密级状态：绝密(　　)　　秘密(　　)　　内部(　　)　　公开(√)

RKNN API For RK356X User Guide

（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：	0.6.0
	作　　者：	HPC
	完成日期：	2021-03-01
	审　　核：	熊伟
	完成日期：	2021-03-01

瑞芯微电子股份有限公司

Rockchips　Semiconductor　Co . , Ltd

更新记录

版本	修改人	修改日期	修改说明	核定人
v0.6.0	HPC	2021-3-1	初始版本	熊伟

目 录

1 主要功能说明

RKNN SDK 为带有 NPU 的 RK3566,RK3568 芯片平台提供编程接口，能够帮助用户部署使用 RKNN-Toolkit2 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：

RK3566、RK3568

注：文档部分地方使用 RK356X 来表示 RK3566、RK3568。

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-Toolkit2 工具将用户的模型转换为 RKNN 模型。

得到 RKNN 模型文件之后，用户可以选择使用 C 接口在 RK3566、RK3568 平台开发应用，后续章节将说明如何在 RK3566、RK3568 平台上基于 RKNN SDK 进行开发。

3.2 RKNN Linux 平台开发说明

3.2.1 Linux 平台 RKNN API 库

对于 RK3566、RK3568，SDK 库文件为<sdk>/rknpu2/下的 librknn_api.so

3.2.1.1 Linux 平台 Demo

SDK 提供了 Linux 平台的 MobileNet 图像分类、SSD 目标检测示例。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>/rknpu2/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
#设置 build-linux.sh 下的 GCC_COMPILER 为正确的编译器路径
./build-linux.sh
```

2） 部署到 RK3566 或 RK3568 设备

```
adb push install/rknn_mobilenet_demo_Linux /userdata/
```

3） 运行 Demo

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

3.3 RKNN ANDROID 平台开发说明

3.3.1 ANDROID 平台 RKNN API 库

对于需要通过 CTS 测试的 Android 设备可以使用 Android 平台的 RKNN API。Android 平台的 RKNN API 位于 Android 系统 SDK 的 vendor/rockchip/hardware/interfaces/neuralnetworks 目录下。当完成 Android 系统编译后，将会生成一些 NPU 相关的库，如下所示：

```
/system/lib64/librknn_api.so
/system/lib64/librknnhal_bridge.rockchip.so
/vendor/lib64/rockchip.hardware.neuralnetworks@1.0.so
/vendor/lib64/rockchip.hardware.neuralnetworks@1.0-adapter-helper.so
/vendor/lib64/librknnrt.so
/vendor/lib64/hw/rockchip.hardware.neuralnetworks@1.0-impl.so
```

其中对于应用只需要链接使用 librknn_api.so 即可调用开发，该库的接口为 RKNN C API。

3.3.2 EXAMPLE 使用说明

目前，SDK 提供了 MobileNet 图像分类、SSD 目标检测示例。Demo 代码位于<sdk>/rknpu2/examples 目录。用户可以使用 NDK 编译 Android 命令行中执行的 demo。下面以 rknn_mobilenet_demo 为例来讲解在 Android 平台上该 demo 如何使用：

1）编译 Demo

```
cd examples/rknn_mobilenet_demo
#设置 build-android.sh 下的 ANDROID_NDK_PATH 为正确的 NDK 路径
./build-android.sh
```

2）部署到 RK3566 或 RK3568 设备

```
adb push install/rknn_mobilenet_demo_Android /data/
```

3）运行 Demo

```
adb shell
cd /data/rknn_mobilenet_demo_Android/
./rknn_mobilenet_demo model/mobilenet_v1.rknn model/dog_224x224.jpg
```

3.4 RKNN C API

3.4.1 API 流程说明

首先，用户使用 rknn_inputs_set 函数设置模型输入，等待推理结束后，使用 rknn_outputs_get 函数获取推理的输出。API 调用流程如图 3-1 所示。

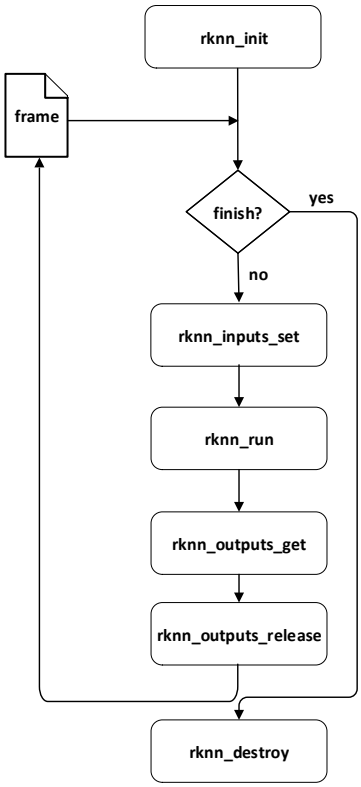

图 3-1 RKNN SDK API 调用流程

3.4.1.1 API 内部处理流程

在推理 RKNN 模型时，原始数据要经过输入处理、NPU 运行模型、输出处理三大流程。在典型的图片推理场景中，假设输入数据 data 是 3 通道的图片且为 NHWC 排布格式，运行时（Runtime）对数据处理的流程如图 3-2 所示。在 API 层面上，rknn_inputs_set 接口（当 pass_through=0 时，详见 rknn_input 结构体）包含了颜色通道交换、归一化、量化、NHWC 转换成 NCHW/NC1HWC2 的过程，rknn_outputs_get 接口（当 want_float=1 时，详见 rknn_output 结构体）包含了 NC1HWC2 转换 NCHW 和反量化的过程。

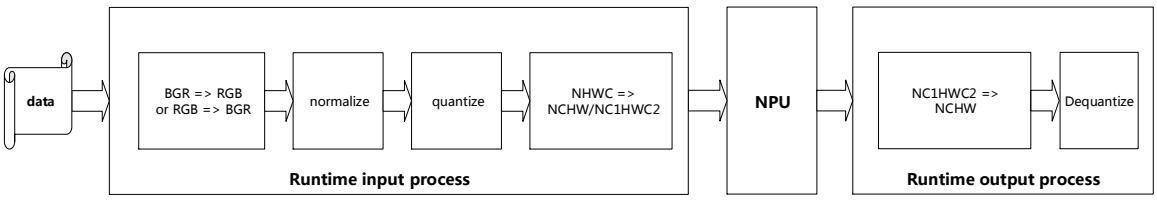

图 3-2 完整的图片数据处理流程

3.4.1.2 量化和反量化

量化和反量化用到的量化方式、量化数据类型以及量化参数，可以通过 rknn_query 接口查询。

目前，RK3566/RK3568 只支持非对称量化，不支持动态定点量化，数据类型和量化方式组合包括：

- int8（非对称量化）
- int16（非对称量化，暂未实现）
- float16

通常，归一化后的数据用 32 位浮点数保存，32 位浮点转换成 16 位浮点数请参考 IEEE-754 标准。假设归一化后的 32 位浮点数据是 D，下面介绍量化流程：

1）float32 转 int8（非对称量化）

假设输入 tensor 的非对称量化参数是 S_q， ZP，数据 D 量化过程表示为下式：

$$D_q = round(clamp(D / S_q + ZP, -128, 127))$$

上式中，clamp 表示将数值限制在某个范围。round 表示做舍入处理。

2）float32 转 int16（非对称量化）

假设输入 tensor 的非对称量化参数是 S_q， ZP，数据 D 量化过程表示为下式：

$$D_q = round(clamp(D / S_q + ZP, -32768, 32767))$$

反量化流程是量化的逆过程，可以根据上述量化公式反推出反量化公式，这里不做赘述。

3.4.2 API 详细说明

3.4.2.1 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 RK3566，RK3568 平台暂不支持设置标志。
返回值	int 错误码（见 rknn 返回值错误码）。

示例代码如下：

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

3.4.2.2 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 rknn 返回值错误码）。

示例代码如下：

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

3.4.2.3 rknn_query

rknn_query 函数能够查询获取到模型输入输出以及 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 rknn 返回值错误码）

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	rknn_input_output_num	查询输入输出 Tensor 个数
RKNN_QUERY_INPUT_ATTR	rknn_tensor_attr	查询输入 Tensor 属性
RKNN_QUERY_OUTPUT_ATTR	rknn_tensor_attr	查询输出 Tensor 属性
RKNN_QUERY_PERF_DETAIL	rknn_perf_detail	查询网络各层运行时间（RK3566,RK3568 暂不支持该功能）
RKNN_QUERY_SDK_VERSION	rknn_sdk_version	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 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);
```

3.4.2.4 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_contex 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 rknn 返回值错误码）

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);
```

3.4.2.5 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 rknn 返回值错误码）

示例代码如下：

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

3.4.2.6 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 rknn 返回值错误码）

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);
```

3.4.2.7 rknn_outputs_release

rknn_outputs_release 函数将释放 rknn_outputs_get 函数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 rknn 返回值错误码）

示例代码如下

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

3.4.3 RKNN 数据结构定义

3.4.3.1 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

3.4.3.2 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： **RKNN_TENSOR_NCHW** **RKNN_TENSOR_NHWC**
type	rknn_tensor_type	Tensor 数据类型，有以下数据类型： **RKNN_TENSOR_FLOAT32** **RKNN_TENSOR_FLOAT16** **RKNN_TENSOR_INT8** **RKNN_TENSOR_UINT8** **RKNN_TENSOR_INT16**
qnt_type	rknn_tensor_qnt_type	Tensor 量化类型，有以下的量化类型： **RKNN_TENSOR_QNT_NONE**：未量化； **RKNN_TENSOR_QNT_DFP**：动态定点量化； **RKNN_TENSOR_QNT_AFFINE_ASYMMETRIC**：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_ASYMMETRI

		C 量化类型的参数。
scale	float	RKNN_TENSOR_QNT_AFFINE_ASYMMETRIC 量化类型的参数。

3.4.3.3 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。（目前，RK3566 和 RK3568 暂不支持 pass_through=1 的配置）
type	rknn_tensor_type	输入数据的类型。
fmt	rknn_tensor_format	输入数据的格式。

3.4.3.4 rknn_tensor_mem

结构体 rknn_tensor_mem 表示 tensor 初始化后的存储状态信息，目前，RK3566 和 RK3568 还未提供使用该结构体的接口，暂时保留以便后续扩展。结构体的定义如下表所示：

成员变量	数据类型	含义
logical_addr	void*	该输入的虚拟地址。
physical_addr	uint64_t	该输入的物理地址。
fd	int32_t	该输入的 fd。
size	uint32_t	该输入 tensor 占用的内存大小。

handle	uint32_t	该输入的 handle。
priv_data	void*	保留的数据。
reserved_flag	uint64_t	保留的标志位。

3.4.3.5 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。

3.4.3.6 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。

3.4.3.7 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。

3.4.4 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_INCOMPATILE_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。