Classification Level: Top Secret( ) Secret( ) Internal( ) Public(√)

RK1808 RKNN SDK DEVELOPER GUIDE

(Technology Department, Graphic Display Platform Center)

Mark: [ ] Changing [√] Released	Version:	v0.9.8
	Author:	Yang Huacong
	Completed Date:	5/Jun/2019
	Reviewer:	Zhuo Hongtian
	Reviewed Date:	5/Jun/2019

福州瑞芯微电子股份有限公司

Fuzhou Rockchip Electronics Co., Ltd.

Revision History

Version	Modifier	Date	Modify Description	Reviewer
v0.9.7	Yang Huacong	25/Jan/2019	Initial version	Zhuo Hongtian
V0.9.8	Yang Huacong	5/Jun/2019	1. Add RKNN API Library description 2. Update rknn example and rknn-toolkit path 3. Fix some mistake	Zhuo Hongtian

Table of Contents

1 Overview

RK1808 RKNN SDK provides programming interfaces for RK1808 NPU. It can help users deploy RKNN model on RK1808 Linux platform. Based on RKNN SDK, developer can quickly develop AI applications with NPU acceleration on RK1808 platform. his SDK mainly includes the following two parts:

1） RKNN C API: Support C/C++ programming, provide RKNN model loading, running, data input and output and debugging interfacesRKNN Python API: Support Python3 programming, also provide RKNN model loading, running, data input and output and debugging interfaces.

2 Dependencies

The RKNN SDK needs to rely on the following:

1）The library included in the RK1808 RKNN SDK supports running on the RK1808 AArch64 linux system.

2） RK1808 RKNN Python SDK needs to rely on Python 3.6 and python-numpy packages.

3 Instructions

3.1 RKNN SDK Development Process

Before using the RK1808 RKNN SDK, users first need to convert the user's trained model to the RKNN model using the RKNN-Toolkit tool on the PC. Users can get the complete installation package of the RKNN-Toolkit in the <rk1808-linux-sdk>/external/rknn-toolkit/ directory. Please refer to the RKNN-Tookit User Guide for the use of RKNN-Toolkit.

After successful conversion to the RKNN model, users can first connect to the RK1808 device via RKNN-Toolkit for online debugging to ensure that the accuracy and performance of the model

meet the requirements.

After getting the RKNN model file, users can choose using C or Python interface to develop the application. The following chapters will explain how to develop application based on the RKNN SDK on RK1808 platform.

3.2 RKNN C API

3.2.1 RK1808 RKNN API Library

The libraries and header files provided by the RK1808 RKNN SDK are located at <rk1808-linux-sdk>/external/rknpu/rknn/ rknn_api/librknn_api directory, developers can use to develop applications.

It should be noted that the RKNN API of the RK1808 and RK3399Pro platforms is compatible, and applications developed by both can be easily ported. However, you need to pay attention to distinguish between the two platforms librknn_api.so, if the developer uses RK3399Pro's librknn_api.so will not be able to run on the RK1808 platform. Developers can use the following methods to distinguish the platform of librknn_api.so.

```
$ strings librknn_api.so |grep version
RK1808 librknn_api version 0.9.8 (58c7577 build: 2019-06-05 17:00:44)
```

3.2.2 EXAMPLES

The SDK provides MobileNet image classification, MobileNet SSD object detection, and Yolo v3 object detection demos. These demos provide reference for developer to develop applications based on the RKNN SDK. The emo code is located in the <rk1808-linux-sdk>/external/rknpu/rknn/rknn_api/examples directory. Let's take rknn_mobilenet_demo as an example to explain how to get started quickly.

1) Compile Demo Source Code

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

2) Deploy to the RK1808 device

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

3) Run Demo

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

3.2.3 API Reference

3.2.3.1 rknn_init

The *rknn_init* function will create a *rknn_context* object, load the RKNN model, and perform specific initialization behavior based on the flag.

API	rknn_init
Description	Initialize rknn
Parameters	*rknn_context *context*：Pointer to *rknn_context* object. After the function is called, the context object will be assigned.
	*void *model*：Binary data for the RKNN model.
	uint32_t size：Model size
	uint32_t flag：A specific initialization flag. Currently supports following flags: **RKNN_FLAG_COLLECT_PERF_MASK**：Open the performance collection debugging switch. After opening, you can query the running time of each layer of the network through the *rknn_query* interface. Note that the running time of *rknn_run* will be

	longer after this flag is set.
Return	int: Error code (See RKNN Error Code).

Sample Code:

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

3.2.3.2 rknn_destroy

The *rknn_destroy* function will release the *rknn_context* object and its associated resources.

API	rknn_destroy
Description	Destroy the rknn_context object and its related resources.
Parameters	*rknn_context context*: The rknn_context object to be destroyed.
Return	int: Error code (See RKNN Error Code).

Sample Code:

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

3.2.3.3 rknn_query

The *rknn_query* function can query the information of model input and output tensor attribute, performance information and SDK version etc.

API	rknn_query
Description	Query the information about the model and the SDK.
Parameters	*rknn_context context*: The object of *rknn_contex*.
	rknn_query_cmd cmd: Query command.
	void info*: Structure object that stores the result of the query.
	uint32_t size: the size of the info Structure object.
Return	int: Error code (See RKNN Error Code).

Currently, the SDK supports the following query commands:

Query command	Return result structure	Function
RKNN_QUERY_IN_OUT_NUM	rknn_input_output_num	Query the number of input and output Tensor.
RKNN_QUERY_INPUT_ATTR	rknn_tensor_attr	Query input Tensor attribute.
RKNN_QUERY_OUTPUT_ATTR	rknn_tensor_attr	Query output Tensor attribute.
RKNN_QUERY_PERF_DETAIL	rknn_perf_detail	Query the running time of each layer of the network.
RKNN_QUERY_SDK_VERSION	rknn_sdk_version	Query the SDK version.

Next we will explain each query command in detail.

1） Query the number of input and output Tensor

The *RKNN_QUERY_IN_OUT_NUM* command can be used to query the number of model input and output Tensor. You need to create the *rknn_input_output_num* structure object first.

Sample Code:

```
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） Query input Tensor attribute

The *RKNN_QUERY_INPUT_ATTR* command can be used to query the attribute of the model input Tensor. You need to create the *rknn_tensor_attr* structure object first.

Sample Code:

```
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） Query output Tensor attribute

The *RKNN_QUERY_OUTPUT_ATTR* command can be used to query the attribute of the model output Tensor. You need to create the *rknn_tensor_attr* structure object first.

Sample Code:

```
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） Query the running time of each layer of the network

If the *RKNN_FLAG_COLLECT_PERF_MASK* flag has been set on the *rknn_init* function, the *RKNN_QUERY_PERF_DETAIL* command can be passed to query the runtime of each layer of the network after *rknn_run* function execution completed. You need to create the *rknn_perf_detail* structure object first.

Sample Code:

```
    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);
```

It should be noted that *rknn_perf_detail.perf_data* does not need to be released. The SDK will automatically manage this buffer memory.

5） Query the SDK version

The *RKNN_QUERY_SDK_VERSION* command can be used to query the version information

of the RKNN SDK. You need to create the *rknn_sdk_version* structure object first.

Sample Code:

```
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

The input data of the model can be set by the *rknn_inputs_set* function. This function can support multiple inputs, each one is a *rknn_input* structure object. Developers needs to set these object field before passing in.

API	rknn_inputs_set
Description	Set the model input data.
Parameter	*rknn_context context*: The object of rknn_contex.
	uint32_t n_inputs: Number of inpus.
	rknn_input inputs[]: Array of rknn_input.
Return	int: Error code (See RKNN Error Code).

Sample Code:

```
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

The *rknn_run* function will perform a model reasoning. The input data need to be set by the *rknn_inputs_set* function before *rknn_run is* called.

API	rknn_run
Description	Perform a model reasoning.
Parameter	*rknn_context context*: The object of rknn_contex.
	rknn_run_extend extend*: Reserved for extension, currently not used, you can pass NULL.
Return	int: Error code (See RKNN Error Code).

Sample Code:

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

3.2.3.6 rknn_outputs_get

The *rknn_outputs_get* function can get the output data of the model reasoning. This function can get multiple output data. Each of these outputs is a *rknn_output* structure object, which needs to be created and set in turn before the function is called.

There are two ways to store buffers for output data:

1) Developer allocate and release buffers themselves. At this time, the *rknn_output.is_prealloc* needs to be set to 1, and the *rknn_output.buf* points to users' allocated buffer;

2) The other is allocated by SDK. At this time, the *rknn_output .is_prealloc* needs to be set to 0. After the function is executed, *rknn_output.buf* will be created and store the output data.

API	rknn_outputs_get
Description	Get model inference output data.
Parameter	*rknn_context context*: The object of rknn_context.
	uint32_t n_outputs: Number of output.
	rknn_output outputs[]: Array of rknn_output.
	rknn_run_extend extend*: Reserved for extension, currently not used, you can pass NULL.
Return	int: Error code (See RKNN Error Code).

Sample Code:

```
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

The *rknn_outputs_release* function will release the relevant resources of the *rknn_output* object.

API	rknn_outputs_release
Description	Release the rknn_output object
Parameter	*rknn_context context*: rknn_context object
	uint32_t n_outputs: Number of output.
	rknn_output outputs[]: rknn_output array to be release.
Return	int: Error code (See RKNN Error Code).

Sample Code:

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

3.2.4 RKNN DataStruct Define

3.2.4.1 rknn_input_output_num

The structure *rknn_input_output_num* represents the number of input and output Tensor，The following table shows the definition:

Field	Type	Meaning
n_input	uint32_t	The number of input tensor
n_output	uint32_t	The number of output tensor

3.2.4.2 rknn_tensor_attr

The structure *rknn_tensor_attr* represents the attribute of the model's Tensor. The following table shows the definition:

Field	Type	Meaning
index	uint32_t	Indicates the index position of the input and output Tensor.
n_dims	uint32_t	The number of Tensor dimensions.
dims	uint32_t[]	Values for each dimension.
name	char[]	Tensor name.
n_elems	uint32_t	The number of Tensor data elements.
size	uint32_t	The memory size of Tensor data.
fmt	rknn_tensor_format	The format of Tensor dimension, has the following format: **RKNN_TENSOR_NCHW** **RKNN_TENSOR_NHWC**

type	rknn_tensor_type	Tensor data type, has the following data types: **RKNN_TENSOR_FLOAT32** **RKNN_TENSOR_FLOAT16** **RKNN_TENSOR_INT8** **RKNN_TENSOR_UINT8** **RKNN_TENSOR_INT16**
qnt_type	rknn_tensor_qnt_type	Tensor Quantization Type, has the following types of quantization: **RKNN_TENSOR_QNT_NONE:** Not quantified; **RKNN_TENSOR_QNT_DFP**: Dynamic fixed point quantization; **RKNN_TENSOR_QNT_AFFINE_ASYMMETRIC**: Asymmetric quantification.
fl	int8_t	RKNN_TENSOR_QNT_DFP quantization parameter
zp	uint32_t	RKNN_TENSOR_QNT_AFFINE_ASYMMETRIC quantization parameter.
scale	float	RKNN_TENSOR_QNT_AFFINE_ASYMMETRIC quantization parameter.

3.2.4.3 rknn_input

The structure *rknn_input* represents a data input to the model, used as a parameter to the *rknn_inputs_set* function. The following table shows the definition:

Field	Type	Meaning
index	uint32_t	The index position of this input.
buf	void*	Point to the input data Buffer.
size	uint32_t	The memory size of the input data Buffer.

pass_through	uint8_t	When set to 1, buf will be directly set to the input node of the model without any pre-processing.
type	rknn_tensor_type	The type of input data.
fmt	rknn_tensor_format	The format of input data.

3.2.4.4 rknn_output

The structure *rknn_output* represents a data output of the model, used as a parameter to the *rknn_outputs_get* function. The following table shows the definition:

Field	Type	Meaning
want_float	uint8_t	Indicates if the output data needs to be converted to float type.
is_prealloc	uint8_t	Indicates whether the Buffer that stores the output data is pre-allocated.
index	uint32_t	The index position of this output.
buf	void*	Pointer which point to the output data Buffer.
size	uint32_t	Output data Buffer memory size.

3.2.4.5 rknn_perf_detail

The structure *rknn_perf_detail* represents the performance details of the model. The following table shows the definition:

Field	Type	Meaning
perf_data	char*	Performance details include the run time of each layer of the network.
data_len	uint64_t	The Length of perf_data.

3.2.4.6 rknn_sdk_version

The structure *rknn_sdk_version* is used to indicate the version information of the RKNN SDK. The following table shows the definition:

Field	Type	Meaning
api_version	char[]	SDK API Version information.
drv_version	char[]	Driver version information.

3.2.5 RKNN Error Code

The return code of the RKNN API function is defined as shown in the following table.

Error Code	Message
RKNN_SUCC	Execution is successful
RKNN_ERR_FAIL	Execution error
RKNN_ERR_TIMEOUT	Execution timeout
RKNN_ERR_DEVICE_UNAVAILABLE	NPU device is unavailable
RKNN_ERR_MALLOC_FAIL	Memory allocation is failed
RKNN_ERR_PARAM_INVALID	Parameter error
RKNN_ERR_MODEL_INVALID	RKNN model is invalid
RKNN_ERR_CTX_INVALID	rknn_context is invalid
RKNN_ERR_INPUT_INVALID	rknn_input object is invalid
RKNN_ERR_OUTPUT_INVALID	rknn_output object is invalid
RKNN_ERR_DEVICE_UNMATCH	Version does not match

3.3 RKNN PYTHON API

3.3.1 RKNN PYTHON API Enable

The RKNN Python API is not enabled by default in the RK1808 Linux SDK. If want to use it,

you need to add the corresponding configuration.

You can add configuration to *"buildroot/configs/rockchip_rk1808_defconfig"* file. Then recompile the firmware.

```
diff --git a/configs/rockchip_rk1808_defconfig
b/configs/rockchip_rk1808_defconfig
index 1659453be8..fb5303bd00 100644
--- a/configs/rockchip_rk1808_defconfig
+++ b/configs/rockchip_rk1808_defconfig
@@ -19,3 +19,15 @@ BR2_PACKAGE_MINIGUI_ENABLE_PNG=y
 BR2_PACKAGE_RKNPU=y
 BR2_PACKAGE_RKNN_DEMO=y
 BR2_PACKAGE_RKWIFIBT_BTUART="ttyS4"
+
+BR2_PACKAGE_PYTHON3=y
+BR2_PACKAGE_PYTHON3_PY_PYC=y
+BR2_PACKAGE_PYTHON_NUMPY_ARCH_SUPPORTS=y
+BR2_PACKAGE_PYTHON_NUMPY=y
+BR2_PACKAGE_OPENCV3=y
+BR2_PACKAGE_OPENCV3_LIB_PYTHON=y
+BR2_PACKAGE_OPENCV3_LIB_IMGCODECS=y
+BR2_PACKAGE_OPENCV3_LIB_IMGPROC=y
+BR2_PACKAGE_OPENCV3_WITH_JPEG=y
+
+BR2_PACKAGE_PYTHON_RKNN=y
```

Alternatively, you can use the "make menuconfig" command to enable the above configuration one by one. After opening, you also need to recompile the firmware.

After compiling and downloading the firmware, you can execute the following command on the shell command line of the RK1808 device to confirm whether RKNN Python is normally turned on:

```
# python
>>> from rknn.api import RKNN
```

3.3.2 EXAMPLES

RKNN Python sample code is in the *<rk1808_linux_sdk>/external/rknpu/rknn/python/example* directory. Let's take mobilenet_v1 as an example to see how to run:

1） Push example to RK1808 device

```
adb push mobilenet_v1 /userdata/
```

2） running

```
adb shell
cd /userdata/mobilenet_v1
python test.py
```

After successful execution, the output is as follows:

```
/userdata/mobilenet_v1 # python test.py
--> Loading RKNN model
done
--> Running model
mobilenet_v1
-----TOP 5-----
[156]: 0.8232421875
[155]: 0.0513916015625
[205]: 0.0250244140625
[188]: 0.0225830078125
[880]: 0.01351165771484375

done
```

3.3.3 API Reference

3.3.3.1 RKNN create and release

When using RKNN Python API, you need to initialize an RKNN object. After doing all the operations, you need to call *release* function to release the RKNN object.

Sample Code:

```
rknn = RKNN()
...
rknn.release()
```

3.3.3.2 load_rknn

The *load_rknn* function can load a RKNN model file.

API	load_rknn
Description	Load the RKNN model.
Parameters	*path*: RKNN model file path.
Return	0: Load successfully
	-1: Load failed

Sample Code:

```
ret = rknn.load(path='./mobilenet_v1.rknn')
if ret != 0:
    print('Load rknn model failed!')
    exit(ret)
```

3.3.3.3 init_runtime

Before model inference or performance evaluation, you must initialize the runtime environment and set required configuration parameters.

API	init_runtime
Description	Initialize the runtime environment.
Parameter	*perf_debug*: Whether to enable debug mode when performing performance evaluation. In debug mode, you can get the running time of each layer. The default is False.
Return	0: Initialize the runtime environment successfully.
	-1: Initialize the runtime environment failed.

Sample Code:

```
ret = rknn.init_runtime()
if ret != 0:
    print('Init runtime environment failed')
    exit(ret)
```

3.3.3.4 inference

The inference function can perform model reasoning. Note that before inference, the *load_rknn* function must be called to load the RKNN model.

API	inference
Description	Perform model reasoning.
Parameter	*inputs*: Input data to be inferred, such as images processed by cv2. The type is ndarray list.
	data_type: The type of input data, can use with the following values: 'float32', 'float16', 'int8', 'uint8', 'ing16'. The default is 'uint8'.
	data_format: Data shape format, you can use the following values: "nchw", "nhwc". The default is 'nhwc'.
	outputs: Specifies the format of the output data. The type is ndarray list, which is used to specify the shape and dtype of the output. The default value is None, and the returned data dtype is float32.
Return	results: The result of the inference, the type is ndarray list.

Sample Code:

```
img = cv2.imread('./dog_224x224.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
outputs = rknn.inference(inputs=[img])
```

3.3.3.5 eval_perf

The *eval_perf* function can get model performance information.

API	eval_perf
Description	Evaluate model performance. If you set perf_debug to False on rknn_runtime function, you get the total time of the model running; If you set perf_debug to True on rknn_runtime function, in addition to returning the total time, it will also return the time-consuming of each layer of network.
Parameter	*inputs*: Input data to be inferred, such as images processed by cv2. The format is ndarray list.
	data_type: The type of input data, can use with the following values: "*float32*", "*float16*", "*int8*", "*uint8*", "*int16*". The default is "*uint8*".
	data_format: Data arrangement format, you can use the following values: "*nchw*", "nhwc". The default is "*nhwc*".
	is_print: Whether to output performance evaluation results in the canonical format. The default is True.
Return	perf_result: Performance information. The type is a dictionary.

Sample Code:

```
rknn.eval_perf(inputs=[img])
```