密级状态：绝密(　　)　　秘密(　　)　　内部(　　)　　公开(√)

RKNN Toolkit Lite 用户使用指南

（技术部，图形计算平台中心）

文件状态： [　] 正在修改 [√] 正式发布	当前版本：	V1.7.5
	作　　者：	饶洪
	完成日期：	2023-07-31
	审　　核：	熊伟
	完成日期：	2023-07-31

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
V1.4.0	饶洪	2020-08-13	初始版本	熊伟
V1.6.0	饶洪	2020-12-31	更新版本号	熊伟
V1.6.1	饶洪	2021-05-21	更新版本号	熊伟
V1.7.0	饶洪	2021-08-08	增加对 ARM 32 位平台的支持	熊伟
V1.7.1	饶洪	2021-11-17	更新版本号	熊伟
V1.7.3	饶洪	2022-08-07	增加 Ubuntu20.04/Python3.8 的支持和相关说明	熊伟
V1.7.5	饶洪	2023-07-31	1. 完善 docker 镜像使用说明； 2. 增加对 ARM64 平台 Python 3.8 / 3.9 / 3.10 的支持和相关包安装说明	熊伟

目 录

1 主要功能说明

RKNN Toolkit Lite 为 RK1808 等带有 Rockchip NPU 的平台提供 Python 编程接口，帮助用户部署 RKNN 模型，加速 AI 应用的落地。

1.1 适用芯片

- RK1806
- RK1808
- RK3399Pro
- RV1109
- RV1126

1.2 适用系统

- Ubuntu: 18.04（x64）及以上
- Windows：7（x64）及以上
- MacOS: 10.13.15（x64）及以上
- Debian: 9.8（aarch64）及以上
- Debian: 10 (armhf / aarch64)

2 开发环境部署

2.1 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 2-1 运行环境

操作系统版本	Ubuntu18.04（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 Debian 10 及以上
Python 版本	3.5 / 3.6 / 3.7 / 3.8 / 3.9 / 3.10
Python 库依赖	'numpy' 'ruamel.yaml' 'psutils'

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. ARM 64 位平台提供 Python 3.5 / 3.6 / 3.7 / 3.8 / 3.9 / 3.10 的安装包。
4. ARM 32 位平台提供 Python3.7 的安装包。

2.2 工具安装

目前提供两种方式安装 RKNN Toolkit Lite：一是通过 pip install 命令安装；二是运行带完整 RKNN Toolkit Lite 工具包的 docker 镜像。下面分别介绍这两种安装方式的具体步骤。

2.2.1 通过 pip 命令安装

1. 创建 virtualenv 环境（如果系统中同时有多个版本的 Python 环境，建议使用 virtualenv 管理 Python 环境）

```
#  以 Ubuntu 18.04 为例
sudo apt install virtualenv
sudo apt-get install libpython3
sudo apt install python3-tk

virtualenv -p /usr/bin/python3 venv
source venv/bin/activate
```

2. 安装依赖模块: opencv-python

```
# Ubuntu 18.04 / 20.04 or Windows 7/10 or MacOS Catalina
pip3 install opencv-python

# Debian 9.8 with Python3.5
pip3 install \
opencv_python_headless-4.0.1.23-cp35-cp35m-linux_aarch64.whl

# Debian 10 with Python3.7
sudo apt-get install python3-dev python3-opencv
```

注: RKNN Toolkit Lite 本身并不依赖 opencv-python，但是在示例中需要用到这个模块，所以安装时也一并安装.

3. 安装 RKNN Toolkit Lite

各平台的安装包都放在 SDK 的 packages 文件夹下。进入 packages 文件夹，执行以下命令安装 RKNN Toolkit Lite：

```
# Ubuntu 18.04
pip3 install rknn_toolkit_lite-1.7.x-cp36-cp36m-linux_x86_64.whl

# Ubuntu 20.04
pip3 install rknn_toolkit_lite-1.7.x-cp38-cp38m-linux_x86_64.whl

# Windows
pip3 install rknn_toolkit_lite-1.7.x-cp36-cp36m-win_amd64.whl

# MacOS python3.6
pip3 install rknn_toolkit_lite-1.7.x-cp36-cp36m-macosx_10_15_x86_64.whl

# MacOS python3.7
pip3 install rknn_toolkit_lite-1.7.x-cp37-cp37m-macosx_10_15_x86_64.whl

# ARM64 python3.5
pip3 install rknn_toolkit_lite-1.7.x-cp35-cp35m-linux_aarch64.whl

# ARM64 python3.6
pip3 install rknn_toolkit_lite-1.7.x-cp36-cp36m-linux_aarch64.whl

# ARM64 python3.7
pip3 install rknn_toolkit_lite-1.7.x-cp37-cp37m-linux_aarch64.whl

# ARM64 python3.8
pip3 install rknn_toolkit_lite-1.7.x-cp38-cp38-linux_aarch64.whl

# ARM64 python3.9
pip3 install rknn_toolkit_lite-1.7.x-cp39-cp39-linux_aarch64.whl

# ARM64 python3.10
pip3 install rknn_toolkit_lite-1.7.x-cp310-cp310-linux_aarch64.whl

# ARM32 python3.7
pip3 install rknn_toolkit_lite-1.7.x-cp37-cp37m-linux_armv7l.whl
```

2.2.2 通过 DOCKER 镜像安装

在 docker 文件夹下提供了一个已打包所有开发环境的 Docker 镜像，用户只需要加载该镜像即可直接上手使用 RKNN Toolkit Lite，使用方法如下：

1. 安装 Docker

   请根据官方手册安装 Docker（https://docs.docker.com/install/linux/docker-ce/ubuntu/）。

2. 加载镜像

执行以下命令加载镜像：

```
docker load --input rknn-toolkit-lite-1.7.x-docker.tar.gz
```

加载成功后，执行“docker images”命令能够看到 rknn-toolkit-lite 的镜像，如下所示：

```
REPOSITORY          TAG         IMAGE ID            CREATED             SIZE
rknn-toolkit-lite   1.7.x       xxxxxxxxx           1 hours ago         1.xxGB
```

3. 运行镜像

执行以下命令运行 docker 镜像，运行后将进入镜像的 bash 环境。

```
docker run -t -i --privileged -v /dev/bus/usb:/dev/bus/usb rknn-toolkit-lite:1.7.x /bin/bash
```

如果想将自己代码映射进去可以加上“-v <host src folder>:<image dst folder>”参数，例如:

```
docker run -t -i --privileged -v /dev/bus/usb:/dev/bus/usb -v /home/rk/test:/test rknn-toolkit-lite:1.7.x /bin/bash
```

注：在运行容器时一定要加上**--privileged -v /dev/bus/usb:/dev/bus/usb**，否则在容器中将无法访问 NPU 设备，导致初始化运行时失败。

4. 运行 demo

```
cd /examples/inference_with_lite
python3 test.py
```

注：该镜像是基于 aarch64 的 Ubuntu18.04 镜像制作而成，所以只能运行在装有 Debian 系统的 RK1808, RK3399Pro 开发板上。只能有一个容器使用 RKNN Toolkit Lite 进行推理，且使用前要先确保宿主机上没有 npu_transfer_proxy 在运行。

3 使用说明

在使用 RKNN Toolkit Lite 之前，用户需要先通过 RKNN Toolkit 将各深度学习框架的模型转成 RKNN 模型。

RKNN Toolkit 完整的安装包和使用文档可以从以下链接获取：

https://github.com/rockchip-linux/rknn-toolkit

3.1 RKNN Toolkit Lite 的使用

3.1.1 使用场景

RKNN Toolkit Lite 的使用场景可以分为两种：

- 运行在 PC 上，此时 PC 需要通过 USB 连接带有 RK1808 等芯片的硬件设备。
- 直接运行在装有 Debian 系统的 RK1808、RK3399Pro、RV1109、RV1126 等开发板上。

3.1.2 使用流程

RKNN Toolkit Lite 使用流程如下：

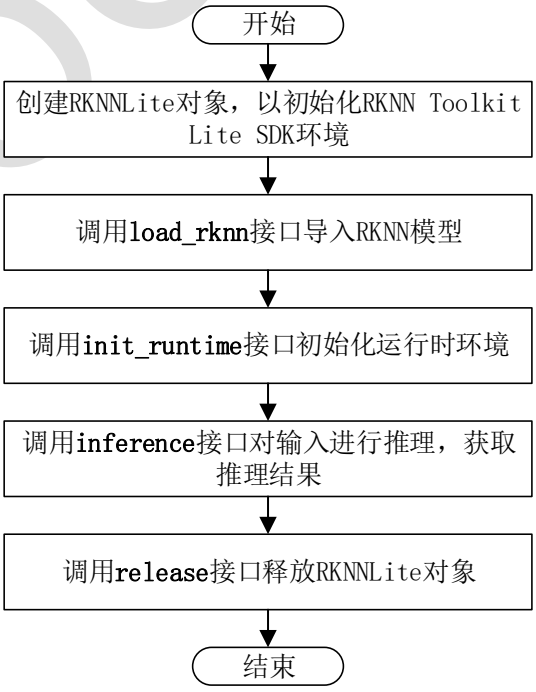

图 3-1 RKNN Toolkit Lite 使用流程

3.2 示例

在 SDK/examples 目录提供了一个使用 RKNN Toolkit Lite 进行模型推理的示例 inference_with_lite。该示例使用的 RKNN 模型可以运行在 RK1806、RK1808、RK3399Pro 上。示例代码如下：

```
import platform
import cv2
import numpy as np
from rknnlite.api import RKNNLite

INPUT_SIZE = 224


def show_top5(result):
    output = result[0].reshape(-1)
    # softmax
    output = np.exp(output)/sum(np.exp(output))
    output_sorted = sorted(output, reverse=True)
    top5_str = 'resnet18\n-----TOP 5-----\n'
    for i in range(5):
        value = output_sorted[i]
        index = np.where(output == value)
        for j in range(len(index)):
            if (i + j) >= 5:
                break
            if value > 0:
                topi = '{}: {}\n'.format(index[j], value)
            else:
                topi = '-1: 0.0\n'
            top5_str += topi
    print(top5_str)


if __name__ == '__main__':
    rknn_lite = RKNNLite()

    # load RKNN model
    print('--> Load RKNN model')
    ret = rknn_lite.load_rknn('./resnet_18.rknn')
    if ret != 0:
        print('Load RKNN model failed')
        exit(ret)
    print('done')

    ori_img = cv2.imread('./space_shuttle_224.jpg')
    img = cv2.cvtColor(ori_img, cv2.COLOR_BGR2RGB)

    # init runtime environment
    print('--> Init runtime environment')
    # run on RK3399Pro/RK1808 with Debian OS, do not need specify
    # target.
    if platform.machine() == 'aarch64':
        target = None
    else:
```

```
        target = 'rk1808'
    ret = rknn_lite.init_runtime(target=target)
    if ret != 0:
        print('Init runtime environment failed')
        exit(ret)
    print('done')

    # Inference
    print('--> Running model')
    outputs = rknn_lite.inference(inputs=[img])
    show_top5(outputs)
    print('done')

    rknn_lite.release()
```

安装 RKNN Toolkit Lite 后执行以下命令运行 demo

```
python3 test.py
```

demo 运行模型预测时输出如下结果：

```
-----TOP 5-----
[812]: 0.999442994594574
[404]: 0.0004096269840374589
[657]: 3.284541890025139e-05
[833]: 2.6112385967280716e-05
[895]: 1.8509887013351545e-05
```

如果要在 RV1109, RV1126 上运行该例子，请使用 RKNN Toolkit 转换适用于这两个平台的 resnet_18 模型。

4 API 详细说明

4.1 RKNNLite 初始化及对象释放

在使用 RKNN Toolkit Lite 时，都需要先调用 RKNNLite()方法初始化一个 RKNNLite 对象，并在用完后调用该对象的 release()方法将资源释放掉。

初始化 RKNNLite 对象时，可以设置 ***verbose*** 和 ***verbose_file*** 参数，以打印详细的日志信息。其中 verbose 参数指定是否要在屏幕上打印详细日志信息；如果设置了 verbose_file 参数，且 verbose 参数值为 True，日志信息还将写到这个参数指定的文件中。

举例如下：

```
# 将详细的日志信息输出到屏幕，并写到 inference.log 文件中
rknn_lite = RKNNLite(verbose=True, verbose_file='./inference.log')
# 只在屏幕打印详细的日志信息
rknn_lite = RKNNLite(verbose=True)
…
rknn_lite.release()
```

4.2 加载 RKNN 模型

API	**load_rknn**
描述	加载 RKNN 模型。
参数	path：RKNN 模型文件路径。
	load_model_in_npu: 是否直接加载 npu 中的 rknn 模型。其中 path 为 rknn 模型在 npu 中的路径。只有当 RKNN Toolkit Lite 运行在连有 NPU 设备的 PC 上或 RK3399Pro Linux 开发板时才可以设为 True。默认值为 False。
返回值	0：加载成功
	-1：加载失败

举例如下：

```
# 从当前目录加载 resnet_18.rknn 模型
ret = rknn_lite.load_rknn('./resnet_18.rknn')
```

4.3 初始化运行时环境

在模型推理之前，必须先初始化运行时环境，确定模型在哪一个芯片平台上运行。

API	**init_runtime**
描述	初始化运行时环境。确定模型运行的设备信息（芯片型号、设备 ID）。
参数	target：目标硬件平台，目前支持“rk3399pro”、“rk1806”、“rk1808”、“rv1109”、“rv1126”。默认为 None，将根据应用所运行的开发板自动选择。
	device_id：设备编号，如果 PC 连接多台智能设备时，需要指定该参数，设备编号可以通过”***list_devices***”接口查看。默认值为 None。
	async_mode：是否使用异步模式。调用推理接口时，涉及设置输入图片、模型推理、获取推理结果三个阶段。如果开启了异步模式，设置当前帧的输入将与推理上一帧同时进行，所以除第一帧外，之后的每一帧都可以隐藏设置输入的时间，从而提升性能。在异步模式下，每次返回的推理结果都是上一帧的。该参数的默认值为 False。
返回值	0：初始化运行时环境成功。
	-1：初始化运行时环境失败。

举例如下：

```
# 初始化运行时环境
ret = rknn_lite.init_runtime(target='rk1808', device_id='012345789AB')
if ret != 0:
    print('Init runtime environment failed')
    exit(ret)
```

4.4 模型推理

API	**inference**
描述	对指定输入进行推理，返回推理结果。
参数	inputs：待推理的输入，如经过 cv2 处理的图片。类型是 list，列表成员是 ndarray。
	data_type：输入数据的类型，可填以下值：’float32’, ‘float16’, ‘uint8’, ‘int8’, ‘int16’。默认值为’uint8’。
	data_format：数据模式，可以填以下值: “nchw”, “nhwc”。默认值为’nhwc’。这两个的不同之处在于 channel 放置的位置。
	inputs_pass_through: 将输入透传给 NPU 驱动。非透传模式下，在将输入传给 NPU 驱动之前，工具会对输入进行减均值、除方差等操作；而透传模式下，不会做这些操作。这个参数的值是一个数组，比如要透传 input0，不透传 input1，则这个参数的值为[1, 0]。默认值为 None，即对所有输入都不透传。
返回值	results：推理结果，类型是 list，列表成员是 ndarray。

举例如下：

以分类模型为例，如 resnet18，代码如下（完整代码参考 examples/inference_with_lite）：

```
# 使用模型对图片进行推理，得到 TOP5 结果
……
outputs = rknn_lite.inference(inputs=[img])
show_top5(outputs)
……
```

输出的 TOP5 结果如下：

```
-----TOP 5-----
[812]: 0.999442994594574
[404]: 0.0004096269840374589
[657]: 3.284541890025139e-05
[833]: 2.6112385967280716e-05
[895]: 1.8509887013351545e-05
```

4.5 查询 SDK 版本

API	**get_sdk_version**
描述	获取 SDK API 和驱动的版本号。 **注：使用该接口前必须完成模型加载和初始化运行环境。**
参数	无
返回值	sdk_version：API 和驱动版本信息。类型为字符串。

举例如下：

```
# 获取 SDK 版本信息
……
sdk_version = rknn_lite.get_sdk_version()
……
```

返回的 SDK 信息如下：

```
=============================================
RKNN VERSION:
    API: 1.7.x (xxxxxx build: 2021-10-28 14:53:41)
    DRV: 1.7.x (xxxxxxx build: 2021-11-12 20:24:57)
=============================================
```

4.6 获取设备列表

API	**list_devices**
描述	列出已连接的带 Rockchip NPU 的设备。 注：目前设备连接模式有两种：ADB 和 NTB。多设备连接时请确保他们的模式都是一样的。
参数	无。
返回值	返回 adb_devices 列表和 ntb_devices 列表，如果设备为空，则返回空列表。 例如我们的环境里插了两个 TB-RK1808 AI 计算棒时，得到的结果如下： adb_devices = [] ntb_devices = ['TB-RK1808S0', ' TB-RK1808S1']

举例如下：

```
from rknnlite.api import RKNNLite

if __name__ == '__main__':
    rknn_lite = RKNNLite()
    rknn_lite.list_devices()
    rknn_lite.release()
```

返回的设备列表信息如下（这里有一个 NTB 模式的 RK1808、RK1109 开发板和一个 ADB 模式的 RK3399Pro 开发板）：

```
**************************
all device(s) with adb mode:
VII67Y7UKQ
all device(s) with ntb mode:
c3d9b8674f4b94f6,515e9b401c060c0b
**************************
```

注：使用多设备时，需要保证它们的连接模式都是一致的，否则会引起冲突，导致初始化运行时环境失败。

4.7 查询模型可运行平台

API	**list_support_target_platform**
描述	查询给定 RKNN 模型可运行的芯片平台。
参数	rknn_model：RKNN 模型路径。如果不指定模型路径，则按类别打印 RKNN Toolkit Lite 当前支持的芯片平台。
返回值	support_target_platform: 返回模型可运行的芯片平台。如果 RKNN 模型路径为空或不存在，返回 None.

参考代码如下所示：

```
rknn_lite.list_support_target_platform(rknn_model=’mobilenet_v1.rknn’)
```

参考结果如下：

```
**************************************************
Target platforms filled in RKNN model:              ['RK1808']
Target platforms supported by this RKNN model: ['RK1806', 'RK1808', 'RK3399PRO']
**************************************************
```

5 附录

5.1 参考文档

RKNN Toolkit 模型转换相关文档请参考以下链接：

https://github.com/rockchip-linux/rknn-toolkit/tree/master/doc

5.2 问题反馈渠道

请通过 RKNN QQ 交流群，Github Issue 或瑞芯微 redmine 将问题反馈给 Rockchip NPU 团队。

RKNN QQ 交流群：1025468710

Github issue: https://github.com/rockchip-linux/rknn-toolkit/issues

Rockchip Redmine: https://redmine.rock-chips.com/

注：Redmine 账号需要通过销售或业务人员开通。如果是第三方开发板，请先找对应厂商反馈问题。