

密级状态：绝密( ) 秘密( ) 内部( ) 公开(√)

## RKNN-Toolkit 使用指南

(技术部，图形显示平台中心)

|                               |       |            |
|-------------------------------|-------|------------|
| 文件状态：<br>[ ] 正在修改<br>[√] 正式发布 | 当前版本： | v0.9.6     |
|                               | 作 者：  | 饶洪         |
|                               | 完成日期： | 2018-11-24 |
|                               | 审 核：  | 卓鸿添        |
|                               | 完成日期： | 2018-11-24 |

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 |
|--------------|-----|------------|---|-----|
| V0.1         | 杨华聪 | 2018-08-25 | 初始版本  | 卓鸿添 |
| V0.9.0       | 饶洪  | 2018-08-29 | RKNN-Toolkit v0.9.0 版本相关内容                                      | 卓鸿添 |
| V0.9.1-beta1 | 饶洪  | 2018-09-07 | 增补 v0.9.1-beta1 版本更新内容  | 卓鸿添 |
| V0.9.1-rc1   | 饶洪  | 2018-09-19 | 增补 v0.9.1-rc1 版本更新内容  | 卓鸿添 |
| V0.9.1-rc1   | 饶洪  | 2018-09-29 | 解决 ssd_mobilenet_fpn inference 阶段 core dump 问题                  | 卓鸿添 |
| V0.9.2       | 卓鸿添 | 2018-10-12 | 优化性能预估方式  | 卓鸿添 |
| V0.9.3       | 杨华聪 | 2018-10-24 | 添加连接开发板硬件说明   | 卓鸿添 |
| V0.9.4       | 杨华聪 | 2018-11-03 | 添加 docker 镜像使用说明  | 卓鸿添 |
| V0.9.5       | 饶洪  | 2018-11-19 | 添加 npy 文件作为量化校正数据的使用说明；添加 vdata 编译选项；完善 reorder_channel 参数的使用说明 | 卓鸿添 |
| V0.9.6       | 饶洪  | 2018-11-24 | 新增接口 get_run_duration 和 get_perf_detail_on_hardware 使用说明        | 卓鸿添 |

## 目 录

|        |  |    |
|--------|--|----|
| 1      | 主要功能说明 .....   | 1  |
| 2      | 系统依赖说明 .....   | 2  |
| 3      | 使用说明 .....   | 3  |
| 3.1    | 安装准备 .....   | 3  |
| 3.2    | DOCKER 镜像使用说明 .....  | 3  |
| 3.3    | RKNN TOOLKIT 的使用 .....   | 4  |
| 3.3.1  | 针对 Caffe、TensorFlow、TensorFlow Lite、ONNX、Darknet 模型的工具使用流程 ..... | 4  |
| 3.3.2  | 针对 RKNN 模型的工具使用流程 .....  | 6  |
| 3.3.3  | RKNN-Toolkit 连接开发板硬件调试 .....                                     | 7  |
| 3.4    | 示例 .....   | 7  |
| 3.5    | API 详细说明 .....   | 10 |
| 3.5.1  | RKNN 初始化及对象释放 .....  | 10 |
| 3.5.2  | 模型加载 .....   | 10 |
| 3.5.3  | RKNN 模型配置 .....  | 13 |
| 3.5.4  | 构建 RKNN 模型 .....   | 13 |
| 3.5.5  | 导出 RKNN 模型 .....   | 14 |
| 3.5.6  | 加载 RKNN 模型 .....   | 15 |
| 3.5.7  | 使用模型对输入进行推理 .....  | 15 |
| 3.5.8  | 预估模型性能 .....   | 17 |
| 3.5.9  | 联机调试获取模型运行时间 .....   | 19 |
| 3.5.10 | 联机调试获取模型每一层的耗时情况 .....   | 19 |

## 1 主要功能说明

RKNN-Toolkit 是为用户提供在 PC 上进行模型转换、模拟运行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，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 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所需时钟数及读、写带宽信息。
- 4) 模型性能调试：可以通过联机调试的方式在指定硬件平台 RK3399Pro 上运行模型，并获取模型在硬件上运行时的总时间和每一层的耗时信息。

## 2 系统依赖说明

本开发套件支持运行于 Ubuntu 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）以上   |
| Python 版本  | 3.5/3.6  |
| Python 库依赖 | 'tensorflow >= 1.11.0'<br>'numpy >= 1.14.3'<br>'scipy >= 1.1.0'<br>'Pillow >= 3.1.2'<br>'h5py >= 2.7.1'<br>'lmdb >= 0.92'<br>'networkx == 1.11'<br>'flatbuffers == 1.9',<br>'protobuf >= 3.5.2'<br>'onnx >= 1.3.0'<br>'flask >= 1.0.2' |

## 3 使用说明

### 3.1 安装准备

注：由于 TensorFlow 有 CPU 和 GPU 两个版本，在安装时，请根据 PC 是否支持 GPU 选择 TensorFlow 的版本。requirements.txt 文件提供两个版本的安装方式，使用时二选一，对于不需要的版本，在执行‘pip install’命令前请删掉。

接着执行以下命令进行安装：

```
sudo apt install virtualenv
sudo apt-get install libpython3.5-dev
sudo apt install python3-tk

virtualenv -p /usr/bin/python3 venv
source venv/bin/activate
pip install -r package/requirements.txt
pip install package/rknn_toolkit-0.9.6-cp35-cp35m-linux_x86_64.whl
```

请根据不同的 python 版本选择不同的安装包文件（位于 package/目录）：

- **Python3.5:** rknn\_toolkit-0.9.6-cp35-cp35m-linux\_x86\_64.whl
- **Python3.6:** rknn\_toolkit-0.9.6-cp36-cp36m-linux\_x86\_64.whl

### 3.2 DOCKER 镜像使用说明

在 docker 文件夹下提供了一个打包了所有开发环境的 docker 镜像，用户只需要加载该镜像即可直接上手使用 RKNN-Toolkit，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-docker-0.9.6.tar.gz
```

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

| REPOSITORY   | TAG   | IMAGE ID     | CREATED       | SIZE   |
|--------------|-------|--------------|---------------|--------|
| rknn-toolkit | 0.9.6 | a43209df54e4 | 2 minutes ago | 1.97GB |

### 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。

```
docker run -t -i --privileged -v /dev/bus/usb:/dev/bus/usb rknn-toolkit:0.9.6 /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:0.9.6 /bin/bash
```

### 4、运行 demo

```
cd /example/mobilenet_v1  
python test.py
```

## 3.3 RKNN Toolkit 的使用

RKNN-Toolkit 有两种基本用法：一是使用原始的 Caffe、TensorFlow、TensorFlow Lite、ONNX、Darknet 模型进行推理和模型性能评估；二是使用 RKNN 模型进行推理和模型性能评估。

### 3.3.1 针对 Caffe、TensorFlow、TensorFlow Lite、ONNX、Darknet 模型的工具使用流程

如果现有的模型是 Caffe、TensorFlow、TensorFlow Lite、ONNX、Darknet 模型，可以按照以下流程使用 RKNN Toolkit。

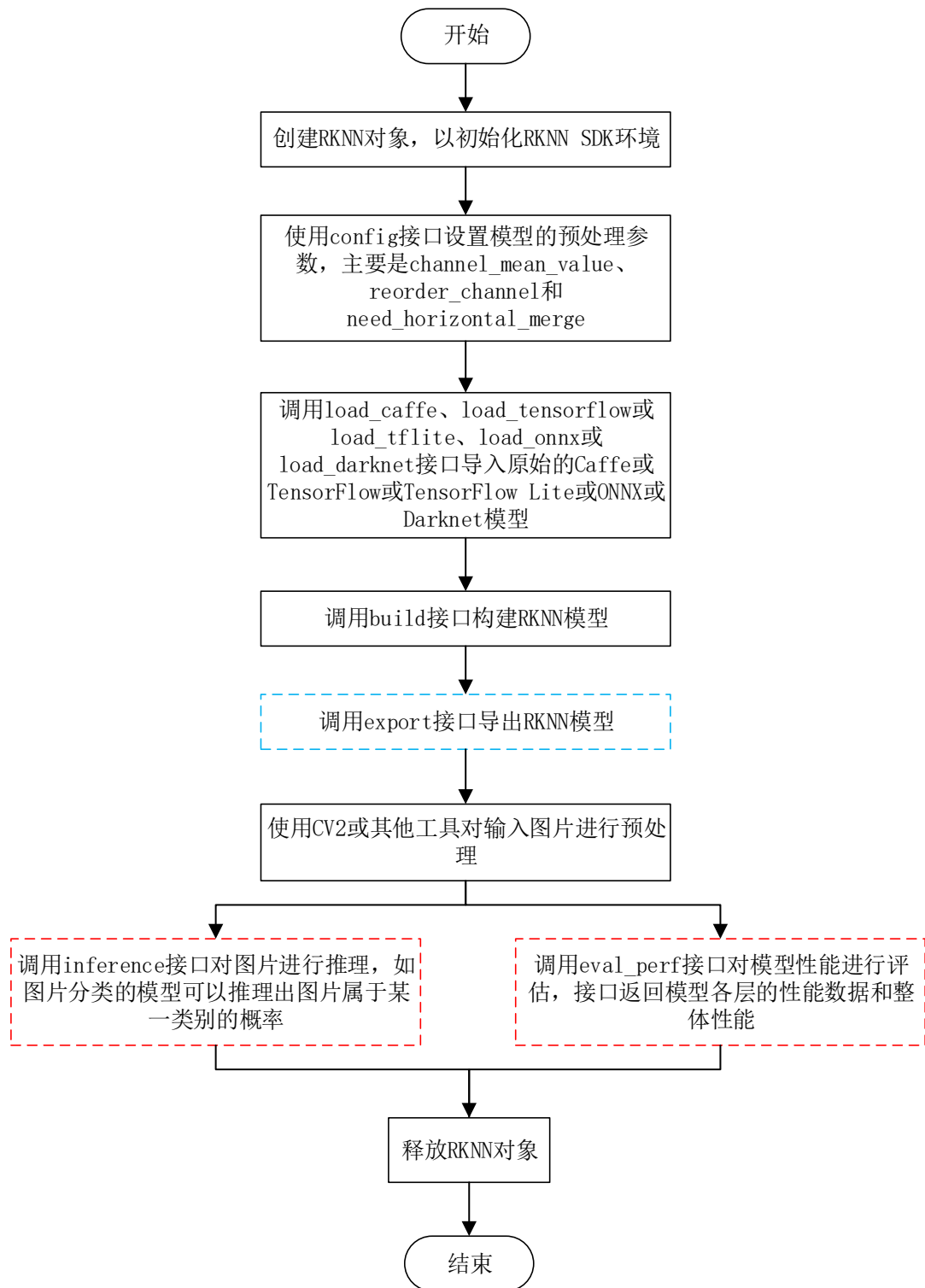


图 3-3-1-1 针对 Caffe、TensorFlow 等原始模型的工具使用流程

注：

- 1、 在使用模型对图片进行推理或性能评估时，前面的步骤除蓝色虚线框标注的导出模型步骤外都要执行。



- 2、上述步骤要按图中的先后顺序执行。
- 3、如果使用 PC 联机调试，第一步初始化 RKNN 对象时需要指定连接的硬件平台类型：  
rk3399pro。
- 4、进行联机调试时，可以通过调用 `get_run_duration` 接口获取模型总耗时，通过调用 `get_perf_detail_on_hardware` 接口获取模型每一层的耗时情况。

### 3.3.2 针对 RKNN 模型的工具使用流程

如果已有 RKNN 模型，可以根据下面的流程使用 RKNN Toolkit。

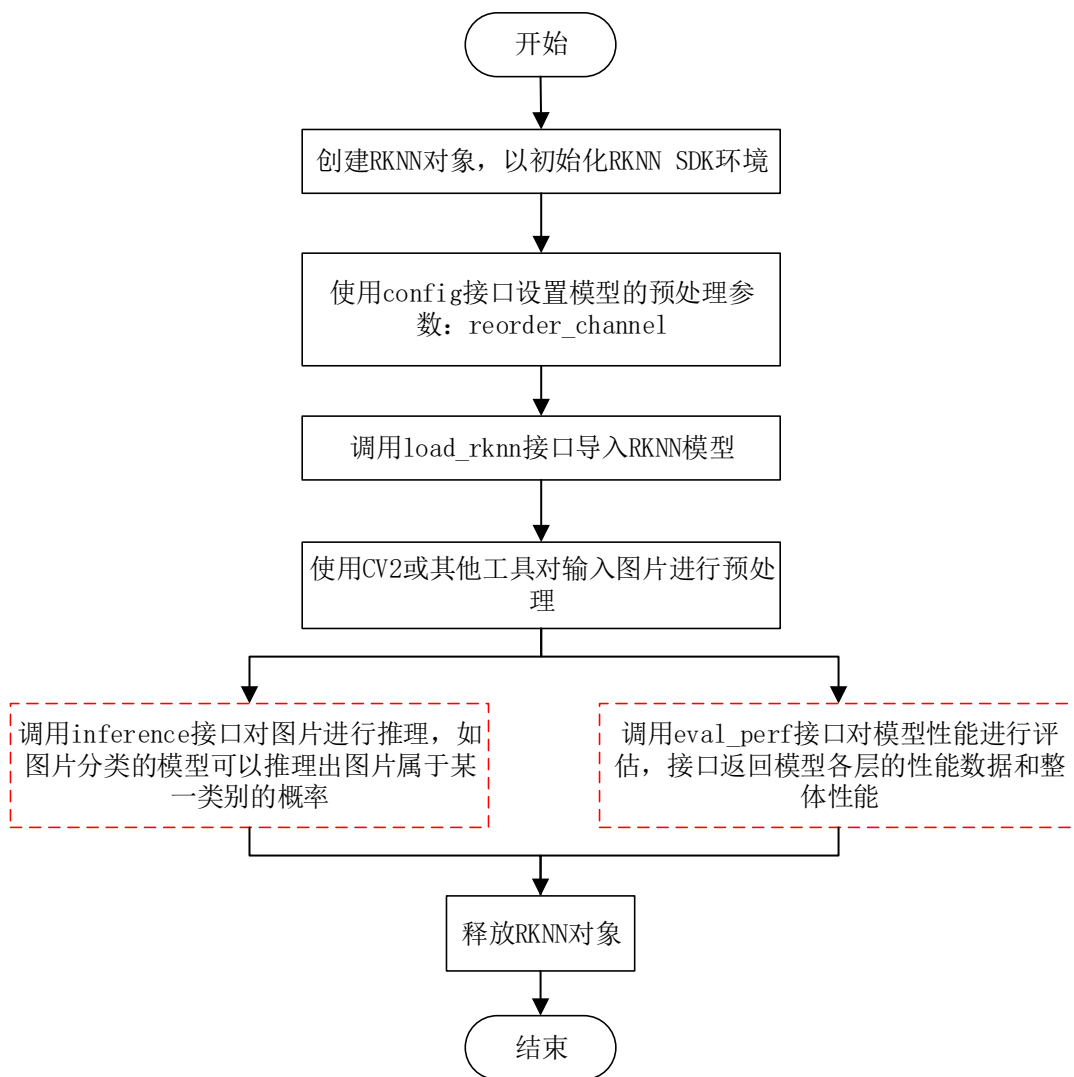


图 3-3-2-1 针对 RKNN 模型的工具使用流程

注：

- 1、在导入 RKNN 模型之前需要指定输入的 `reorder_channel` 信息。在推理时输入的图片按照

RGB 形式输入。

2、联机调试模式同 3.3.1 场景。

### 3.3.3 RKNN-Toolkit 连接开发板硬件调试

RKNN-Toolkit 能够直接通过 USB 连接开发板硬件，在硬件上运行模型推理或获取模型性能信息。步骤如下：

- 1、确保开发板的 USB OTG 连接到 PC，并且 ADB 能够正常识别到设备
- 2、修改 Example 并运行

创建 RKNN 对象时候指定 target 和 device\_id

```
rknn = RKNN(target='rk3399pro', device_id='xxxxxx')
```

## 3.4 示例

以下是加载 TensorFlow Lite 模型的示例代码（详细参见 example/mobilenet\_v1 目录）：

```
import numpy as np
import cv2
from rknn.api import RKNN

def show_outputs(outputs):
    output = outputs[0][0]
    output_sorted = sorted(output, reverse=True)
    top5_str = 'mobilenet_v1\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 = '{}: {}'.format(index[j], value)
            else:
                topi = '-1: 0.0\n'
            top5_str += topi
    print(top5_str)

def show_perfs(perfs):
```

```
#total_cycles = outputs['total_cycles']
perfs = 'perfs: {}'.format(outputs)
print(perfs)

if __name__ == '__main__':

    # Create RKNN object
    rknn = RKNN()

    # pre-process config
    print('--> config model')
    rknn.config(channel_mean_value='103.94 116.78 123.68 58.82',
reorder_channel='0 1 2')

    # Load tensorflow model
    print('--> Loading model')
    ret = rknn.load_tflite(model='./mobilenet_v1.tflite')
    if ret != 0:
        print('Load mobilenet_v1 failed!')
        exit(ret)
    print('done')

    # Build model
    print('--> Building model')
    ret = rknn.build(do_quantization=True, dataset='./dataset.txt')
    if ret != 0:
        print('Build mobilenet_v1 failed!')
        exit(ret)
    print('done')

    # Export rknn model
    print('--> Export RKNN model')
    ret = rknn.export_rknn('./mobilenet_v1.rknn')
    if ret != 0:
        print('Export mobilenet_v1.rknn failed!')
        exit(ret)
    print('done')

    # Set inputs
    img = cv2.imread('./dog_224x224.jpg')
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # Inference
    print('--> Running model')
    outputs = rknn.inference(inputs=[img])
    show_outputs(outputs)
    print('done')
    #print('inference result: ', outputs)

    # perf
```

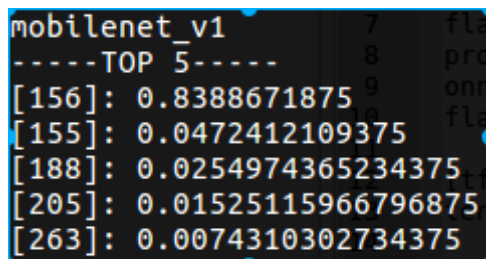
```
print('--> Begin evaluate model performance')
perf_results = rknn.eval_perf(inputs=[img])
print('done')

rknn.release()
```

其中 dataset.txt 是一个包含测试图片路径的文本文件，例如我们在 example/mobilenet\_v1 目录下有一张 dog\_224x224.jpg 的图片，那么对应的 dataset.txt 内容如下

dog\_224x224.jpg

demo 运行模型预测时输出如下结果：



```
mobilenet_v1
-----TOP 5-----
[156]: 0.8388671875
[155]: 0.0472412109375
[188]: 0.0254974365234375
[205]: 0.01525115966796875
[263]: 0.0074310302734375
```

评估模型性能时输出如下结果（仅供参考）：

```
=====
                Performances Summary
=====
ID      CYCLESREADBW(Byte) WRITEBW(Byte)   NODE
0       289789    1194602           0 convolution.relu.pooling.layer2
1       127021     3115             0 convolution.relu.pooling.layer2
2       147384     8960             0 convolution.relu.pooling.layer2
3       211488     5797             0 convolution.relu.pooling.layer2
4        75632    12239            0 convolution.relu.pooling.layer2
5       114474    11992            0 convolution.relu.pooling.layer2
6       118682    20454            0 convolution.relu.pooling.layer2
7        91877    12567            0 convolution.relu.pooling.layer2
8        71918    35279            0 convolution.relu.pooling.layer2
9        76158    39867            0 convolution.relu.pooling.layer2
10      132834    68071            0 convolution.relu.pooling.layer2
11       76158    39995            0 convolution.relu.pooling.layer2
12       81312    133913           0 convolution.relu.pooling.layer2
13       86392    136521           0 convolution.relu.pooling.layer2
14      156286    265004           0 convolution.relu.pooling.layer2
15       86392    136457           0 convolution.relu.pooling.layer2
16      156286    265004           0 convolution.relu.pooling.layer2
17       86392    136585           0 convolution.relu.pooling.layer2
18      156286    265004           0 convolution.relu.pooling.layer2
19       86392    136585           0 convolution.relu.pooling.layer2
20      156286    265004           0 convolution.relu.pooling.layer2
21       86392    136905           0 convolution.relu.pooling.layer2
22      156286    265004           0 convolution.relu.pooling.layer2
23       86392    137097           0 convolution.relu.pooling.layer2
```

```

24      130900      528734      0 convolution.relu.pooling.layer2
25      161835      506276      0 convolution.relu.pooling.layer2
26      267833      1053039      50239 convolution.relu.pooling.layer2
27          0          0          0 pooling.layer2
28      332360      1380079      0 fullyconnected.relu.layer
29        4165        128      1001 tensor.transpose
30          0          0          0 softmax.layer
Total Cycles: 3911393
Total Read BandWidth(MiB): 6.87
Total Write BandWidth(KiB): 50.04
FPS(@800MHz): 204.53
=====

```

## 3.5 API 详细说明

### 3.5.1 RKNN 初始化及对象释放

在使用 RKNN Toolkit 的所有 API 接口时,都需要先调用 RKNN()方法初始化一个 RKNN 对象,并在用完后调用该对象的 release()方法将对象释放掉。

举例如下:

```

rknn = RKNN() # for simulator
#或 rknn = RKNN(target='rk3399pro', device_id='xxxxxx') # for rk3399pro
...
rknn.release()

```

### 3.5.2 模型加载

RKNN-Toolkit 目前支持 Caffe、TensorFlow、TensorFlow Lite、ONNX、Darknet 五种模型,它们在加载时调用的接口不同,下面详细说明这五种模型的加载接口。

#### 3.5.2.1 Caffe 模型加载接口

|     |  |
|-----|--|
| API | load_caffe   |
| 功能  | 加载 caffe 模型  |
| 参数  | model: caffe 模型文件 (.prototxt 后缀文件) 所在路径。                     |
|     | proto: caffe 模型的格式 (默认值为'caffe', 可选值为'lstmc_caffe'), 由于某些模型可 |

|     |   |
|-----|---|
|     | 能根据 <b>caffe</b> 模型进行了一定的修改，需要使用特殊的格式，目前支持 <b>lstm_caffe</b> 。        |
|     | <b>blobs</b> : <b>caffe</b> 模型的二进制数据文件（ <b>.caffemodel</b> 后缀文件）所在路径。 |
| 返回值 | 0: 导入成功   |
|     | -1: 导入失败  |

举例如下：

```
# 从当前路径加载 mobilent_v2 模型
ret = rknn.load_caffe(model='./mobilenet_v2.prototxt',
                      proto='caffe',
                      blobs='./mobilenet_v2.caffemodel')
```

### 3.5.2.2 TensorFlow 模型加载接口

|     |  |
|-----|--|
| API | load_tensorflow  |
| 功能  | 加载 TensorFlow 模型   |
| 参数  | <b>tf-pb</b> : TensorFlow 模型文件（ <b>.pb</b> 后缀）所在路径。  |
|     | <b>inputs</b> : 模型输入节点，目前只支持一个输入。输入节点字符串放在列表中。   |
|     | <b>outputs</b> : 模型的输出节点，支持多个输出节点。所有输出节点放在一个列表中。   |
|     | <b>input_size_list</b> : 每个输入节点对应的图片的尺寸和通道数。如示例中的 <b>mobilenet-v1</b> 模型，其输入节点对应的输入尺寸是[224, 224, 3]。 |
| 返回值 | 0: 导入成功  |
|     | -1: 导入失败   |

举例如下：

```
# 从当前目录加载 ssd_mobilenet_v1_coco_2017_11_17 模型
ret = rknn.load_tensorflow(
    tf_pb='./ssd_mobilenet_v1_coco_2017_11_17.pb',
    inputs=['FeatureExtractor/MobilenetV1/MobilenetV1/Conv2d_0/
            BatchNorm/batchnorm/mul_1'],
    outputs=['concat', 'concat_1'],
    input_size_list=[[300, 300, 3]])
```

### 3.5.2.3 TensorFlow Lite 模型加载接口

|     |  |
|-----|--|
| API | load_tflite                                    |
| 功能  | 加载 TensorFlow Lite 模型                          |
| 参数  | model: TensorFlow Lite 模型文件 (.tflite 后缀) 所在路径。 |
| 返回值 | 0: 导入成功  |
|     | -1: 导入失败                                       |

举例如下：

```
# 从当前目录加载 mobilenet_v1 模型
ret = rknn.load_tflite(tflite_model = './mobilenet_v1.tflite')
```

### 3.5.2.4 ONNX 模型加载

|     |                                   |
|-----|-----------------------------------|
| API | load_onnx                         |
| 功能  | 加载 ONNX 模型                        |
| 参数  | model: ONNX 模型文件 (.onnx 后缀) 所在路径。 |
| 返回值 | 0: 导入成功                           |
|     | -1: 导入失败                          |

举例如下：

```
# 从当前目录加载 arcface 模型
ret = rknn.load_onnx(model = './arcface.onnx')
```

### 3.5.2.5 Darknet 模型加载接口

|     |                                     |
|-----|-------------------------------------|
| API | load_darknet                        |
| 功能  | 加载 Darknet 模型                       |
| 参数  | model: Darknet 模型文件 (.cfg 后缀) 所在路径。 |
|     | weight: 权重文件 (.weights 后缀) 所在路径     |
| 返回值 | 0: 导入成功                             |

|  |          |
|--|----------|
|  | -1: 导入失败 |
|--|----------|

举例如下：

```
# 从当前目录加载 yolov3-tiny 模型
ret = rknn.load_darknet(model = './yolov3-tiny.cfg',
                        weight='./yolov3.weights')
```

### 3.5.3 RKNN 模型配置

在模型加载之前，需要先对模型进行配置，这可以通过 config 接口完成。

|     |   |
|-----|---|
| API | config  |
| 功能  | 设置模型参数  |
| 参数  | batch_size: 每一批数据集的大小，默认值为 100。   |
|     | channel_mean_value: 均值，不同的均值对模型推理结果的准确性有影响。   |
|     | epochs: 推理或性能评估时，对同一批数据集处理的次数，默认值为 1。   |
|     | reorder_channel: 图像通道顺序。'0 1 2'代表 RGB，'2 1 0'代表 BGR。  |
|     | need_horizontal_merge: 是否需要合并 Horizontal，默认值为 False。如果模型是 inception v1/v3/v4，建议开启该选项。   |
|     | quantized_dtype: 量化类型，目前支持的量化类型有 asymmetric_quantized-u8、dynamic_fixed_point-8、dynamic_fixed_point-16，默认值为 asymmetric_quantized-u8。 |
| 返回值 | 无   |

举例如下：

```
# model config
rknn.config(channel_mean_value='103.94 116.78 123.68 58.82',
            reorder_channel='0 1 2',
            need_horizontal_merge=True)
```

### 3.5.4 构建 RKNN 模型

|     |  |
|-----|--|
| API | build  |
| 功能  | 根据导入的 Caffe、TensorFlow、TensorFlow Lite 模型，构建对应的 RKNN 模型。 |



|     |   |
|-----|---|
| 参数  | <b>do_quantization:</b> 是否对模型进行量化，值为 True 或 False。  |
|     | <b>dataset:</b> 量化校正数据的数据集。目前支持文本文件格式，用户可以把用于校正的图片或 npy 文件路径放到一个.txt 文件中。文本文件里每一行一条路径信息。如：<br><br>a.jpg<br><br>b.jpg<br><br>或<br><br>a.npy<br><br>b.npy |
|     | <b>pre_compile:</b> 预编译开关，如果设置成 True，可以减小模型大小，及模型在硬件设备上的首次启动速度，但是打开这个开关后，构建出来的模型就只能在硬件平台上运行，无法通过模拟器进行推理或性能评估。   |
| 返回值 | 0: 构建成功   |
|     | -1: 构建失败  |

举例如下：

```
# 构建 RKNN 模型，并且做量化
ret = rknn.build(do_quantization=True, dataset='./dataset.txt')
```

### 3.5.5 导出 RKNN 模型

前一个接口构建的 RKNN 模型可以保存成一个文件，之后如果想要再使用该模型进行结果预测或性能分析，直接通过模型导入接口加载模型即可，无需再用原始模型构建对应的 RKNN 模型。

|     |                                |
|-----|--------------------------------|
| API | export                         |
| 功能  | 将 RKNN 模型保存到指定文件中（.rknn 后缀）。   |
| 参数  | <b>export_path:</b> 导出模型文件的路径。 |
| 返回值 | 0: 导出成功                        |
|     | -1: 导出失败                       |

举例如下：

```
.....
# 将构建好的 RKNN 模型保存到当前路径的 mobilenet_v1.rknn 文件中
```

```
ret = rknn.export_rknn(export_path = './mobilenet_v1.rknn')
.....
```

### 3.5.6 加载 RKNN 模型

|     |                    |
|-----|--------------------|
| API | load_rknn          |
| 功能  | 加载 RKNN 模型。        |
| 参数  | path: RKNN 模型文件路径。 |
| 返回值 | 0: 加载成功            |
|     | -1: 加载失败           |

举例如下：

```
# 从当前路径加载 mobilenet_v1.rknn 模型
ret = rknn.load(path='./mobilenet_v1.rknn')
```

### 3.5.7 使用模型对输入进行推理

在使用模型进行推理前，必须先构建或加载一个 RKNN 模型。

|     |  |
|-----|--|
| API | inference  |
| 功能  | 使用模型对指定的输入进行推理，得到推理结果。如果设置 target 为 RK3399Pro，则得到的是模型在硬件平台上运行得到的推理结果；否则为模拟器上运行得到的推理结果。 |
| 参数  | inputs: 待推理的输入，如经过 cv2 处理的图片。格式是 ndarray list。   |
| 返回值 | results: 推理结果，类型是 ndarray list。  |

举例如下：

对于推理模型，如 mobilenet\_v1，代码如下（完整代码参考 example/mobilenet\_v1）：

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

输出的 TOP5 结果如下：

```
mobilenet_v1
-----TOP 5-----
[156]: 0.8388671875
[155]: 0.0472412109375
[188]: 0.0254974365234375
[205]: 0.01525115966796875
[263]: 0.0074310302734375
```

对于目标检测的模型，如 mobilenet\_ssd，代码如下(完整代码参考 example/mobilenet-ssd):

```
# 使用模型对图片进行推理，得到目标检测结果
.....
outputs = rknn.inference(inputs=[image])
.....
```

输出的结果经过后处理后得到如下结果:

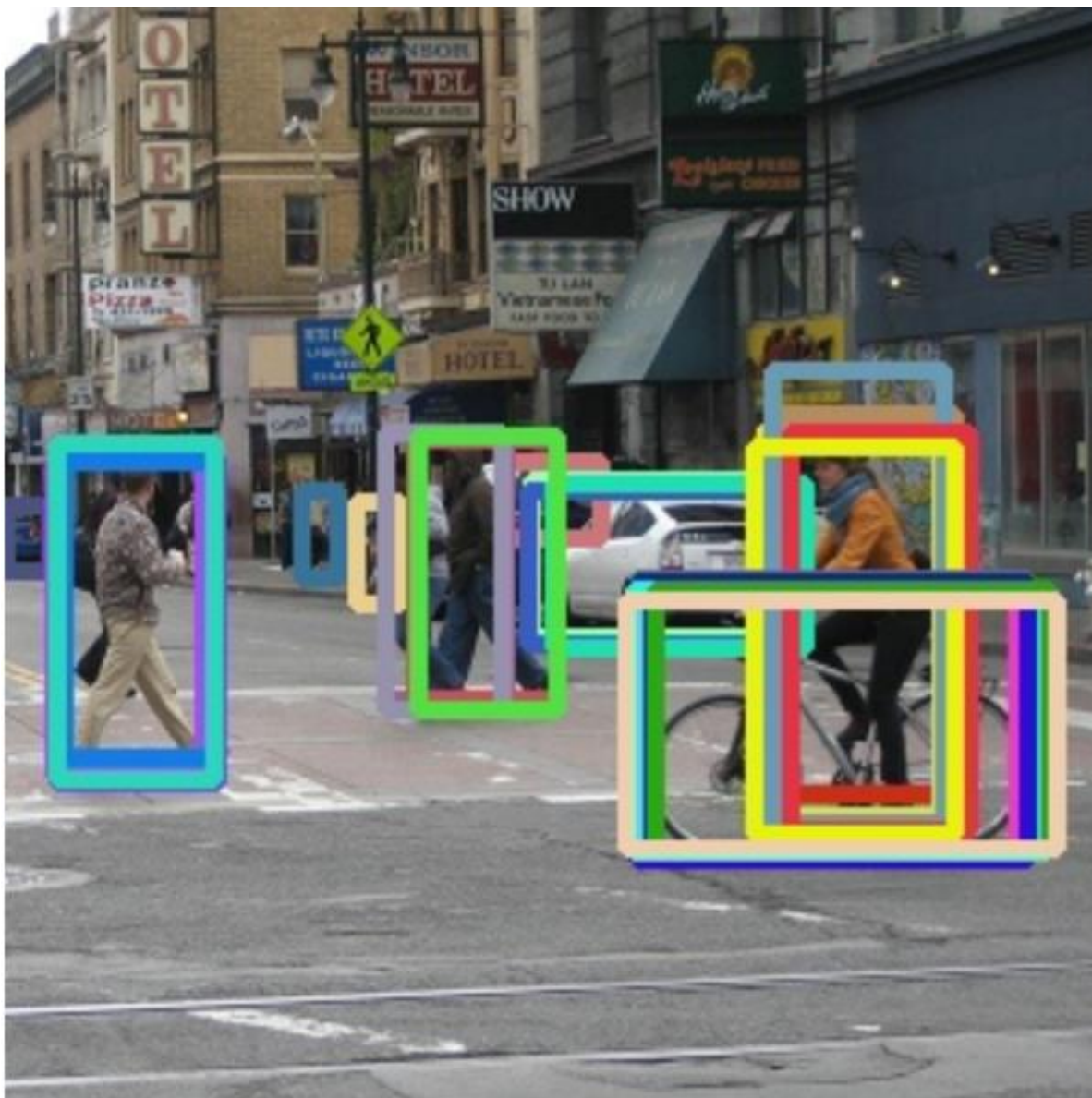


图 3-5-7-1 mobilenet-ssd inference 结果

### 3.5.8 预估模型性能

|     |  |
|-----|--|
| API | eval_perf  |
| 功能  | 预估模型的性能。无论设置 <b>target</b> 与否，得到的都是模型在模拟器上运行时的性能。  |
| 参数  | inputs: 输入数据，如经过 cv2 处理得图片。格式是 ndarray list。<br><br>is_print: 是否以规范格式输出性能评估结果。默认值为 True。 |
| 返回值 | perf_result: 性能预估结果。类型为字典。   |

举例如下：

```
# 对模型性能进行评估
.....
rknn.eval_perf(inputs=[image], is_print=True)
.....
```

如 example/mobilenet-ssd，其性能评估结果如下：

```
=====
                Performances Summary
=====
ID      CYCLESREADBW(Byte) WRITEBW(Byte)   NODE
0       101258      273740           0 convolution.relu.pooling.layer2
0       158968      16004            0 convolution.relu.pooling.layer2
1       270507      15979      864075 convolution.relu.pooling.layer2
2       347479     1438990            0 convolution.relu.pooling.layer2
3       370919       7653            0 convolution.relu.pooling.layer2
4       195422      14735            0 convolution.relu.pooling.layer2
5       264581      12823            0 convolution.relu.pooling.layer2
6       349869      22950     1428750 convolution.relu.pooling.layer2
7       281845     778880            0 convolution.relu.pooling.layer2
7       141052      19994            0 convolution.relu.pooling.layer2
8       123624      35919            0 convolution.relu.pooling.layer2
9       193265      40635            0 convolution.relu.pooling.layer2
10      231398      68711            0 convolution.relu.pooling.layer2
11      193265      40763            0 convolution.relu.pooling.layer2
12      120069     134041            0 convolution.relu.pooling.layer2
13      204465     136585            0 convolution.relu.pooling.layer2
14      232529     265132            0 convolution.relu.pooling.layer2
15      204465     136585            0 convolution.relu.pooling.layer2
16      232529     265132            0 convolution.relu.pooling.layer2
17      204465     136649            0 convolution.relu.pooling.layer2
18      232529     265132            0 convolution.relu.pooling.layer2
19      204465     136457            0 convolution.relu.pooling.layer2
20      232529     265132            0 convolution.relu.pooling.layer2
21      204465     136713            0 convolution.relu.pooling.layer2
```

|    |        |         |       |                                 |
|----|--------|---------|-------|---------------------------------|
| 22 | 232529 | 265132  | 0     | convolution.relu.pooling.layer2 |
| 23 | 127537 | 141808  | 0     | convolution.relu.pooling.layer2 |
| 24 | 36153  | 7306    | 0     | convolution.relu.pooling.layer2 |
| 25 | 92033  | 128     | 0     | convolution.relu.pooling.layer2 |
| 25 | 140338 | 242825  | 0     | convolution.relu.pooling.layer2 |
| 26 | 39040  | 128     | 98553 | tensor.transpose                |
| 27 | 5356   | 128     | 4332  | tensor.transpose                |
| 28 | 155316 | 548515  | 0     | convolution.relu.pooling.layer2 |
| 29 | 244497 | 505764  | 0     | convolution.relu.pooling.layer2 |
| 30 | 385994 | 1597021 | 0     | convolution.relu.pooling.layer2 |
| 31 | 165273 | 615061  | 0     | convolution.relu.pooling.layer2 |
| 32 | 23273  | 25437   | 0     | convolution.relu.pooling.layer2 |
| 33 | 80643  | 263993  | 0     | convolution.relu.pooling.layer2 |
| 34 | 23327  | 128     | 54600 | tensor.transpose                |
| 35 | 4666   | 128     | 2400  | tensor.transpose                |
| 36 | 16620  | 128     | 0     | convolution.relu.pooling.layer2 |
| 36 | 333148 | 1356691 | 0     | convolution.relu.pooling.layer2 |
| 37 | 71215  | 282085  | 0     | convolution.relu.pooling.layer2 |
| 38 | 7897   | 13083   | 0     | convolution.relu.pooling.layer2 |
| 39 | 19010  | 66553   | 0     | convolution.relu.pooling.layer2 |
| 40 | 8687   | 128     | 13650 | tensor.transpose                |
| 41 | 4022   | 128     | 600   | tensor.transpose                |
| 42 | 6370   | 128     | 0     | convolution.relu.pooling.layer2 |
| 42 | 86420  | 351697  | 0     | convolution.relu.pooling.layer2 |
| 43 | 37297  | 142291  | 0     | convolution.relu.pooling.layer2 |
| 44 | 4900   | 6934    | 0     | convolution.relu.pooling.layer2 |
| 45 | 10832  | 33779   | 0     | convolution.relu.pooling.layer2 |
| 46 | 5564   | 128     | 4914  | tensor.transpose                |
| 47 | 3889   | 128     | 216   | tensor.transpose                |
| 48 | 5089   | 128     | 0     | convolution.relu.pooling.layer2 |
| 48 | 86633  | 352591  | 0     | convolution.relu.pooling.layer2 |
| 49 | 36996  | 142291  | 0     | convolution.relu.pooling.layer2 |
| 50 | 4751   | 6934    | 0     | convolution.relu.pooling.layer2 |
| 51 | 6655   | 17017   | 0     | convolution.relu.pooling.layer2 |
| 52 | 4588   | 128     | 2184  | tensor.transpose                |
| 53 | 3861   | 128     | 96    | tensor.transpose                |
| 54 | 13369  | 45191   | 0     | convolution.relu.pooling.layer2 |
| 55 | 16031  | 29526   | 0     | fullyconnected.relu.layer       |
| 56 | 6959   | 14761   | 0     | fullyconnected.relu.layer       |
| 57 | 4002   | 128     | 546   | tensor.transpose                |
| 58 | 3843   | 128     | 24    | tensor.transpose                |

Total Cycles: 7860619

Total Read BandWidth(MiB): 11.20

Total Write BandWidth(MiB): 2.36

FPS(@800MHz): 101.77

=====

### 3.5.9 联机调试获取模型运行时间

|     |                                     |
|-----|-------------------------------------|
| API | get_run_duration                    |
| 功能  | 联机调试时，获取模型在指定硬件平台 RK3399Pro 上的运行时间。 |
| 参数  | 无                                   |
| 返回值 | time: 单位 us。如果获取失败，返回小于 0 的错误码。     |

举例如下：

```
# 联机调试时获取模型在指定硬件平台 RK3399Pro 上的运行时间
.....
rknn.get_run_duration ()
.....
```

### 3.5.10 联机调试获取模型每一层的耗时情况

|     |  |
|-----|--|
| API | get_perf_detail_on_hardware            |
| 功能  | 联机调试时，获取模型在指定硬件平台 RK3399Pro 上每一层的耗时情况。 |
| 参数  | 无                                      |
| 返回值 | perf_detail: 性能详情，如果获取失败，返回 None       |

举例如下：

```
# 联机调试时获取模型在指定硬件平台 RK3399Pro 上每一层的耗时情况
.....
rknn.get_perf_detail_on_hardware ()
.....
```

如 mobilenet\_v1 在 RK3399Pro 上每一层的性能如下：

| Performances Detail   |                              |              |                |        |          |
|-----------------------|------------------------------|--------------|----------------|--------|----------|
| Layer ID              | Name                         | Operation ID | Operator       | Target | Time(us) |
| 0                     | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 2567     |
| 1                     | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 345      |
| 2                     | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 283      |
| 3                     | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 381      |
| 4                     | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 205      |
| 5                     | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 255      |
| 6                     | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 233      |
| 7                     | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 291      |
| 8                     | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 207      |
| 9                     | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 303      |
| 10                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 305      |
| 11                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 303      |
| 12                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 226      |
| 13                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 227      |
| 14                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 304      |
| 15                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 228      |
| 16                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 303      |
| 17                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 250      |
| 18                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 311      |
| 19                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 227      |
| 20                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 303      |
| 21                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 204      |
| 22                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 309      |
| 23                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 227      |
| 24                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 227      |
| 25                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 431      |
| 26                    | ConvolutionReluPoolingLayer2 | 0            | CONVOLUTION    | NN     | 335      |
| 27                    | PoolingLayer2                | 0            | POOLING        | SH     | 513      |
| 28                    | FullyConnectedReluLayer      | 0            | FULLYCONNECTED | TP     | 396      |
| 29                    | TensorTranspose              | 0            | TENSOR_TRANS   | TP     | 124      |
| 30                    | SoftmaxLayer                 | 0            | SOFTMAX        | SH     | 459      |
| Total Time(us): 11282 |                              |              |                |        |          |