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

RKNN Toolkit Lite User Guide

(Technology Department, Graphic Computing Platform Center)

Mark: [ ] Editing [√] Released	Version	V1.7.0
	Author	Rao Hong
	Completed Date	2021-08-08
	Reviewer	Vincent
	Reviewed Date	2021-08-08

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd

Revision History

Version	Modifier	Date	Modify description	Reviewer
V1.4.0	Rao Hong	2020-08-13	Initial version.	Vincent
V1.6.0	Rao Hong	2020-12-31	Update version.	Vincent
V1.6.1	Rao Hong	2021-05-21	Update version.	Vincent
V1.7.0	Rao Hong	2021-08-08	Update version.	Vincent

Table of Contents

1 Overview

RKNN Toolkit Lite is a simplified version of RKNN Toolkit, providing users with a development kit for model inference on PC, RK3399Pro, RK1808, RK1806, RV1109, RV1126. Users can easily and quickly complete the development of AI applications and deployment on hardware devices such as RK1808 through the Python interface provided by the SDK.

1.1 Applicable chip

- RK1806
- RK1808
- RK3399Pro(D/X)
- RV1109
- RV1126

1.2 Applicable system

- Ubuntu: 16.04(x64) or later
- Windows：7(x64) or later
- MacOS: 10.13.15(x64) or later.
- Debian: 9.8(aarch64) or later.
- Debian: 10(armhf)

2 Requirements/Dependencies

This software development kit supports running on the Ubuntu, Windows, Mac OS X or Debian operating system. It is recommended to meet the following requirements in the operating system environment:

Table 1 Operating system environment

Operating system version	Ubuntu16.04（x64）or later Windows 7 (x64) or later Mac OS X 10.13.5 (x64) or later Debian 9.8 (x64) or later Debian 10(armhf)
Python version	3.5/3.6
Python library dependencies	'numpy == 1.16.3' 'ruamel.yaml == 0.15.81' 'psutils == 5.6.2'

Note:

1. Windows only support Python 3.6 currently.
2. MacOS support python 3.6 and python 3.7.
3. Arm 64bit platform support python 3.5 and python 3.7.
4. Arm 32bit platform support python 3.7.

3 User Guide

3.1 Installation

There are two ways to install RKNN Toolkit Lite: one is via pip install command, the other is running docker image with full RKNN Toolkit Lite environment. The specific steps of the two installation ways are described below.

3.1.1 Install by pip command

1. Create virtualenv environment. If there are multiple versions of the Python environment in the system, it is recommended to use virtualenv to manage the Python environment.

```
sudo apt install virtualenv
sudo apt-get install libpython3.5-dev
sudo apt install python3-tk

virtualenv -p /usr/bin/python3 venv
source venv/bin/activate
```

2. Install dependent libraries: opencv-python

```
# Ubuntu 16.04 / 18.04 or Windows 7/10 or MacOS Catalina
pip3 install opencv-python

# Debian 9.8 with Python3.5
pip3 install \
opencv_python_headless-4.0.1.23-cp35-cp35m-linux_aarch64.whl

# Debian 10 with Python3.7
sudo apt-get install python3-dev python3-opencv
```

Note: RKNN Toolkit Lite itself does not rely on opencv-python, but the example will use this library to load image, so the library is also installed here.

3. Install RKNN Toolkit Lite

   The installation packages of each platform are placed in the *packages* folder. Enter *packages* folder and execute command below to install RKNN Toolkit Lite:

```
# Ubuntu 16.04
pip3 install rknn_toolkit_lite-1.7.0-cp35-cp35m-linux_x86_64.whl

# Ubuntu 18.04
pip3 install rknn_toolkit_lite-1.7.0-cp36-cp36m-linux_x86_64.whl

# Windows
pip install rknn_toolkit_lite-1.7.0-cp36-cp36m-win_amd64.whl

# MacOS python3.6
pip3 install rknn_toolkit_lite-1.7.0-cp36-cp36m-macosx_10_15_x86_64.whl

# MacOS python3.7
pip3 install rknn_toolkit_lite-1.7.0-cp37-cp37m-macosx_10_15_x86_64.whl

# Debian9
pip3 install rknn_toolkit_lite-1.7.0-cp35-cp35m-linux_aarch64.whl

# Debian10 aarch64
pip3 install rknn_toolkit_lite-1.7.0-cp37-cp37m-linux_aarch64.whl

# Debian10 armhf
pip3 install rknn_toolkit_lite-1.7.0-cp37-cp37m-linux_armv7l.whl
```

3.1.2 Install by the Docker Image

In docker folder, there is a Docker image that has been packaged for all development requirements, Users only need to load the image and can directly use RKNN Toolkit Lite, detailed steps are as follows:

1. Install Docker

   Please install Docker according to the official manual:

   https://docs.docker.com/install/linux/docker-ce/ubuntu/

2. Load Docker image

   Execute the following command to load Docker image:

```
docker load --input rknn-toolkit-lite-1.7.0-docker.tar.gz
```

After loading successfully, execute "docker images" command and the image of rknn-toolkit appears as follows:

```
REPOSITORY          TAG          IMAGE ID              CREATED          SIZE
rknn-toolkit-lite   1.7.0        607159f28bd3          1 hours ago      1.31GB
```

3. Run image

Execute the following command to run the docker image. After running, it will enter the bash environment.

```
docker run -t -i --privileged -v /dev/bus/usb:/dev/bus/usb rknn-toolkit-lite:1.7.0 /bin/bash
```

If you want to map your own code, you can add the “-v <host src folder>:<image dst folder>” parameter, for example:

```
docker run -t -i --privileged -v /dev/bus/usb:/dev/bus/usb -v /home/rk/test:/test rknn-toolkit-lite:1.7.0 /bin/bash
```

4. Run demo

```
cd /examples/inference_with_lite
python3 test.py
```

Note: The image is based on the Ubuntu 18.04 of aarch64, so it can only run on the RK1808 or RK3399Pro development board with the Debian system. There can only be one container at a time using RKNN Toolkit Lite for inference. And make sure that no npu_transfer_proxy is running on the host machine before using.

3.2 Usage of RKNN Toolkit Lite

3.2.1 usage scenarios

The usage scenarios of RKNN Toolkit Lite can be divided into two types：

- Running on a PC：at this time the PC needs to connect to a hardware device with a chip such as RK1808 via USB.
- Run directly on RK1808, RK3399Pro and other development boards equipped with Debian system.

3.2.2 Use process

The use process of RKNN Toolkit Lite is as follows：

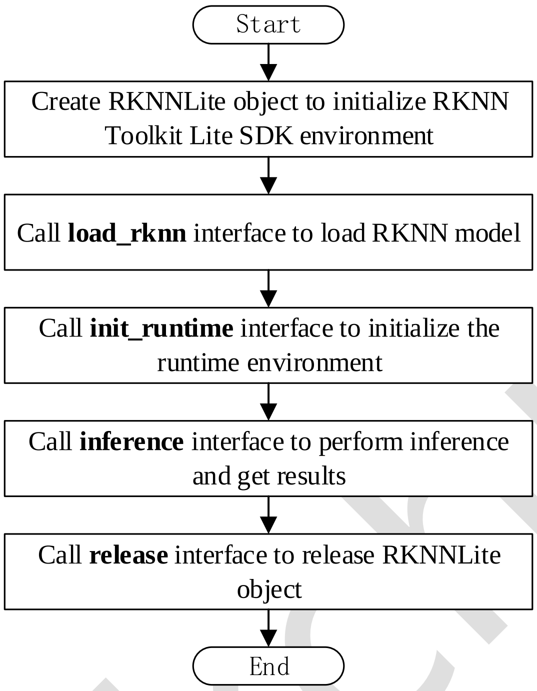
Start
Create RKNNLite object to initialize RKNN Toolkit Lite SDK environment
Call **load_rknn** interface to load RKNN model
Call **init_runtime** interface to initialize the runtime environment
Call **inference** interface to perform inference and get results
Call **release** interface to release RKNNLite object
End

Figure 3-2-2-1 Usage flow of RKNN Toolkit Lite

3.3 Example

An example “inference_with_lite” using RKNN Toolkit Lite for model inference is provided in the SDK/examples directory:

```
import platform
import cv2
import numpy as np
from rknnlite.api import RKNNLite

INPUT_SIZE = 224


def show_top5(result):
    output = result[0].reshape(-1)
    # softmax
    output = np.exp(output)/sum(np.exp(output))
    output_sorted = sorted(output, reverse=True)
    top5_str = 'resnet18\n-----TOP 5-----\n'
    for i in range(5):
        value = output_sorted[i]
```

```
            index = np.where(output == value)
            for j in range(len(index)):
                if (i + j) >= 5:
                    break
                if value > 0:
                    topi = '{}: {}\n'.format(index[j], value)
                else:
                    topi = '-1: 0.0\n'
                top5_str += topi
        print(top5_str)


if __name__ == '__main__':
    rknn_lite = RKNNLite()

    # load RKNN model
    print('--> Load RKNN model')
    ret = rknn_lite.load_rknn('./resnet_18.rknn')
    if ret != 0:
        print('Load RKNN model failed')
        exit(ret)
    print('done')

    ori_img = cv2.imread('./space_shuttle_224.jpg')
    img = cv2.cvtColor(ori_img, cv2.COLOR_BGR2RGB)

    # init runtime environment
    print('--> Init runtime environment')
    # run on RK3399Pro/RK1808 with Debian OS, do not need specify
    # target.
    if platform.machine() == 'aarch64':
        target = None
    else:
        target = 'rk1808'
    ret = rknn_lite.init_runtime(target=target)
    if ret != 0:
        print('Init runtime environment failed')
        exit(ret)
    print('done')

    # Inference
    print('--> Running model')
    outputs = rknn_lite.inference(inputs=[img])
    show_top5(outputs)
    print('done')

    rknn_lite.release()
```

After installing RKNN Toolkit Lite execute the following command to run the demo：

```
python3 test.py
```

When performing model inference, the result of this demo is as follows:

```
-----TOP 5-----
[812]: 0.999442994594574
[404]: 0.0004096269840374589
[657]: 3.284541890025139e-05
[833]: 2.6112385967280716e-05
[895]: 1.8509887013351545e-05
```

4 RKNN Toolkit Lite API description

4.1 RKNNLite object initialization and release

The initialization/release function group consists of API interfaces to initialize and release the RKNNLite object as needed. The **RKNNLite()** must be called before using all the API interfaces of RKNN Toolkit Lite, and call the **release()** method to release the object when task finished.

When we init RKNNLite object, we can set ***verbose*** and ***verbose_file*** parameters, used to show detailed log information of model loading, inferencing and so on. The data type of verbose parameter is bool. If we set the value of this parameter to True, the RKNN Toolkit Lite will show detailed log information on screen. The data type of verbose_file is string. If we set the value of this parameter to a file path, the detailed log information will be written to this file (**the verbose also need be set to True**).

The sample code is as follows:

```
# Show the detailed log information on screen, and saved to
# mobilenet_build.log
rknn_lite = RKNNLite(verbose=True, verbose_file='./inference.log')
# Only show the detailed log information on screen.
rknn_lite = RKNNLite(verbose=True)
…
rknn_lite.release()
```

4.2 Loading RKNN model

API	**load_rknn**
Description	Load RKNN model
Parameter	path: The path of RKNN model file.
	load_model_in_npu: Whether to load RKNN model in NPU directly. The path parameter should fill in the path of the model in NPU. It can be set to True only when RKNN Toolkit Lite run on RK3399Pro Linux Develop board or PC with NPU device is connected. Default value is False.
Return Value	0: Load successfully
	-1: Load failed

The sample code is as follows:

```
# Load the resnet_18 RKNN model in the current path
ret = rknn_lite.load_rknn('./resnet_18.rknn')
```

4.3 Initialize the runtime environment

Before inference or performance evaluation, the runtime environment must be initialized. This interface determines which type of runtime hardware is specified to run model.

API	**init_runtime**
Description	Initialize the runtime environment. Set the device information (hardware platform, device ID).
Parameter	target: Target hardware platform, now supports “rk3399pro”, “rk1806”, “rk1808”, “rv1109”, “rv1126”. The default value is None, which indicates model runs on default hardware platform and system. Specifically, if RKNN Toolkit Lite is used in RK3399Pro or RK1808 Linux development board, the default device is NPU that comes with RK3399Pro or RK1808. The “rk1808” includes TB-RK1808 AI Compute Stick.
	device_id: Device identity number, if multiple devices are connected to PC, this parameter needs to be specified which can be obtained by calling “***list_devices***” interface. The default value is “None “. Note: Mac OS X platform does not supple multiple devices.
	async_mode: Whether to use asynchronous mode. When calling the inference interface, it involves setting the input picture, model running, and fetching the inference result. If the asynchronous mode is enabled, setting the input of the current frame will be performed simultaneously with the inference of the previous frame, so in addition to the first frame, each subsequent frame can hide the setting input time, thereby improving performance. In asynchronous mode, the inference result returned each time is the previous frame. The default value for this parameter is False.

Return Value	0: Initialize the runtime environment successfully
	-1: Initialize the runtime environment failed

The sample code is as follows:

```
# Initialize the runtime environment
ret = rknn_lite.init_runtime(target='rk1808', device_id='012345789AB')
if ret != 0:
        print('Init runtime environment failed')
        exit(ret)
```

4.4 Inference with RKNN model

Before using the model for inference, you must load a RKNN model first.

API	**inference**
Description	Use the model to perform inference with specified input and get the inference result.
Parameter	inputs: Inputs to be inferred, such as images processed by cv2. The object type is list. List members are ndarray.
	data_type: The numerical type of input data. Optional values are 'float32', 'float16', 'int8', 'uint8', 'ing16'. The default value is 'uint8'.
	data_format: The shape format of input data. Optional values are "nchw", "nhwc". The default value is 'nhwc'.
	inputs_pass_through: Pass the input transparently to the NPU driver. In non-transparent mode, the tool will reduce the mean, divide the variance, etc. before the input is passed to the NPU driver; in transparent mode, these operations will not be performed. The value of this parameter is an array. For example, to pass input0 and not input1, the value of this parameter is [1, 0]. The default value is None, which means that all input is not transparent.
Return Value	results: The result of inference, the object type is list. List members are ndarray.

The sample code is as follows:

For classification model, such as resnet18, the code is as follows (refer to *examples/*

inference_with_lite for the complete code):

```
# Preform inference for a picture with a model and get a top-5 result
……
outputs = rknn_lite.inference(inputs=[img])
show_top5(outputs)
……
```

The result of top-5 is as follows:

```
-----TOP 5-----
[812]: 0.999442994594574
[404]: 0.00040962698403745 89
[657]: 3.284541890025139e-05
[833]: 2.6112385967280716e-05
[895]: 1.8509887013351545e-05
```

4.5 Get SDK version

API	**get_sdk_version**
Description	Get API version and driver version of referenced SDK. Note: Before we use this interface, we must load model and initialize runtime first.
Parameter	None
Return Value	sdk_version： API and driver version. Data type is string.

The sample code is as follows：

```
# Get SDK version
……
sdk_version = rknn_lite.get_sdk_version()
……
```

The SDK version looks like below：

```
==============================================
RKNN VERSION:
      API: 1.7.0 (f75fb8e build: 2021-07-20 16:23:11)
      DRV: 1.7.0 (e40021e build: 2021-08-05 09:50:13)
==============================================
```

4.6 List Devices

API	**list_devices**
Description	List connected RK3399PRO/RK1806/RK1808/RV1109/RV1126. Note: There are currently two device connection modes: ADB and NTB. RK1806/RK1808 and RV1109/RV1126 support both ADB and NTB, RK3399Pro only support ADB, TB-RK1808 AI Compute Stick only support NTB. Make sure their modes are the same when connecting multiple devices
Parameter	None

Return Value	Return adb_devices list and ntb_devices list. If there are no devices connected to PC, it will return two empty list.
	For example, there are two TB-RK1808 AI Compute Sticks connected to PC, it`s return looks like below:
	adb_devices = []
	ntb_devices = ['TB-RK1808S0', '515e9b401c060c0b']

The sample code is as follows：

```
from rknnlite.api import RKNNLite

if __name__ == '__main__':
        rknn_lite = RKNNLite()
        rknn_lite.list_devices()
        rknn_lite.release()
```

The devices list looks like below：

```
*************************
all device(s) with adb mode:
VII67Y7UKQ
all device(s) with ntb mode:
```

```
c3d9b8674f4b94f6,515e9b401c060c0b
*************************
```

Note: When using multiple devices, you need to ensure that their connection modes are consistent, otherwise it will cause conflicts, resulting in failure to initialize the runtime environment.

4.7 Query RKNN model runnable platform

API	**list_support_target_platform**
Description	Query the chip platform that a given RKNN model can run on.
Parameter	rknn_model: RKNN model path. If the model path is not specified, the chip platforms currently supported by the RKNN Toolkit Lite are printed by category
Return Value	support_target_platform (dict): Return the chip platform where the model can run. If the RKNN model path is empty or does not exist, return None

The sample code is as follows:

```
rknn_lite.list_support_target_platform(rknn_model='mobilenet_v1.rknn')
```

The runnable chip platforms look like below：

```
**************************************************
Target platforms filled in RKNN model:             ['RK1808']
Target platforms supported by this RKNN model: ['RK1806', 'RK1808', 'RK3399PRO']
**************************************************
```