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

RKNN-Toolkit Quick Start

(Technology Department, Graphic Display Platform Center)

Mark: [ ] Editing [√] Released	Version	V1.3.0
	Author	Rao Hong
	Completed Date	2019-12-23
	Auditor	Randall
	Reviewed Date	2019-12-23

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
V1.2.1	Rao Hong	2019-09-26	Synchronize the modification contents of RKNN-Toolkit-V1.2.1	Randall
V1.3.0	Rao Hong	2019-12-23	Synchronize the modification contents of RKNN-Toolkit-V1.3.0	Randall

Content

1 Main Features Introduction

RKNN-Toolkit is a software development kit for users to perform model conversion, inference and performance evaluation on PC, RK3399Pro, RK1808, TB-RK1808S0 AI Compute Stick or RK3399Pro Linux development board users can easily complete the following functions through the provided python interface:

1） Model conversion: support to convert Caffe、TensorFlow、TensorFlow Lite、ONNX、Darknet、Pytorch、MXNetmodel to RKNN model, support RKNN model import/export, which can be used on hardware platform later. Support for multiple input models starting with version 1.2.0. Support for Pytorch and MXNet since version 1.3.0, these two features are currently experimental.

2） Quantization: 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 1.0.0, RKNN-Toolkit began to support hybrid quantization.

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: Evaluate system and NPU memory consumption at runtime of the model. It can 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. This feature is supported starting with version 0.9.9

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 0.9.5, and the pre-compilation method has been upgraded in 1.0.0. 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. Please refer to the <Rockchip_Developer_Guide_RKNN_-Toolkit_Custom_OP_CN> document for the use and development of custom OP.

9） Quantitative error analysis: This function will give the Euclidean or cosine distance of each layer of inference results before and after the model is quantized. This can be used to analyze how quantitative error occurs, and provide ideas for improving the accuracy of quantitative models. This feature is supported from version 1.3.0.

10）Visualization: This function presents various functions of RKNN-Toolkit in the form of a graphical interface, simplifying the user's operation steps. Users can complete model conversion

and inference by filling out forms and clicking function buttons, and no need to write scripts manually. Please refer to the < Rockchip_User_Guide_RKNN_Toolkit_Visualization_EN> document for the use of visualization.

11）Model optimization level: RKNN-Toolkit optimizes the model during model conversion. The default optimization selection may have some impact on model accuracy. By setting the optimization level, you can turn off some or all optimization options to analyze the impact of RKNN-Toolkit model optimization options on accuracy. For specific usage of optimization level, please refer to the description of optimization_level option in config interface. This feature is supported from version 1.3.0.

2 System Dependency Introduction

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
Python version	3.5/3.6
Python library dependencies	'numpy == 1.16.3' 'scipy == 1.3.0' 'Pillow == 5.3.0' 'h5py == 2.8.0' 'lmdb == 0.93' 'networkx == 1.11' 'flatbuffers == 1.10', 'protobuf == 3.6.1' 'onnx == 1.4.1' 'onnx-tf == 1.2.1' 'flask == 1.0.2' 'tensorflow == 1.11.0' or 'tensorflow-gpu' 'dill==0.2.8.2' 'ruamel.yaml == 0.15.81' 'psutils == 5.6.2' 'ply == 3.11' 'requests == 2.22.0' 'pytorch == 1.2.0' 'mxnet == 1.5.0'

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==1.11.0
```

```
pip3 install mxnet==1.5.0
pip3 install torch==1.2.0 torchvision==0.4.0
pip3 install opencv-python
pip3 install gluoncv
```

c) Install RKNN-Toolkit

```
sudo pip3 install rknn_toolkit-1.3.0-cp35-cp35m-linux_x86_64.whl
```

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

```
rk@rk:~/rknn-toolkit-v1.3.0/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 examples/lite/mobilenet_v1 directory

```
rk@rk:~/rknn-toolkit-v1.3.0/package$ cd ../examples/lite/mobilenet_v1
rk@rk:~/rknn-toolkit-v1.3.0/examples/lite/mobilenet_v1$
```

2. Execute test.py script

```
rk@rk:~/rknn-toolkit-v1.3.0/examples/lite/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]: 0.85107421875
[155]: 0.09173583984375
[205]: 0.01358795166015625
[284]: 0.006465911865234375
[194]: 0.002239227294921875

done
--> Begin evaluate model performance
============================================================================
                              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               104
62          convolution.relu.pooling.layer2_2               120
48          convolution.relu.pooling.layer2_2               72
63          convolution.relu.pooling.layer2_2               101
49          convolution.relu.pooling.layer2_2               92
64          convolution.relu.pooling.layer2_2               99
50          convolution.relu.pooling.layer2_2               110
65          convolution.relu.pooling.layer2_2               107
51          convolution.relu.pooling.layer2_2               212
66          convolution.relu.pooling.layer2_2               107
52          convolution.relu.pooling.layer2_2               212
67          convolution.relu.pooling.layer2_2               107
53          convolution.relu.pooling.layer2_2               212
68          convolution.relu.pooling.layer2_2               107
54          convolution.relu.pooling.layer2_2               212
69          convolution.relu.pooling.layer2_2               107
55          convolution.relu.pooling.layer2_2               212
70          convolution.relu.pooling.layer2_2               107
56          convolution.relu.pooling.layer2_2               174
```

```
71          convolution.relu.pooling.layer2_2           220
57          convolution.relu.pooling.layer2_2           353
28          pooling.layer2_1                            36
58          fullyconnected.relu.layer_3                 110
30          softmaxlayer2.layer_1                       90
Total Time(us): 4694
FPS(800MHz): 213.04
=============================================================

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.

Other demos in the examples directory are executed the same way as mobilenet_v1. These models are mainly used for classification, target detection.

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 examples/lite/mobilenet_v1 directory

```
rk@rk:~/rknn-toolkit-v1.3.0/examples/lite/mobilenet_v1$
```

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

```
rk@rk:~/rknn-toolkit-v1.3.0/examples/lite/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-v1.3.0/examples/lite/mobilenet_v1$ python test.py
--> config model
done
--> Loading model
```

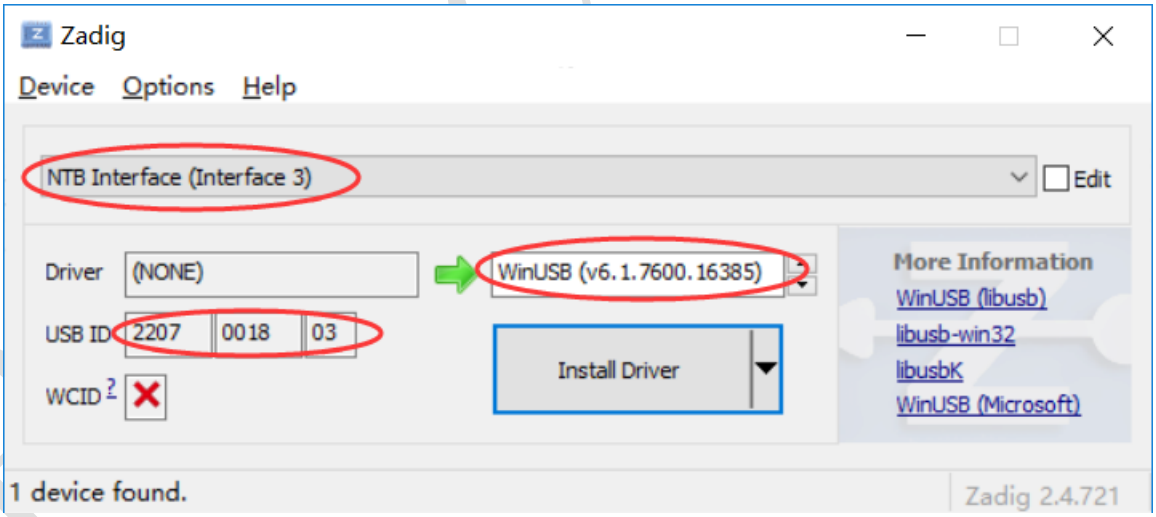
```
done
--> Building model
done
--> Export RKNN model
done
--> Init runtime environment
done
--> Running model
mobilenet_v1
-----TOP 5-----
[156]: 0.85107421875
[155]: 0.09173583984375
[205]: 0.01358795166015625
[284]: 0.006465911865234375
[194]: 0.002239227294921875

done
--> Begin evaluate model performance
========================================================================
                            Performance
========================================================================
Total Time(us): 5805
FPS: 172.27
========================================================================

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

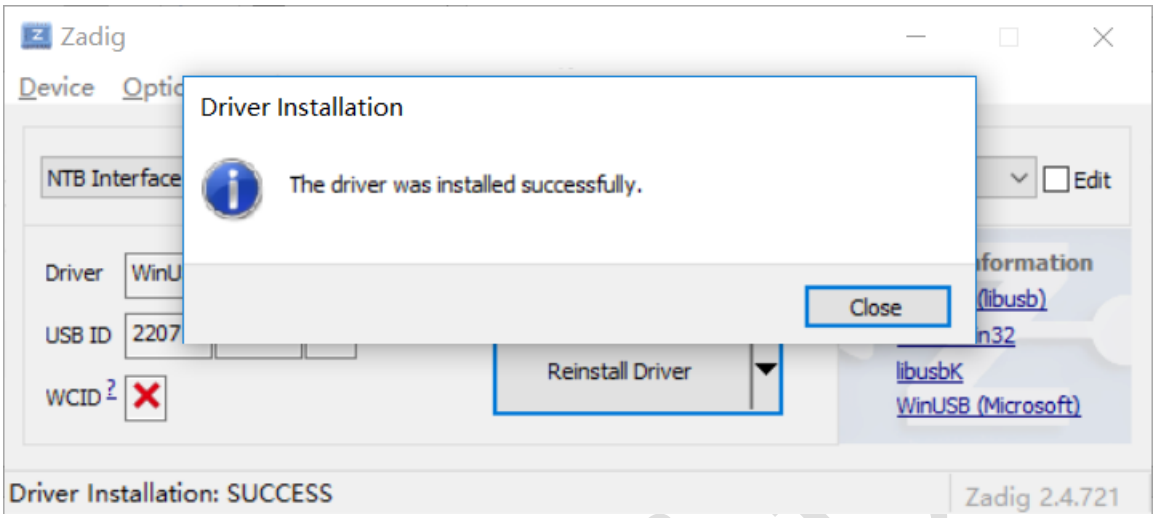
- One pc with Windows 7 (64bit) or Windows 10 (64bit).
- One TB-RK1808 AI Compute Stick (Windows platform currently only supports computing sticks).
- Connect TB-RK1808 AI Compute Stick to PC through USB. If this is first time to use TB-RK1808 AI Compute Stick, we need install driver first. Installation 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:



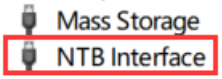
    Note: The USB ID should be **2207:0018**; the driver choose default: WinUSB.

    2. Click Install Driver.

    3. If the installation is successful, the following interface will appear:



- 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-v1.3.0/packages

```
D:\workspace\rknn-toolkit-v1.3.0>cd packages
```

2. Install Python dependency.

```
pip install tensorflow==1.11.0
pip install torch==1.2.0+cpu torchvision==0.4.0+cpu -f
https://download.pytorch.org/whl/torch_stable.html --user
pip install mxnet==1.5.0
pip install opencv-python
```

Note: opencv-python and gluoncv are used in example.

3. Manually install lmdb, in directory：

packages\required-packages-for-win-python36

```
D:\workspace\rknn-toolkit-v1.3.0\packages\required-packages-for-win-python36>pip install lmdb-0.95-cp36-cp36m-win_amd64.whl
```

4. Install RKNN-Toolkit.

```
pip install rknn_toolkit-1.3.0-cp36-cp36m-win_amd64.whl
```

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

```
D:\workspace\rknn-toolkit-v1.3.0\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 examples/lite/mobilenet_v1 directory.

```
D:\workspace\rknn-toolkit-v1.3.0\packages>cd ..\

D:\workspace\rknn-toolkit-v1.3.0>cd examples\lite\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-v1.3.0\examples\lite\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]: 0.8828125
[155]: 0.06768798828125
[188 205]: 0.0086669921875
[188 205]: 0.0086669921875
[263]: 0.006366729736328125

done
--> Begin evaluate model performance
============================================
                Performance
============================================
Total Time(us): 6032
FPS: 165.78
============================================

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.

Other demos in the examples directory are executed the same way as mobilenet_v1. These models are mainly used for classification, target detection.

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-v1.3.0/packages

```
cd packages/
```

2. Install Python dependency.

```
pip3 install tensorflow==1.11.0
pip3 install mxnet==1.5.0
pip3 install torch==1.2.0 torchvision==0.4.0
pip3 install opencv-python
pip3 install gluoncv
```

Note: opencv-python and gluoncv are used in example.

3. Install RKNN-Toolkit.

```
pip3 install rknn_toolkit-1.3.0-cp36-cp36m-macosx_10_9_x86_64.whl
```

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

```
(rknn-venv)macmini:rknn-toolkit-v1.3.0 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 examples/lite/mobilenet_v1 directory.

```
(rknn-venv)macmini:rknn-toolkit-v1.3.0 rk$ cd examples/lite/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]: 0.85107421875
[155]: 0.09173583984375
[205]: 0.01358795166015625
[284]: 0.006465911865234375
[194]: 0.002239227294921875

done
--> Begin evaluate model performance
============================================
                Performance
============================================
Total Time(us): 6046
FPS: 165.40
============================================

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.

Other demos in the examples directory are executed the same way as mobilenet_v1. These models are mainly used for classification, target detection.

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-v1.3.0\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
```

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-v1.3.0\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-v1.3.0\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-v1.3.0\packages directory

```
pip3 install rknn_toolkit-1.3.0-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 examples/lite/mobilenet_v1 directory

```
linaro@linaro-alip:~/rknn-toolkit-v1.3.0/ $ cd examples/lite/mobilenet_v1
```

2. Run test.py script

```
linaro@linaro-alip:
~/rknn-toolkit-v1.3.0/examples/lite/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]: 0.85107421875
[155]: 0.09173583984375
[205]: 0.01358795166015625
[284]: 0.006465911865234375
[194]: 0.002239227294921875

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.

Other demos in the examples directory are executed the same way as mobilenet_v1. These models are mainly used for classification, target detection.

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_V1.3.0_EN.pdf >.