

密级状态：绝密( ) 秘密( ) 内部( ) 公开(√)

## Rockchip User Guide RKNN API

(技术部，图形计算平台中心)

文件状态： [ ] 正在修改 [√] 正式发布	当前版本：	1.4.0
	作 者：	杨华聪
	完成日期：	2020-09-10
	审 核：	卓鸿添
	完成日期：	2020-09-10

瑞芯微电子股份有限公司

Rockchips Semiconductor Co., Ltd

(版本所有, 翻版必究)

## 更新记录

版本	修改人	修改日期	修改说明	核定人
v0.9.6	杨华聪	2018-12-20	初始版本	熊伟
V0.9.7	杨华聪	2019-01-16	1) 添加 RKNN SDK 开发流程说明 2) 添加 RKNN Python API 说明	熊伟
V0.9.8	杨华聪	2019-06-05	1) 添加 RKNN API 库的说明 2) 更新 rknn example 和 rknn-toolkit 路径 3) 修正文档中错误	卓鸿添
V1.3.3	卓鸿添	2020-06-02	1) 支持 RV1126/RV1109 2) 删除 python 支持说明，关于 python 支持见 RKNN Toolkit Lite 相关文档。	卓鸿添
V1.4.0	卓鸿添	2020-09-10	1) 增加错误码	卓鸿添

## 目 录

1 主要功能说明.....	4
2 硬件平台.....	4
3 使用说明.....	4
3.1 RKNN SDK 开发流程.....	4
3.2 RKNN C API.....	5
3.2.1 RKNN API 库.....	5
3.2.2 EXAMPLE 使用说明.....	5
3.2.3 API 详细说明.....	6
3.2.3.1 <i>rknn_init</i> .....	6
3.2.3.2 <i>rknn_destroy</i> .....	6
3.2.3.3 <i>rknn_query</i> .....	7
3.2.3.4 <i>rknn_inputs_set</i> .....	9
3.2.3.5 <i>rknn_run</i> .....	10
3.2.3.6 <i>rknn_outputs_get</i> .....	10
3.2.3.7 <i>rknn_outputs_release</i> .....	11
3.2.4 RKNN 数据结构定义.....	12
3.2.4.1 <i>rknn_input_output_num</i> .....	12
3.2.4.2 <i>rknn_tensor_attr</i> .....	12
3.2.4.3 <i>rknn_input</i> .....	13
3.2.4.4 <i>rknn_output</i> .....	14
3.2.4.5 <i>rknn_perf_detail</i> .....	14
3.2.4.6 <i>rknn_sdk_version</i> .....	14
3.2.5 RKNN 返回值错误码.....	15

## 1 主要功能说明

RKNN SDK 为 RK1808 等带有 NPU 的平台提供编程接口，能够帮助用户部署使用 RKNN-Toolkit 导出的 RKNN 模型，加速 AI 应用的落地。

## 2 硬件平台

本文档适用如下硬件平台：

1) RK1808、RK1806

2) RV1126、RV1109

注意：本文档不适用 RK3399Pro

下面的说明以 RK1808 为例，也同时适用上述其他平台。

## 3 使用说明

### 3.1 RKNN SDK 开发流程

在使用 RKNN SDK 之前，用户首先需要使用 RKNN-Toolkit 工具将用户的模型转换为 RKNN 模型，用户可以在 <https://github.com/rockchip-linux/rknn-toolkit> 获取工具的完整安装包及使用文档。

成功转换生成 RKNN 模型之后，用户可以先通过 RKNN-Toolkit 连接 RK1808 等开发板进行联机调试，确保模型的精度性能符合要求。

得到 RKNN 模型文件之后，用户可以选择使用 C 或 Python 接口在 RK1808 等平台开发应用，后续章节将说明如何在 RK1808 等平台上基于 RKNN SDK 进行开发。

## 3.2 RKNN C API

### 3.2.1 RKNN API 库

RKNN SDK 所提供的库和头文件位于<sdk>/external/rknpu/rknn/  
rknn\_api/librknn\_api 目录下，开发者可以在自己应用中引用即可开发应用。

需要注意的是，RK1808 和 RK3399Pro 平台的 RKNN API 是兼容的，两者开发的应用程序可以很方便地移植。但是使用过程中需要注意要区分两个平台的 librknn\_api.so，如果开发者使用 RK3399Pro 的 librknn\_api.so 将无法在 RK1808 平台上运行。开发者可以使用以下方法来区分 librknn\_api.so 的平台：

```
$ strings librknn_api.so |grep version  
librknn_api version 1.4.0 (60ffbd build: 2020-09-10 19:24:19)
```

### 3.2.2 EXAMPLE 使用说明

SDK 提供了 Linux 平台的 MobileNet 图像分类、MobileNet SSD 目标检测以及 YoloV3 Tiny 目标检测 Demo。这些 Demo 能够为客户基于 RKNN SDK 开发自己的 AI 应用提供参考。Demo 代码位于<sdk>/external/rknpu/rknn/rknn\_api/examples 目录。下面以 rknn\_mobilenet\_demo 为例来讲解如何快速上手运行。

#### 1) 编译 Demo

```
cd examples/rknn_mobilenet_demo  
mkdir build && cd build  
cmake ..  
make && make install  
cd -
```

#### 2) 部署到 RK1808 设备

```
adb push install/rknn_mobilenet_demo /userdata/
```

#### 3) 运行 Demo

```
adb shell
cd /userdata/rknn_mobilenet_demo/
./rknn_mobilenet_demo mobilenet_v1.rknn dog_224x224.jpg
```

### 3.2.3 API 详细说明

#### 3.2.3.1 rknn\_init

rknn\_init 初始化函数将创建 rknn\_context 对象、加载 RKNN 模型以及根据 flag 执行特定的初始化行为。

API	rknn_init
功能	初始化 rknn
参数	rknn_context *context: rknn_context 指针。函数调用之后，context 将会被赋值。
	void *model: RKNN 模型的二进制数据。
	uint32_t size: 模型大小。
	uint32_t flag: 特定的初始化标志。目前 RK1808 平台仅支持以下标志：  <b>RKNN_FLAG_COLLECT_PERF_MASK</b> : 打开性能收集调试开关，打开之后能够通过 rknn_query 接口查询网络每层运行时间。需要注意，该标志被设置后 rknn_run 的运行时间将会变长。
返回值	int 错误码（见 <a href="#">rknn 返回值错误码</a> ）。

示例代码如下：

```
rknn_context ctx;
int ret = rknn_init(&ctx, model_data, model_data_size, 0);
```

#### 3.2.3.2 rknn\_destroy

rknn\_destroy 函数将释放传入的 rknn\_context 及其相关资源。

API	rknn_destroy
-----	--------------

功能	销毁 rknn_context 对象及其相关资源。
参数	rknn_context context: 要销毁的 rknn_context 对象。
返回值	int 错误码（见 <a href="#">rknn 返回值错误码</a> ）。

示例代码如下：

```
int ret = rknn_destroy (ctx);
```

### 3.2.3.3 rknn\_query

rknn\_query 函数能够查询获取到模型输入输出、运行时间以及 SDK 版本等信息。

API	rknn_query
功能	查询模型与 SDK 的相关信息。
参数	rknn_context context: rknn_context 对象。
	rknn_query_cmd cmd: 查询命令。
	void* info: 存放返回结果的结构体变量。
	uint32_t size: info 对应的结构体变量的大小。
返回值	int 错误码（见 <a href="#">rknn 返回值错误码</a> ）

当前 SDK 支持的查询命令如下表所示：

查询命令	返回结果结构体	功能
RKNN_QUERY_IN_OUT_NUM	<a href="#">rknn_input_output_num</a>	查询输入输出 Tensor 个数
RKNN_QUERY_INPUT_ATTR	<a href="#">rknn_tensor_attr</a>	查询输入 Tensor 属性
RKNN_QUERY_OUTPUT_ATTR	<a href="#">rknn_tensor_attr</a>	查询输出 Tensor 属性
RKNN_QUERY_PERF_DETAIL	<a href="#">rknn_perf_detail</a>	查询网络各层运行时间
RKNN_QUERY_SDK_VERSION	<a href="#">rknn_sdk_version</a>	查询 SDK 版本

接下来的将依次详解各个查询命令如何使用。

#### 1) 查询输入输出 Tensor 个数

传入 RKNN\_QUERY\_IN\_OUT\_NUM 命令可以查询模型输入输出 Tensor 的个数。其中需

要先创建 rknn\_input\_output\_num 结构体对象。

示例代码如下：

```
rknn_input_output_num io_num;
ret = rknn_query(ctx, RKNN_QUERY_IN_OUT_NUM, &io_num,
                 sizeof(io_num));
printf("model input num: %d, output num: %d\n", io_num.n_input,
       io_num.n_output);
```

## 2) 查询输入 Tensor 属性

传入 RKNN\_QUERY\_INPUT\_ATTR 命令可以查询模型输入 Tensor 的属性。其中需要先创建 rknn\_tensor\_attr 结构体对象。

示例代码如下：

```
rknn_tensor_attr input_attrs[io_num.n_input];
memset(input_attrs, 0, sizeof(input_attrs));
for (int i = 0; i < io_num.n_input; i++) {
    input_attrs[i].index = i;
    ret = rknn_query(ctx, RKNN_QUERY_INPUT_ATTR, &(input_attrs[i]),
                     sizeof(rknn_tensor_attr));
}
```

## 3) 查询输出 Tensor 属性

传入 RKNN\_QUERY\_OUTPUT\_ATTR 命令可以查询模型输出 Tensor 的属性。其中需要先创建 rknn\_tensor\_attr 结构体对象。

示例代码如下：

```
rknn_tensor_attr output_attrs[io_num.n_output];
memset(output_attrs, 0, sizeof(output_attrs));
for (int i = 0; i < io_num.n_output; i++) {
    output_attrs[i].index = i;
    ret = rknn_query(ctx, RKNN_QUERY_OUTPUT_ATTR, &(output_attrs[i]),
                     sizeof(rknn_tensor_attr));
}
```

## 4) 查询网络各层运行时间

如果在 rknn\_init 函数调用时有设置 RKNN\_FLAG\_COLLECT\_PERF\_MASK 标志，那么在执行 rknn\_run 完成之后，可以传入 RKNN\_QUERY\_PERF\_DETAIL 命令来查询网络每层运



行时间。其中需要先创建 rknn\_perf\_detail 结构体对象。

示例代码如下：

```
rknn_perf_detail perf_detail;  
ret = rknn_query(ctx, RKNN_QUERY_PERF_DETAIL, &perf_detail,  
                sizeof(rknn_perf_detail));  
printf("%s", perf_detail.perf_data);
```

注意，用户不需要释放 rknn\_perf\_detail 中的 perf\_data，SDK 会自动管理该 Buffer 内存。

### 5) 查询 SDK 版本

传入 RKNN\_QUERY\_SDK\_VERSION 命令可以查询 RKNN SDK 的版本信息。其中需要先创建 rknn\_sdk\_version 结构体对象。

示例代码如下：

```
rknn_sdk_version version;  
ret = rknn_query(ctx, RKNN_QUERY_SDK_VERSION, &version,  
                sizeof(rknn_sdk_version));  
printf("sdk api version: %s\n", version.api_version);  
printf("driver version: %s\n", version.drv_version);
```

#### 3.2.3.4 rknn\_inputs\_set

通过 rknn\_inputs\_set 函数可以设置模型的输入数据。该函数能够支持多个输入，其中每个输入是 rknn\_input 结构体对象，在传入之前用户需要设置该对象。

API	rknn_inputs_set
功能	设置模型输入数据。
参数	rknn_context context: rknn_context 对象。
	uint32_t n_inputs: 输入数据个数。
	rknn_input inputs[]: 输入数据数组，数组每个元素是 rknn_input 结构体对象。
返回值	int 错误码（见 <a href="#">rknn 返回值错误码</a> ）

示例代码如下：

```
rknn_input inputs[1];
memset(inputs, 0, sizeof(inputs));
inputs[0].index = 0;
inputs[0].type = RKNN_TENSOR_UINT8;
inputs[0].size = img_width*img_height*img_channels;
inputs[0].fmt = RKNN_TENSOR_NHWC;
inputs[0].buf = in_data;

ret = rknn_inputs_set(ctx, 1, inputs);
```

### 3.2.3.5 rknn\_run

rknn\_run 函数将执行一次模型推理，调用之前需要先通过 rknn\_inputs\_set 函数设置输入数据。

API	rknn_run
功能	执行一次模型推理。
参数	rknn_context context: rknn_context 对象。
	rknn_run_extend* extend: 保留扩展，当前没有使用，传入 NULL 即可。
返回值	int 错误码（见 <a href="#">rknn 返回值错误码</a> ）

示例代码如下：

```
ret = rknn_run(ctx, NULL);
```

### 3.2.3.6 rknn\_outputs\_get

rknn\_outputs\_get 函数可以获取模型推理的输出数据。该函数能够一次获取多个输出数据。其中每个输出是 rknn\_output 结构体对象，在函数调用之前需要依次创建并设置每个 rknn\_output 对象。

对于输出数据的 buffer 存放可以采用两种方式：一种是由用户自行申请和释放，此时 rknn\_output 对象的 is\_prealloc 需要设置为 1，并且将 buf 指针指向用户申请的 buffer；另一种是由 rknn 来进行分配，此时 rknn\_output 对象的 is\_prealloc 设置为 0 即可，函数执行之后 buf 将指向输出数据。

API	rknn_outputs_get
功能	获取模型推理输出
参数	rknn_context context: rknn_context 对象。
	uint32_t n_outputs: 输出数据个数
	rknn_output outputs[]: 输出数据的数组，其中数组每个元素为 rknn_output 结构体对象，代表模型的一个输出。
	rknn_output_extend* extend: 保留扩展，当前没有使用，传入 NULL 即可。
返回值	int 错误码（见 <a href="#">rknn 返回值错误码</a> ）

示例代码如下：

```
rknn_output outputs[io_num.n_output];
memset(outputs, 0, sizeof(outputs));
for (int i = 0; i < io_num.n_output; i++) {
    outputs[i].want_float = 1;
}
ret = rknn_outputs_get(ctx, io_num.n_output, outputs, NULL);
```

### 3.2.3.7 rknn\_outputs\_release

rknn\_outputs\_release 函数将释放之前获取得到的输出的相关资源。

API	rknn_outputs_release
功能	释放 rknn_output 对象
参数	rknn_context context: rknn_context 对象
	uint32_t n_outputs: 输出数据个数
	rknn_output outputs[]: 要销毁的 rknn_output 数组
返回值	int 错误码（见 <a href="#">rknn 返回值错误码</a> ）

示例代码如下

```
ret = rknn_outputs_release(ctx, io_num.n_output, outputs);
```

### 3.2.4 RKNN 数据结构定义

#### 3.2.4.1 rknn\_input\_output\_num

结构体 rknn\_input\_output\_num 表示输入输出 Tensor 个数，其结构体成员变量如下表所示：

成员变量	数据类型	含义
n_input	uint32_t	输入 Tensor 个数
n_output	uint32_t	输出 Tensor 个数

#### 3.2.4.2 rknn\_tensor\_attr

结构体 rknn\_tensor\_attr 表示模型的 Tensor 的属性，结构体的定义如下表所示：

成员变量	数据类型	含义
index	uint32_t	表示输入输出 Tensor 的索引位置。
n_dims	uint32_t	Tensor 维度个数。
dims	uint32_t[]	Tensor 各维度值。
name	char[]	Tensor 名称。
n_elems	uint32_t	Tensor 数据元素个数。
size	uint32_t	Tensor 数据所占内存大小。
fmt	rknn_tensor_format	Tensor 维度的格式，有以下格式：  <b>RKNN_TENSOR_NCHW</b>  <b>RKNN_TENSOR_NHWC</b>
type	rknn_tensor_type	Tensor 数据类型，有以下数据类型：  <b>RKNN_TENSOR_FLOAT32</b>  <b>RKNN_TENSOR_FLOAT16</b>  <b>RKNN_TENSOR_INT8</b>

		<b>RKNN_TENSOR_UINT8</b>  <b>RKNN_TENSOR_INT16</b>
qnt_type	rknn_tensor_qnt_type	Tensor 量化类型，有以下的量化类型：  <b>RKNN_TENSOR_QNT_NONE</b> ：未量化；  <b>RKNN_TENSOR_QNT_DFP</b> ：动态定点量化；  <b>RKNN_TENSOR_QNT_AFFINE_ASYMMETRIC</b> ：非对称量化。
fl	int8_t	RKNN_TENSOR_QNT_DFP 量化类型的参数。
zp	uint32_t	RKNN_TENSOR_QNT_AFFINE_ASYMMETRIC 量化类型的参数。
scale	float	RKNN_TENSOR_QNT_AFFINE_ASYMMETRIC 量化类型的参数。

### 3.2.4.3 rknn\_input

结构体 rknn\_input 表示模型的一个数据输入，用来作为参数传入给 rknn\_inputs\_set 函数。

结构体的定义如下表所示：

成员变量	数据类型	含义
index	uint32_t	该输入的索引位置。
buf	void*	输入数据 Buffer 的指针。
size	uint32_t	输入数据 Buffer 所占内存大小。
pass_through	uint8_t	设置为 1 时会将 buf 存放的输入数据直接设置给模型的输入节点，不做任何预处理。
type	rknn_tensor_type	输入数据的类型。
fmt	rknn_tensor_format	输入数据的格式。

#### 3.2.4.4 rknn\_output

结构体 `rknn_output` 表示模型的一个数据输出，用来作为参数传入给 `rknn_outputs_get` 函数，在函数执行后，结构体对象将会被赋值。结构体的定义如下表所示：

成员变量	数据类型	含义
want_float	uint8_t	标识是否需要将输出数据转为 float 类型输出。
is_prealloc	uint8_t	标识存放输出数据的 Buffer 是否是预分配。
index	uint32_t	该输出的索引位置。
buf	void*	输出数据 Buffer 的指针。
size	uint32_t	输出数据 Buffer 所占内存大小。

#### 3.2.4.5 rknn\_perf\_detail

结构体 `rknn_perf_detail` 表示模型的性能详情，结构体的定义如下表所示：

成员变量	数据类型	含义
perf_data	char*	性能详情包含网络每层运行时间，能够直接打印出来查看。
data_len	uint64_t	存放性能详情的字符串数组的长度。

#### 3.2.4.6 rknn\_sdk\_version

结构体 `rknn_sdk_version` 用来表示 RKNN SDK 的版本信息，结构体的定义如下：

成员变量	数据类型	含义
api_version	char[]	SDK 的版本信息。
drv_version	char[]	SDK 所基于的驱动版本信息。

### 3.2.5 RKNN 返回值错误码

RKNN API 函数的返回值错误码定义如下表所示

错误码	错误详情
RKNN_SUCC (0)	执行成功
RKNN_ERR_FAIL (-1)	执行出错
RKNN_ERR_TIMEOUT (-2)	执行超时
RKNN_ERR_DEVICE_UNAVAILABLE (-3)	NPU 设备不可用
RKNN_ERR_MALLOC_FAIL (-4)	内存分配失败
RKNN_ERR_PARAM_INVALID (-5)	传入参数错误
RKNN_ERR_MODEL_INVALID (-6)	传入的 RKNN 模型无效
RKNN_ERR_CTX_INVALID (-7)	传入的 rknn_context 无效
RKNN_ERR_INPUT_INVALID (-8)	传入的 rknn_input 对象无效
RKNN_ERR_OUTPUT_INVALID (-9)	传入的 rknn_output 对象无效
RKNN_ERR_DEVICE_UNMATCH (-10)	版本不匹配
RKNN_ERR_INCOMPATIBLE_PRE_COMPILE_MODEL (-11)	RKNN 模型使用 pre_compile 模式，但是和当前驱动不兼容
RKNN_ERR_INCOMPATIBLE_OPTIMIZATION_LEVEL_VERSION (-12)	RKNN 模型设置了优化等级的选项，但是和当前驱动不兼容
RKNN_ERR_TARGET_PLATFORM_UNMATCH (-13)	RKNN 模型和当前平台不兼容，一般是将 RK1808 的平台 RKNN 模型放到了 RV1109/RV1126 上。