Security Class: Top-Secret ( )　　Secret ( )　　Internal ( )　　Public (√)

Rockchip Quick Start RKNN-Toolkit EN

(Technology Department, Graphic Display Platform Center)

Mark: [ ] Editing [√] Released	Version	V1.2.0
	Author	Rao Hong
	Completed Date	2019-08-21
	Auditor	Randall
	Reviewed Date	2019-08-21

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Electronics Co., Ltd

Revision History

Version no.	Author	Revision Date	Revision description	Auditor
V0.9.9	Rao Hong	2019-03-25	Initial version release	Randall
V1.0.0	Rao Hong	2019-05-08	Synchronize the modification contents of RKNN-Toolkit-V1.0.0	Randall
V1.1.0	Rao Hong	2019-06-28	1. Synchronize the modification contents of RKNN-Toolkit-V1.1.0 2. Rename document, from <RKNN-Toolkit Quick Setup Guide> to <RKNN-Toolkit Quick Start> 3. Add quick start for Windows/Mac OS X/ARM64 platform.	Randall
V1.2.0	Rao Hong	2019-08-21	Synchronize the modification contents of RKNN-Toolkit-V1.2.0	Randall

Content

1 Main Features Introduction

RKNN-Toolkit provides for users the development kit of model conversion, inference and performance evaluation based on PC, RK3399Pro, RK1808, TB-RK1808 AI Compute Stick or RK3399Pro Linux development board. Users can easily implement below features with the provided python interface:

1） Model conversion: support to convert Caffe、TensorFlow、TensorFlow Lite、ONNX、Darknet model to RKNN model, support RKNN model import/export, which can be used on hardware platform later.

2） Quantization function: support to convert float model to quantization model, currently support quantized methods including asymmetric quantization (asymmetric_quantized-u8) and dynamic fixed point quantization (dynamic_fixed_point-8 and dynamic_fixed_point-16). Starting with V1.0.0, RKNN-Toolkit began to support hybrid quantization. For a detailed description of hybrid quantization, please refer to Section 3.3.

3） Model inference: able to simulate running model on PC and obtain the inference results. Also able to run model on specific hardware platform RK3399Pro (or RK3399Pro Linux development board), RK1808, TB-RK1808 AI Compute Stick and obtain the inference results.

4） Performance evaluation: able to simulate running on PC and obtain the total time consumption and each layer's time consumption of the model. Also able to run model with on-line debugging method on specific hardware platform RK3399Pro, RK1808, TB-RK1808 AI Compute Stick or directly run on RK3399Pro Linux development board to obtain the total time consumption and each layer's time consumption when the model runs completely once on the hardware.

5） Memory evaluation: obtain the memory usage through on-line debugging method when the model is running on specific hardware platform such as RK3399Pro, RK1808, TB-RK1808 AI Compute Stick or RK3399Pro Linux development board.

6） Model pre-compilation: with pre-compilation techniques, model loading time can be reduced, and for some models, model size can also be reduced. However, the pre-compiled RKNN model can only be run on a hardware platform with an NPU, and this feature is currently only supported by the x86_64 Ubuntu platform. RKNN-Toolkit supports the model pre-compilation feature from version V0.9.5, and the pre-compilation method has been upgraded in V1.0.0. The upgraded precompiled model is not compatible with the old driver.

7） Model segmentation: This function is used in a scenario where multiple models run simultaneously. A single model can be divided into multiple segments to be executed on the NPU, thereby adjusting the execution time of multiple models occupying the NPU, and avoiding other models because one model occupies too much execution time. RKNN-Toolkit supports this feature from version 1.2.0. This feature must be used on hardware with an NPU and the NPU driver version is greater than 0.9.8.

8） Custom OP: If the model contains an OP that is not supported by RKNN-Toolkit, it will fail during the model conversion phase. At this time, you can use the custom layer feature to define an unsupported OP so that the model can be converted and run normally. RKNN-Toolkit supports this feature from version 1.2.0.

2 System Dependency Introduction

This development kit supports running on Ubuntu / Windows / MacOS / Debian operation system with the following environment requirements:

Table 1 Running environment

Operation system version	Ubuntu16.04（x64）or higher Windows 7（x64）or higher Mac OS X 10.13.5（x64）or higher Debian 9.8（x64）or higher
Python version	3.5/3.6
Python library dependency	'numpy >= 1.16.1' 'scipy >= 1.1.0' 'Pillow >= 3.1.2' 'h5py >= 2.7.1' 'lmdb >= 0.92' 'networkx == 1.11' 'flatbuffers == 1.9', 'protobuf >= 3.5.2' 'onnx == 1.4.1' 'onnx-tf == 1.2.1' 'flask >= 1.0.2' 'tensorflow >= 1.11.0' 'dill==0.2.8.2' 'opencv-python>=3.4.3.18' 'ruamel.yaml==0.15.82' 'psutils>=5.6.2'

Note: Only support python3.6 wheel package for Windows and Mac OS X.

3 Ubuntu platform Quick Start Guide

This chapter mainly describes how to quickly setup and use RKNN-Toolkit based on Ubuntu 16.04, Python3.5.

3.1 Environment Preparation

- One x86_64 bit computer with ubuntu16.04
- One RK1808 EVB board.
- Connect RK1808 device to PC through USB, use 'adb devices' command to check, and the result is as below:

```
rk@rk:~$ adb devices
List of devices attached
0123456789ABCDEF         device
```

Note: "0123456789ABCDEF" is device id.

3.2 Install RKNN-Toolkit（Take Python3.5 as example）

1. Install Python3.5

```
sudo apt-get install python3.5
```

2. Install pip3

```
sudo apt-get install python3-pip
```

3. Obtain RKNN-Toolkit install package, and then execute below steps:

   a) Enter package directory:

```
cd package/
```

   b) Install Python dependency

```
pip3 install tensorflow
```

```
pip3 install opencv-python
```

c) Install RKNN-Toolkit

```
sudo pip3 install rknn_toolkit-1.2.0-cp35-cp35m-linux_x86_64.whl
```

d) Check if RKNN-Toolkit is installed successfully or not

```
rk@rk:~/rknn-toolkit-v1.2.0/package$ python3
>>> from rknn.api import RKNN
>>>
```

The installation is successful if the import of RKNN module doesn't fail.

3.3 Execute the example attached in the install package

3.3.1 Simulate the running example on PC

RKNN-Toolkit has a built-in RK1808 simulator which can be used to simulate the action of the model running on RK1808.

Here take mobilenet_v1 as example. mobilenet_v1 in the example is a Tensorflow Lite model, used for picture classification, and it is running on simulator.

The running steps are as below:

1. Enter example/mobilenet_v1 directory

```
rk@rk:~/rknn-toolkit-v1.2.0/package$ cd ../example/mobilenet_v1
rk@rk:~/rknn-toolkit-v1.2.0/example/mobilenet_v1$
```

2. Execute test.py script

```
rk@rk:~/rknn-toolkit-v1.2.0/example/mobilenet_v1$ python3 test.py
```

3. Get the results after the script execution as below:

```
--> config model
done
--> Loading model
done
```

```
--> Building model
done
--> Export RKNN model
done
--> Init runtime environment
W [RK_nn_softmax_compute:45]Softmax's beta is 0. Set beta to 1
done
--> Running model
mobilenet_v1
-----TOP 5-----
[156]: 0.8837890625
[155]: 0.0677490234375
[188 205]: 0.00867462158203125
[188 205]: 0.00867462158203125
[263]: 0.0057525634765625

done
--> Begin evaluate model performance
W [RK_nn_softmax_compute:45]Softmax's beta is 0. Set beta to 1
=====================================================================
                            Performance
=====================================================================
Layer ID    Name                                     Time(us)
0          tensor.transpose_3                          72
44         convolution.relu.pooling.layer2_2           363
59         convolution.relu.pooling.layer2_2           201
45         convolution.relu.pooling.layer2_2           185
60         convolution.relu.pooling.layer2_2           243
46         convolution.relu.pooling.layer2_2           98
61         convolution.relu.pooling.layer2_2           149
47         convolution.relu.pooling.layer2_2           152
62         convolution.relu.pooling.layer2_2           120
48         convolution.relu.pooling.layer2_2           116
63         convolution.relu.pooling.layer2_2           101
49         convolution.relu.pooling.layer2_2           185
64         convolution.relu.pooling.layer2_2           101
50         convolution.relu.pooling.layer2_2           111
65         convolution.relu.pooling.layer2_2           109
51         convolution.relu.pooling.layer2_2           213
66         convolution.relu.pooling.layer2_2           109
52         convolution.relu.pooling.layer2_2           213
67         convolution.relu.pooling.layer2_2           109
53         convolution.relu.pooling.layer2_2           213
68         convolution.relu.pooling.layer2_2           109
54         convolution.relu.pooling.layer2_2           213
69         convolution.relu.pooling.layer2_2           109
55         convolution.relu.pooling.layer2_2           213
70         convolution.relu.pooling.layer2_2           109
56         convolution.relu.pooling.layer2_2           174
71         convolution.relu.pooling.layer2_2           219
```

```
57          convolution.relu.pooling.layer2_2           353
58          fullyconnected.relu.layer_3                 110
Total Time(us): 4772
FPS(800MHz): 209.56
=====================================================================

done
```

The main operations of this example include: create RKNN object, model configuration, load TensorFlow Lite model, structure RKNN model, export RKNN model, load pictures and infer to get TOP5 result, evaluate model performance, release RKNN object.

The execution method of mobilenet_v2 and mobilenet-ssd in example directory is the same as mobilenet_v1, except that the execution script of mobilenet-ssd is ssd.py and after execution it will output one out.jpg picture where the detected object will be marked out.

3.3.2 Example running on RK1808

Here take mobilenet_v1 as example. mobilenet_v1 example in the tool package is running on PC simulator. If want to run the example on RK1808 EVB board, you can refer to below steps:

1. Enter example/mobilenet_v1 directory

```
rk@rk:~/rknn-toolkit-v1.2.0/example/mobilenet_v1$
```

2. Modify the parameter of initializing environment variable in test.py script

```
rk@rk:~/rknn-toolkit-v1.2.0/example/mobilenet_v1$ vim test.py
# find the method of initializing environment variable in script init_runtime,
as below
ret = rknn.init_runtime()
# modify the parameter of the method
ret = rknn.init_runtime(target=’rk1808’, device_id=’ 0123456789ABCDEF’)
# save and exit
```

3. Execute test.py script, and then get the result as below:

```
rk@rk:~/rknn-toolkit-v1.2.0/example/mobilenet_v1$ python test.py
--> config model
done
--> Loading model
done
```

```
--> Building model
done
--> Export RKNN model
done
--> Init runtime environment
done
--> Running model
mobilenet_v1
-----TOP 5-----
[156]: 0.8837890625
[155]: 0.0677490234375
[188 205]: 0.00867462158203125
[188 205]: 0.00867462158203125
[263]: 0.0057525634765625

done
--> Begin evaluate model performance
======================================================================
                            Performance
======================================================================
Total Time(us): 6098
FPS: 163.99
======================================================================

done
```

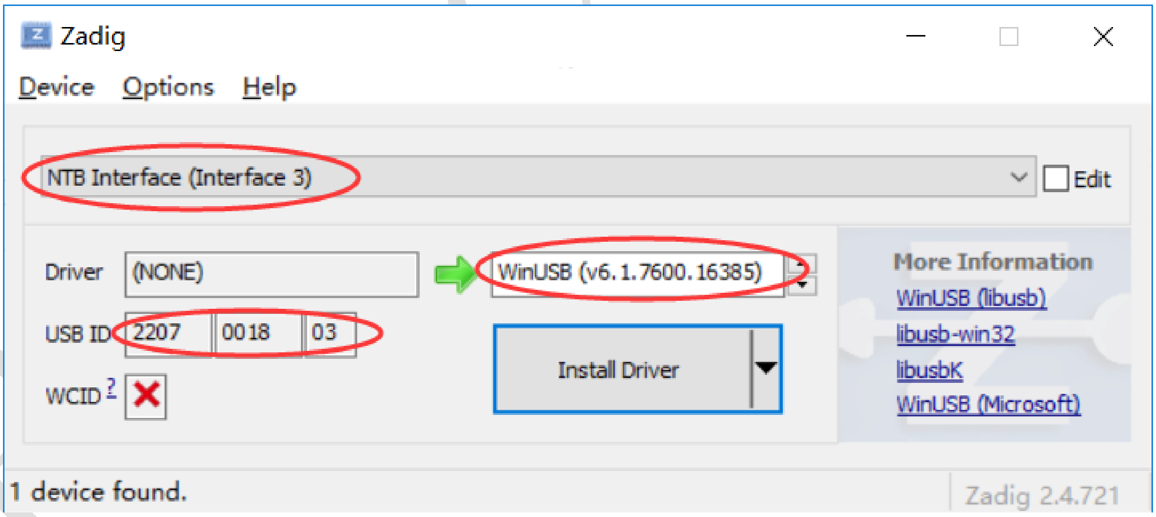
4 Windows platform Quick Start Guide

This chapter introduces how to use RKNN-Toolkit on Windows platforms with python 3.6.

4.1 Environmental preparations

- One pc with Windows 7 (64bit) or Windows 10 (64bit).
- One TB-RK1808 AI Compute Stick (Windows platform currently only supports computing sticks).
- Connect TB-RK1808 AI Compute Stick to PC through USB. If this is first time to use TB-RK1808 AI Compute Stick, we need install driver first. Installation method is as follows:
  - Open SDK package, and enter directory: platform-tools/drivers_installer/windows-x86_64, run the zadig-2.4.exe program as an administrator to install the computing stick driver:

    1. Confirm the equipment and the driver to be installed:



NTB Interface (Interface 3)

Driver (NONE) → WinUSB (v6.1.7600.16385)

USB ID 2207 0018 03

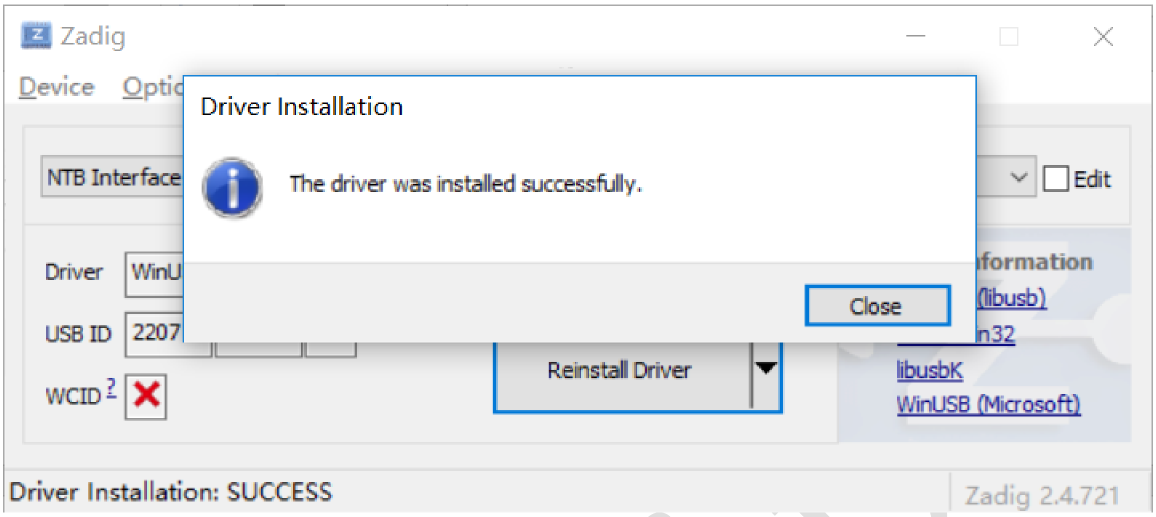
WCID ? ☒

1 device found.

Zadig 2.4.721

Note: The USB ID should be **2207:0018**; the driver choose default: WinUSB.

2. Click Install Driver.
3. If the installation is successful, the following interface will appear:

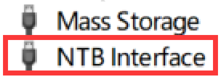


Driver Installation

The driver was installed successfully.

Driver Installation: SUCCESS

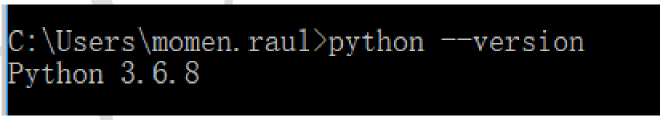
- After installation, if the TB-RK1808 AI Compute Stick in the Windows Device Manager does not have an exclamation point, and as shown below, the installation is successful.



Note: Please reboot compute after installing driver.

4.2 Install RKNN-Toolkit

Before install RKNN-Toolkit, make sure python3.6 has been installed. This can be determined by executing python –version in cmd, as explained below. Python 3.6 is already installed on the system.



```
C:\Users\momen.raul>python --version
Python 3.6.8
```

Get RKNN-Toolkit SDK package, then perform the following steps:

1. Enter directory: rknn-toolkit-v1.2.0/packages

```
D:\workspace\rknn-toolkit-v1.2.0>cd packages
```

2. Install Python dependency.

```
D:\workspace\rknn-toolkit-v1.2.0\packages>pip install tensorflow==1.13.1
D:\workspace\rknn-toolkit-v1.2.0\packages>pip install opencv-python
```

Note: opencv-python is used in example.

3. Manually install lmdb, in directory：

rknn-toolkit-v1.2.0\packages\required-packages-for-win-python36

```
    D:\workspace\rknn-toolkit-v1.2.0\packages\required-packages-for-win-pyt
hon36>pip install lmdb-0.95-cp36-cp36m-win_amd64.whl
```

4. Install RKNN-Toolkit.

```
    pip install rknn_toolkit-1.2.0-cp36-cp36m-win_amd64.whl
```

5. Check if RKNN-Toolkit is installed successfully or not.

```
    D:\workspace\rknn-toolkit-v1.2.0\packages>python
    Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC
v.1916 64 bit (AMD64)] on win32
    Type "help", "copyright", "credits" or "license" for more information.
    >>> from rknn.api import RKNN
    >>>
```

4.3 Running the sample attached in the installation package

Take mobilenet_v1 as an example, which is a Tensorflow Lite model for image classification。

The running steps are as below:

1. Enter example/mobilenet_v1 directory.

```
    D:\workspace\rknn-toolkit-v1.2.0\packages>cd ..\

    D:\workspace\rknn-toolkit-v1.2.0>cd example\mobilenet_v1
```

2. Modify the parameter of initializing environment variable in test.py script.

```
    #Befor modifying:
    ret = rknn.init_runtime()
    #After modifying:
    ret = rknn.init_runtime(target='rk1808')
```

3. Run test.py script

```
    D:\workspace\rknn-toolkit-v1.2.0\example\mobilenet_v1>python test.py
```

4. Get the TOP5 and performance after the script execution as below:

```
--> config model
done
--> Loading model
done
--> Building model
done
--> Export RKNN model
done
--> Init runtime environment
done
--> Running model
mobilenet_v1
-----TOP 5-----
[156]: 0.8837890625
[155]: 0.0677490234375
[188 205]: 0.00867462158203125
[188 205]: 0.00867462158203125
[263]: 0.0057525634765625

done
--> Begin evaluate model performance
================================================
                        Performance
================================================
Total Time(us): 6063
FPS: 164.93
================================================

done
```

The main operations of this example include: create RKNN object, model configuration, load TensorFlow Lite model, structure RKNN model, export RKNN model, load pictures and infer to get TOP5 result, evaluate model performance, release RKNN object.

The execution method of mobilenet_v2 and mobilenet-ssd in example directory is the same as mobilenet_v1, except that the execution script of mobilenet-ssd is ssd.py and after execution it will output one out.jpg picture where the detected object will be marked out.

Note:

1. Simulator can not run on Windows platform, so we must have a TB-RK1808 AI Compute Stick.
2. For more detail about TB-RK1808 AI Compute Stick, please refer to this link:

http://t.rock-chips.com/wiki.php?mod=view&pid=28

5 Mac OS X platform Quick Start Guide

This chapter introduces how to use RKNN-Toolkit on Mac OS X platforms with python 3.6.

5.1 Environmental preparations

- One pc with MacOS High Sierra.
- One TB-RK1808 AI Compute Stick.
- Connect TB-RK1808 AI Compute Stick to PC through USB, execute program 'npu_transfer_proxy' in directory 'platform-tools/ntp/mac-osx-x86_64', check weather TB-RK1808 AI Compute Stick has connected. Result should looks like below:

```
macmini:ntp rk$ ./npu_transfer_proxy devices
List of ntb devices attached
TS018080000000013        2bed0cc1        USB_DEVICE
```

Note: The red line is the TB-RK1808 AI Compute Stick. Device id is "**TS018080000000013**".

5.2 Install RKNN-Toolkit

Get RKNN-Toolkit SDK package, then perform the following steps:

1. Enter directory: rknn-toolkit-v1.2.0/packages

```
cd packages/
```

2. Install Python dependency.

```
pip3 install tensorflow
pip3 install opencv-python
```

Note: opencv-python is used in example.

3. Install RKNN-Toolkit.

```
pip3 install rknn_toolkit-1.2.0-cp36-cp36m-macosx_10_9_x86_64.whl
```

4. Check if RKNN-Toolkit is installed successfully or not.

```
(rknn-venv)macmini:rknn-toolkit-v1.2.0 rk$ python3
>>> from rknn.api import RKNN
>>>
```

5.3 Running the sample attached in the installation package

Take mobilenet_v1 as an example, which is a Tensorflow Lite model for image classification

The running steps are as below:

1. Enter example/mobilenet_v1 directory.

```
(rknn-venv)macmini:rknn-toolkit-v1.2.0 rk$ cd example/mobilenet_v 1
```

2. Modify the parameter of initializing environment variable in test.py script.

```
#Befor modifying:
ret = rknn.init_runtime()
#After modifying:
ret = rknn.init_runtime(target='rk1808')
```

3. Run test.py script

```
(rknn-venv)macmini:mobilenet_v1 rk$ python3 test.py
```

4. Get the TOP5 and performance after the script execution as below:

```
--> config model
done
--> Loading model
done
--> Building model
done
--> Export RKNN model
done
--> Init runtime environment
done
--> Running model
mobilenet_v1
-----TOP 5-----
[156]: 0.8837890625
[155]: 0.0677490234375
```

```
[188 205]: 0.00867462158203125
[188 205]: 0.00867462158203125
[263]: 0.0057525634765625

done
--> Begin evaluate model performance
===================================================
                     Performance
===================================================
Total Time(us): 6097
FPS: 164.02
===================================================

done
```

The main operations of this example include: create RKNN object, model configuration, load TensorFlow Lite model, structure RKNN model, export RKNN model, load pictures and infer to get TOP5 result, evaluate model performance, release RKNN object.

The execution method of mobilenet_v2 and mobilenet-ssd in example directory is the same as mobilenet_v1, except that the execution script of mobilenet-ssd is ssd.py and after execution it will output one out.jpg picture where the detected object will be marked out.

Note:

1. Simulator can not run on Mac OS X platform, so we must have a TB-RK1808 AI Compute Stick.
2. For more detail about TB-RK1808 AI Compute Stick, please refer to this link: http://t.rock-chips.com/wiki.php?mod=view&pid=28

6 ARM64 platform (Python 3.5) Quick Start Guide

This chapter introduces how to use RKNN-Toolkit on ARM64 platforms (Debian 9.8 systems) with python3.5.

6.1 Environmental preparations

- An RK3399Pro with Debian 9.8 operating system. Make sure that the remaining space of the root partition is greater than 5GB.
- Ensure that the NPU driver version is greater than 0.9.6.
- If can not find npu_transfer_proxy or npu_transfer_proxy.proxy in /usr/bin directory, we need copy the npu_transfer_proxy in rknn-toolkit-v1.2.0\platform-tools\ntp\linux_aarch64 directory to /usr/bin/ directory, and go to the directory and execute the following command (you have to start the program after each reboot, so please add it to boot script):

```
sudo ./npu_transfer_proxy &
```

6.2 Install RKNN-Toolkit

1. Execute the following command to update the system packages which will be used later when installing Python dependencies.

```
sudo apt-get update
sudo apt-get install cmake gcc g++ libprotobuf-dev protobuf-compiler
sudo apt-get install liblapack-dev libjpeg-dev zlib1g-dev
sudo apt-get install python3-dev python3-pip python3-scipy
```

2. Execute the following command to update pip.

```
pip3 install --upgrade pip
```

You also need to modify /usr/bin/pip3 after update, otherwise it will report pip3 error when installing other dependencies. Modify /usr/bin/pip3 as follows:

```
from pip import main      -->      from pip import __main__
…
    sys.exit(main())      -->         sys.exit(__main__._main())
```

3. Install Python package tool.

```
pip3 install wheel setuptools
```

4. Install dependency package h5py.

```
sudo apt-get build-dep python3-h5py && \
pip3 install h5py
```

5. Install TensorFlow and the corresponding whl package is in the rknn-toolkit-v1.2.0\packages\required-packages-for-arm64-debian9-python35 directory.

```
pip3 install tensorflow-1.11.0-cp35-none-linux_aarch64.whl --user
```

Note: Since some libraries that TensorFlow relies on need compile and install on the ARM64 platform after downloading the source code, this step will take a long time.

6. Install opencv-python and the corresponding whl package is in the `rknn-toolkit-v1.2.0\packages\required-packages-for-arm64-debian9-python35’ directory.

```
pip3 install \
opencv_python_headless-4.0.1.23-cp35-cp35m-linux_aarch64.whl
```

7. Install RKNN-Toolkit and the corresponding whl package is in the rknn-toolkit-v1.2.0\packages directory

```
pip3 install rknn_toolkit-1.2.0-cp35-cp35m-linux_aarch64.whl --user
```

Note: Since some libraries that RKNN-Toolkit relies on need compile and install on the ARM64 platform after downloading the source code, this step will take a long time.

6.3 Running the sample attached in the installation package

Take mobilenet_v1 as an example, which is a Tensorflow Lite model for image classification.

The running steps are as below:

1. Enter example/mobilenet_v1 directory

```
    linaro@linaro-alip:~/rknn-toolkit-v1.2.0/ $ cd example/mobilenet_v1
```

2. Run test.py script

```
    linaro@linaro-alip: ~/rknn-toolkit-v1.2.0/example/mobilenet_v1$ python3
test.py
```

3. Get the results after the script execution as below:

```
    --> config model
    done
    --> Loading model
    done
    --> Building model
    done
    --> Export RKNN model
    done
    --> Init runtime environment
    done
    --> Running model
    mobilenet_v1
    -----TOP 5-----
    [156]: 0.8837890625
    [155]: 0.0677490234375
    [188 205]: 0.00867462158203125
    [188 205]: 0.00867462158203125
    [263]: 0.0057525634765625

    done
    --> Begin evaluate model performance
    ===================================================
                                  Performance
    ===================================================
    Total Time(us): 5761
    FPS: 173.58
    ===================================================

    done
```

The main operations of this example include: create RKNN object, model configuration, load TensorFlow Lite model, structure RKNN model, export RKNN model, load pictures and infer to get

TOP5 result, evaluate model performance, release RKNN object.

The execution method of mobilenet_v2 and mobilenet-ssd in example directory is the same as mobilenet_v1, except that the execution script of mobilenet-ssd is ssd.py and after execution it will output one out.jpg picture where the detected object will be marked out.

Note:

1. Simulator can not run on ARM64 platform, these models in example are running on built-in NPU of RK3399Pro.
2. Currently, we can only run RKNN-Toolkit on ARM64 Plarform with RK3399 and RK3399Pro. If the EVB board is RK3399, we need connect a TB-RK1808 AI Compute Stick.
3. For more detail about TB-RK1808 AI Compute Stick, please refer to this link: http://t.rock-chips.com/wiki.php?mod=view&pid=28

7 Reference Document

For more detailed usage and interface descriptions of RKNN-Toolkit, please refer to <Rockchip_User_Guide_RKNN_Toolkit_V1.2.0_EN.pdf >.