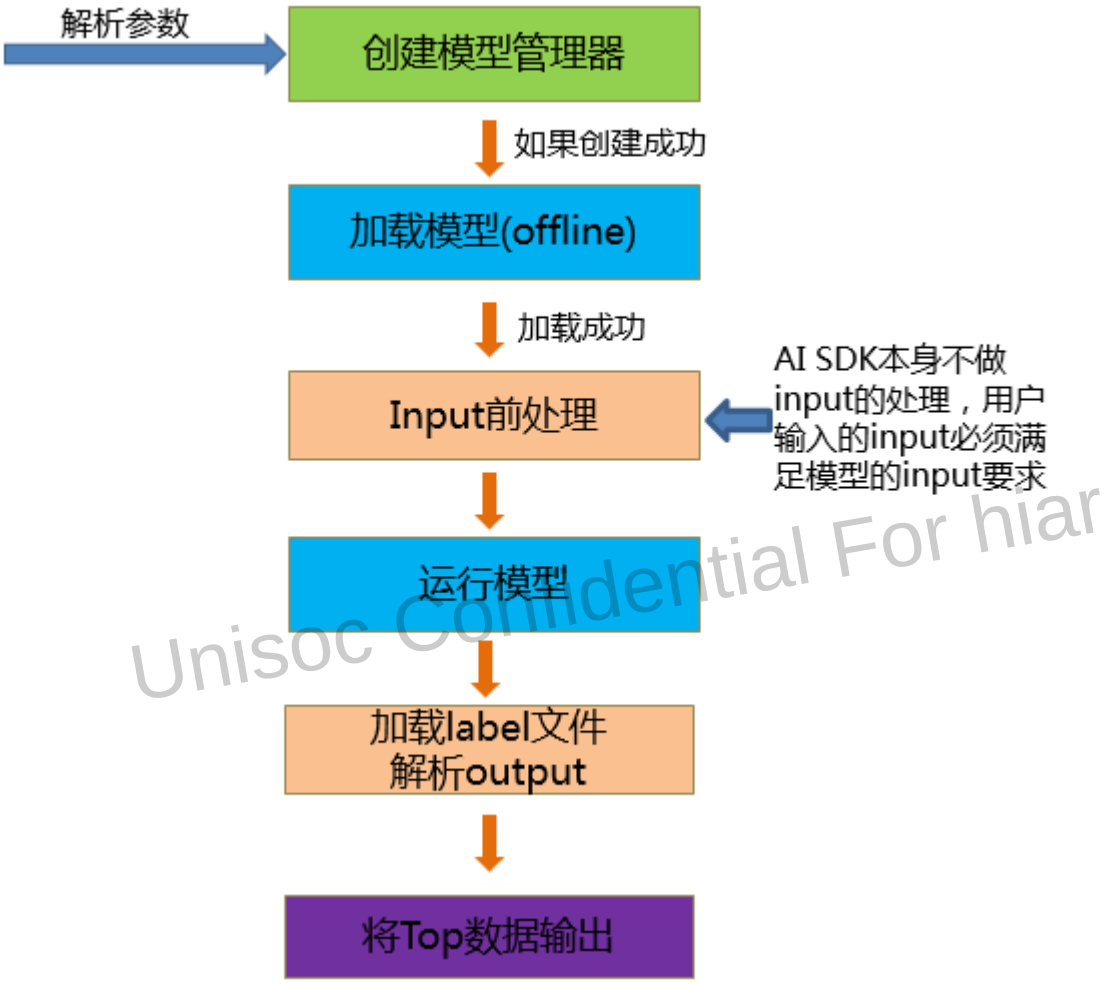
三 参考文献　References

N/A。

目　录 Contents

1 AISDK 概览

AI SDK demo（下文简称 demo）是基于 AI SDK API 开发的一个示例程序，用于向开发者演示 AI SDK 的使用流程。下面就 demo 的流程和使用分别来说明。Ps：文档中所示的 demo 流程和代码，基于 AI SDK V2 开发。Demo 的流程如下图:



2 Demo 流程

demo 演示的是图片分类的流程，如果是其它模型，去掉“加载 label 文件解析 output”及之后的步骤，更改为自己需要的处理即可。

备注：参数解析使用的是 tclap，这一个开源库，它可以将程序运行时输入的参数做解析，如有疑问自行查找资料。

2.1 代码流程

```
1.  int main(int argc, char* argv[]) {
2.    // parse cmdline
3.    cli.Parse(argc, argv);//解析输入的参数，包括输入数据的格式，模型文件等。
4.
5.    // 1. create model manager
6.    ModelMgr* modelManager = CreateModelManager();
7.    if (modelManager == NULL) {
8.      std::cout << "UNISOC_AISDK odelManager is NULL " << std::endl;
9.      return -1;
10.   }
11.
12.   // 2. load model  加载模型
13.   int ret = LoadModel(modelManager, cli.modelfile().c_str(), HIGH_PERF);
14.   if (ret != AI_SUCCESS) {
15.     std::cout << "UNISOC_AISDK load model fail " << std::endl;
16.     return ret;
17.   }
18.
19.   // 3. input image files/buffers;  初始化输入输出数据的信息
20.   DataFormat dataformat;
21.   InitDataFormat(&dataformat);
22.   dataformat.model_shape.input_node_dim_size = 4;
23.   dataformat.model_shape.output_node_dim_size = 2;
24.   // the  input_node_name && output_node_name only for mobilenet v1 tensorflow
25.   char input_node_name[] = "input";
26.   char output_node_name[] = "MobilenetV1/Predictions/Reshape_1";
27.   dataformat.model_shape.input_node_name = input_node_name;
28.   dataformat.model_shape.output_node_name = output_node_name;
29.   // input image buffers
30.   unsigned int inbuf_size = 1;
31.   unsigned int outbuf_size = 1;
32.   int input_count = cli.infiles().size();
```

```
33.   // if the model's input node is same as output node, set output count as input
34.   // count. Otherwise, give the valid output size.
35.   int output_count = input_count;
36.   void* input_buffer[input_count];
37.   void* output_buffer[output_count];
38.   if (!cli.usebuffer()) {
39.     std::cout << "UNISOC_AISDK input files, size: " << input_count << std::endl;
40.   } else {
41.     int count = 0;
42.     for (auto file : cli.infiles()) {
43.         std::ifstream is(file.c_str(), std::ios_base::binary);
44.         if (!is.is_open()) {
45.           return AI_NO_EXIST_ERROR;
46.         }
47.         for (unsigned i = 0; i < dataformat.model_shape.input_node_dim_size;
48.              ++i) {
49.           dataformat.model_shape.input_node_shape[i] = cli.inshape[i];
50.           inbuf_size *= dataformat.model_shape.input_node_shape[i];
51.         }
52.         for (unsigned i = 0; i < dataformat.model_shape.output_node_dim_size;
53.              ++i) {
54.           dataformat.model_shape.output_node_shape[i] = cli.outshape[i];
55.           outbuf_size *= dataformat.model_shape.output_node_shape[i];
56.         }
57.         if (dataformat.input_type == AISDK_FLOAT32) {
58.           inbuf_size *= sizeof(float);
59.           outbuf_size *= sizeof(float);
60.         }
61. #ifdef USE_PREPARED_INPUT
62.         //读取预处理好的图片到 buffer
63.         is.seekg(0, is.end);
64.         unsigned int file_len = is.tellg();
65.         if (file_len < 0U) {
66.           std::cout << "UNISOC_AISDK invalid file size:" << file_len  << std::endl;
67.           return -1;
68.         }
69.         is.seekg(0, is.beg);
70.         if (inbuf_size != file_len) {
71.           return -1;
72.         }
73. #endif
74.         input_buffer[count] = reinterpret_cast<void*>(malloc(inbuf_size));
75.         memset(input_buffer[count], 0, inbuf_size);
```

```
76.         output_buffer[count] = reinterpret_cast<void*>(malloc(outbuf_size));
77.         memset(output_buffer[count], 0, outbuf_size);
78. #ifdef USE_PREPARED_INPUT
79.         is.read(reinterpret_cast<char*>(input_buffer[count]), file_len);
80.         is.close();
81. #else
82.         return -1;
83. #endif
84.         count++;
85.       }
86.     }
87.   }  // end if
88.   // 4. run model
89.   if (!cli.usebuffer()) {
90.     // to do use buffer
91.   } else {
92.       RunModel(modelManager, input_buffer, input_count, &dataformat,
93.                output_buffer, output_count, 1000);
94.   }
95.
96.   // 5. parse output, print top-n result
97.   if (!cli.usebuffer()) {
98.     // to do use buffer
99.   } else {
100.     for (int i = 0; i < output_count; i++) {
101.       if (dataformat.input_type == AISDK_Q8A) {
102.         GetTopnResult<uint8_t>(reinterpret_cast<uint8_t*>(output_buffer[i]),
103.                                outbuf_size, false);
104.       } else {
105.         GetTopnResult<float>(reinterpret_cast<float*>(output_buffer[i]),
106.                              outbuf_size / sizeof(float), true);
107.       }
108.     }
109.     for (int i = 0; i < input_count; i++) {
110.       free(input_buffer[i]);
111.     }
112.     for (int i = 0; i < output_count; i++) {
113.       free(output_buffer[i]);
114.     }
115.   }
116.   DestroyModelManager(modelManager);
117.   return 0;
118. }
```

2.2 demo 使用说明

Demo 运行的时候需要输入参数，其中必须输入的参数包括离线模型路径，label 文件，input 文件，更多参数可以输入./aisdk_demo –h 查询。

USAGE:

aisdk_demo [--unpackedout] [--unpackedin] [-b] [-t <unsigned integer>] [--outshape <n,c,h,w>] [--inshape <n,c,h,w>] -l <std::string> [--outputformat <std::string>] [--inputformat <std::string>] [-f <std::string>] [-o <std::string>] -m <std::string> [--] [--version] [-h] <input image file(s)> ...

Where:

--unpackedout //output 结果是否为 NPU 格式，通常不用设置，默认设置为 false

output unpacked

--unpackedin // input 是否为 NPU 格式，通常不用设置，默认设置为 false

input unpacked

-b, --usebuffer //demo 与 AI SDK 之间以 buffer 的形式传递 input/output 数据，默认设置为 true；设置为 false 则以文件形式，input 需传递文件路径，output 结果也会写入到给定路径的文件中。

input/output use buffer

-t <unsigned integer>, --timeout <unsigned integer>//模型运行的超时设置，默认 1000ms

timeout value

--outshape <n,c,h,w> //模型 output 层的 shape，请按照模型实际情况填写，不一定是 NCHW。

output tensor shape

--inshape <n,c,h,w> //模型 input 层的 shape ,请按照模型实际情况填写 ,不一定是 NCHW。

input tensor shape

-l <std::string>, --labelfile <std::string>//图片分类的 label 文件路径

(required) label file path

--outputformat <std::string>//output 数据的类型，float32 or Q8A（u_int8）

output buffer/file format

--inputformat <std::string>//input 数据的类型，float32 or Q8A（u_int8）

input buffer/file format

-f <std::string>, --fileorder <std::string>//如果是已经预处理好的图片（在 PC 端通过 python 预处理过的 raw 图片），需给出图片数据存储顺序

input file order, rgb or bgr

-o <std::string>, --outputpath <std::string>//将 output 数据写入到文件中时，文件存放路径

output file path

-m <std::string>, --modelfile <std::string>//离线模型存放路径

(required) model file path

2.2.1 demo 执行示例

下面分别以 googlenet 和 inception v3 的离线模型执行命令来说明 demo 如何使用。在执行之前，需要将用到的文件 push 到手机上（文件会随文档一并发布）。先将 libaisdk64.so push 到 vendor/lib64，其它文件 push 到/data/aisdk_test。

首先是 Googlenet 识别一张哈士奇的图片，这里可以看到只提供了最基本的必选参数。

./aisdk_demo –m /data/aisdk_test/googlenet.mbs.bin -l /data/aisdk_test/imagenet_labels.txt /data/aisdk_test /dog_nchw.f32

inshape: 1,3,224,224

outshape: 1,1,1000,1

========== Execute Result ==========

[Confidence] [Index] [Label]

0.478616 250 Siberian husky

0.441568 248 Eskimo dog, husky

0.076883 249 malamute, malemute, Alaskan malamute

0.001276 537 dogsled, dog sled, dog sleigh

继续使用 demo 来推理一张猫的图片，这里就必须提供 inshape 参数了，因为默认的是 1，3，224，224，如果使用默认参数就会出错。

./aisdk_demo -m /data/aisdk/i3_mapped_csim_16-16-16.mbs.bin -l /data/aisdk/inception_v3_labels.txt --inshape 1,3,299,299 /data/aisdk/299_nchw_std128.raw

inshape: 1,3,299,299

outshape: 1,1,1000,1

========== Execute Result ==========

[Confidence] [Index] [Label]

0.711061 283 283:tiger cat

0.111141 286 286:Egyptian cat

0.078427 282 282:tabby, tabby cat

0.009678 264 264:Pembroke, Pembroke Welsh corgi

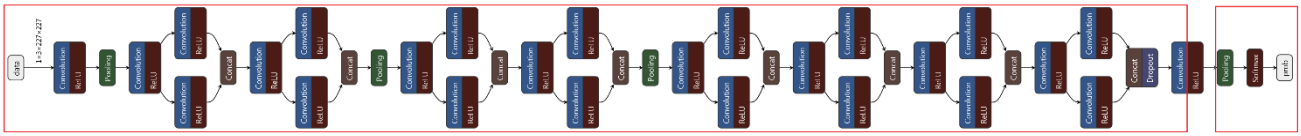
0.002746 905 905:window screen

3 分段 demo 使用说明

3.1 流程说明

某些模型在离线转换过程中，因部分算子不能支持，需要对模型做分段处理，下面以 squeezenet 为例就分段模型的执行做个说明。

这里将 squeezenet 分为三段，第一段从 data 到 conv10，第二段是 relu_conv10，第三段是余下部分。事实上 squeezenet 本身是可以全部转换成功的，这里只是做个例子。关于模型分段的方法，可以参考 imgtec 的文档和示例。



第一段输入是一张图片，使用 opencv 接口对图片做解码/resize 等，接着通过调用 aisdk，完成模型的运行。

第二段是 relu_conv10，它的输入是第一段的输出，我们在代码中实现 relu，如下：

```
template <class T>
void relu(T *buffer, unsigned buffer_len) {
    buffer_len /= sizeof(T);

    for (int i=0; i<buffer_len; i++) {
        buffer[i] = std::max(buffer[i], T(0));
    }
    tmp = nullptr;
}
```

其中 buffer 就是第一段的输出，而 buffer_len 是 buffer 的长度（多少 byte）。第三段输入是 relu 的输出，继续调用 aisdk 的接口，完成模型加载和推理动作。接下来说下代码实现。

3.2 代码解读

```
int main(int argc, char* argv[]) {
  std::string modelfile_first("/data/aisdk_test/squeezenet_net0.mbs.bin");
  std::string modelfile_second("/data/aisdk_test/squeezenet_net1.mbs.bin");
  std::string input_image("/data/aisdk_test/cat.jpg");
  //std::string infile("/data/aisdk_test/cat.raw");
  bool usebuffer = false;

  // 1. create model manager  创建 modelmanager
  std::cout << "UNISOC_AISDK create model manager start " << std::endl;
  ModelMgr* modelManager = CreateModelManager();
  if (modelManager == NULL) {
    std::cout << "UNISOC_AISDK odelManager is NULL " << std::endl;
    return -1;
  }

  // 2. load model 加载第一段模型
  int ret = LoadModel(modelManager, modelfile_first.c_str(), HIGH_PERF);
  if (ret != AI_SUCCESS) {
    std::cout << "UNISOC_AISDK load model fail " << std::endl;
    return ret;
  }

  // 3. input image files/buffers;
```

```
33.   DataFormat dataformat;
34.   InitDataFormat(&dataformat);
35.   dataformat.model_shape.input_node_dim_size = 4;
36.   dataformat.model_shape.output_node_dim_size = 4;
37.
38.   // input image buffers
39.   unsigned int inbuf_size = 1;
40.   unsigned int outbuf_size = 1;
41.   int input_count = 1;
42.   // if the model's input node is same as output node, set output count as input
43.   // count. Otherwise, give the valid output size.
44.   int output_count = input_count;
45.   void* input_buffer[input_count];
46.   void* output_buffer[output_count];
47.
48.   std::cout << "UNISOC_AISDK input files, size: " << input_count << std::endl;
49.   int count = 0;
50.   //设置第一段模型的 input/output 节点信息
51.   dataformat.model_shape.input_node_dim_size = 4;
52.   dataformat.model_shape.input_node_shape[0] = 1;
53.   dataformat.model_shape.input_node_shape[1] = 3;
54.   dataformat.model_shape.input_node_shape[2] = 227;
55.   dataformat.model_shape.input_node_shape[3] = 227;
56.   dataformat.model_shape.output_node_dim_size = 4;
57.   dataformat.model_shape.output_node_shape[0] = 1;
58.   dataformat.model_shape.output_node_shape[1] = 1000;
59.   dataformat.model_shape.output_node_shape[2] = 14;
60.   dataformat.model_shape.output_node_shape[3] = 14;
61.   for (unsigned i = 0; i < dataformat.model_shape.input_node_dim_size;
62.        ++i) {
63.     inbuf_size *= dataformat.model_shape.input_node_shape[i];
64.   }
65.
66.   for (unsigned i = 0; i < dataformat.model_shape.output_node_dim_size;
67.        ++i) {
68.     outbuf_size *= dataformat.model_shape.output_node_shape[i];
69.   }
70.   //计算 input/output buffer 的大小
71.   if (dataformat.input_type == AISDK_FLOAT32) {
72.     inbuf_size *= sizeof(float);
73.     outbuf_size *= sizeof(float);
74.   }
75.
```

```
76.   std::cout << "UNISOC_AISDK malloc buffer, inbuf: " << inbuf_size
77.             << " , outbuf :" << outbuf_size << std::endl;
78.   input_buffer[count] = reinterpret_cast<void*>(malloc(inbuf_size));
79.   memset(input_buffer[count], 0, inbuf_size);
80.   output_buffer[count] = reinterpret_cast<void*>(malloc(outbuf_size));
81.   memset(output_buffer[count], 0, outbuf_size);
82.   //读图片，解码之后将图片保存到 input_buffer 中
83.   get_img<float>((float*)input_buffer[count], input_image, dataformat.model_shape.input_nod
    e_shape);
84.
85.   // 4. run model  执行第一段模型
86.   if (usebuffer) {
87.     // to do use buffer
88.   } else {
89.     RunModel(modelManager, input_buffer, input_count, &dataformat,
90.              output_buffer, output_count, 1000);
91.   }
92.
93.   std::cout << "UNISOC_AISDK do relu " << std::endl;
94.   sleep(2);
95.   for (int i = 0; i < output_count; i++) {
96.     //调用 relu，实现自定义算子
97.     if (dataformat.input_type == AISDK_FLOAT32) {
98.       my_relu<float>((float*)output_buffer[i], outbuf_size);
99.     } else {
100.       my_relu<uint8_t>((uint8_t*)output_buffer[i], outbuf_size);
101.     }
102.   }
103.   //销毁 modelmanager
104.   DestroyModelManager(modelManager);
105.
106.   modelManager = CreateModelManager();  //创建 modelmanager
107.   if (modelManager == NULL) {
108.     std::cout << "UNISOC_AISDK odelManager is NULL " << std::endl;
109.     return -1;
110.   }
111.   //modelmanager 加载第二段模型
112.   ret = LoadModel(modelManager, modelfile_second.c_str(), HIGH_PERF);
113.   if (ret != AI_SUCCESS) {
114.     std::cout << "UNISOC_AISDK load model fail " << std::endl;
115.     return ret;
116.   }
117.   //设置第二段模型的 input/output 信息
```

```
118.   DataFormat dataformat_2nd;
119.   InitDataFormat(&dataformat_2nd);
120.   dataformat_2nd.model_shape.input_node_dim_size = 4;
121.   dataformat_2nd.model_shape.input_node_shape[0] = 1;
122.   dataformat_2nd.model_shape.input_node_shape[1] = 1000;
123.   dataformat_2nd.model_shape.input_node_shape[2] = 14;
124.   dataformat_2nd.model_shape.input_node_shape[3] = 14;
125.   dataformat_2nd.model_shape.output_node_dim_size = 2;
126.   dataformat_2nd.model_shape.output_node_shape[0] = 1;
127.   dataformat_2nd.model_shape.output_node_shape[1] = 1000;
128.   //分配空间，执行第二段模型
129.   void* output_buffer_2nd[output_count];
130.   output_buffer_2nd[0] = malloc(1000*sizeof(float));
131.   RunModel(modelManager, output_buffer, input_count, &dataformat_2nd,
132.            output_buffer_2nd, output_count, 1000);
133.
134.   // 5. parse output, print top-n result
135.   if (usebuffer) {
136.     // to do use buffer
137.   } else {
138.     for (int i = 0; i < output_count; i++) {
139.       if (dataformat.input_type == AISDK_Q8A) {
140.         GetTopnResult<uint8_t>(reinterpret_cast<uint8_t*>(output_buffer_2nd[i]),
141.                                1000, false);
142.       } else {
143.         GetTopnResult<float>(reinterpret_cast<float*>(output_buffer_2nd[i]),
144.                              1000, true);
145.       }
146.     }
147.     for (int i = 0; i < input_count; i++) {
148.       free(input_buffer[i]);
149.     }
150.     for (int i = 0; i < output_count; i++) {
151.       free(output_buffer[i]);
152.     }
153.
154.     for (int i = 0; i < output_count; i++) {
155.       free(output_buffer_2nd[i]);
156.     }
157.   }
158.   DestroyModelManager(modelManager);
159.   return 0;
160. }
```