密级状态：绝密(　　)　　秘密(　　)　　内部(　　)　　公开(√)

# RKNN Toolkit 快速上手指南

（技术部，图形计算平台中心）

| | | |
|---|---|---|
| 文件状态：<br>[　] 正在修改<br>[√] 正式发布 | 当前版本： | V1.6.1 |
| | 作　　者： | 饶洪 |
| | 完成日期： | 2021-05-21 |
| | 审　　核： | 熊伟 |
| | 完成日期： | 2021-05-21 |



瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd

# 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 | 熊伟 |
| V1.0.0 | 饶洪 | 2019-05-08 | 同步 RKNN-Toolkit-V1.0.0 修改内容 | 熊伟 |
| V1.1.0 | 饶洪 | 2019-06-28 | 1. 同步 RKNN-Toolkit-V1.1.0 修改内容<br>2. 新增 Windows/MacOS/ARM64 等平台的快速上手指南 | 熊伟 |
| V1.2.0 | 饶洪 | 2019-08-21 | 同步 RKNN-Toolkit-V1.2.0 修改内容 | 熊伟 |
| V1.2.1 | 饶洪 | 2019-09-26 | 同步 RKNN-Toolkit-V1.2.1 修改内容 | 熊伟 |
| V1.3.0 | 饶洪 | 2019-12-23 | 同步 RKNN-Toolkit-V1.3.0 修改内容 | 熊伟 |
| V1.3.2 | 饶洪 | 2020-04-03 | 同步 RKNN-Toolkit-V1.3.2 修改内容 | 熊伟 |
| V1.4.0 | 饶洪 | 2020-08-13 | 同步 RKNN-Toolkit-V1.4.0 修改内容 | 熊伟 |
| V1.6.0 | 饶洪 | 2020-12-31 | 同步 RKNN-Toolkit-V1.6.0 修改内容 | 熊伟 |
| V1.6.1 | 饶洪 | 2021-05-21 | 同步 RKNN-Toolkit-V1.6.1 修改内容 | 熊伟 |

# 目 录

# 1 主要功能说明

RKNN-Toolkit 是为用户提供在 PC、 Rockchip NPU 平台上进行模型转换、推理和性能评估的开发套件，用户通过该工具提供的 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、PyTorch、MXNet 和 Keras 模型转为 RKNN 模型，并支持 RKNN 模型导入导出，RKNN 模型能够在 Rockchip NPU 平台上加载使用。从 1.2.0 版本开始支持多输入模型。从 1.3.0 版本开始支持 PyTorch 和 MXNet。从 1.6.0 版本开始支持 Keras 框架模型，并支持 TensorFlow 2.0 导出的 H5 模型。
2) 量化功能：支持将浮点模型量化为定点模型，目前支持的量化方法为非对称量化（asymmetric_quantized-u8），动态定点量化（dynamic_fixed_point-8 和 dynamic_fixed_point-16）。从 1.0.0 版本开始，RKNN-Toolkit 开始支持混合量化功能。从 1.6.1 版本开始，RKNN Toolkit 提供量化参数优化算法 MMSE。
3) 模型推理：能够在 PC 上模拟 Rockchip NPU 运行 RKNN 模型并获取推理结果；或将 RKNN 模型分发到指定的 NPU 设备上进行推理。
4) 性能评估：能够在 PC 上模拟 Rockchip NPU 运行 RKNN 模型，并评估模型性能（包括总耗时和每一层的耗时）；或将 RKNN 模型分发到指定 NPU 设备上运行，以评估模型在实际设备上运行时的性能。
5) 内存评估：评估模型运行时内存的占用情况。使用该功能时，必须将 RKNN 模型分发到 NPU 设备中运行，并调用相关接口获取内存使用信息。从 0.9.9 版本开始支持该功能。
6) 模型预编译：通过预编译技术生成的 RKNN 模型可以减少 NPU 加载模型的时间。通过预编译技术生成的 RKNN 模型只能在 NPU 硬件上运行，不能在模拟器中运行。当前只有 x86_64 Ubuntu 平台支持直接从原始模型生成预编译 RKNN 模型。RKNN Toolkit 从 0.9.5 版本开始支持模型预编译功能，并在 1.0.0 版本中对预编译方法进行了升级，升级后的预编译模型无法与旧驱动兼容。从 1.4.0 版本开始，支持通过 NPU 设备将普通 RKNN 模型转为预编译 RKNN 模型，详情请参考接口 export_rknn_precompile_model 的使用说明。
7) 模型分段：该功能用于多模型同时运行的场景。将单个模型分成多段在 NPU 上执行，借

此来调节多个模型占用 NPU 的时间，避免因为一个模型占用太多 NPU 时间，而使其他模型无法及时执行。RKNN Toolkit 从 1.2.0 版本开始支持该功能。目前，只有 RK1806/RK1808/RV1109/RV1126 芯片支持该功能，且 NPU 驱动版本要大于 0.9.8。

8) 自定义算子功能：如果模型含有 RKNN Toolkit 不支持的算子（operator），那么在模型转换阶段就会失败。这时候可以使用自定义算子功能以添加 RKNPU 不支持的算子，从而使模型能正常转换和运行。RKNN-Toolkit 从 1.2.0 版本开始支持该功能。自定义算子的使用和开发请参考《Rockchip_Developer_Guide_RKNN_Toolkit_Custom_OP_CN》文档。自定义算子目前只支持 TensorFlow 框架。

9) 量化精度分析功能：该功能将给出模型量化前后每一层推理结果与浮点模型推理结果的欧氏距离和余弦距离，以便于分析量化误差是如何出现的，为提高量化模型的精度提供思路。该功能从 1.3.0 版本开始支持。1.4.0 版本增加逐层量化精度分析子功能，将每一层运行时的输入指定为正确的浮点值，以排除逐层误差积累，能够更准确的反映每一层自身受量化的影响。

10) 可视化功能：该功能以图形界面的形式呈现 RKNN-Toolkit 的各项功能，简化用户操作步骤。用户可以通过填写表单、点击功能按钮的形式完成模型的转换和推理等功能，而不需要再去手动编写脚本。有关可视化功能的具体使用方法请参考《Rockchip_User_Guide_RKNN_Toolkit_Visualization_CN》文档。1.3.0 版本开始支持该功能。1.4.0 版本完善了对多输入模型的支持，并且支持 RK1806, RV1109, RV1126 等新的 RK NPU 设备。1.6.0 版本增加对 Keras 框架的支持。

11) 模型优化等级功能：RKNN-Toolkit 在模型转换过程中会对模型进行优化，默认的优化选项可能会对模型精度产生一些影响。通过设置优化等级，可以关闭部分或全部优化选项。有关优化等级的具体使用方法请参考 config 接口中 optimization_level 参数的说明。该功能从 1.3.0 版本开始支持。

12) 模型加密功能：使用指定的加密方法将 RKNN 模型整体加密。RKNN Toolkit 从 1.6.0 版本开始支持模型加密功能。

# 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<br>Windows 7（x64）及以上<br>Mac OS X 10.13.5（x64）及以上<br>Debian 9.8（aarch64）及以上 |
| Python 版本 | 3.5/3.6/3.7 |
| Python 库依赖 | 'numpy == 1.16.3'<br>'scipy == 1.3.0'<br>'Pillow == 5.3.0'<br>'h5py == 2.8.0'<br>'lmdb == 0.93'<br>'networkx == 1.11'<br>'flatbuffers == 1.10',<br>'protobuf == 3.11.2'<br>'onnx == 1.6.0'<br>'onnx-tf == 1.2.1'<br>'flask == 1.0.2'<br>'tensorflow == 1.11.0' or 'tensorflow-gpu'<br>'dill==0.2.8.2'<br>'ruamel.yaml == 0.15.81'<br>'psutils == 5.6.2'<br>'ply == 3.11'<br>'requests == 2.22.0'<br>'torch == 1.2.0' or 'torch == 1.5.1' or 'torch==1.6.0'<br>'mxnet == 1.5.0'<br>'sklearn == 0.0'<br>'opencv-python == 4.0.1.23' |

注：

1. Windows 只提供 Python3.6 的安装包。
2. MacOS 提供 python3.6 和 python3.7 的安装包。

3. ARM64 平台（安装 Debian 9 或 10 操作系统）提供 Python3.5（Debain 9）和 Python3.7（Debian10）的安装包。
4. 因为 Pytorch / TensorFlow 等逐渐停止对 Python3.5 的支持，RKNN Toolkit 下一个大版本将移除 Linux x86 平台上 Python3.5 的安装包，转而提供 Python3.6 和 Python3.7 的安装包。
5. 除 MacOS 平台外，其他平台的 scipy 依赖为>=1.1.0。
6. ARM64 平台不需要依赖 sklearn 和 opencv-python。

# 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 或 RV1126 EVB 板。
- 将 RK1808 或 RV1126 EVB 板通过 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
515e9b401c060c0b        device
c3d9b8674f4b94f6        device
```

其中标红的为设备 ID，第一个是 RK1808 开发板，第二个是 RV1126 开发板。

## 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==1.14.0
pip3 install mxnet==1.5.0
```

```
pip3 install torch==1.5.1 torchvision==0.4.0
pip3 install gluoncv
```

c) 安装 RKNN-Toolkit

```
sudo pip3 install rknn_toolkit-1.6.1-cp35-cp35m-linux_x86_64.whl
```

d) 检查 RKNN-Toolkit 是否安装成功

```
rk@rk:~/rknn-toolkit-v1.6.1/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

Linux x86_64 上的 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。如果要模拟 RV1126，需要在 config 里设置 target_platform=[‘rv1126’]。

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 examples/tflite/mobilenet_v1 目录

```
rk@rk:~/rknn-toolkit-v1.6.1/package$ cd ../examples/tflite/mobilenet_v1
rk@rk:~/rknn-toolkit-v1.6.1/examples/tflite/mobilenet_v1$
```

2. 执行 test.py 脚本

```
rk@rk:~/rknn-toolkit-v1.6.1/examples/tflite/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]: 0.8642578125
[155]: 0.083740234375
[205]: 0.01241302490234375
[284]: 0.006565093994140625
[194]: 0.002044677734375

done
--> Begin evaluate model performance
W When performing performance evaluation, inputs can be set to None to use fake inputs.
=============================================================
                        Performance
=============================================================
Layer ID    Name                                Time(us)
60          openvx.tensor_transpose_3           72
1           convolution.relu.pooling.layer2_2   370
3           convolution.relu.pooling.layer2_2   213
5           convolution.relu.pooling.layer2_2   186
7           convolution.relu.pooling.layer2_2   299
9           convolution.relu.pooling.layer2_2   99
11          convolution.relu.pooling.layer2_2   140
13          convolution.relu.pooling.layer2_2   103
15          convolution.relu.pooling.layer2_2   133
17          convolution.relu.pooling.layer2_2   102
19          convolution.relu.pooling.layer2_2   110
21          convolution.relu.pooling.layer2_2   169
23          convolution.relu.pooling.layer2_2   112
25          convolution.relu.pooling.layer2_2   108
27          convolution.relu.pooling.layer2_2   127
29          convolution.relu.pooling.layer2_2   210
31          convolution.relu.pooling.layer2_2   127
33          convolution.relu.pooling.layer2_2   210
35          convolution.relu.pooling.layer2_2   127
37          convolution.relu.pooling.layer2_2   210
39          convolution.relu.pooling.layer2_2   127
41          convolution.relu.pooling.layer2_2   210
43          convolution.relu.pooling.layer2_2   127
45          convolution.relu.pooling.layer2_2   210
47          convolution.relu.pooling.layer2_2   109
49          convolution.relu.pooling.layer2_2   172
51          convolution.relu.pooling.layer2_2   220
```

```
53              convolution.relu.pooling.layer2_2       338
55              pooling.layer2                          34
56              fullyconnected.relu.layer_3             110
58              softmaxlayer2.layer                     39
Total Time(us): 4923
FPS(600MHz): 152.35
FPS(800MHz): 203.13
Note: Time of each layer is converted according to 800MHz!
============================================================

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。

examples 目录中的其他示例的执行方式与 mobilenet_v1 相同，这些模型主要用于分类、目标检测和图像分割。

### 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 EVB 板上运行这个示例，可以参考以下步骤：

1. 进入 examples/tflite/mobilenet_v1 目录

```
rk@rk:~/rknn-toolkit-v1.6.1/examples/tflite/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-v1.6.1/examples/tflite/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='515e9b401c060c0b')
# 保存修改并退出
```

3. 执行 test.py 脚本，得到如下结果：

```
rk@rk:~/rknn-toolkit-v1.6.1/examples/tflite/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]: 0.85205078125
[155]: 0.09185791015625
[205]: 0.012237548828125
[284]: 0.006473541259765625
[194]: 0.0024929046630859375

done
--> Begin evaluate model performance
W When performing performance evaluation, inputs can be set to None to use fake inputs.
========================================================================
                                Performance
========================================================================
Total Time(us): 5499
FPS: 181.85
========================================================================

done
```

### 3.3.3 在 RV1126 上运行示例

RV1126 与 RK1808 EVB 板类似，但是在调用 config 接口时，需要指定 target_platform 为 RV1126，在 init_runtime 时，target 也要填 RV1126。具体步骤如下：

1. 进入 examples/tflite/mobilenet_v1 目录

```
rk@rk:~/rknn-toolkit-v1.6.1/examples/tflite/mobilenet_v1$
```

2. 修改 test.py 脚本里的 config 参数和 init_runtime 参数

```
rk@rk:~/rknn-toolkit-v1.6.1/examples/tflite/mobilenet_v1$ vim test.py
# 找到脚本里调用模型配置接口 config 的地方，如下
rknn.config(channel_mean_value='128 128 128 128', reorder_channel='0 1 2')
```

```
    # 修改该接口的参数
    rknn.config(channel_mean_value='128 128 128 128', reorder_channel='0 1 2',
target_platform=['rv1126'])
    # 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='rv1126', device_id='c3d9b8674f4b94f6')
    # 保存修改并退出
```

3. 执行 test.py 脚本，得到如下结果：

```
    rk@rk:~/rknn-toolkit-v1.6.1/examples/tflite/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
```

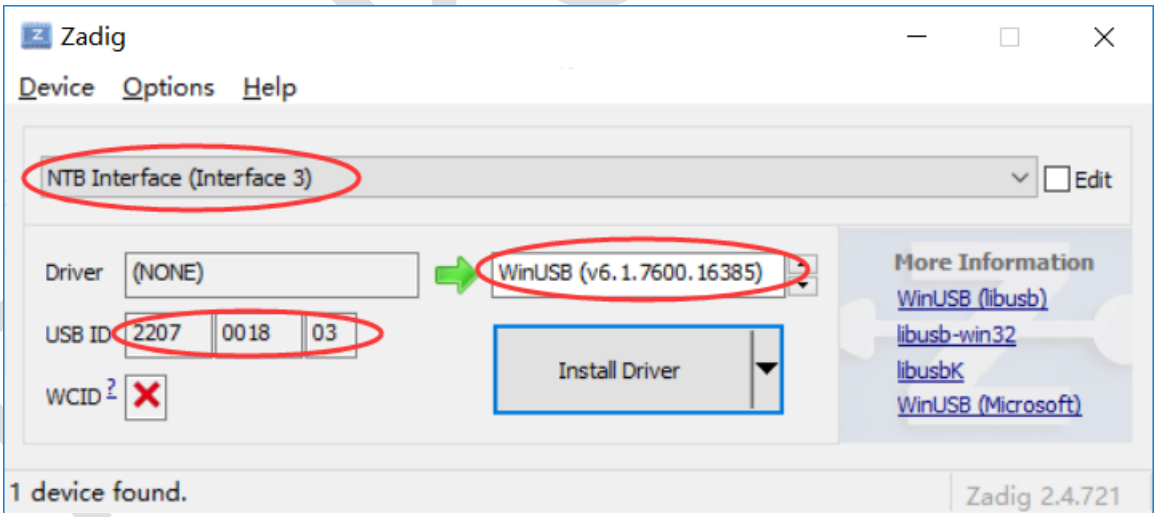
# 4 Windows 平台（Python3.6）快速上手指南

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

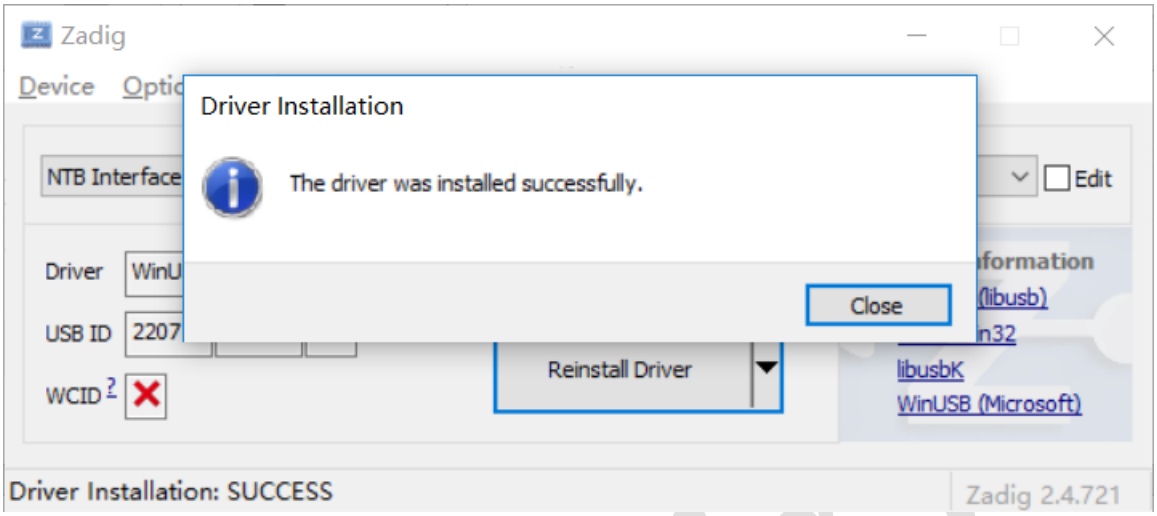
- 一台安装有 Windows 10（或 Windows7）操作系统的 PC。
- 一个 RK1808 或 RV1126 开发板。
- 将 RK1808 或 RV1126 开发板通过 USB 连接到 PC 上。第一次使用开发板时需要安装相应的驱动，安装方式如下：
  - 进入 SDK 包 platform-tools/drivers_installer/windows-x86_64 目录，以管理员身份运行 zadig-2.4.exe 程序安装计算棒的驱动，如下图所示：

    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 应该以 **2207** 开头；安装的驱动选择默认的 WinUSB 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 NTB Interface 设备没有感叹号，且如下所示，说明安装成功：


**注：安装完驱动后需要重启计算机。**

## 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\rk>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-v1.6.1>cd packages
```

2. 安装 Python 依赖：

```
pip install tensorflow==1.14.0
pip install torch==1.6.0+cpu torchvision==0.7.0+cpu -f
https://download.pytorch.org/whl/torch_stable.html --user
pip install mxnet==1.5.0
pip install gluoncv
```

```

```

注：gluoncv 在运行 example 中的例子时会用到。

3. 安装 RKNN-Toolkit

```
pip install rknn_toolkit-1.6.1-cp36-cp36m-win_amd64.whl
```

4. 检查 RKNN-Toolkit 是否安装成功

```
D:\workspace\rknn-toolkit-v1.6.1\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 在 RK1808 上运行示例

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 examples/tflite/mobilenet_v1 目录

```
D:\workspace\rknn-toolkit-v1.6.1\packages>cd ..\

D:\workspace\rknn-toolkit-v1.6.1>cd examples\tflite\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-v1.6.1\examples\tflite\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]: 0.8828125
[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
```

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。

examples 目录中的其他示例的执行方式与 mobilenet_v1 相同，这些模型主要用于图像分类、目标检测和图像分割。

注：

1. Windows 平台并不提供 NPU 模拟器，所以在 Windows 平台上必须接 Rockchip NPU 设备才可以使用推理/性能评估/内存评估等功能。

## 4.4 在 RV1126 上运行示例

Windows 平台上使用 RV1126 运行示例时需要修改的地方和运行步骤与 Ubuntu 平台相同，这里不再赘述。

# 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 RK1808 或 RV1126 开发板。
- 将 RK1808 或 RV1126 开发板通过 USB（OTG 口）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 Rockchip NPU 设备，命令如下：

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

上图标红的这一行即为我们插入的 RK1808 开发板。设备 ID 为“515e9b401c060c0b”。

## 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-v1.6.1/packages 目录：

```
cd packages/
```

2. 安装 Python 依赖

```
pip3 install tensorflow==1.14.0
pip3 install mxnet==1.5.0
pip3 install torch==1.6.0 torchvision==0.7.0
pip3 install gluoncv
```

注：gluoncv 在运行 example 中的例子时会用到。

3. 安装 RKNN-Toolkit

```
pip3 install rknn_toolkit-1.6.1-cp36-cp36m-macosx_10_15_x86_64.whl
```

4. 检查 RKNN-Toolkit 是否安装成功

```
(rknn-venv)macmini:rknn-toolkit-v1.6.1 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 在 RK1808 上运行示例

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 examples/tflite/mobilenet_v1 目录

```
(rknn-venv)macmini:rknn-toolkit-v1.6.1 rk$ cd examples/tflite/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]: 0.85107421875
[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
```

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。

examples 目录中的其他示例的执行方式与 mobilenet_v1 相同，这些模型主要用于图像分类、目标检测和图像分割。

注：

1. Mac OS X 平台并不提供 NPU 模拟器功能，所以在 Windows 平台上必须接 Rockchip NPU 设备才可以使用推理/性能评估/内存评估等功能。

## 5.4 在 RV1126 上运行示例

Mac OS 平台上使用 RV1126 运行示例时需要修改的地方和运行步骤与 Ubuntu 平台相同，这里不再赘述。

# 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。
- 如果/usr/bin 目录下没有 npu_transfer_proxy 或 npu_transfer_proxy.proxy 程序，则将 rknn-toolkit-v1.6.1\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
```

3. 安装 Python 打包工具

```
pip3 install wheel setuptools
```

4. 安装依赖包 h5py/gluoncv

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

```
pip3 install h5py
pip3 install gluoncv
```

5. 安装 TensorFlow，由于 debian9 系统上没有相应的源，需要在网上自行搜索 whl 包安装。

6. 安装 torch 和 torchvision

```
pip3 install torch==1.2.0 torchvision==0.2.2 --user
```

7. 安装 mxnet

```
pip3 install mxnet==1.5.0
```

8. 安装 opencv-python，由于 debian9 系统上没有相应的源，需要在网上自行搜索 whl 包安装。

9. 安装 RKNN-Toolkit，相应的 whl 包在 rknn-toolkit-v1.6.1\packages\目录下：

```
pip3 install rknn_toolkit-1.6.1-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 examples/tflite/mobilenet_v1 目录

```
linaro@linaro-alip:~/rknn-toolkit-v1.6.1/ $ cd examples/tflite/mobilenet_v1
```

2. 执行 test.py 脚本

```
linaro@linaro-alip: ~/rknn-toolkit-v1.6.1/examples/tflite/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]: 0.85107421875
[155]: 0.09173583984375
[205]: 0.013587951660156255
[284]: 0.006465911865234375
[194]: 0.002239227294921875

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

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。

examples 目录中的其他示例的执行方式与 mobilenet_v1 相同，这些模型主要用于图像分类、目标检测和图像分割。

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。

# 7 参考文档

有关 RKNN-Toolkit 更详细的用法和接口说明，请参考《Rockchip_User_Guide_RKNN_Toolkit_v1.6.1_CN.pdf》手册。