

密级状态：绝密( ) 秘密( ) 内部( ) 公开(√)

## RKNN-Toolkit 快速上手指南

(技术部，图形显示平台中心)

文件状态： [ ] 正在修改 [√] 正式发布	当前版本：	V1.1.0
	作 者：	饶洪
	完成日期：	2019-06-28
	审 核：	卓鸿添
	完成日期：	2019-06-28

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Semiconductor Co., Ltd

(版本所有, 翻版必究)

## 更新记录

版本	修改人	修改日期	修改说明	核定人
V0.9.9	饶洪	2019-03-25	初始版本	卓鸿添
V1.0.0	饶洪	2019-05-08	同步 RKNN-Toolkit-V1.0.0 修改内容	卓鸿添
V1.1.0	饶洪	2019-06-28	1. 同步 RKNN-Toolkit-V1.1.0 修改内容 2. 新增 Windows/MacOS/ARM64 等平台的快速上手指南	卓鸿添

## 目 录

<b>1</b>	<b>主要功能说明.....</b>	<b>1</b>
<b>2</b>	<b>系统依赖说明.....</b>	<b>2</b>
<b>3</b>	<b>UBUNTU 平台快速上手.....</b>	<b>3</b>
3.1	环境准备 .....	3
3.2	安装 RKNN-TOOLKIT（以 PYTHON3.5 为例） .....	4
3.3	运行安装包中附带的示例.....	4
3.3.1	在 PC 上仿真运行示例.....	4
3.3.2	在 RK1808 上运行示例.....	6
<b>4</b>	<b>WINDOWS 平台（PYTHON3.6）快速上手指南 .....</b>	<b>8</b>
4.1	环境准备 .....	8
4.2	安装 RKNN-TOOLKIT.....	9
4.3	运行安装包中附带的示例.....	10
<b>5</b>	<b>MAC OS X 平台（PYTHON3.6）快速上手指南 .....</b>	<b>13</b>
5.1	环境准备 .....	13
5.2	安装 RKNN-TOOLKIT.....	13
5.3	运行安装包中附带的示例.....	14
<b>6</b>	<b>ARM64 平台（PYTHON3.5）快速上手指南 .....</b>	<b>16</b>
6.1	环境准备 .....	16
6.2	安装 RKNN-TOOLKIT.....	16
6.3	运行安装包中附带的示例.....	17
<b>7</b>	<b>参考文档.....</b>	<b>20</b>

## 1 主要功能说明

RKNN-Toolkit 是为用户提供在 PC、RK3399Pro、RK1808、TB-RK1808 AI 计算棒或 RK3399Pro Linux 开发板上进行模型转换、推理和性能评估的开发套件，用户通过提供的 python 接口可以便捷地完成以下功能：

1) 模型转换：支持 Caffe、TensorFlow、TensorFlow Lite、ONNX、Darknet 模型转成 RKNN 模型，支持 RKNN 模型导入导出，后续能够在硬件平台上加载使用。

2) 模型推理：能够在 PC 上模拟运行模型并获取推理结果；也可以在指定硬件平台 RK3399Pro（或 RK3399Pro Linux 开发板）、RK1808、TB-RK1808 AI 计算棒上运行模型并获取推理结果。

3) 性能评估：能够在 PC 上模拟运行并获取模型总耗时及每一层的耗时信息；也可以通过联机调试的方式在指定硬件平台 RK3399Pro、RK1808、TB-RK1808 AI 计算棒上运行模型，或者直接在 RK3399Pro Linux 开发板上运行，以获取模型在硬件上完整运行一次所需的总时间和每一层的耗时情况。

4) 获取模型运行时的内存使用情况：通过联机调试的方式获取模型在指定硬件平台 RK3399Pro、RK1808、TB-RK1808 AI 计算棒或 RK3399Pro Linux 开发板上运行时的内存使用情况。

5) 量化功能：支持将浮点模型转成量化模型，目前支持的量化方法有非对称量化（asymmetric\_quantized-u8），动态定点量化（dynamic\_fixed\_point-8 和 dynamic\_fixed\_point-16）。

## 2 系统依赖说明

本开发套件支持运行于 Ubuntu、Windows、MacOS、Debian 等操作系统。需要满足以下运行环境要求：

表 1 运行环境

操作系统版本	Ubuntu16.04（x64）及以上 Windows 7（x64）及以上 Mac OS X 10.13.5（x64）及以上 Debian 9.8（x64）及以上
Python 版本	3.5/3.6
Python 库依赖	'numpy >= 1.16.1' 'scipy >= 1.1.0' 'Pillow >= 3.1.2' 'h5py >= 2.7.1' 'lmbd >= 0.92' 'networkx == 1.11' 'flatbuffers == 1.9', 'protobuf >= 3.5.2' 'onnx >= 1.3.0' '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'

注：Windows 及 MacOS 只提供 Python3.6 的安装包。

### 3 Ubuntu 平台快速上手

本章节以 Ubuntu 16.04、Python3.5 为例说明如何快速上手使用 RKNN-Toolkit。

#### 3.1 环境准备

- 一台安装有 ubuntu16.04 操作系统的 x86\_64 位计算机。
- RK1808 EVB 板。
- 将 RK1808 设备通过 USB 连接到 PC 上，使用 `adb devices` 命令查看，结果如下：

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

其中标红的为设备 ID。

- 如果 RK1808 是以 NTB 模式运行（TB-RK1808 AI 计算棒也是以这种模式运行），在第一次使用之前需要以 `sudo` 权限执行 `platform-tools/update_rk_usb_rule/linux` 目录下的脚本（只需要执行一次即可），让普通用户获得该设备的 USB 读写权限：

```
rk@rk: ~/rknn-toolkit-v1.1.0/platform-tools/update_rk_usb_rule/linux$
sudo ./update_rk1808_usb_rule.sh
```

执行完后，可以通过以下方式确认普通用户是否已获得该设备的 USB 读写权限：

```
rk@rk:~$ ll /dev/bus/usb/003/075
crw-rw-rw- 1 root root 189, 330 May 15 09:37 /dev/bus/usb/003/075
```

注：其中 003 是设备的 bus 号，075 是 device 号，这些数值可以通过“`lsusb`”命令查看。

更多有关 TB-RK1808 AI 计算棒的资料，请参考链接：

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

## 3.2 安装 RKNN-Toolkit（以 Python3.5 为例）

### 1. 安装 Python3.5

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

### 2. 安装 pip3

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

### 3. 获取 RKNN-Toolkit 安装包，然后执行以下步骤：

#### a) 进入 package 目录：

```
cd package/
```

#### b) 安装 Python 依赖

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

#### c) 安装 RKNN-Toolkit

```
sudo pip3 install rknn_toolkit-1.1.0-cp35-cp35m-linux_x86_64.whl
```

#### d) 检查 RKNN-Toolkit 是否安装成功

```
rk@rk:~/rknn-toolkit-v1.1.0/package$ python3  
>>> from rknn.api import RKNN  
>>>
```

如果导入 RKNN 模块没有失败，说明安装成功。

## 3.3 运行安装包中附带的示例

### 3.3.1 在 PC 上仿真运行示例

RKNN-Toolkit 自带了一个 RK1808 的模拟器，可以用来仿真模型在 RK1808 上运行时的行为。

这里以 mobilenet\_v1 为例。示例中的 mobilenet\_v1 是一个 Tensorflow Lite 模型，用于图片分

类，它是在模拟器上运行的。

运行该示例的步骤如下：

1. 进入 example/mobilenet\_v1 目录

```
rk@rk:~/rknn-toolkit-v1.1.0/package$ cd ../example/mobilenet_v1
rk@rk:~/rknn-toolkit-v1.1.0/example/mobilenet_v1$
```

2. 执行 test.py 脚本

```
rk@rk:~/rknn-toolkit-v1.1.0/example/mobilenet_v1$ python3 test.py
```

3. 脚本执行完后得到如下结果：

```
--> 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
=====
```

Layer ID	Name	Time(us)
0	tensor.transpose_3	72
45	convolution.relu.pooling.layer2_2	363
60	convolution.relu.pooling.layer2_2	200
46	convolution.relu.pooling.layer2_2	185
61	convolution.relu.pooling.layer2_2	242
47	convolution.relu.pooling.layer2_2	98
62	convolution.relu.pooling.layer2_2	149



```
48      convolution.relu.pooling.layer2_2      152
63      convolution.relu.pooling.layer2_2      120
49      convolution.relu.pooling.layer2_2      116
64      convolution.relu.pooling.layer2_2      101
50      convolution.relu.pooling.layer2_2      185
65      convolution.relu.pooling.layer2_2      101
51      convolution.relu.pooling.layer2_2      111
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      213
71      convolution.relu.pooling.layer2_2      109
57      convolution.relu.pooling.layer2_2      174
72      convolution.relu.pooling.layer2_2      219
58      convolution.relu.pooling.layer2_2      353
59      fullyconnected.relu.layer_3           110
30      tensor.transpose_3                    5
Total Time(us): 4775
FPS(800MHz): 209.42
=====
done
```

这个例子涉及到的主要操作有：创建 RKNN 对象；模型配置；加载 TensorFlow Lite 模型；构建 RKNN 模型；导出 RKNN 模型；加载图片并推理，得到 TOP5 结果；评估模型性能；释放 RKNN 对象。

example 目录中的 mobilenet\_v2 和 mobilenet-ssd 示例的执行方式与 mobilenet\_v1 相同，只是 mobilenet-ssd 示例中要执行的脚本是 ssd.py，且该示例执行完后会输出一张 out.jpg 图像，它会把检测到的对象在图像中用方框框出来。

### 3.3.2 在 RK1808 上运行示例

这里以 mobilenet\_v1 为例。工具包中带的 mobilenet\_v1 示例是在 PC 模拟器上运行的，如果要在 RK1808 设备上运行这个示例，可以参考以下步骤：

1. 进入 example/mobilenet\_v1 目录

```
rk@rk:~/rknn-toolkit-v1.1.0/example/mobilenet_v1$
```

2. 修改 test.py 脚本里的初始化环境变量时带的参数

```
rk@rk:~/rknn-toolkit-v1.1.0/example/mobilenet_v1$ vim test.py
# 找到脚本里初始化环境变量的方法 init_runtime, 如下
ret = rknn.init_runtime()
# 修改该方法的参数
ret = rknn.init_runtime(target='rk1808', device_id='0123456789ABCDEF')
# 保存修改并退出
```

3. 执行 test.py 脚本, 得到如下结果:

```
rk@rk:~/rknn-toolkit-v1.1.0/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): 5840
FPS: 171.23
=====

done
```

## 4 Windows 平台（Python3.6）快速上手指南

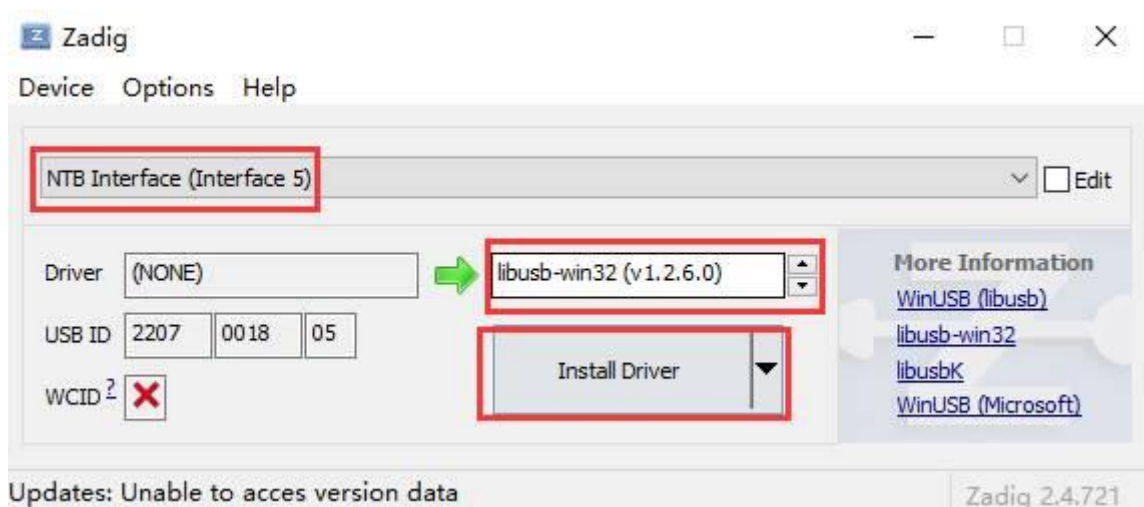
本章节说明如何在 Windows 系统、Python3.6 环境中使用 RKNN-Toolkit。

### 4.1 环境准备

- 一台安装有 Windows 7（或更高版本）操作系统的 PC。
- 一个 TB-RK1808 AI 计算棒（Windows 平台目前只支持计算棒）。
- 将 TB-RK1808 AI 计算棒通过 USB 连接到 PC 上。第一次使用计算棒时需要安装相应的驱动，安装方式如下：

- 进入 SDK 包 platform-tools/drivers\_installer/windows-x86\_64 目录，以管理员身份运行 zadig-2.4.exe 程序安装计算棒的驱动，如下图所示：

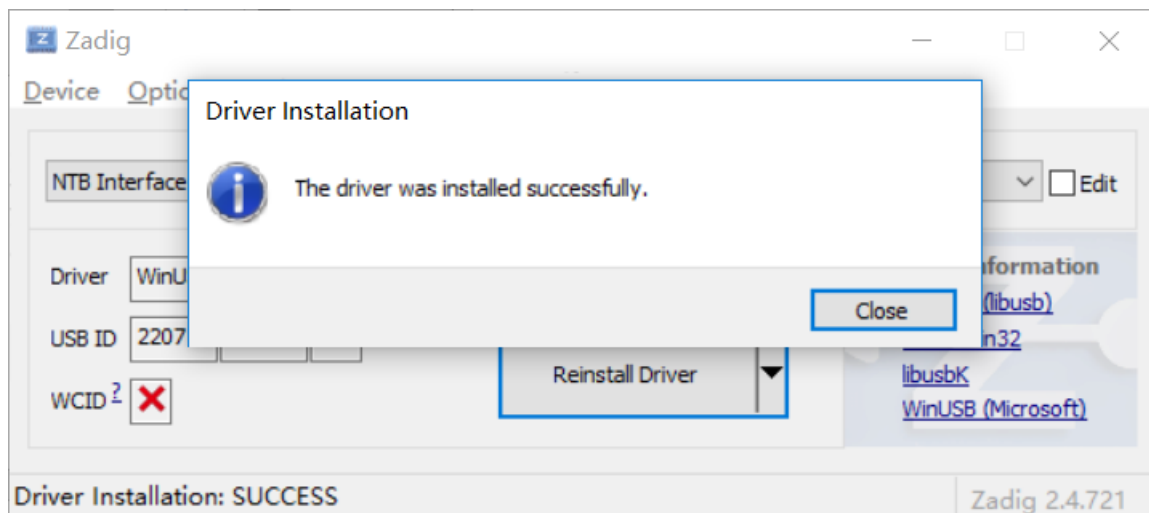
1. 确认待安装的设备及需要安装的驱动



注：待安装的设备其 USB ID 应该是 2207:0018；安装的驱动选择 libusb-win32(v1.2.6.0)。

确认完后点 Install Driver 开始安装驱动。

2. 安装成功后会出现如下界面：



- 安装完后如果 windows 设备管理器的中的 TB-RK1808 AI 计算棒没有感叹号, 且如下所示, 说明安装成功:



注:

1. 安装完驱动后需要重启计算机。
2. 安装驱动和使用 TB-RK1808 AI 计算棒时请关闭杀毒软件、安全卫士或电脑管家之类的工具; 如果关闭后安装仍失败, 请将它们卸载掉后再安装。

## 4.2 安装 RKNN-Toolkit

安装 RKNN-Toolkit 前需要确保系统里已经安装有 Python3.6。这可以通过 cmd 里执行 `python --version` 确定, 如下说明系统已经安装有 Python3.6:

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

获取 RKNN-Toolkit SDK 包, 然后执行以下步骤:

1. 在 sdk 根目录以管理员权限执行 cmd, 然后进入 package 目录:

```
D:\workspace\rknn-toolkit-v1.1.0>cd packages
```

2. 安装依赖 (tensorflow 和 opencv-python):

```
D:\workspace\rknn-toolkit-v1.1.0\packages>pip install tensorflow==1.13.1
D:\workspace\rknn-toolkit-v1.1.0\packages>pip install opencv-python
```

注：opencv-python 在运行 example 中的例子时会用到。

3. 手动安装 lmbd，该 wheel 包放在 packages\required-packages-for-win7-python36 目录下：

```
D:\workspace\rknn-toolkit-v1.1.0\packages\required-packages-for-win-python36>pip install lmbd-0.95-cp36-cp36m-win_amd64.whl
```

4. 安装 RKNN-Toolkit

```
pip install rknn_toolkit-1.1.0-cp36-cp36m-win_amd64.whl
```

5. 检查 RKNN-Toolkit 是否安装成功

```
D:\workspace\rknn-toolkit-v1.1.0\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
>>>
```

如果导入 RKNN 模块没有失败，说明安装成功。

## 4.3 运行安装包中附带的示例

这里以 mobilenet\_v1 为例。示例中的 mobilenet\_v1 是一个 Tensorflow Lite 模型，用于图片分类。

运行该示例的步骤如下：

1. 进入 example/mobilenet\_v1 目录

```
D:\workspace\rknn-toolkit-v1.1.0\packages>cd ..\
D:\workspace\rknn-toolkit-v1.1.0>cd example\mobilenet_v1
```

2. 修改脚本 test.py 脚本，找到调用 init\_runtime 接口的地方，添加参数 target='rk1808'：

```
#修改前：
```

```
ret = rknn.init_runtime()
# 修改后:
ret = rknn.init_runtime(target='rk1808')
```

3. 执行 test.py 的脚本:

```
D:\workspace\rknn-toolkit-v1.1.0\example\mobilenet_v1>python test.py
```

4. 脚本执行完后得到概率最高的前 5 类结果及模型运行的参考性能:

```
--> 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): 7419
FPS: 134.79
=====

done
```

这个例子涉及到的主要操作有：创建 RKNN 对象；模型配置；加载 TensorFlow Lite 模型；构建 RKNN 模型；导出 RKNN 模型；加载图片并推理，得到 TOP5 结果；评估模型性能；释放 RKNN 对象。

example 目录中的 mobilenet\_v2 和 mobilenet-ssd 示例的执行方式与 mobilenet\_v1 相同，只是

mobilenet-ssd 示例中要执行的脚本是 `ssd.py`，且该示例执行完后会输出一张 `out.jpg` 图像，它会把检测到的对象在图像中用方框框出来。

注：

1. Windows 平台并不提供 NPU 模拟器功能，所以在 Windows 平台上必须接 TB-RK1808 计算棒进行使用。
2. 更多有关 TB-RK1808 AI 计算棒的资料，请参考链接：

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

## 5 Mac OS X 平台（Python3.6）快速上手指南

本章节说明如何在 Mac OS X 系统、Python3.6 环境中使用 RKNN-Toolkit。

### 5.1 环境准备

- 一台安装有 MacOS High Sierra（或更高版本）操作系统的 Mac PC。
- 一个 TB-RK1808 AI 计算棒。
- 将 TB-RK1808 AI 计算棒通过 USB 连接到 PC 上。在 pc 上进入 SDK 包 platform-tools/ntp/mac-osx-x86\_64 目录，运行 npu\_transfer\_proxy 程序查看是否存在可用的 RK1808 计算棒，命令如下：

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

上图标红的这一行即为我们插入的 TB-RK1808 AI 计算棒。设备 ID 为“TS018080000000013”。

### 5.2 安装 RKNN-Toolkit

获取 RKNN-Toolkit SDK 包，然后执行以下步骤：

1. 进入 rknn-toolkit-1.1.0/packages 目录：

```
cd packages/
```

2. 安装 tensorflow、opencv-python 依赖

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

注：opencv-python 在运行 example 中的例子时会用到。

3. 安装 RKNN-Toolkit



```
pip3 install rknn_toolkit-1.1.0-cp36-cp36m-macosx_10_9_x86_64.whl
```

4. 检查 RKNN-Toolkit 是否安装成功

```
(rknn-venv)macmini:rknn-toolkit-1.1.0 rk$ python3  
>>> from rknn.api import RKNN  
>>>
```

如果导入 RKNN 模块没有失败，说明安装成功。

## 5.3 运行安装包中附带的示例

这里以 mobilenet\_v1 为例。示例中的 mobilenet\_v1 是一个 Tensorflow Lite 模型，用于图片分类。

运行该示例的步骤如下：

1. 进入 example/mobilenet\_v1 目录

```
(rknn-venv)macmini:rknn-toolkit-1.1.0 rk$ cd example/mobilenet_v1
```

2. 修改脚本 test.py 脚本，找到调用 init\_runtime 接口的地方，添加参数 target='rk1808'：

```
#修改前：  
ret = rknn.init_runtime()  
#修改后：  
ret = rknn.init_runtime(target='rk1808')
```

3. 执行 test.py 脚本

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

4. 脚本执行完后得到 Top5 结果：

```
--> 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): 7419
FPS: 134.79
=====

done
```

这个例子涉及到的主要操作有：创建 RKNN 对象；模型配置；加载 TensorFlow Lite 模型；构建 RKNN 模型；导出 RKNN 模型；加载图片并推理，得到 TOP5 结果；评估模型性能；释放 RKNN 对象。

example 目录中的 mobilenet\_v2 和 mobilenet-ssd 示例的执行方式与 mobilenet\_v1 相同，只是 mobilenet-ssd 示例中要执行的脚本是 ssd.py，且该示例执行完后会输出一张 out.jpg 图像，它会把检测到的对象在图像中用方框框出来。

注：

1. Mac OS X 平台并不提供 NPU 模拟器功能，所以在 Mac OS X 平台上必须接 TB-RK1808 计算棒进行使用。
2. 更多有关 TB-RK1808 AI 计算棒的资料，请参考链接：

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

## 6 ARM64 平台（Python3.5）快速上手指南

本章节说明如何在 ARM64 平台（Debian 9.8 系统）、Python3.5 环境中使用 RKNN-Toolkit。

### 6.1 环境准备

- 一台安装有 Debian 9.8 操作系统的 RK3399Pro，并且确保 root 分区剩余空间大于 5GB。
- 确保 NPU 驱动版本大于 0.9.6。
- 如果 /usr/bin 目录下没有 npu\_transfer\_proxy 或 npu\_transfer\_proxy.proxy 程序，则将 rknn-toolkit-1.1.0\platform-tools\ntp\linux\_aarch64 目录下的 npu\_transfer\_proxy 拷贝到 /usr/bin/ 目录下，并进到该目录执行以下命令（每次重启后都要启动该程序，可以将它加到开机脚本中）：

```
sudo ./npu_transfer_proxy &
```

### 6.2 安装 RKNN-Toolkit

1. 执行以下命令更新系统包，这些包在后面安装 Python 依赖包时会用到。

```
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. 执行以下命令更新 pip

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

更新完还需要修改 /usr/bin/pip3，否则安装其他依赖包时会报 pip3 的错误。修改 /usr/bin/pip3，如下：

```
from pip import main    -->    from pip import __main__
...
```

```
sys.exit(main())    -->    sys.exit(__main__.__main__())
```

### 3. 安装 Python 打包工具

```
pip3 install wheel setuptools
```

### 4. 安装依赖包 h5py

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

### 5. 安装 TensorFlow，相应的 whl 包在

rknn-toolkit-1.1.0\packages\required-packages-for-arm64-debian9-python35 目录下：

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

注：由于 TensorFlow 依赖的一些库在 ARM64 平台上需要下载后源码后进行编译安装，所以这一步会耗费较长时间。

### 6. 安装 opencv-python，相应的 whl 包在

rknn-toolkit-1.1.0\packages\required-packages-for-arm64-debian9-python35 目录下：

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

### 7. 安装 RKNN-Toolkit，相应的 whl 包在 rknn-toolkit-1.1.0\packages\目录下：

```
pip3 install rknn_toolkit-1.1.0-cp35-cp35m-linux_aarch64.whl --user
```

注：由于 RKNN-Toolkit 依赖的一些库在 ARM64 平台上需要下载源码后编译安装，所以这一步会耗费较长时间。

## 6.3 运行安装包中附带的示例

这里以 mobilenet\_v1 为例。示例中的 mobilenet\_v1 是一个 Tensorflow Lite 模型，用于图片分类。

运行该示例的步骤如下：

1. 进入 example/mobilenet\_v1 目录

```
linaro@linaro-alip:~/rknn-toolkit-1.1.0/ $ cd example/mobilenet_v1
```

2. 执行 test.py 脚本

```
linaro@linaro-alip: ~/rknn-toolkit-1.1.0/example/mobilenet_v1$ python3 test.py
```

3. 脚本执行完后得到如下结果:

```
--> 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): 5790
FPS: 172.71
=====

done
```

这个例子涉及到的主要操作有：创建 RKNN 对象；模型配置；加载 TensorFlow Lite 模型；构建 RKNN 模型；导出 RKNN 模型；加载图片并推理，得到 TOP5 结果；评估模型性能；释放 RKNN 对象。

example 目录中的 mobilenet\_v2 和 mobilenet-ssd 示例的执行方式与 mobilenet\_v1 相同，只是 mobilenet-ssd 示例中要执行的脚本是 ssd.py，且该示例执行完后会输出一张 out.jpg 图像，它会把检测到的对象在图像中用方框框出来。

注：

1. ARM64 平台不支持模拟器功能，所以这些示例都是跑在 RK3399Pro 自带的 NPU 上。
2. ARM64 平台目前只支持 RK3399 和 RK3399Pro，如果是 RK3399，需要外接一个 TB-RK1808 AI 计算棒。
3. 有关 TB-RK1808 AI 计算棒的更多资料，请参考链接：

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

## 7 参考文档

有关 RKNN-Toolkit 更详细的用法和接口说明，请参考《RKNN-Toolkit 使用指南\_V1.1.0.pdf》手册。