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

RKNN-Toolkit Quick Start

(Technology Department, Graphic Computing Platform Center)

Mark: [ ] Editing [√] Released	Version	V1.4.0
	Author	Rao Hong
	Completed Date	2020-08-13
	Auditor	Randall
	Reviewed Date	2020-08-13

瑞芯微电子股份有限公司

Rockchip 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
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
V1.3.2	Rao Hong	2020-04-03	Synchronize the modification contents of RKNN-Toolkit-V1.3.2	Randall
V1.4.0	Rao Hong	2020-08-13	Synchronize the modification contents of RKNN-Toolkit-V1.4.0	Randall

Content

1 Main Features Introduction

RKNN-Toolkit is a development kit that provides users with model conversion, inference and performance evaluation on PC and Rockchip NPU platforms. Users can easily complete the following functions through the Python interface provided by the tool:

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

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 version 1.0.0, RKNN-Toolkit began to support hybrid quantization.

3） Model inference: Able to simulate Rockchip NPU to run RKNN model on PC and get the inference result. This tool can also distribute the RKNN model to the specified NPU device to run, and get the inference results.

4） Performance evaluation: Able to simulate Rockchip NPU to run RKNN model on PC, and evaluate model performance (including total time and time-consuming information of each layer). This tool can also distribute the RKNN model to the specified NPU device to run, and evaluate the model performance in the actual device.

5） Memory evaluation: Evaluate system and NPU memory consumption at runtime of the model. When using this function, he RKNN model must be distributed to the NPU device to run, and then call the relevant interface to obtain memory information. 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. Starting from version 1.4.0, ordinary RKNN models can also be converted into precompiled models through NPU device.

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. Starting from version 1.4.0, new feature called individual quantization accuracy analysis provided. The tool assigns the input of each layer at runtime as the correct floating point value, and then calculates the quantized error of the layer. This can avoid misjudgments caused by the accumulation of errors layer by layer, and more accurately reflect the influence of quantization on each layer itself.

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.Version 1.4.0 improves the support for multi-inputs models and supports new NPU devices such as RK1806/RV1109/RV1126 as target.

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:

1. Windowsonly support Python 3.6 currently.
2. MacOS support Python 3.6 and Python 3.7.
3. Arm64 platform support Python 3.5(Debian 9) and Python 3.7(Debian10).

4. Scipy version on MacOS should be 1.3.0, other platform is >=1.1.0.

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 or RV1126 EVB board.
- Connect RK1808 (or RV1126) device to PC through OTG USB, use 'adb devices' command to check, and the result is as below:

```
rk@rk:~$ adb devices
List of devices attached
515e9b401c060c0b          device
c3d9b8674f4b94f6          device
```

The content marked in red is the device ID, the first is the RK1808 development board, and the second is the RV1126 development board.

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-1.4.0-cp35-cp35m-linux_x86_64.whl
```

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

```
rk@rk:~/rknn-toolkit-v1.4.0/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/RV1126 simulator which can be used to simulate the action of the model running on RK1808 or RV1126. If users want to simulate RV1126, it`s needed to set target_platform=['rv1126'] in config interface.

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-v1.4.0/package$ cd ../examples/lite/mobilenet_v1
rk@rk:~/rknn-toolkit-v1.4.0/examples/lite/mobilenet_v1$
```

2. Execute test.py script

```
rk@rk:~/rknn-toolkit-v1.4.0/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-v1.4.0/examples/lite/mobilenet_v1$
```

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

```
    rk@rk:~/rknn-toolkit-v1.4.0/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-v1.4.0/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
```

3.3.3 Example running on RV1126

Runing on RV1126 is similar to RK1808 EVB board. But when calling the config interface, it`s needed to specify target_platform as RV1126. In init_runtime, target also fills in RV1126. Specific steps are as follows：

1. Enter examples/lite/mobilenet_v1 directory

```
rk@rk:~/rknn-toolkit-v1.4.0/examples/lite/mobilenet_v1$
```

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

```
    rk@rk:~/rknn-toolkit-v1.4.0/examples/lite/mobilenet_v1$ vim test.py
    # find the method of config in script, as below
    rknn.config(channel_mean_value='128 128 128 128', reorder_channel='0 1
2')
    # modify the parameter of this interface
    rknn.config(channel_mean_value='128 128 128 128', reorder_channel='0 1
2', target_platform=['rv1126'])
    # 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='rv1126', device_id='c3d9b8674f4b94f6')
    # save and exit
```

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

```
    rk@rk:~/rknn-toolkit-v1.4.0/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]: 0.8603515625
    [155]: 0.0833740234375
    [205]: 0.0123443603515625
    [284]: 0.00726318359375
    [260]: 0.002262115478515625

    done
    --> Begin evaluate model performance
    =====================================================================
                                Performance
    =====================================================================
    Total Time(us): 4759
    FPS: 210.13
    =====================================================================

    done
```

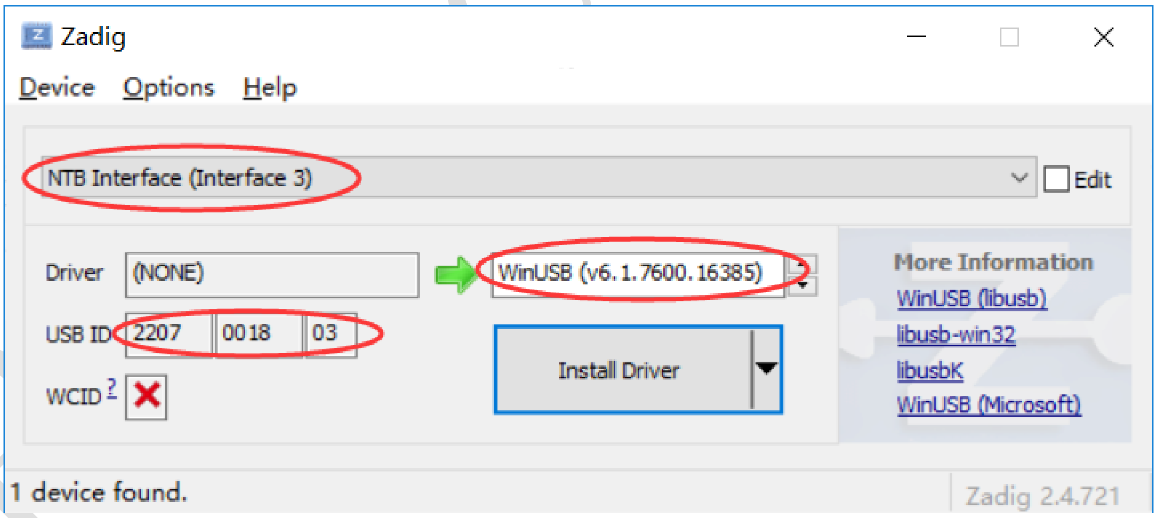
4 Windows platform Quick Start Guide

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 RK1808 or RV1126 EVB board.
- Connect RK1808 (or RV1126) EVB board to PC through USB(OTG). If this is first time to use RK1808 (or RV1126) 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:



Zadig

NTB Interface (Interface 3)

Driver (NONE) → WinUSB (v6.1.7600.16385)

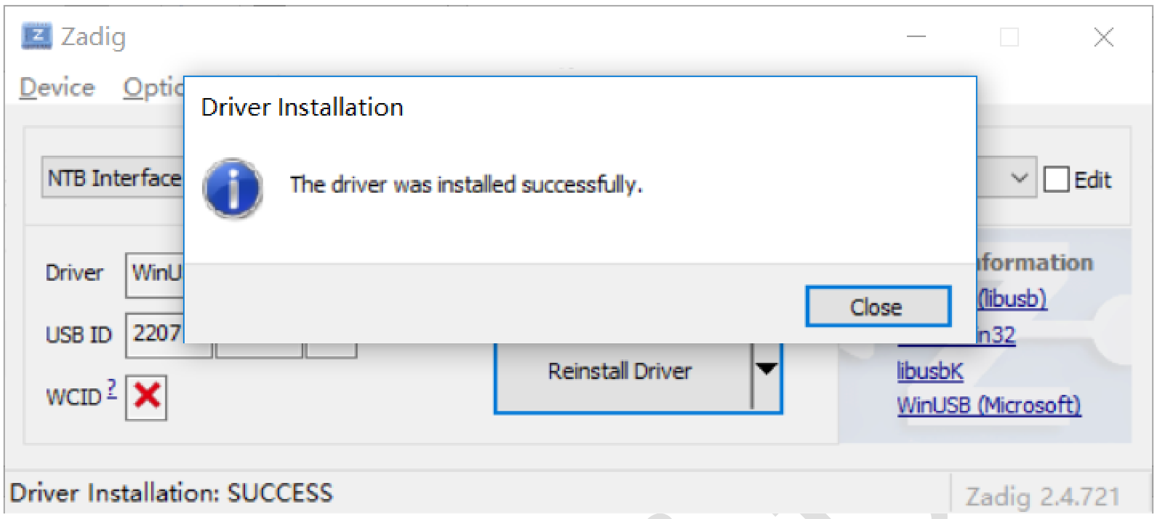
USB ID 2207 0018 03

1 device found.

Zadig 2.4.721

Note: The USB ID should start with **2207**; the driver choose default: WinUSB.

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



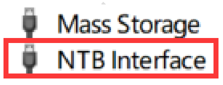
Driver Installation

The driver was installed successfully.

Driver Installation: SUCCESS

Zadig 2.4.721

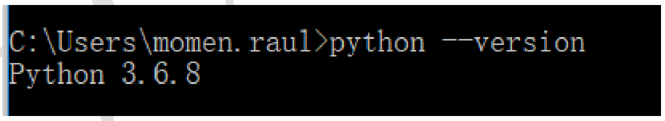
- After installation, if the NTB Interface in the Windows Device Manager does not have an exclamation point, and as shown below, the installation is successful.



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-v1.4.0/packages

```
D:\workspace\rknn-toolkit-v1.4.0>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-v1.4.0\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-1.4.0-cp36-cp36m-win_amd64.whl
```

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

```
    D:\workspace\rknn-toolkit-v1.4.0\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 Example running on RK1808

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-v1.4.0\packages>cd ..\

    D:\workspace\rknn-toolkit-v1.4.0>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-v1.4.0\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.

4.4 Example running on RV1126

When using RV1126 to run the example on the Windows platform, the modifications and the running steps are the same as the Ubuntu platform, so it won't be repeated here.

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 RK1808 or RV1126 EVB board.
- Connect RK1808 (or RV1126) EVB board to PC through USB(OTG), execute program 'npu_transfer_proxy' in directory 'platform-tools/ntp/mac-osx-x86_64', check weather EVB board has connected. Result should looks like below:

```
macmini:ntp rk$ ./npu_transfer_proxy devices
List of ntb devices attached
515e9b401c060c0b        2bed0cc1        USB_DEVICE
```

Note: The red line is the RK1808 EVB board. Device id is "515e9b401c060c0b".

5.2 Install RKNN-Toolkit

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

1. Enter directory: rknn-toolkit-v1.4.0/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-1.4.0-cp36-cp36m-macosx_10_15_x86_64.whl
```

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

```
(rknn-venv)macmini:rknn-toolkit-v1.4.0 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-v1.4.0 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.

5.4 Example running on RV1126

When using RV1126 to run the example on the Mac OS platform, the modifications and the running steps are the same as the Ubuntu platform, so it won’t be repeated here.

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.
- 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-v1.4.0\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-v1.4.0\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-v1.4.0\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-v1.4.0\packages directory

```
pip3 install rknn_toolkit-1.4.0-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-v1.4.0/ $ cd examples/lite/mobilenet_v1
```

2. Run test.py script

```
    linaro@linaro-alip:
~/rknn-toolkit-v1.4.0/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_V1.4.0_EN.pdf >.