NN SDK API Description

Revision: 1.9. 4

Release Date: 2022-02–10

Copyright

Trademarks

Amlogic, and other Amlogic icons are trademarks of Amlogic companies. All other trademarks and registered trademarks are property of their respective companies.

Disclaimer

Amlogic may make improvements and/or changes in this document or in the product described in this document at any time.

This product is not intended for use in medical, life saving, or life sustaining applications.

Circuit diagrams and other information relating to products of Amlogic are included as a means or illustrating typical applications. Consequently, complete information sufficient for production design is not necessarily given. Amlogic makes no representations or warranties with respect to the accuracy or completeness of the contents presented in this document.

Contact Information

- Website: www.amlogic.com
- Pre-sales consultation: contact@amlogic.com
- Technical support: support@amlogic.com

Reversion History

Issue 1.9.4 (2021-12-15)

Twelfth edition.

1. Add nnsdk workflow flowchart to Chapter 1.

2. Added the introduction of the model performance debug interface in Section 3.5.

3. Add the introduction of the save NPU input information in section 3.3.9.

Issue 1.8.2 (2021-06-30)

Eleventh version.

1. Add aml_util_switchInputBuffer and aml_util_switchOutputBuffer API descriptions

2. Modify the nn_input interface definition

Issue 1.8.0 (2021-03-22)

Tenth version.

1. Delete the SDK network model support list and demo API data structure table in Chapter 1

2. Modify the description of assign_user_address_t, aml_output_config_t, aml_perf_mode_t, input_info in section 2.2.

3. Delete the original Chapter 3.

Issue 1.7.2 (2020–12–15)

ninth version.

1. Modify table 1.2 SDK API interface table, add table 1.3 SDK API data structure table, add table 1.4 SDK Demo API data structure table.

2. Modify the sample code in subsections 2.1.3, 2.1.4, 2.1.7, 2.1.10, 2.1.11, 2.1.13.

3.Add aml_util_flushTensorHandle and aml_util_getHardwareStatus API descriptions

4.Modify the description of the SDK integration demo call process description in section 3.1.1

5.Add the description of the demo frame type in Section 3.1.2

6.Modify the description of opencv dependency environment in subsection 3.1.3

7.Modify the reference code in section 3.1.4

8.Add sample code in subsection 3.2.4

9.Add the sample code of section 3.2.5 usb camera

10.Modify Section 3.2.8 SDK-HardwareStatus

11.Modify the reference code in section 3.2.11

Issue 1.7 (2020-09-25)

Eighth version.

1. Added the introduction of ffmpeg interface.

2. Modified the introduction of aml_module_output_get interface.

Issue 1.6 (2020-07-31)

Seventh version.

1. Added the introduction of setProfile interface.

2. Added the introduction of setPowerPolicy interface.

3. Added the introduction of using usb_camera.

4. Modified the introduction of calling method of user-defined model.

5. Added introduction of tensor_info and info structure.

Issue 1.5 (2020-06-23)

Sixth version.

1. Added the related introduction of aml_util_getInputTensorInfo / aml_util_getOutputTensorInfo interface.

2. Added the related introduction of aml_util_freeTensorInfo interface.

3. Added the related introduction of aml_util_flushTensorHandle interface.

4. Removed the original chapter 3 and 4 and added the introduction of using SDK API.

5. Supported face_emotion, head_detection, person_detection model.

Issue 1.4 (2020-05-25)

Fifth version.

1. Modified aml_config definition.

2. Modified sdk demo and custom_network reference code.

3. Supported face_landmark_68 and image_segmentation model.

Issue 1.3 (2020-04-22)

Fourth version.

1. Modified outBuf_t definition.

2. Modified sdk demo and custom_network reference code.

Issue 1.2 (2020-03-24)

Third version.

1. Added the function introduction of input dma swap.

2. Added the introduction of aml_util_swapInputBuffer interface.

3. Modified the process introduction of custom_network basic call.

4. Modified demo and custom_network reference code.

Issue 1.1 (2020-02-27)

Second version.

1. Added the function introduction of input/output dma.

2. Added the introduction of API aml_util_mallocAlignedBuffer and aml_util_freeAlignedBuffer interface.

3. Added the introduction of assign_user_address_t, aml_output_config_t and aml_output_format_t data structure.

4. Modified face_gender_out_t structure.

5. Added the control function introduction of micro-opencv.

Issue 1.0 (2019-12-12)

First version.

Contents

1. Introduction

This document introduces the Amlogic SDK API interface and related data structure to guide amlogic developers and amlogic customers.

SDK API interface information is shown in table 1.1, SDK data structure information is shown in table 1.2,SDK workflow information is shown in picture 1.1.

Chapter one: The first chapter introduces the structure information and main contents of this document.

Chapter two: The SDK API and data structure are introduced, including the functions, parameters and return values of the SDK API. The data structure defined by the SDK is introduced in detail.

Chapter three: How to use the SDK API, this chapter introduces how to use the SDK API, and attaches a sample code for reference.

Table 1.1 SDK API interface table

SDK API	void* aml_module_create(aml_config* config);
	int aml_module_input_set(void* context,nn_input *pInput);
	void* aml_module_output_get(void* context,aml_output_config_t outconfig)
	void* aml_module_output_get_simple(void* context);
	int aml_module_destroy(void* context);
	unsigned char * aml_util_mallocAlignedBuffer(int mem_size);
	void aml_util_freeAlignedBuffer(unsigned char *addr);
	int aml_util_swapInputBuffer(void *context,void *newBuffer,unsigned int inputId);
	int aml_util_swapOutputBuffer(void *context,void *newBuffer,unsigned int outputId);
	tensor_info* aml_util_getInputTensorInfo(const char* nbgdata);
	tensor_info* aml_util_getOutputTensorInfo(const char* nbgdata);
	void aml_util_freeTensorInfo(tensor_info* tinfo);
	int aml_util_flushTensorHandle(void* context,aml_flush_type_t type);
	int aml_util_setProfile(aml_profile_type_t type,const char *savepath);

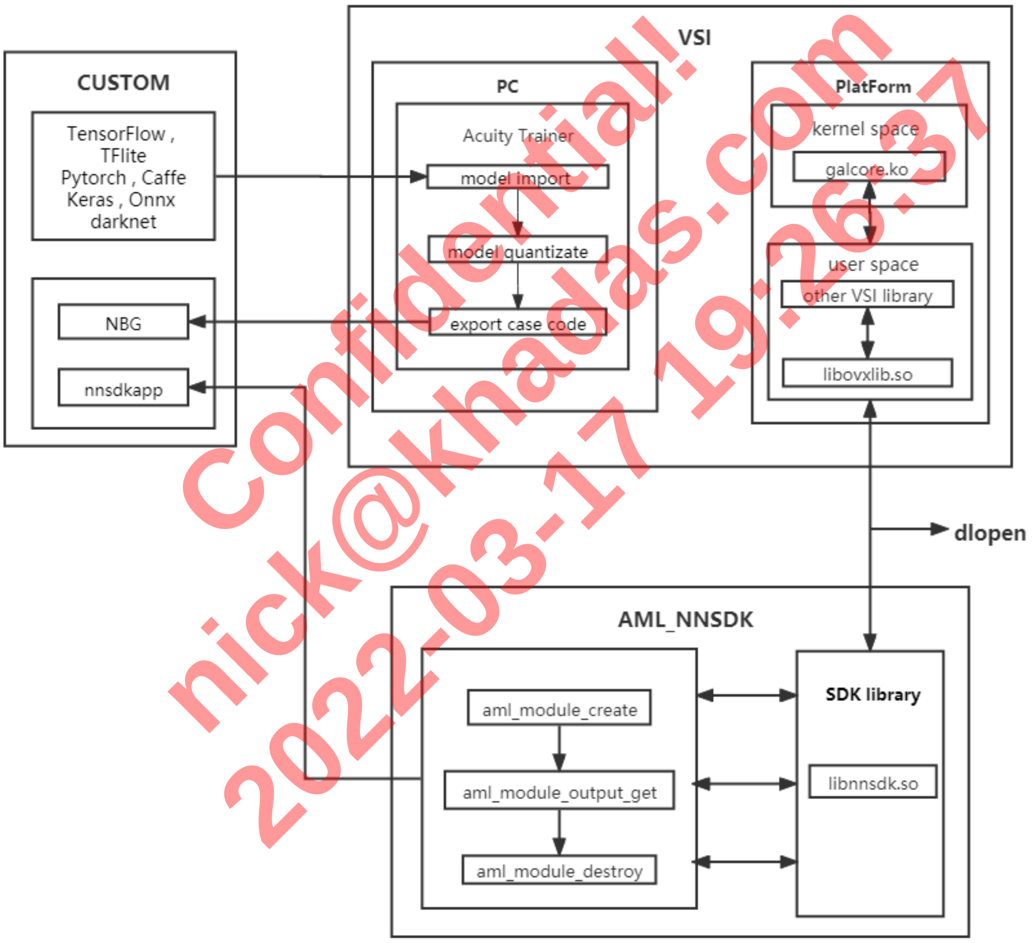
	int aml_util_setPowerPolicy(aml_policy_type_t type);
	int aml_util_switchInputBuffer(void *context,void *newBuffer,unsigned int inputId);
	int aml_util_switchOutputBuffer(void *context,void *newBuffer,unsigned int outputId);
	int aml_util_getHardwareStatus(int *customID,int *powerStatus);

Table 1.2 SDK API structure table

structure	function
aml_config	Output config information
assign_user_address_t	Set user defined memory address
aml_output_config_t	Set output model and data type information
aml_output_format_t	Set output data type
amlnn_input_mode_t	Set input mode
amlnn_model_type	Set framework type
amlnn_nbg_type	Set read nbg file type
amlnn_input_type	Set input data type
amlnn_query_cmd	Set query type
nn_buffer_format_e	Set output buffer type
nn_buffer_quantize_format_e	Set quantize type
aml_module_t	Set demo type
aml_profile_type_t	Set profile type

aml_policy_type_t	Set policy type
aml_flush_type_t	Set flush tensor type
assign_user_address_t	Set assign address type
image_out_t	Output image results
tensor_info	Output tensor information

picture 1.1 SDK workflow



2. Introduce SDK API and data structure

2.1 Detailed description of SDK API

void* aml_module_create(aml_config* config);

int aml_module_input_set(void* context,nn_input *pInput);

void* aml_module_output_get(void* context,aml_output_config_t outconfig);

void* aml_module_output_get_simple(void* context);

int aml_module_destroy(void* context);

unsigned char * aml_util_mallocAlignedBuffer(int mem_size);

void aml_util_freeAlignedBuffer(unsigned char *addr);

int aml_util_swapInputBuffer(void *context,void *newBuffer,unsigned int inputId);

int aml_util_swapOutputBuffer(void *context,void *newBuffer,unsigned int outputId);

int aml_util_switchInputBuffer(void *context,void *newBuffer,unsigned int inputId);

int aml_util_switchOutputBuffer(void *context,void *newBuffer,unsigned int outputId);

tensor_info* aml_util_getInputTensorInfo(const char* nbgdata);

tensor_info* aml_util_getOutputTensorInfo(const char* nbgdata);

void aml_util_freeTensorInfo(tensor_info* tinfo);

int aml_util_flushTensorHandle(void* context,aml_flush_type_t type);

int aml_util_setProfile(aml_profile_type_t type,const char *savepath);

int aml_util_setPowerPolicy(aml_policy_type_t type);

int aml_util_getHardwareStatus(int *customID,int *powerStatus);

2.1.1 aml_module_create

API	Void* aml_module_create(aml_config* config);
Function	Initialize the handle and load the generated nbg model
Parameter	Aml_config* config：path and type information of nbg model
Return value	Void*：handle context object pointer

Code example：

```
void *context;
context = aml_module_create(&config);
```

2.1.2 aml_module_input_set

API	Int aml_module_input_set(void* context,nn_input *pInput);
Function	Set the input data according to the nbg model
Parameter	void* context: handle context pointer
	nn_input *pInput： input data. Currently, only one input can be set at a time. If it is a multiple input model, you need to call amlnn_inputs_set interface several times, which will be corrected later.
Return value	int: success returns zero,and failure returns nonzero.

Code example：

```
    nn_input inData;
    unsigned char *rawdata;

    rawdata = get_jpeg_rawData(jpeg_path,224,224);
    inData.input_index = 0;     //this value is index of input,begin from 0
    inData.size = 224*224*3;
    inData.input = rawdata;
    inData.input_type = RGB24_RAW_DATA;
    ret = aml_module_input_set(context,&inData);
```

2.1.3 aml_module_output_get

API	void* aml_module_output_get(void* context,aml_output_config_t outconfig)	
Function	obtain the result of one inference.	
Parameter	void* context: handle context pointer.	
	aml_output_config_t outconfig:model type and output type.	
	aml_module_t mdType:model type.	aml_output_format_t format: output type.
Return value	void*: Inference result structure pointer.	

Code example：

```
    img_classify_out_t *cls_out = NULL;
    modelType = IMAGE_CLASSIFY;
    outconfig.format = AML_OUTDATA_FLOAT32;
```

```
outdata = (nn_output*)aml_module_output_get(context,outconfig);
cls_out = (img_classify_out_t*)post_process_all_module(modelType,outdata);
for(i = 0; i < 5; i++)
{
        printf("top %d:score--%f,class--%d\n",i,cls_out->score[i],cls_out->topClass[i]);
}
```

2.1.4 aml_module_output_get_simple

API	void* aml_module_output_get_simple(void* context);
Function	obtain the result of one inference,ouput tensor type is float32
Parameter	void* context: handle context pointer.
Return value	void*：Inference result structure pointer.

Code example：

```
nn_output *outdata = NULL;
outdata = (nn_output *)aml_module_output_get_simple(qcontext);
process_top5((float*)outdata->out[0].buf,outdata->out[0].size/sizeof(float),NULL);
```

2.1.5 aml_module_destroy

API	int aml_module_destroy(void* context);
Function	unload the nbg model, and destroy the context and related resources.
Parameter	void* context：handle context pointer.
Return value	int: Success returns zero,and failure returns nonzero.

Code example：

```
int ret = aml_module_destroy(context);
```

2.1.6 aml_util_mallocAlignedBuffer

API	unsigned char * aml_util_mallocAlignedBuffer(int mem_size);
Function	Request memory space for inbuf or outbuf to use the input / output DMA function. See examples.
Parameter	int mem_size: Size of allocated memory space.
Return value	unsigned char *: Returns a pointer to the requested address.

Code example：

```
    unsigned char * inbuf = aml_util_mallocAlignedBuffer(224*224*3);
    unsigned char * outbuf = aml_util_mallocAlignedBuffer(1000*2)；
    //Set the input and output dma addresses. After setting, the input data can be directly
copied to the inbuf, which will map directly to the npu space without additional operation.
Similarly, after setting outbuf, when aml_module_output_get the system will directly
parse the outbuf data. The process of copying the output data from NPU to CPU is
omitted, so as to reduce the overall running time of the system.
    config.inOut.outAddr[0] = outbuf;
    config.inOut.inAddr[0] = inbuf;
    config.modelType = TENSORFLOW;
    context = aml_module_create(&config);
```

2.1.7 aml_util_freeAlignedBuffer

API	void aml_util_freeAlignedBuffer(unsigned char *addr);
Function	Release inbuf dma or outbuf dma memory space allocated by using aml_util_mallocAlignedBuffer.
Parameter	unsigned char *addr：aml_util_mallocAlignedBuffer pointer returned
Return value	NULL。

Code example：

```
    modelType = IMAGE_CLASSIFY;
    outconfig.format = AML_OUTDATA_FLOAT32;
    outdata = (nn_output*)aml_module_output_get(qcontext,outconfig);
    cls_out = (img_classify_out_t*)post_process_all_module(modelType,outdata);
    aml_util_freeAlignedBuffer(inbuf);
    aml_util_freeAlignedBuffer(outbuf);
    aml_module_destroy(context);
```

2.1.8 aml_util_swapInputBuffer

API	int aml_util_swapInputBuffer(void *context,void *newBuffer,unsigned int inputId);
Function	Modify the start pointer of input DMA buffer to realize the fast switch of input.
Parameter	void *context: handle context pointer.
	void *newBuffer: Input buffer information.
	unsigned int inputId：Input ID, starting from 0.

Return value	int：Success returns zero,and failure returns nonzero.

Code example：

```
        unsigned char * outbuf = aml_util_mallocAlignedBuffer(1000*2);
        unsigned char * inbuf1 = aml_util_mallocAlignedBuffer(224*224*3);
        unsigned char * inbuf2 = aml_util_mallocAlignedBuffer(224*224*3);
        config.inOut.outAddr[0] = outbuf;
        config.inOut.inAddr[0] = inbuf1;
        config.inOut.io_type = AML_IO_VIRTUAL;

        config.modelType = TENSORFLOW;
        context = aml_module_create(&config);
        memcpy(inbuf1,rawdata1,224*224*3);
        ///After executing forward inference and post processing, switch the second picture.
        memcpy(inbuf2,rawdata2,224*224*3);    //This operation can be ready during the
execution of the first frame, forming an effect similar to ping-pang buffer to save input
time.
        aml_util_swapInputBuffer(context,(void*)inbuf2,0);
```

2.1.9 aml_util_swapOutputBuffer

API	int aml_util_swapOutputBuffer(void *context,void *newBuffer,unsigned int outputId);
Function	Modify the start pointer of output DMA buffer to realize the fast switch of input.
Parameter	void *context: handle context pointer.
	void *newBuffer: Output buffer information.
	unsigned int outputId：Output ID, starting from 0.
Return value	int：Success returns zero,and failure returns nonzero.

Code example：

```
        unsigned char * outbuf = aml_util_mallocAlignedBuffer(1000*2);
        config.inOut.outAddr[0] = outbuf;
        config.inOut.io_type = AML_IO_VIRTUAL;
        context = aml_module_create(&config);
        aml_util_swapOutputBuffer(context,(void*)outbuf,0);
        aml_module_output_get(context,outconfig);
```

2.1.10 aml_util_flushTensorHandle

API	int aml_util_flushTensorHandle(void* context,aml_flush_type_t type);
Function	flush tensor handle
Parameter	void *context: handle context pointer
	aml_flush_type_t type:tensor type，set input or output
Return value	NULL

Code example：

```
    memcpy(inbuf,rawdata,224 * 224 * 3);
    aml_util_flushTensorHandle(context,AML_INPUT_TENSOR);
```

2.1.11 aml_util_getInputTensorInfo

API	tensor_info* aml_util_getInputTensorInfo(const char* nbgdata);
Function	Obtain input information of NBG file.
Parameter	const char* nbgdata：nbg file data information
Return value	tensor_info*：Return a pointer to the tensor information in nbg.

Code example：

```
    inptr = aml_util_getInputTensorInfo(config.pdata);
    if(inptr->valid == 1)
    {
        printf("the input buffer info is get success\n");
        printf("the input tensor number is %d\n",inptr->num);
        printf("the input[0] data_format:%d\n",inptr->info[0].data_format);
        printf("the input[0] quantization_format:%d\n",inptr->info[0].quantization_format);
        printf("the input[0] sizes_of_dim:%d,%d,%d,%d\n",inptr->info[0].sizes_of_dim[0],
        inptr->info[0].sizes_of_dim[1],inptr->info[0].sizes_of_dim[2],inptr->info[0].sizes_o
f_dim[3]);
    }
    aml_util_freeTensorInfo(inptr);
```

2.1.12 aml_util_getOutputTensorInfo

API	tensor_info* aml_util_getOutputTensorInfo(const char* nbgdata);

Function	Obtain input information of NBG file.
Parameter	const char* nbgdata：nbg file data information
Return value	tensor_info*：Return a pointer to the tensor information in nbg.

Code example：

```
     outptr = aml_util_getOutputTensorInfo(config.pdata);
     if(outptr->valid == 1)
     {
          printf("the output buffer info is get success\n");
          printf("the output tensor number is %d\n",outptr->num);
          printf("the output[0] data_format:%d\n",outptr->info[0].data_format);
          printf("the output[0] quantization_format:%d\n",outptr->info[0].quantization_format);
          printf("the output[0] sizes_of_dim:%d,%d,%d,%d\n",outptr->info[0].sizes_of_dim[0],
          outptr->info[0].sizes_of_dim[1],outptr->info[0].sizes_of_dim[2],outptr->info[0].sizes_o
f_dim[3]);
     }
     aml_util_freeTensorInfo(outptr);
```

2.1.13 aml_util_freeTensorInfo

API	void aml_util_freeTensorInfo(tensor_info* tinfo);
Function	Release the allocated memory space of inptr or outptr.
Parameter	tensor_info*：aml_util_getInputTensorInfo or aml_util_getOutputTensorInfo obtained inptr or outptr data.
Return value	NULL

Code example：

```
     inptr = aml_util_getInputTensorInfo(config.pdata);
     aml_util_freeTensorInfo(inptr);
```

2.1.14 aml_util_setProfile

API	int aml_util_setProfile(aml_profile_type_t type,const char *savepath);
Function	Set Profile to get performance, bandwidth and memory information.
Parameter	aml_profile_type_t: Get Profile information type.
	const char *:profile log save path

Return value	NULL

```
typedef enum {
      AML_PROFILE_NONE                   = 0,
      AML_PROFILE_PERFORMANCE      = 1,
      AML_PROFILE_BANDWIDTH         = 2,
      AML_PROFILE_MEMORY              = 3
} aml_profile_type_t;
```

Code example：

```
      aml_util_setProfile(AML_PROFILE_MEMORY,"/media/test/sdk_save.log")
      aml_util_setProfile(AML_PROFILE_BANDWIDTH,"/media/test/bandwidth_save.log")
      aml_util_setProfile(AML_PROFILE_PERFORMANCE,NULL)
      aml_util_setProfile(AML_PROFILE_NONE,NULL)
      context = aml_module_create(&config);
```

2.1.15 aml_util_setPowerPolicy

API	int aml_util_setPowerPolicy(aml_policy_type_t type);
Function	Set power policy
Parameter	aml_policy_type_t: Power policy type AML_PERFORMANCE_MODE，AML_POWER_SAVE_MODE， AML_MINIMUM_POWER_MODE
Return value	NULL

```
typedef enum {
      AML_PERFORMANCE_MODE              = 1,
      AML_POWER_SAVE_MODE                 = 2,
      AML_MINIMUM_POWER_MODE          = 3
} aml_policy_type_t;
```

Code example：

```
      aml_util_setPowerPolicy(AML_MINIMUM_POWER_MODE);
      context = aml_module_create(&config);
```

2.1.16 aml_util_getHardwareStatus

API	int aml_util_getHardwareStatus(int *customID,int *powerStatus);

Function	Get board customID and power status info
Parameter	int *customID:chip info pointer
	int *powerStatus:power status info pointer
Return value	NULL

Code example：

```
int customID,powerStatus;
aml_util_getHardwareStatus(&customID,&powerStatus);
printf("customID=%x,powerStatus=%x\n",customID,powerStatus);
context = init_network(argc,argv);
```

2.1.17 aml_util_switchInputBuffer

API	int aml_util_switchInputBuffer(void *context,void *newBuffer,unsigned int inputId);
功能	修改输入物理地址对应 DMA buffer 的起始指针，实现 input 的快速切换
参数	void *context:句柄 context 指针。
	void *newBuffer:输入的 buffer 信息。
	unsigned int inputId：输入 ID，从 0 开始。
返回值	int：成功返回零，失败返回非零。

示例代码：

```
config.inOut.inAddr[0] = input_phy_addr;
config.inOut.outAddr[0] = output_phy_addr;
config.inOut.io_type = AML_IO_PHYS;
context = aml_module_create(&config);

aml_util_switchInputBuffer(qcontext,(void*)new_in_addr,0);
aml_util_switchOutputBuffer(qcontext,(void*)new_out_addr,0);
aml_module_output_get(context,outconfig);
```

2.1.18 aml_util_switchOutputBuffer

API	int aml_util_switchOutputBuffer(void *context,void *newBuffer,unsigned int outputId);

功能	修改输出物理地址对应 DMA buffer 的起始指针，实现 output 的快速切换
参数	void *context:句柄 context 指针。
	void *newBuffer:输出的 buffer 信息。
	unsigned int outputId：输出 ID，从 0 开始。
返回值	int：成功返回零，失败返回非零。

示例代码：

```
    config.inOut.inAddr[0] = input_phy_addr;
    config.inOut.outAddr[0] = output_phy_addr;
    config.inOut.io_type = AML_IO_PHYS;
    context = aml_module_create(&config);

    aml_util_switchInputBuffer(qcontext,(void*)new_in_addr,0);
    aml_util_switchOutputBuffer(qcontext,(void*)new_out_addr,0);
    aml_module_output_get(context,outconfig);
```

2.2 Detailed description of data structure of SDK

2.2.1 aml_config

Member variable	Data type	Meaning
typeSize	int	Number of structure member variables
path	const char *	load nbg from file pointer
pdata	const char *	load nbg from memory pointer
length	int	nbg size
modelType	amlnn_model_type	Nbg model type
nbgType	amlnn_nbg_type	load nbg method
inOut	assign_user_address_t	assign user address

```
typedef struct __aml_nn_config
{
    int typeSize;
    const char *path;
    const char *pdata;
    int length;
    amlnn_model_type modelType;
    amlnn_nbg_type nbgType;
    assign_user_address_t inOut;
}aml_config;
typedef enum _amlnn_nbg_type_ {
    NN_NBG_FILE      = 0,   //load nbg from file
    NN_NBG_MEMORY     //load nbg from memory
} amlnn_nbg_type;
```

amlnn_model_type

Member variable	Model type
CAFFE	Caffe model
TENSORFLOW	Tensorflow model
TENSORFLOWLITE	TensorflowLite model
DARKNET	Darknet model
ONNX	Onnx model
KERAS	Keras model
PYTORCH	Pytorch model

2.2.2 assign_user_address_t

Member variable	Data type	Meaning
io_type	aml_io_format_t	dma io type
inAddr	unsigned char*	input address
outAddr	unsigned char*	output address

```
typedef struct __assign_address
{
    unsigned char* inAddr[ADDRESS_MAX_NUM];
    unsigned char* outAddr[ADDRESS_MAX_NUM];
}assign_user_address_t;
typedef enum {
    AML_IO_VIRTUAL       = 0,
    AML_IO_PHYS          = 1,
} aml_io_format_t;
```

2.2.3 aml_output_config_t

Member variable	Data type	Meaning
typeSize	int	Number of structure member variables
mdType	aml_module_t	Network model
perfMode	aml_perf_mode_t	Model inference type
format	aml_output_format_t	Output data type

```
typedef   struct __amlnn_module_out_data_t
{
    int typeSize;
    aml_module_t mdType;
    aml_perf_mode_t perfMode;
    aml_output_format_t format;
} aml_output_config_t;
```

2.2.4 aml_perf_mode_t

成员变量	含义
AML_NO_PERF	Performance test mode off
AML_PERF_INFERRENCE	Inference step
AML_PERF_OUTPUT	inverse quantization step

```
typedef enum {
    AML_NO_PERF              = 0,
    AML_PERF_INFERRENCE   = 1,
    AML_PERF_OUTPUT           = 2
} aml_perf_mode_t;
```

2.2.5 aml_output_format_t

Member variable	Meaning
AML_OUTDATA_FLOAT32	Output data of type float32
AML_OUTDATA_RAW	Output raw data
AML_OUTDATA_DMA	Output dma data

```
typedef enum {
    AML_OUTDATA_FLOAT32         = 0,
    AML_OUTDATA_RAW             = 1,
    AML_OUTDATA_DMA             = 2
} aml_output_format_t;
```

2.2.6 image_out_t

Member variable	Data type	Meaning
height	int	Output image height value
width	int	Output image width value
channel	int	Number of output image channels
data	unsigned char *	Output image buffer information

```
typedef struct __nn_image_out
{
    int height;
    int width;
    int channel;
    unsigned char *data;    //this buffer is returned by aml_module_output_get
}image_out_t;
```

2.2.7 tensor_info

Member variable	Data type	Meaning
valid	unsigned int	Get tensor information flag
num	unsigned int	Tensor number
info	info *	tensor information pointer

```
typedef struct {
    unsigned int valid;
    unsigned int num;
    info *info;
} tensor_info;
```

2.2.8 Info

Member variable	Data type	Meaning
dim_count	unsigned int	Number of dimensions
sizes_of_dim	unsigned int[]	Dimension value
data_format	unsigned int	buffer format type
data_type	unsigned int	Data type
quantization_format	unsigned int	Quantitative type
fixed_point_pos	int	int8/int16 Quantified parameter values
TF_scale	float	Scale value
TF_zeropoint	int	Zeropoint value
name	char[]	name

```
typedef struct {
    unsigned int dim_count;          /*dim count*/
    unsigned int sizes_of_dim[4]; /*dim value,just support 4-d dim*/
    unsigned int data_format;        /*see as nn_buffer_format_e*/
    unsigned int data_type;          /*not use*/
    unsigned int quantization_format; /*see as nn_buffer_quantize_format_e*/
    int fixed_point_pos;              /*for int8/int16
QUANTIZE_DYNAMIC_FIXED_POINT*/
    float TF_scale;                   /*as tf define,scale*/
    int TF_zeropoint;                 /*as tf define,zeropoint*/
    char name[MAX_NAME_LEGTH];        /*not use,will used in future*/
} info;
```

2.2.9 detBox

Member variable	Data type	Meaning
x	float	Output the x-axis coordinate value of the object box
y	float	Output the Y-axis coordinate value of the object box
w	float	Output the width information of the object box
h	float	Output the height information of the object box
score	float	Output the score information of the object
objectClass	float	Output the category information of the object

2.2.10 point_t

Member variable	Data type	Meaning
x	float	Output the X-axis coordinate value of the object point
y	float	Output the Y-axis coordinate value of the object point

2.2.11 nn_output

Member variable	Data type	Meaning
typeSize	int	Number of structure member variables

num	unsigned int	Number of output tensors
out	outBuf_t[]	Output tensor structure array

```
typedef struct __nnout
{
        int typeSize;
        unsigned int num;      /*===========output tensor number============*/
        outBuf_t out[OUTPUT_MAX_NUM];
}nn_output;
```

2.2.12 outBuf_t

Member variable	Data type	Meaning
size	unsigned int	Number of output tensors
buf	unsigned char *	Output tensor data address
param	nn_buffer_params_t *	Output properties of tensor
name	char	Output tensor name

```
typedef struct out_buf
{
        unsigned int size;
        unsigned char *buf;
        nn_buffer_params_t *param;
        char    name[MAX_NAME_LEGTH];        //output tensor name
        aml_output_format_t out_format;
}outBuf_t;
```

2.2.13 nn_buffer_params_t

Member variable	Data type	Meaning
num_of_dims	unsigned int	Number of tensor dimensions
sizes	unsigned int[]	Values of tensor dimensions

data_format	nn_buffer_format_e	There are several data types of tensor： NN_BUFFER_FORMAT_FP32 = 0, NN_BUFFER_FORMAT_FP16 NN_BUFFER_FORMAT_UINT8 NN_BUFFER_FORMAT_INT8 NN_BUFFER_FORMAT_UINT16 NN_BUFFER_FORMAT_INT16
quant_format	nn_buffer_quantize_format_e	The quantization format has the following values： NN_BUFFER_QUANTIZE_NONE=0， NN_BUFFER_QUANTIZE_DYNAMIC_FIXED_POINT（dynamic fixed point quantization） NN_BUFFER_QUANTIZE_TF_ASYMM （Asymmetric quantization）
dfp	struct	DYNAMIC_FIXED_POINT quantitative type parameter： fl = dfp.fixed_point_pos

affine	struct	TF_ASYMM quantitative type parameter
		scale=affine.scale
		zeroPoint=affine.zeroPoint

```
typedef struct _nn_buffer_create_params_t
{
    unsigned int        num_of_dims;
    unsigned int         sizes[4];
    nn_buffer_format_e              data_format;
    nn_buffer_quantize_format_e quant_format;
    union {
        struct {
            unsigned char fixed_point_pos;
        } dfp;
        struct {
            float              scale;
            unsigned int zeroPoint;
            } affine;
        }
        quant_data;
} nn_buffer_params_t;
```

2.2.14 nn_input

Member variable	Data type	Meaning
typeSize	int	Number of structure member variables
input_index	int	The input data corresponds to the subscript of tensor, and the single input is 0
size	int	Input data size
input	unsigned char*	Input data address
input_type	amlnn_input_type	Input data types include the following types: RGB24_RAW_DATA = 0, TENSOR_RAW_DATA, QTENSOR_RAW_DATA, BINARY_RAW_DATA INPUT_DMA_DATA
info	input_info	Input buffer information

```
typedef struct {
    int valid;
    float mean[INPUT_CNANNEL];
    float scale;
    aml_input_format_t input_format;
    int int16_type ; //when int16 quantize model mean value is not 128,128,128,set
1
}input_info;
typedef enum {
    AML_INPUT_DEFAULT     = 0,     //channle format: caffe 2 1 0 ,others 0 1 2
    AML_INPUT_MODEL_1     = 1,     //channle format: 0 1 2
    AML_INPUT_MODEL_2     = 2,     //channle format: 2 1 0
} aml_input_format_t;
```

3. Introduce how to use SDK API

Currently, the SDK package uses a specific network ID. CUSTOM_NETWORK represents the new network, so the output data will retain the output of the original network, and no post-processing operation will be done for this type. In multiple network application,customers can manage multiple network instances according to the context value returned by network aml_module_createa.

Please refer to GitHub for the demo demo of each function usage mode: https://github.com/Amlogic-NN/AML_NN_SDK/tree/master/Linux/Demo

3.1 Basic call process of SDK custom model

1. set input/output dma（These interfaces are not required and are called as needed）

```
unsigned char * inbuf = aml_util_mallocAlignedBuffer(224*224*3);
unsigned char * outbuf = aml_util_mallocAlignedBuffer(1000*2);
memcpy(inbuf,rawdata,224*224*3);
config.inOut.outAddr[0] = outbuf;
config.inOut.inAddr[0] = inbuf;
```

2. Get input and output tensor information（These interfaces are not required and are called as needed）

```
inptr = aml_util_getInputTensorInfo(config.pdata);
outptr = aml_util_getOutputTensorInfo(config.pdata);
aml_util_freeTensorInfo(inptr);
aml_util_freeTensorInfo(outptr );
```

3. Initialization model：void* context = aml_module_create(&config);

4. set Input buffer swap（These interfaces are not required and are called as needed）

```
unsigned char * inbuf2 = aml_util_mallocAlignedBuffer(224*224*3);
memcpy(inbuf2,rawdata,224*224*3);
aml_util_swapInputBuffer(context,(void*)inbuf2,0);
```

5. flush cache（These interfaces are not required and are called as needed）

```
aml_util_flushTensorHandle(context,AML_INPUT_TENSOR); //use with input dma
```

6. Prepare input data and set input data

```
inData.input_index = 0;
inData.size = 224*224*3;
inData.input = rawdata;
inData.input_type = INPUT_DMA_DATA; //use with input dma
int ret = aml_module_input_set(context,&inData);
```

7. Carry out forward inference and obtain the inference results

```
outdata = (nn_output *)aml_module_output_get_simple(context);
```

8. Multiple inference: repeat steps 2 and 3 several times

9. Unload model：int ret = aml_module_destroy(context);

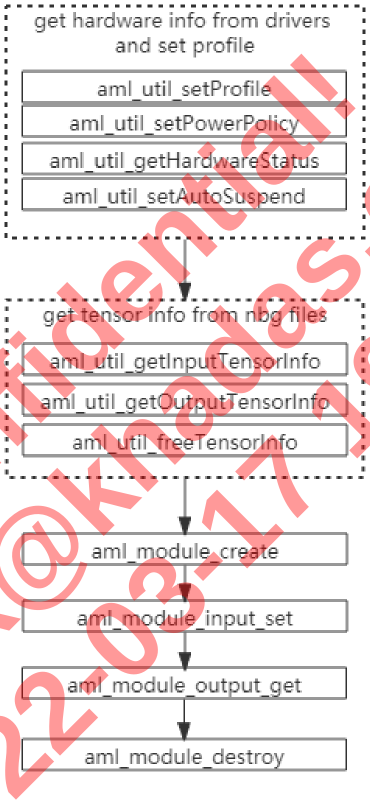
10. Free input/output memory（These interfaces are not required and are called as needed）

aml_util_freeAlignedBuffer(outbuf);

aml_util_freeAlignedBuffer(inbuf);

3.2 SDK demo call process description

Picture 3.1 SDK API call process



3.2.1 Model initialization

aml_config config;

//model nbg has two load methods

Mode one：load nbg from memory

fp = fopen(argv[1],"rb");

fseek(fp,0,SEEK_END);

size = (int)ftell(fp);

rewind(fp);

config.pdata = (char *)calloc(1,size);

```
        fread((void*)config.pdata,1,size,fp);
        config.nbgType = NN_NBG_MEMORY;
        config.length = size;
        fclose(fp);
```

Mode two：load nbg from file

```
        config.path= (const char *)argv[1];
        config.nbgType = NN_NBG_FILE;
```

```
    config.modelType = TENSORFLOW;            //Integration model framework
established
    void *context = aml_module_create(&config);
```

Note：

1. The nbg file of integration model is provided in package SDK_Rlease for users to use. Nbg needs to be pushed into the specified path of the platform.

https://github.com/Amlogic-NN/AML_NN_SDK

At present, nbg files are different from platforms, and the following methods can be used to determine the required nbg files.

Execute the order：cat /proc/cpuinfo

Check the first four characters of the sequence code corresponding to the last two lines of serial, and determine the nbg file according to the following corresponding relationship.

PID	Serial number
0X7D	Serial: 290a70004
0X88	Serial: 290b70004
0X99	Serial: 2b0a0f00d
0XA1	Serial: 300a01024

For example, if the first four digits of the serial number are 290b, you need to change XXX_ 88. NB push into the specified path of the platform.

2. modelType：Model framework type. modelType:Specify the corresponding frame type. Different frame types correspond to different preprocessing methods.The framework type of the integration model is fixed and no further modification is required.

caffe, pytorch, darknet, onnx default to nchw format, tensorflow, tensorflowlite, keras default to nhwc format, and the format conversion process has been integrated in libnnsdk.so, users do not need to modify the format specially. The reorder channel of caffe is set to '2,1,0' by default, and the rest of the frames are set to '0,1,2' by default. If the channel format of the pytorch model is '2,1,0', the frame type needs to be set to caffe.

Caffe:	BBBGGGRRR
Pytorch、darknet、onnx:	RRRGGGBBB
tensorflow、tensorflowlite、keras:	RGBRGBRGB

If the combination of the user's model frame type and reorder channel is not within the above range, the channel format can be set by selecting the info.valid mode.

inData.info.valid = 2;

inData.info.input_format = AML_INPUT_MODEL_1;   // reorder channel ‘0,1,2’

inData.info.input_format = AML_INPUT_MODEL_2;   // reorder channel ‘2,1,0’

3. file mode, the input parameter is the path of the nbg file. Among them, config.path is a pointer to the specific information of nbg, when specified config.nbgType is NN_ NBG_ FILE, its assignment is the path of the nbg file.

4. When nbg content has been saved in memory, you can choose to load from memory.Among them, config.pdata is a pointer to the specific information of nbg, when specified config.nbgType is NN_ NBG_ MEMORY, whose assignment represents the first memory address of nbg. Different fromNN_NBG_FILE，This mode requires additional settings config.size as the actual length of nbg's memory.

5. Using memory to load nbg requires memory release，free((void*)config.pdata).

3.2.2 set input

nn_input inData;

unsigned char *rawdata = get_jpeg_rawData(jpeg_path,416,416);

inData.input_index = 0;   //Input data index value to support multiple data input

inData.size = 416*416*3;   //Integrated model input data, size determined

inData.input = rawdata;

inData.input_type = RGB24_RAW_DATA;   //set input data format

Note：

1. get_jpeg_rawData：read jpeg_path image. The current function only supports the image input which is consistent with the specified size.Users can customize the relevant interface to input image information.

2. inData.size：Integrated model input data, size determined.

3. input_index：Input the data index value. Single input can be set to 0. In case of multiple inputs, the interface needs to be called multiple times, and the index value is increased by 1 in turn.

4. inData.input_type：Input data types include the following types:

In addition, type INPUT_DMA_DATA is supported，if using input dma function，inData.input_type needs to be set to this type.

Input_type type description:

●RGB24_RAW_DATA：Input RGB format data. After data input, the system will do reorder, mean, scale and other related operations. The values of these parameters come from the information set when acuity generates nbg.

●TENSOR_RAW_DATA：Float input of raw tensor data. After getting the data buffer, the system will perform the following operations:

Float fval;

Memcpy(&fval,rawdata,sizeof(float));

convertFloatToDtype(fval,tensordata);

The above completes a float data processing. Where rawdata is the input raw data, tensordata is the final input data into the model.

●QTENSOR_RAW_DATA：Be similar to TENSOR_RAW_DATA . The input data is of float type, and the processing method is uint8：

Float fval;

Memcpy(&fval,rawdata,sizeof(float));

Tensordata = (uint8)fval;

The above completes a float data processing. Where rawdata is the input raw data, tensordata is the final input data into the model.

●BINARY_RAW_DATA:The raw data is input without any processing and conversion. The process is approximately as follows:

Memcpy(tensordata_all,rawdata_all,all_size);

Copy all the input data intact into the input buffer of the model.

●INPUT_DMA_DATA：Input obtained using DMA function.

data = _get_rgb_data(tensor, pmeta, input, format_type);

*memcpy(input_dma,data,sz*stride)*

Input data type	meaning	Processing method
RGB24_RAW_DATA	Input RGB data, (0,255)	Reorder,mean,scale

TENSOR_RAW_DATA	Input tensor float data,
	Processing method is flaot32
		Memcpy(&fval,rawdata,sizeof(float));
		convertFloatToDtype(fval,tensordata)

QTENSOR_RAW_DATA	Input tensor float data,
	Processing method is uint8
		Memcpy(&fval,rawdata,sizeof(float));
		Tensordata = (uint8)fval

BINARY_RAW_DATA	Input rawdata	Memcpy(tensordata_all,rawdata_all,all_size);

3.2.3 Get model results

Use the output structure of nn_output to receive the result returned by the aml_module_output_get interface

```
nn_output *outdata = NULL;
outconfig.typeSize = sizeof(aml_output_config_t);
modelType = CUSTOM_NETWORK;                          //Set output model type
outconfig.format = AML_OUTDATA_FLOAT32;               //Set output data type
outdata = (nn_output*)aml_module_output_get(context,outconfig);
if(outdata->out[0].param->data_format == NN_BUFFER_FORMAT_FP32)
{
    process_top5((float*)outdata->out[0].buf,outdata->out[0].size/sizeof(float),NULL);
}
```

Outdata contains complete output information, including output dim, output format, output content and so on. The output content is float32 format by default, which can be implemented after relevant network post-processing. Such as mobilenet：

```
process_top5((float*)pout->out[0].buf,pout->out[0].size/sizeof(float));
```

Note：

1、Outconfig.format can choose AML_OUTDATA_FLOAT32, AML_OUTDATA_RAW, AML_OUTDATA_DMA according to actual needs.

2、AML_OUTDATA_FLOAT32 indicates that the output data type is f32, and the type conversion has been completed inside the SDK; AML_OUTDATA_RAW indicates the output of the original type of data; AML_OUTDATA_DMA indicates the output dma output buffer data type, which needs to be used with DMA.

In order to facilitate users to use custom models, a simple parameter interface is provided for forward reasoning and obtaining reasoning results.

```
void* aml_module_output_get_simple(void* context);
```

This interface only needs to pass in the context, and the default output data type is float32. usage:

```
nn_output * outdata = (nn_output *)aml_module_output_get_simple(context);
```

Users can flexibly choose the reasoning interface according to their needs.

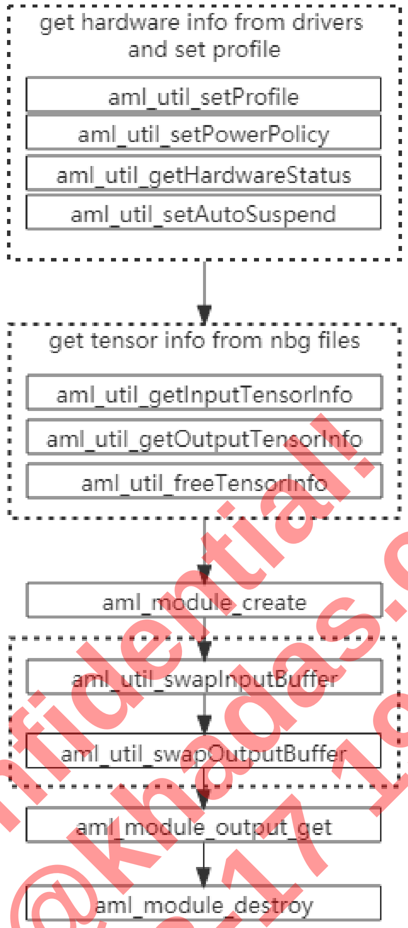
4、If the user needs to obtain the forward reasoning time of the model, the performance test mode can be enabled through outconfig.perfMode to obtain the forward reasoning and dequantization time respectively. outconfig.perfMode is not set by default, and the test mode is closed by default.

```
outconfig.typeSize = sizeof(aml_output_config_t);
modelType = CUSTOM_NETWORK;
outconfig.format = AML_OUTDATA_FLOAT32;
tmsStart = get_perf_count();
outconfig.perfMode = AML_PERF_INFERRENCE;
for (i=0;i<100;i++)
    outdata = (nn_output*)aml_module_output_get(qcontext,outconfig);
tmsEnd = get_perf_count();
msVal = (tmsEnd - tmsStart)/1000000;
usVal = (tmsEnd - tmsStart)/1000;
printf("AML_PERF_INFERRENCE: %ldms or %ldus\n", msVal, usVal);

tmsStart = get_perf_count();
outconfig.perfMode = AML_PERF_OUTPUT;
outdata = (nn_output*)aml_module_output_get(qcontext,outconfig);
tmsEnd = get_perf_count();
msVal = (tmsEnd - tmsStart)/1000000;
usVal = (tmsEnd - tmsStart)/1000;
printf("AML_PERF_OUTPUT: %ldms or %ldus\n", msVal, usVal);
```

3.3 Expansion function introduction

Picture 3.2 SDK API DMA function process


get hardware info from drivers and set profile
aml_util_setProfile
aml_util_setPowerPolicy
aml_util_getHardwareStatus
aml_util_setAutoSuspend
get tensor info from nbg files
aml_util_getInputTensorInfo
aml_util_getOutputTensorInfo
aml_util_freeTensorInfo
aml_module_create
aml_util_swapInputBuffer
aml_util_swapOutputBuffer
aml_module_output_get
aml_module_destroy

3.3.1 Introduction of SDK - DMA function

1. input dma set

1.1 Virtual Address

```
inbuf = aml_util_mallocAlignedBuffer(224*224*3);      //malloc buffer
config.inOut.inAddr[0] = inbuf;                       //inAddr information set
context = aml_module_create(&config);
```

Mode 1：

```
memcpy(inbuf,rawdata,224*224*3);                      //rawdata Input data for
read
```

Mode 2：

```
inData.input_index = 0;
inData.size = 224*224*3;
inData.input = rawdata;
inData.input_type = INPUT_DMA_DATA;              //Set input data type
aml_module_input_set(context,&inData);
```

Note：

1. aml_util_mallocAlignedBuffer The size of the corresponding image needs to be input when allocating memory.

2. mallocAlignedBuffer and config.inOut.inAddr setting must be completed before aml_module_create,otherwise DMA function can not be used normally.

3. after setting the input pointer，Method 1can directly copy the input data to the inputbuf. There is no need to do aml_module_input_set operation.**This method must ensure that the input data meets the preprocessing requirements.**

4. If you are not clear about the preprocessing operation to be done, you can use method 2. set inData.input_type = INPUT_DMA_DATA. The system will do the corresponding preprocessing and directly copy to the inputbuf. There is no memory interaction between NPU and CPU.

1.2 Physical Address

```
config.inOut.inAddr[0] = (unsigned char*)0x10000000;//set physical address
config.inOut.io_type = AML_IO_PHYS;
context = aml_module_create(&config);
```

2. input dma swap set

```
inbuf1 = aml_util_mallocAlignedBuffer(224*224*3);
inbuf2 = aml_util_mallocAlignedBuffer(224*224*3);
config.inOut.inAddr[0] = inbuf1;          //set inAddr
context = aml_module_create(&config);
memcpy(inbuf2,rawdata,224*224*3);
aml_util_swapInputBuffer(context,(void*)inbuf2,0);
```

note：

1. aml_util_swapInputBuffer the third parameter is inputId，starting from 0.

2. This operation can be prepared during the execution of the first frame, forming an effect similar to ping-pang buffer, so as to save input time.

3. output dma set

```
outbuf = aml_util_mallocAlignedBuffer(1000*2);
config.inOut.outAddr[0] = outbuf;
context = aml_module_create(&config);
```

note：

1. aml_util_swapInputBuffer needs to set output buffer size，For example, the classification result of the classification model is 1000 categories, and each category is a float 16 type, so outbuf needs a de length of 1000 * 2.

2. mallocAlignedBuffer and config.inOut.outAddr must be completed before aml_module_create， otherwise dma function will not be used normally.

3. If you use the input dma function, you need to flush cach. Otherwise, you may get the wrong result. When the input is small, you can choose not to do this operation. When the input size exceeds 224 * 224 * 3, you must perform the flush operation.

```
flush cache
memcpy(inbuf,rawdata,224*224*3)
aml_util_flushTensorHandle(context,AML_INPUT_TENSOR)
```

3.3.2 Introduction of SDK - GetTensorInfo function

```
inptr = aml_util_getInputTensorInfo(config.pdata);
outptr = aml_util_getOutputTensorInfo(config.pdata);
if(inptr->valid == 1)
{
    printf("the input buffer info is get success\n");
    printf("the input tensor number is %d\n",inptr->num);
    printf("the input[0] data_format:%d\n",inptr->info[0].data_format);
    printf("the input[0] quantization_format:%d\n",inptr->info[0].quantization_format);
    printf("the input[0] sizes_of_dim:%d,%d,%d,%d\n",inptr->info[0].sizes_of_dim[0],
inptr->info[0].sizes_of_dim[1],inptr->info[0].sizes_of_dim[2],inptr->info[0].sizes_of_dim[3]);
}

if(outptr->valid == 1){
    for (i=0;i<outptr->num;i++) {
        printf("the outptr tensor name is %s\n",outptr->info[i].name);
        printf("the outptr buffer info is get success\n");
        printf("the outptr tensor number is %d\n",outptr->num);
        printf("the outptr[%d] data_format:%d\n",i,outptr->info[i].data_format);
        printf("the outptr[%d] quantization_format:%d\n",i,outptr->info[i].
    quantization_format);
        printf("the outptr[%d] sizes_of_dim:%d,%d,%d,%d\n",i,
    outptr->info[i].sizes_of_dim[0],outptr->info[i].sizes_of_dim[1],outptr->info[i].sizes_of_dim[2],outptr->info[i].sizes_of_dim[3]);
    }
}
aml_util_freeTensorInfo(inptr);          //free inptr or outptr
aml_util_freeTensorInfo(outptr);
```

note：

1. config.pdata：Data information of nbg file.

2. The function of obtaining tensor information needs to be used in conjunction with the memory read nbg method to be used normally.

3. The information obtained includes dim_count , sizes_of_dim[4] , data_format, data_type, quantization_format, fixed_point_pos , TF_scale , TF_zeropoint, name

3.3.3 Introduction of SDK - mean value inconsistent setting mode

mean value set

```
inData.info.mean[0] = 123;      //the input mean will set as this;
inData.info.mean[1] = 116;
inData.info.mean[2] = 103;
```

note：

1.At present, the three channels of uint8 type mean are consistent by default. If the user's model three channel mean values are inconsistent, the user can set the mean value before aml_module_input_set.

2.The current mean value of int8 / int16 type is set to 128 by default. If the user model mean is not 128, you can set mean value before aml_module_input_set .

Code example:

```
context = aml_module_create(&config);
inData.typeSize = sizeof(inData);
inData.input_index = 0;
inData.size = 320*320*3;
inData.input_type = RGB24_RAW_DATA;
inData.info.valid = 1;
inData.info.mean[0] = 123;
inData.info.mean[1] = 116;
inData.info.mean[2] = 103;
aml_module_input_set(context,&inData);
```

3.3.4 Introduction of SDK - usb camera function

USB camera can be displayed on framebuffer in real time by calling camera_thread_func and net_thread_func two threads.

```
context = init_network(argc,argv);
pthread_mutex_init(&mutex_data,NULL);
init_fb();
if (0 != pthread_create(&tid[0],NULL,camera_thread_func,NULL))
```

```
{
        fprintf(stderr, "Couldn't create thread func\n");
        return -1;
}
thread_args = (void*)argv[3];
if (0 != pthread_create(&tid[1],NULL,net_thread_func,thread_args))
{
        fprintf(stderr, "Couldn't create thread func\n");
        return -1;
}
while(1)
{
        for (i=0;i<2;i++)
                pthread_join(tid[i], NULL);
}
```

Among them, camera_thread_func integrates the calling mode of usb camera. net_thread_func copies the video stream information read from the camera, and uses it as the input information of the model operation after resize.

```
dup_rgbbuf = (unsigned char*)malloc(display_width * display_high * 3);
input_data = (unsigned char*)malloc(input_width * input_high * 3);
while(1)
{
        if(rgbbuf != NULL)
        {
                pthread_mutex_lock(&mutex_data);
                memcpy(dup_rgbbuf,rgbbuf,display_width*display_high*3);
                pthread_mutex_unlock(&mutex_data);
        }
        resize_input_data(dup_rgbbuf,input_data);
        ret = run_network(netType,context,input_data,AML_IN_CAMERA,dup_rgbbuf);
}
```

note：

1. The current resolution is set to 640 * 480 by default, and supports 1920 * 1080, 1280 * 720 and 640 * 480 resolutions.

3.3.5 Introduction of SDK - profile control mode

aml_util_setProfile(AML_PROFILE_MEMORY,"save_log_path");

Four modes can be set in setProfile：

AML_PROFILE_NONE means that Profile information is not output.
AML_PROFILE_PERFORMANCE means the running time of the output model, that is, the performance index.
AML_PROFILE_BANDWIDTH means the bandwidth information consumed by the output model.
AML_PROFILE_MEMORY means the memory occupied by the output model.

aml_util_setProfile operation needs to be completed before init_network. By setting different modes, the corresponding information can be output and saved to the specified path. AML_PROFILE_PERFORMANCE can not save the log currently.

note：

1. save_ log_ path must be set to a path with permissions at the board.

2. If different modes are set at the same time, a refresh operation is required after the setProfile, that is aml_util_setProfile(AML_PROFILE_NONE,NULL). Then set another mode.

3. To obtain the forward reasoning time, set the number of cycles by setting environment variables

```
export NN_LOOP_TIME=100
```

3.3.6 Introduction of SDK - PowerPolicy control mode

aml_util_setPowerPolicy(AML_MINIMUM_POWER_MODE);

setPowerPolicy can set three modes:

AML_PERFORMANCE_MODE means the full computing power operation model.

AML_POWER_SAVE_MODE means the half computing power operation model.
AML_MINIMUM_POWER_MODE means the quarter computing power operation model.

setPowerPolicy operation needs to be completed before init_network.

note：

1. The running time of the three models is approximately linear, that is, the relationship of two times and four times of the total computing power.

2. If different modes are set at the same time, a refresh operation is required after the setProfile, that is aml_util_setProfile(AML_PROFILE_NONE,NULL). Then set another mode.

3.3.7 Introduction of SDK - get HardwareStatus info

int aml_util_getHardwareStatus(int* customID,int *powerStatus)

customID is the chip PID ,which is divided into six platforms: 7d, 88, 99, a1, b9, be.

powerStatus is the current four state of the power supply, POWER_IDLE，POWER_ON，POWER_OFF，POWER_RESET.

Through this interface, chip information and power status can be obtained to facilitate user debugging.

3.3.8 Introduction of ffmpeg interface

SDK supports input video data, and uses ffmpeg to decode the video for network input. The image obtained by parsing the video and resize is fed into the model as rawdata input data.

```
sprintf(cmd,"ffmpeg -i %s -qscale:v 2 -r 24 ",jpath);

ptr=strcat(cmd,"tmp/image%5d.bmp");

system(ptr);

sprintf(img_name,"tmp/image%05d.bmp",index);

while (1)

{

    sprintf(img_name,"tmp/image%05d.bmp",index);

    if((access(img_name,F_OK)) < 0)

    {

        break;

    }

    img=imread(img_name,199);

    resize(img,dst,dst.size());

    ret = run_network(netType,context,dst.data,AML_IN_VIDEO,(unsigned char*)img_name);

    index++;

}

system("ffmpeg -f image2 -i tmp/image%5d.bmp -b:v 5626k videoout.mp4");
```

3.3.9 SDK - save NPU input data

Currently inData.input_type supports input data in multiple formats of RGB24_RAW_DATA, TENSOR_RAW_DATA, QTENSOR_RAW_DATA, BINARY_RAW_DATA, INPUT_DMA_DATA, in order to facilitate confirmation that the pre-processing process is normal and the input data passed to the NPU is expected. You can set the debug flag for data pre-processing and save the input information.

```
inData.info.preprocess_debug=1;
```

After running the model, a text file named "input_data_[input_num]_[input size].txt" will be saved, which can be compared with the PC simulation results.

3.4 reference code

```
int main(int argc,char **argv)
{
    void *context;
    const char *jpeg_path = NULL;
    unsigned char *rawdata;
    aml_config config;
    nn_input inData;
    nn_output *outdata = NULL;
    int ret;
    int i;
    nn_query *query;
    jpeg_path = (const char *)argv[2];
    config.modelType = TENSORFLOW;

    /****************    load nbg from memory    **************/
    fp = fopen(argv[1],"rb");
    fseek(fp,0,SEEK_END);
    size = (int)ftell(fp);
    rewind(fp);
    config.pdata = (char *)calloc(1,size);
    fread((void*)config.pdata,1,size,fp);
    config.nbgType = NN_NBG_MEMORY;
    config.size = size;
    fclose(fp);
    /****************    load nbg from file    **************/
    config.path= (const char *)argv[1];
    config.nbgType = NN_NBG_FILE;


    context = aml_module_create(&config);
    rawdata = get_jpeg_rawData(jpeg_path,224,224);
    inData.input_index = 0;    //this value is index of input,begin from 0
    inData.size = 224*224*3;
    inData.input = rawdata;
    inData.input_type = BINARY_RAW_DATA;
    ret = aml_module_input_set(context,&inData);
    if(rawdata != NULL)
    {
        free(rawdata);
        rawdata = NULL;
    }
    outdata = (nn_output *)aml_module_output_get_simple(context);
    if(outdata->out[0].param->data_format == NN_BUFFER_FORMAT_FP32)
    {
        process_top5((float*)outdata->out[0].buf,outdata->out[0].size/sizeof(float),NULL)
    ;}
    ret = aml_module_destroy(context);
    return ret;
}
```

Outdata contains complete output information, including output dim, output format, output content and so on. The output content is float32 format by default, which can be implemented after relevant network post-processing. Such as mobilenet：

```
process_top5((float*)pout->out[0].buf,pout->out[0].size/sizeof(float));
```

```
void process_top5(float *buf,unsigned int num,img_classify_out_t* clsout)
{
    int j = 0;
    unsigned int MaxClass[5]={0};
    float fMaxProb[5]={0.0};

    float *pfMaxProb = fMaxProb;
    unsigned int   *pMaxClass = MaxClass,i = 0;

    for (j = 0; j < 5; j++)
    {
        for (i=0; i<num; i++)
        {
            if ((i == *(pMaxClass+0)) || (i == *(pMaxClass+1)) || (i == *(pMaxClass+2)) ||
                (i == *(pMaxClass+3)) || (i == *(pMaxClass+4)))
            {
                continue;
            }
            if (buf[i] > *(pfMaxProb+j))
            {
                *(pfMaxProb+j) = buf[i];
                *(pMaxClass+j) = i;
            }
        }
    }
    for (i=0; i<5; i++)
    {
        if (clsout == NULL)
        {
            printf("%3d: %8.6f\n", MaxClass[i], fMaxProb[i]);
        }
        else
        {
            clsout->score[i] = fMaxProb[i];
            clsout->topClass[i] = MaxClass[i];
        }
    }
}
```

3.5 Model performance debug interface

1. aml_util_setProfile(aml_profile_type_t type,const char *savepath);

Obtain model operation indicators

1.1. Savepath

Specify the save path of the board, DDK saves the relevant information to the specified path

1.2 aml_profile_type_t type

Profile debug way

A. AML_PROFILE_PERFORMANCE

Obtain model run_graph time

B. AML_PROFILE_BANDWIDTH

Obtain the bandwidth information when the model is running and write it to the file

C. AML_PROFILE_MEMORY

Obtain the memory usage information of the model when it is running and write it to the file

D. AML_PERLAYER_RUNTIME

Get the running time of each layer of the model and write it to the file

E. AML_PERLAYER_BANDWIDTH

Obtain the bandwidth information of each layer of the model and write it to the file

F. AML_PERLAYER_OUTPUT

Obtain the running results of each layer of the model and write them to the file

Note: D/E/F function will be improved in subsequent versions, and it is not currently supported

2. aml_util_setPowerPolicy(aml_policy_type_t type);

Set power strategy by controlling NPU frequencytypedef enum {

```
        AML_PERFORMANCE_MODE              = 1,      // nanoqFreq
        AML_POWER_SAVE_MODE               = 2,     // nanoqFreq/2
        AML_MINIMUM_POWER_MODE            = 3      // nanoqFreq/4
    } aml_policy_type_t;
```

3. aml_util_getHardwareStatus(int* customID,int *powerStatus,int* version);

Obtain hardware info

A.Obtain hardware ID
B.Get current power status
C.Get DDK version information

4. aml_util_setAutoSuspend(int timeout);

Set the sleep time, when the NPU has no tasks within the specified time, the system will automatically enter the power-off state