

密级状态：绝密( ) 秘密( ) 内部( ) 公开(√)

## RKNN-Toolkit 问题排查手册

(技术部, 图形显示平台中心)

文件状态:	当前版本:	1.3.2
[ ] 正在修改	作 者:	HPC
[√] 正式发布	完成日期:	2020-04-13
	审 核:	卓鸿添
	完成日期:	2020-04-13

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
0.9	HPC	2019-04-01	初稿	卓鸿添
1.0	HPC	2019-07-18	增加一些常见问题	卓鸿添
1.1	HPC	2019-08-22	增加深度神经网络模型设计建议	卓鸿添
1.2	HPC	2019-10-11	增加一些常见问题	卓鸿添
1.3	HPC	2019-12-25	增加一些常见问题，重新整理目录结构	卓鸿添
1.3.2	HPC	2020-04-13	增加一些常见问题	卓鸿添

## 目录

1	RKNN TOOLKIT 用法相关问题 .....	4
2	关于量化精度的问题.....	18
3	CAFFE 模型转换常见问题 .....	19
4	TENSORFLOW 模型转换常见问题 .....	23
5	PYTORCH 模型转换常见问题.....	24
6	深度神经网络模型设计建议.....	26

## 1 RKNN Toolkit 用法相关问题

### 1.1 RKNN Toolkit 安装问题

#### 1.1.1 安装 RKNN Toolkit 时出现“undefined symbol: PyFPE\_jbuf”错误

出现这个错误的原因是 Python 环境不干净，比如在两个不同的路径里都安装了 numpy。可以重新创建一个干净的 Python 环境后再试。

#### 1.1.2 在 Toybrick 上安装 RKNN Toolkit 出现“Permission Denied”错误。

原因是没有 root 权限，安装的时候需要加上“-user”选项。

### 1.2 模型配置问题

#### 1.2.1 rknn.config 函数,为什么 channel\_mean\_value 有 4 个值? 如果是 rgb 图像, 还是 4 个值吗?

rknn.config 里面的 channel-mean-value: 用来设置预处理的命令行参数。包括四个值(M0 M1 M2 S0), 前三个值为均值参数, 后面一个值为 Scale 参数。对于输入数据是三通的(Cin0, Cin1, Cin2)数据来讲, 经过预处理后, 输出的数据为(Cout0, Cout1, Cout2), 计算过程如下:

$$Cout0 = (Cin0 - M0)/S0$$

$$Cout1 = (Cin1 - M1)/S0$$

$$Cout2 = (Cin2 - M2)/S0$$

例如,如果需要将输入数据归一化到[-1, 1]之间, 则可以设置这个参数为(128 128 128 128)。如果需要将输入数据归一化到[0, 1]之间, 则可以设置这个参数为 (0 0 0 255)。

#### 1.2.2 当输入图像是单通道灰度图片时, rknn.config 接口如何设定

对于 1.3.0 之前的版本, 请参考 1.2.1 的回答, 当输入图像是单通道时, 只用到“Cout0 = (Cin0 - M0)/S0”, 因此你可以设置为(M0, 0, 0, S0), M1、M2 的值不会被用到。

从 1.3.0 开始, 单通道图片的 channel\_mean\_value 应该设成(M0, S0)。

### 1.2.3 rknn.config 函数，怎么设定 scale 参数，即把输入的 range 压缩到一定的范围。e.g. from (0-255) to (0-1)

参考 1.2.1 的回答。

### 1.2.4 当输入的 channel 大于 3 时，rknn.config 接口如何设定

对于 1.3.0 之前的版本，比如输入维度为 1x25x25x96（NHWC 格式）时，channel\_mean\_value 及 reorder\_channel 均不要设置。这种情况下默认所有通道的 mean 为 0，scale 为 1。

从 1.3.0 版本开始，如果输入维度为 1x25x25x4（NHWC）时，channel\_mean\_value 应该被设置成（M0, M1, M2, M3, S0）。如果维度大于 5，如 1x25x25x96（NHWC 格式），channel\_mean\_value 及 reorder\_channel 均不要设置，这种情况下默认所有通道的 mean 为 0，scale 为 1。

### 1.2.5 channel\_mean\_value 及 reorder\_channel 的执行先后顺序是什么？

在 RKNN 内部处理是先做了通道转换，然后做 mean 和 scale 的处理。因此，比如对于 Caffe 的模型，reorder\_channel 设置成"2 1 0"，mean 需要按照 BGR 的顺序设置。

## 1.3 模型转换问题

### 1.3.1 RKNN Toolkit 支持哪些深度学习框架？支持这些深度学习框架的哪些版本？

RKNN Toolkit 支持的深度学习框架包括 TensorFlow, TensorFlow Lite, Caffe, ONNX 和 Darknet。

它和各深度学习框架的版本对应关系如下：

RKNN Toolkit	TensorFlow	TF Lite	Caffe	ONNX	Darknet	Pytorch	MXNet
1.0.0	>=1.10.0, <=1.13.2	Schema version = 3	1.0	Release version	官方最新提交:	不支持	不支持

				1.3.0	810d7f7		
1.1.0	>=1.10.0, <=1.13.2	Schema version = 3	1.0	Release version 1.3.0	官 方 最 新提交: 810d7f7	不支持	不支持
1.2.0	>=1.10.0, <=1.13.2	Schema version = 3	1.0	Release version 1.4.1	官 方 最 新提交: 810d7f7	不支持	不支持
1.2.1	>=1.10.0, <=1.13.2	Schema version = 3	1.0	Release version 1.4.1	官 方 最 新提交: 810d7f7	不支持	不支持
1.3.0	>=1.10.0, <=1.13.2	Schema version = 3	1.0	Release version 1.4.1	官 方 最 新提交: 810d7f7	>=1.0.0, <=1.2.0	>=1.4.0, <=1.5.1
1.3.2	>=1.10.0, <=1.13.2	Schema version = 3	1.0	Release version 1.4.1	官 方 最 新提交: 810d7f7	>=1.0.0, <=1.2.0	>=1.4.0, <=1.5.1

注:

1. 依照语义版本，用某一版本 TensorFlow 写出的任何图或检查点，都可以通过相同主要版本中更高（次要或补丁）版本的 TensorFlow 来进行加载和评估，所以理论上，1.14.0 之前版本的 TensorFlow 生成的 pb 文件，RKNN Toolkit 1.0.0 及之后的版本都是支持的。关于 TensorFlow 版本兼容性的更多信息，可以参考官方资料：  
[https://www.tensorflow.org/guide/version\\_compat?hl=zh-CN](https://www.tensorflow.org/guide/version_compat?hl=zh-CN)
2. 因为 tflite 不同版本的 schema 之间是互不兼容的，所以构建 tflite 模型时使用与 RKNN Toolkit 不同版本的 schema 可能导致加载失败。目前 RKNN Toolkit 使用的 tflite schema 是基于 TensorFlow Lite 官方 GitHub master 分支上的如下提交：  
0c4f5dfea4ceb3d7c0b46fc04828420a344f7598。具体的 schema 链接如下：  
<https://github.com/tensorflow/tensorflow/commits/master/tensorflow/lite/schema/schema.hema.fbs>
3. RKNN Toolkit 所使用的 caffe protocol 有两种，一种是基于 berkeley 官方修改的

protocol，一种是包含 LSTM 层的 protocol。其中基于 berkeley 官方修改的 protocol 来自：<https://github.com/BVLC/caffe/tree/master/src/caffe/proto>，commit 值为 21d0608，RKNN Toolkit 在这个基础上新增了一些 OP。而包含 LSTM 层的 protocol 参考以下链接：<https://github.com/xmfbit/warpctc-caffe/tree/master/src/caffe/proto>，commit 值为 bd6181b。这两种 protocol 通过 load\_caffe 接口中的 proto 参数指定。

4. ONNX release version 和 opset version、IR version 之间的关系参考官网说明：

<https://github.com/microsoft/onnxruntime/blob/master/docs/Versioning.md>

ONNX release version	ONNX opset version	Supported ONNX IR version
1.3.0	8	3
1.4.1	9	3

5. Darknet 官方 Github 链接：<https://github.com/pjreddie/darknet>，RKNN Toolkit 现在的转换规则是基于 master 分支的最新提交（commit 值：810d7f7）制定的。

### 1.3.2 RKNN Toolkit 是否支持多输入的模型转换？

RKNN Toolkit 需要升级到 1.2.0 或之后的版本。

### 1.3.3 什么时候能支持 pytorch 和 mxnet 模型直接转成 rknn？

请升级到 RKNN Toolkit 1.3.0 及之后的版本。

### 1.3.4 加载模型时，numpy 模块报错：Object arrays cannot be loaded when allow\_pickle=False.

错误信息如下：

```
E Catch exception when building RKNN model! Add RKNN-Toolkit FAQ document.
T Traceback (most recent call last):
T   File "rknn/api/rknn_base.py", line 459, in rknn.api.rknn_base.RKNNBase.build
T   File "rknn/api/rknn_base.py", line 952, in rknn.api.rknn_base.RKNNBase.quantize
T   File "rknn/base/RKNNlib/app/tensorzone/workspace.py", line 231, in rknn.base.RKNNlib.app.tensorzone.workspace.Workspace.load_data
T   File "rknn/base/RKNNlib/app/tensorzone/graph.py", line 32, in rknn.base.RKNNlib.app.tensorzone.graph.Graph.load_data
T   File "rknn/base/RKNNlib/RKNNnet.py", line 379, in rknn.base.RKNNlib.RKNNnet.RKNNnet.load_data
T   File "rknn/base/RKNNlib/RKNNnet.py", line 391, in rknn.base.RKNNlib.RKNNnet.RKNNnet.load_old_data
T   File "rknn/base/RKNNlib/RKNNnet.py", line 392, in rknn.base.RKNNlib.RKNNnet.RKNNnet.load_old_data
T   File "/home/raul/work/python-env/rknn-package-tenv/lib/python3.5/site-packages/numpy/lib/npio.py", line 447, in load
T     pickle_kwargs=pickle_kwargs)
T   File "/home/raul/work/python-env/rknn-package-tenv/lib/python3.5/site-packages/numpy/lib/format.py", line 692, in read_array
T     raise ValueError("Object arrays cannot be loaded when ")
T ValueError: Object arrays cannot be loaded when allow_pickle=False
```

这个错误是由于 numpy 升级到 1.16.3 以后，加载 numpy 文件时的参数 allow\_pickle 默认值发生变化导致的（从 True 改成了 False）。解决方法有两种：一是将 numpy 版本降到 1.16.2 或更低的版本；二是将 RKNN Toolkit 更新到 1.0.0 或更新的版本。

### 1.3.5 加载模型出错时常见的排查步骤。

首先确认原始深度学习框架是否可以加载该模型并进行正确的推理；其次如果模型有 RKNN Toolkit 不支持的层（或 OP），通过打开调试日志开关，在日志中可以看到是哪一层或 OP 是 RKNN Toolkit 不支持的，这类错误日志通常包含”Try match xxx failed”或”Not match xxx”等。

## 1.4 模型量化问题

### 1.4.1 RKNN Toolkit 支持的量化方式

RKNN 支持两种量化机制：

- **Quantization-aware training**

可以参考 Tensorflow quantization-aware training

(<https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/quantize>)，这种方法要求用户有一定的 fine tune 重训练的基础。使用 RKNN Toolkit 导入量化后的模型时使用 `rknn.build(do_quantization=False)`，这时 RKNN Toolkit 将使用模型自身的量化参数，因此在量化精度上不会有损失。目前只支持导入 TensorFlow 和 TensorFlow Lite 两种深度学习框架的已量化模型，且 TensorFlow 的已量化模型在 Windows 上受限于 TensorFlow 功能而无法被加载。

- **Post training quantization**

使用这种方式时，用户导入已训练好的 float point 模型，RKNN Toolkit 根据用户提供的 dataset 进行量化。Dataset 应尽量覆盖模型可能的输入类型。官方提供的 example 为了简单一般只放一张图片，建议多放一些。

目前 RKNN Toolkit 支持 3 种量化方式：

- ✓ **asymmetric\_quantized-u8 (default)**

这是 tensorflow 支持的量化方式，也是 google 推荐的。根据”[Quantizing deep convolutional networks for efficient inference: A whitepaper](#)” 论文的描述，对于大部分网络，这种量化方式对精度的损失最小。



其计算公式如下：

$$\text{quant} = \text{round}\left(\frac{\text{float\_num}}{\text{scale}}\right) + \text{zero\_point}$$
$$\text{quant} = \text{cast\_to\_bw}$$

其中 quant 代表量化后的数，float\_num 代表 float，scale 是一个 float32 类型，zero-points 是一个 int32 类型，代表实数为 0 时对应的量化值，最后把 quant 饱和到【range\_min, range\_max】

$$\text{range\_max} = 255$$
$$\text{range\_min} = 0$$

因为当前只支持 u8

对应的反量化

$$\text{float\_num} = \text{scale}(\text{quant} - \text{zero\_point})$$

✓ dynamic\_fixed\_point-8

对于有些模型而言，dynamic\_fixed\_point-8 量化的精度比 asymmetric\_quantized-u8 高。

其公式如下

$$\text{quant} = \text{round}(\text{float\_num} * 2^{\text{fl}})$$
$$\text{quant} = \text{cast\_to\_bw}$$

其中 quant 代表量化后的数，float\_num 代表 float，fl 是左移的 bits，最后把 quant 饱和到【range\_min, range\_max】

$$\text{range\_max} = 2^{bw-1} - 1$$
$$\text{range\_min} = -(2^{bw-1} - 1)$$

若 bw=8，则范围在[-127, 127]

✓ dynamic\_fixed\_point-16

dynamic\_fixed\_point-16 的量化公式与 dynamic\_fixed\_point-8 一样，只不过 bw=16。对于 rk3399pro/rk1808 而言，NPU 里面都带有 300Gops int16 的计算单元，对于某些量化到 8 位精度损失较大的网络，可以考虑使用此量化方式。

#### 1.4.2 转换模型时如果 do\_quantization 为 False，是否也会进行量化，量化精度是什么？（因为转换后模型体积小了接近一半）

分两种情况，当导入的模型是量化的模型时，do\_quantization=False 会使用该模型里面的量化参数，具体请参考 1.9 的回答。当导入的模型是非量化模型时，do\_quantization=False 不会做量化的操作，但是会把权重从 float32 转成 float16，这块不会有精度损失。

### 1.4.3 构建 RKNN 模型（调用 build 接口）时，设置 do\_quantization=False 能构建成功，但是设成 True，构建失败

错误信息如下：

```
Traceback (most recent call last):
  File "test.py", line 52, in <module>
    ret = rknn.build(do_quantization=True, dataset='./dataset.txt')
  File "/home/ljq/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/rknn/api/rknn.py", line 162, in build
    ret = self.rknn_base.build(do_quantization=do_quantization, dataset=dataset, pack_vdata=pre_compile)
  File "/home/ljq/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/rknn/base/rknnlib/app/tensorzone/tensorprovider.py", line 154, in get_output
    return self.queue_task.queue.dequeue_many(batch_size)
  File "/home/ljq/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/tensorflow/python/ops/data_flow_ops.py", line 478, in dequeue_many
    self._queue_ref, n=n, component_types=self._dtypes, name=name)
  File "/home/ljq/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/tensorflow/python/ops/gen_data_flow_ops.py", line 3487, in queue_dequeue_many_v2
    component_types=component_types, timeout_ms=timeout_ms, name=name)
  File "/home/ljq/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/tensorflow/python/framework/op_def_library.py", line 787, in _apply_op_helper
    op_def=op_def)
  File "/home/ljq/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/tensorflow/python/util/deprecation.py", line 488, in new_func
    return func(*args, **kwargs)
  File "/home/ljq/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/tensorflow/python/framework/ops.py", line 3274, in create_op
    op_def=op_def)
  File "/home/ljq/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/tensorflow/python/framework/ops.py", line 1770, in __init__
    self.traceback = tf_stack.extract_stack()
tensorflow.python.framework.errors_implOutOfRangeError (see above for traceback): FIFOQueue '_0_fifo_queue' is closed and has insufficient elements (requested 1, current size 0)
[[node TfFifoQueueDequeueMany (defined at /home/ljq/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/rknn/base/rknnlib/app/tensorzone/tensorprovider.py:154) = QueueDequeueManyV2[ _class=["loc:@input_116/cond/Switch_2"], component_types=[DT_FLOAT, DT_INT32], timeout_ms=-1, _device="/job:localhost/replica:0/task:0/device:CPU:0"]](fifo_queue, fifo_queue_DequeueMany/n)]]
Build onnx failed!
```

这是由于 dataset.txt 里没有数据，或者数据是 RKNN Toolkit 不支持的格式，建议用 jpg 或 npy。

### 1.4.4 RKNN 量化过程中的 dataset 起什么作用？为什么量化需要和 dataset 关联？

RKNN 量化过程中，需要找到合适的量化参数，比如 scale 或者 zero point，这些量化参数的选择需要根据实际的输入做 inference 来确定。

### 1.4.5 升级到 RKNN Toolkit 1.2.0 后 dataset.txt 里明明填了 200 张图片，量化校正却很快就完成，内存占用也很低，得到的量化模型精度下降很多，这些图片是否都被用于量化校正？

RKNN Toolkit 1.2.0 对 config 接口中的 batch\_size 默认值做了调整，在该版本下如果想要使用多张图片进行量化，这个参数的值就要设成相应的图片数量。如果这个值设得过大，可能因系统内存被耗尽而导致程序异常。此时，需要将 RKNN Toolkit 升级到 1.2.1 或之后的

版本。在 1.2.1 版本，batch\_size 默认值恢复成 100。量化校正时使用到的图片为 batch\_size 和 epochs 的乘积。假如 dataset.txt 中填写了 200 张图片，那么 batch\_size 设成 100，epochs 设成 2，或者 batch\_size 设成 200，epochs 设成 1，这两百张图片都能用与量化校正，但前者的内存使用峰值将小于后者。如果只想用其中的 100 张，那可以把 batch\_size 设成 100，epochs 设成 1。

**1.4.6 量化校正时，dataset 里填的是 npy 文件，文件中 numpy 数组的 shape 是 (4, 640, 480)，但量化校正时，log 提示 shape 为 (640, 480, 480)，导致量化校正失败。**

在使用 numpy 文件作为校正数据时，数据需要按 NHWC 的格式排列。

**1.4.7 量化校正时，dataset 里的图片是否需要和模型输入的尺寸一样大？**

不需要。RKNN Toolkit 会对自动对这些图片进行缩放处理。但是由于缩放会使图片信息发生改变，可能会对精度产生一些影响，所以最好使用尺寸相近的图片。

## 1.5 模型推理问题

**1.5.1 rknn.Inference()接口调用多次后出错或者卡住**

如果出错信息类似：

```
Traceback (most recent call last):
  File "rknn_pic_to_emb.py", line 63, in <module>
  File "rknn_pic_to_emb.py", line 42, in get_embedding
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/site-packages/rknn/
api/rknn.py", line 234, in inference
  File "rknn/api/redirect_stdout.py", line 76, in rknn.api.redirect_stdout.redir
ect_stdout.redirect_stdout.func_wrapper
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/contextlib.py", lin
e 81, in __enter__
  File "rknn/api/redirect_stdout.py", line 48, in stdout_redirector
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/tempfile.py", line
622, in TemporaryFile
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/tempfile.py", line
262, in _mkstemp_inner
OSError: [Errno 24] Too many open files: '/tmp/tmp5yw4m_22'
Traceback (most recent call last):
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/weakref.py", line 6
24, in _exitfunc
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/weakref.py", line 5
48, in _call_
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/tempfile.py", line
799, in _cleanup
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/shutil.py", line 48
2, in rmtree
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/shutil.py", line 48
0, in rmtree
OSError: [Errno 24] Too many open files: '/tmp/tmp_d63w4jh'
```

请将 RKNN Toolkit 升级到 0.9.9 及以后。

### 1.5.2 rknn.inference()推理速度慢的问题

这个问题有两方面的现象：

- 1) 进行前向推理测试速度慢，经测试 mobilenet-ssd 有的图片耗时在 0.5 秒以上
- 2) 模型 rknn.inference 的时间和 rknn.eval\_perf()时间相差较大，比如

理论计算时间(单图)	1.79ms	8.23ms	7.485ms	30.55ms
实际计算时间(单图)	21.37ms	39.82ms	33.12ms	76.13ms

实测帧率慢的问题，有两方面的原因：

1. 使用 pc + adb 的方式传图片比较慢，这种对高帧率的网络影响很大比如理论 1.79ms 的网络。
2. 在 0.9.8 及之前版本的实现中，推理包含了一些额外的时间，0.9.9 及之后的版本已经做了优化。

对于更真实的实测帧率，可以直接在板子上使用 c/c++ api 进行测试。

### 1.5.3 RKNN Toolkit 0.9.9 版本第一次 inference 很慢

RKNN Toolkit 0.9.9 版本将加载模型推迟到第一次 inference 时，因此第一次 inference 比



较慢，这个问题在 1.0.0 及之后的版本中已经解决。

#### 1.5.4 YOLO 前向测试返回的 **outputs** 为[**array1** , **array2**], 长度分别为[**10140** , **40560**], 返回值含义是什么

`rknn.inference` 返回的 `outputs` 是一个 `numpy ndarray` 的列表，每个模型输出数据大小个数都不一样，用户需要自行查找模型的对应输出和解析规则。

#### 1.5.5 `rknn.inference()`是否支持同时输入多张图片？或者说支持 **batch** 输入。

RKNN Toolkit 需要升级到 1.2.0 或之后的版本，并且需要在构建 RKNN 模型时就指定输入图片的数量，详细用法参考《Rockchip\_User\_Guide\_RKNN\_Toolkit\_CN》中关于 `build` 接口的说明。

另外，当 `rknn_batch_size` 大于 1（如等于 4 时），python 里推理的调用要由：

```
outputs = rknn.inference(inputs=[img])
```

修改为：

```
img = np.concatenate((img, img, img, img), axis=0)
```

```
outputs = rknn.inference(inputs=[img])
```

#### 1.5.6 同样的图片，为什么 `rknn_batch_size=1` 的模型与 `rknn_batch_size=4` 的模型结果不一样？

因为有些 OP 在 `batch size` 为 4 时走的指令经过了优化，它的运算精度会更高，所以导致有些模型结果看起来有些不同。

#### 1.5.7 批量推理得到的结果，如何与输入对应？

如果输入图像（`shape` 是 [224, 224, 3]）对应的输出是 `output`（`shape` 是 [1, 1001]），那当 `batch_size` 为 2 时，它的输入是两张图像做 `concat` 的结果，`shape` 是 [448, 224, 3]，对应的输出是 [2, 1001]。

### 1.5.8 RKNN Toolkit python 接口推理得到的结果和 RKNN API C 接口推理得到的结果有细微差别是为什么？

有可能是 cv2 解码和 jpeg 解码不一致导致。python 端可以使用 numpy 的 tofile() 方法将图片数据导出成二进制，C API 直接读取二进制的图像数据作为输入，这样保证输入一样的情况下进行对比。

### 1.5.9 模型的输入是单通道的，用 opencv 读图后作为输入进行推理，得到的结果是错的？

用 opencv 默认的 imread 接口读取图片时，得到的是一个 3 通道的图，所以结果不正确。使用以下方法读取结果，就能得到正确的结果：

```
img = cv2.imread('32x32_gray.jpg', 0)
```

### 1.5.10 使用 1.1.0 版本推理时，有些模型 softmax 得到的结果和不为 1？

请更新到 1.2.1 或之后的版本。

## 1.6 预编译问题

### 1.6.1 在开发板上用 RKNN Toolkit 转换模型时开启 pre\_compile=true 出错

Arm64 版本的 RKNN Toolkit 暂时还不支持 pre\_compile，如果需要打开 pre\_compile，建议在开发机上用 x86 版本 RKNN Toolkit 进行转换。

### 1.6.2 RKNN Toolkit 0.9.9 版本生成的 pre-compile 模型在 0.9.6 版本驱动上无法运行？

RKNN Toolkit 1.0.0 版本生成的预编译模型不能在 NPU 驱动版本小于 0.9.6 的设备上运行；旧版本 RKNN Toolkit (0.9.9 及之前的版本) 生成的预编译模型不能在安装了新版本 NPU 驱动的设备上运行。驱动版本号可以通过 get\_sdk\_version 接口查询。

### 1.6.3 调用 `rknn.build()` 时如果设置 `pre_compile=True` 报错，不设置可以转换成功。

错误信息如下：

```
E Catch exception when building RKNN model!
T Traceback (most recent call last):
T   File "rknn/api/rknn_base.py", line 515, in rknn.api.rknn_base.RKNNBase.build
T   File "rknn/api/rknn_base.py", line 439, in rknn.api.rknn_base.RKNNBase._build
T   File "rknn/base/ovxconfiggenerator.py", line 187, in
rknn.base.ovxconfiggenerator.generate_vx_config_from_files
T   File "rknn/base/RKNNlib/app/exporter/ovxlib_case/casegenerator.py", line 380, in
rknn.base.RKNNlib.app.exporter.ovxlib_case.casegenerator.CaseGenerator.generate
T   File "rknn/base/RKNNlib/app/exporter/ovxlib_case/casegenerator.py", line 352, in
rknn.base.RKNNlib.app.exporter.ovxlib_case.casegenerator.CaseGenerator._gen_special_case
T   File "rknn/base/RKNNlib/app/exporter/ovxlib_case/casegenerator.py", line 330, in
rknn.base.RKNNlib.app.exporter.ovxlib_case.casegenerator.CaseGenerator._gen_nb_file
T AttributeError: 'CaseGenerator' object has no attribute 'nbg_graph_file_path'
```

请确认：

- 1) 系统装有 gcc 编译工具链
- 2) 模型的名称只包含“字母”、“数字”、“\_”，特别注意有些复制过来的模型名称会自动加上“(1)”，这时会失败。也可以升级到 1.3.0 或之后的版本，由 RKNN Toolkit 自动处理这些特殊字符。

## 1.7 初始化运行环境问题

### 1.7.1 联机调试时，`rknn_init` 失败，出现 `RKNN_ERR_MODEL_INVALID` 的错误

错误信息如下：

```
E RKNNAPI: rknn_init, msg_load_ack fail, ack = 1, expect 0!  
E Catch exception when init runtime!  
T Traceback (most recent call last):  
T   File "rknn/api/rknn_base.py", line 646, in rknn.api.rknn_base.RKNNBase.init_runtime  
T   File "rknn/api/rknn_runtime.py", line 378, in  
rknn.api.rknn_runtime.RKNNRuntime.build_graph  
T Exception: RKNN init failed. error code: RKNN_ERR_MODEL_INVALID
```

出现该错误一般有两种情况：

- 1) 在生成 rknn 模型时，使用了 `pre_compile=True` 的选项，而不同版本的 RKNN Toolkit 和驱动是有对应关系的，建议将 RKNN Toolkit 和板子的固件都升级到最新的版本。
- 2) 在生成 rknn 模型时，没有使用 `pre_compile=True` 的选项，这时一般是系统固件太老了，建议将板子的固件升级到最新的版本。

### 1.7.2 联 机 调 试 时 ， rknn\_init 失 败 ， 出 现 RKNN\_ERR\_DEVICE\_UNAVAILABLE 的错误

错误信息如下：

```
E RKNNAPI: rknn_init, driver open fail! ret = -9!  
E Catch exception when init runtime!  
T Traceback (most recent call last):  
T File "rknn/api/rknn_base.py", line 617, in rknn.api.rknn_base.RKNNBase.init_runtime  
T File "rknn/api/rknn_runtime.py", line 378, in rknn.api.rknn_runtime.RKNNRuntime.build_graph  
T Exception: RKNN init failed. error code: RKNN_ERR_DEVICE_UNAVAILABLE
```

这个问题的原因比较复杂，请按以下方式排查：

- 1) 确保 RKNN Toolkit 及板子的系统固件都已经升级到最新版本。各版本 RKNN Toolkit 和系统固件各组件之间的对应关系如下：

RKNN Toolkit	rknn_server	NPU 驱动	librknn_runtime
1.0.0	0.9.6/0.9.7	6.3.3:203718	0.9.8/0.9.9
1.1.0	0.9.8	6.3.3:203718	1.0.0



1.2.0	0.9.9	6.4.0.213404	1.1.0
1.2.1	1.2.0	6.4.0.213404	1.2.0
1.3.0	1.3.0	6.4.0, 227915	1.3.0
1.3.2	1.3.2	6.4.0, 227915	1.3.2

在 RK1808 上，这些组件的版本查询方法如下：

```
# execute these commands on RK1808

dmesg | grep -i galcore      # 查询 NPU 驱动版本

strings /usr/bin/rknn_server | grep build    # 查询 rknn_server 版本

strings /usr/lib/librknn_runtime.so | grep version    # 查询 librknn_runtime 版本
```

也可以通过 `get_sdk_version` 接口查询版本信息，其中的 DRV 版本对应的就是 `rknn_server` 的版本。

- 2) 确保 `adb devices` 能看到设备，并且 `init_runtime()` 的 `target` 和 `device_id` 设置正确
- 3) 如果使用 RKNN Toolkit 1.1.0 及以上版本，请确保 `rknn.list_devices()` 能看到设备，并且 `init_runtime()` 的 `target` 和 `device_id` 设置正确
- 4) 如果使用的是计算棒或者 RK1808 EVB 版的 NTB 模式，请确保已经调用 `update_rk1808_usb_rule.sh`（在 RKNN Toolkit 发布包中）来获得 USB 设备的读写权限。  
(可以先使用 `sudo` 运行该 `python` 脚本看是否还出现问题)
- 5) 如果是直接在 RK3399/RK3399Pro 上运行 AARCH64 版本的 RKNN Toolkit，请确保系统固件都已经升级到最新版本。

## 1.8 日志问题

### 1.8.1 在使用 RKNN Toolkit 时，如果脚本里也使用 `logging` 模块输出日志，会报错退出。

请升级到 1.2.1 及之后的版本。

## 1.8.2 升级到 RKNN Toolkit 1.2.0 后，在 Windows 系统中加载模型时直接退出，且没有任何日志信息，现象如下：

```
_np Quint8 = np.dtype(["Quint8", np.int8, 1])
E:\python3.6\lib\site-packages\tensorflow\python\framework\dtypes.py:527: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a fu
)) / (1,)type.
_np Quint8 = np.dtype(["Quint8", np.int8, 1])
E:\python3.6\lib\site-packages\tensorflow\python\framework\dtypes.py:528: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a fu
)) / (1,)type.
_np Quint16 = np.dtype(["Quint16", np.int16, 1])
E:\python3.6\lib\site-packages\tensorflow\python\framework\dtypes.py:529: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a fu
)) / (1,)type.
_np Quint16 = np.dtype(["Quint16", np.int16, 1])
E:\python3.6\lib\site-packages\tensorflow\python\framework\dtypes.py:530: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a fu
)) / (1,)type.
_np Quint32 = np.dtype(["Quint32", np.int32, 1])
E:\python3.6\lib\site-packages\tensorflow\python\framework\dtypes.py:535: FutureWarning: Passing (type, 1) or 'ltype' as a synonym of type is deprecated; in a fu
)) / (1,)type.
_np Resource = np.dtype(["Resource", np.ubyte, 1])

WARNING: The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
* https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
* https://github.com/tensorflow/addons
If you depend on functionality not listed there, please file an issue.

E:\python3.6\lib\site-packages\onnx_tf\common\_init_.py:87: UserWarning: FrontendHandler.get_outputs_names is deprecated. It will be removed in future release.
warnings.warn(message)
D save dump info to: ./build.log
--> Loading model
E:\python3.6\lib\site-packages\onnx_tf\common\_init_.py:87: UserWarning: FrontendHandler.get_outputs_names is deprecated. It will be removed in future release.
warnings.warn(message)
```

在 Windows 系统中使用 RKNN Toolkit 1.2.0 时需要新建环境变量“PYTHONLEGACYWINDOWSSTDIO”，并设置值为 1。也可以更新到 RKNN Toolkit 1.2.1 及之后的版本，这些版本不需要手动设置该环境变量。

## 2 关于量化精度的问题

### 2.1 量化后精度与原来模型对不上，如何调试？

#### ➤ 首先确保 float 类型的精度和原始平台测试结果相近：

- (1) 使用 RKNN Toolkit 导入量化后的模型时使 `rknn.build(do_quantization=False)`;
- (2) 参考 1.1 设置 `channel_mean_value` 参数，确保其和训练模型时使用的参数相同；
- (3) 务必确保测试时输入图像通道顺序为 R,G,B。（不论训练时使用的图像通道顺序如何，使用 RKNN 做测试时都按 R,G,B 输入）
- (4) 在 `rknn.config` 函数里面设置 `reorder_channel` 参数，'0 1 2'代表 RGB，'2 1 0'代表 BGR，务必和训练时候图像通道顺序一致

#### ➤ 量化后的精度测试

- (1) 使用多张图进行量化，确保量化精度稳定。

在 `rknn.config` 中设置 `batch_size` 参数（建议设置 `batch_size = 200`）并且在 `dataset.txt` 中给出大于 200 张图像路径用于量化。

如果显存不够，可以设置 `batch_size=1`, `epochs=200` 代替 `batch_size = 200` 进行量化

- (2) 精度对比，尽量用较大数据集进行测试。分类网络比较 top-1, top-5 精度，检测网络

比较数据集的 mAP, Recall 等。

(3) 如果是人脸识别, 不能使用 float 模型的结果和量化模型的结果进行特征比较。比如 A1、A2 两张图片, 用 float 模型跑出来的结果 A1fp、A2fp, 用量化模型跑的结果 A1u8、A2u8, 这时可以算 A1fp 和 A2fp 的欧式距离来计算两个图片相似度, 也可以算 A1u8 和 A2u8 的欧式距离来计算两个图片相似度, 但是不能算 A1fp 和 A2u8 的欧式距离来计算两个图片相似度。

## 2.2 如何 dump 网络每层输出?

目前 PC 模拟器可以支持 dump 出每一层网络的数据, 在执行 inference 的脚本前需要设置一个环境变量, 命令如下:

```
export NN_LAYER_DUMP=1
```

```
python xxx.py
```

执行完之后, 会在当前目录生成每层网络的 tensor 数据文件, 这样可以和别的框架的数据进行逐层比对。

注意: 有些层会被合并, 比如 conv+bn+scale 会合并成一个 conv, 这时候就需要和原来模型的 scale 层的输出进行对比。

## 2.3 RKNN Toolkit 目前支持哪些框架的已量化模型?

RKNN Toolkit 目前支持 TensorFlow 和 TensorFlow Lite 这两种框架的已量化模型。

## 3 Caffe 模型转换常见问题

### 3.1 转换模型时, 出现 “Deprecated caffe input usage” 错误

该模型是旧版的 caffe 的模式, 需要修改输入层成如下类似格式。

```
layer {  
  name: "data"  
  type: "Input"  
  top: "data"
```

```
input_param {  
  shape {  
    dim: 1  
    dim: 3  
    dim: 224  
    dim: 224  
  }  
}
```

### 3.2 转换模型时，出现 “Message type "caffe.PoolingParameter" has no field named "round\_mode"” 错误

Pool 层的 round\_mode 字段不能识别，可以改成 ceil\_model，比如原来是 round\_mode: CEIL，那么可以删掉（默认 ceil\_mode 为 True）或者改成 ceil\_mode: True。

### 3.3 在进行 caffe 或者其他模型转换时，出现 “ValueError(“'%s' is not a valid scope name" % name)” 的错误

详细的错误信息类似如下：

```
T      raise ValueError(“'%s' is not a valid scope name" % name)  
T  ValueError: '_plus0_17' is not a valid scope name
```

对于这种情况是因为：layer name '\_plusxxx' 用下划线开头不合法，要遵循 tensorflow 的命名规则：

[A-Za-z0-9.][A-Za-z0-9\_.\-/\*]\* (for scopes at the root)

[A-Za-z0-9\_.\-/\*]\* (for other scopes)

### 3.4 Caffe 版本的 SSD 转换失败，出现 “Invalid tensor id(1), tensor(@mbox\_conf\_flatten\_188:out0)” 的错误

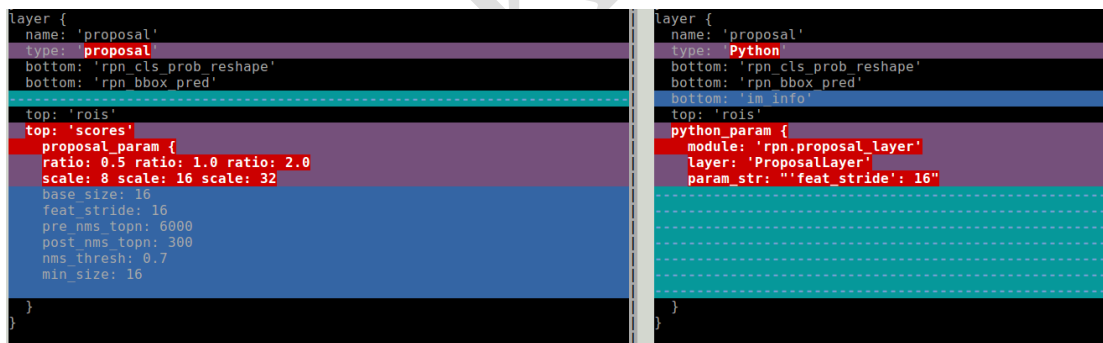
不支持 detectionoutput 这个 layer，可以删掉，改成在 CPU 做。

### 3.5 Caffe 版本的 SSD 模型去掉 detectionoutput 后应该有 3 个 output tensor，但 RKNN 推理时实际只返回两个 tensor

这个缺失的 tensor 是先验框，它在训练、推理阶段都一样，且对所有输入都一样，为了提高性能，RKNN Toolkit 在模型中将相关的层优化掉了。而要得到该先验框 tensor，可以在训练阶段将先验框的 tensor 保存下，或者用 Caffe 先 inference 一次。

### 3.6 py-faster-rcnn 模型转换时出现 “ValueError: Invalid tensor id(1), tensor(@rpn\_bbox\_pred\_18:out0)” 错误

与官方相比需要修改 prototxt 中的 'proposal' 层如下：



The image shows a side-by-side comparison of two Caffe prototxt files for the 'proposal' layer. The left side shows the original configuration from py-faster-rcnn, which includes a 'proposal\_param' block with various parameters like ratio, scale, base\_size, feat\_stride, and nms settings. The right side shows the modified configuration for RKNN Toolkit, where the 'proposal\_param' block is replaced by a 'python\_param' block that references a custom 'PropoalLayer' module and its parameters.

```
layer {
  name: 'proposal'
  type: 'proposal'
  bottom: 'rpn_cls_prob_reshape'
  bottom: 'rpn_bbox_pred'
  top: 'rois'
  top: 'scores'
  proposal_param {
    ratio: 0.5 ratio: 1.0 ratio: 2.0
    scale: 8 scale: 16 scale: 32
    base_size: 16
    feat_stride: 16
    pre_nms_topn: 6000
    post_nms_topn: 300
    nms_thresh: 0.7
    min_size: 16
  }
}
```

```
layer {
  name: 'proposal'
  type: 'Python'
  bottom: 'rpn_cls_prob_reshape'
  bottom: 'rpn_bbox_pred'
  bottom: 'im_info'
  top: 'rois'
  python_param {
    module: 'rpn.proposal_layer'
    layer: 'PropoalLayer'
    param_str: "'feat_stride': 16"
  }
}
```

```
layer {
  name: 'proposal'
  type: 'proposal'
  bottom: 'rpn_cls_prob_reshape'
  bottom: 'rpn_bbox_pred'
  top: 'rois'
  top: 'scores'
  proposal_param {
```

```
ratio: 0.5 ratio: 1.0 ratio: 2.0

scale: 8 scale: 16 scale: 32

base_size: 16

feat_stride: 16

pre_nms_topn: 6000

post_nms_topn: 300

nms_thresh: 0.7

min_size: 16

}

}
```

### 3.7 Caffe 模型转换时出现 “E Not supported caffe net model version(v0 layer or v1 layer)” 错误

详细错误如下：

```
E Not supported caffe net model version(v0 layer or v1 layer). Please use the newest model version.
E Catch exception when loading caffe model: ../model/vgg16.prototxt!
T Traceback (most recent call last):
  File "rknn/api/rknn_base.py", line 288, in rknn.api.rknn_base.RKNNBase.load_caffe
  File "rknn/base/RKNNlib/converter/caffe_loader.py", line 997, in rknn.base.RKNNlib.converter.caffe_loader.CaffeLoader.load_blobs
  File "rknn/base/RKNNlib/converter/caffe_loader.py", line 893, in rknn.base.RKNNlib.converter.caffe_loader.CaffeLoader.parse_blobs
  File "rknn/base/RKNNlib/RKNNLog.py", line 105, in rknn.base.RKNNlib.RKNNLog.e valid scope name
T ValueError: Not supported caffe net model version(v0 layer or v1 layer). Please use the newest model version.
Load vgg16 failed!
```

主要是由于 caffe 模型的版本太旧，需要更新，更新方法如下（以 VGG16 为例）：

- 1) 从 <https://github.com/BVLC/caffe.git> 下载 Caffe 源码
- 2) 编译 Caffe
- 3) 将模型转为新的格式

```
./build_release/tools/upgrade_net_proto_text vgg16_old/vgg16.prototxt vgg16_new/vgg16.prototxt

./build_release/tools/upgrade_net_proto_binary vgg16_old/vgg16.caffemodel vgg16_new/vgg16.caffemodel
```

## 4 Tensorflow 模型转换常见问题

### 4.1 转换 google 官方的 ssd\_mobilenet\_v2 模型出现 “AttributeError: ‘NoneType’ object has no attribute op” 错误

一个可能的原因是 input 节点没有取对，可以改成如下：

```
rknn.load_tensorflow(tf_pb='./ssd_mobilenet_v2_coco_2018_03_29/frozen_inference_graph.pb',  
                     inputs=['FeatureExtractor/MobilenetV2/MobilenetV2/input'],  
                     outputs=['concat', 'concat_1'],  
                     input_size_list=[[INPUT_SIZE, INPUT_SIZE, 3]])
```

### 4.2 转换 SSD\_Resnet50\_v1\_FPN\_640x640 模型出现 “Cannot convert value dtype ([‘resource’, ‘u1’]) to a Tensorflow Dtype” 错误。

需更新 RKNN Toolkit 到 0.9.8 及以后版本。

### 4.3 RKNN Toolkit 1.0.0 版本下，TensorFlow 模型的输出结果 shape 发生了变化？

1.0.0 以前的版本如果模型输出的数据是按” NHWC” 排列的，将转成” NCHW” 。从 1.0.0 版本开始，output 的 shape 将与原始模型保持一致，不再进行” NHWC” 到” NCHW” 的转换。进行后处理时请注意 channel 所在的位置。

### 4.4 转换模型时，出现”E ValueError: NodeDef mentions attr ‘explicit\_paddings’ not in Op<name=Conv2D; ”的错误

这是由于该模型训练时用的 tensorflow 的版本大于等于 1.14.0 版本，而转换成 rknn 模型时，

采用的 tensorflow 版本小于等于 1.13.2 版本导致的。解决方法是保持训练模型和转换 rknn 模型使用相同的 tensorflow 版本。比如要么都用大于等于 1.14.0 版本，要么都用小于等于 1.13.2 版本。

## 5 Pytorch 模型转换常见问题

在 1.3.0 版本之前，RKNN Toolkit 通过 ONNX 间接支持 pytorch，因此需要将 pytorch 先转成 ONNX。在 1.3.0 及之后的版本中，RKNN Toolkit 支持直接加载 Pytorch 模型。如果转换过程遇到问题，请先将 RKNN Toolkit 升级到最新版本。

### 5.1 转换时遇到类似 “assert(tsr.op\_type == 'Constant')” 的错误

这是 pytorch 0.4.5 以后的版本引入的问题，在你的模型中，有类似 “x = x.view(x.size(0), -1)” 这样的语句，需要改成 “x = x.view(int(x.size(0)), -1)”。

### 5.2 转换时遇到 “PyTorchStreamReader failed reading zip archive” 的错误

详细错误如下：

```
E Catch exception when loading pytorch model: /home/chh/my_code/RKNN_TOOLKIT/1/pytorch_model_convert/yinbao/RetinaFace_PyTorch/PyTorch_RetinaFace/weights/mobilenet0.25_Final.pth!
E Traceback (most recent call last):
E   File "rknn/api/rknn_base.py", line 567, in rknn.api.rknn_base.RKNNBase.load_pytorch
E   File "rknn/base/RKNNlib/app/importer/import_pytorch.py", line 95, in rknn.base.RKNNlib.app.importer.import_pytorch.ImportPytorch.run
E   File "rknn/base/RKNNlib/converter/convert_pytorch.py", line 541, in rknn.base.RKNNlib.converter.convert_pytorch.convert_pytorch.__init__
E   File "/home/chh/chenhao/anaconda3/envs/rknn_toolkit/lib/python3.6/site-packages/torch/jit/_init_.py", line 162, in load
E     cpp_module = torch._C.import_ir_module(cu, f, map_location._extra_files)
E RuntimeError: [enforce fail at inline_container.cc:137] . PyTorchStreamReader failed reading zip archive: failed finding central directory
E frame #0: c10::ThrowEnforceNotMet(char const*, int, char const*, std::string const&, void const*) + 0x47 (0x7f80824bfe17 in /home/chh/chenhao/anaconda3/envs/rknn_toolkit/lib/python3.6/site-packages/torch/lib/libc10.so)
E frame #1: caffe2::serialize::PyTorchStreamReader::valid(char const*) + 0x6b (0x7f802b291ceb in /home/chh/chenhao/anaconda3/envs/rknn_toolkit/lib/python3.6/site-packages/torch/lib/libtorch.so)
E frame #2: caffe2::serialize::PyTorchStreamReader::init() + 0x9a (0x7f802b29579a in /home/chh/chenhao/anaconda3/envs/rknn_toolkit/lib/python3.6/site-packages/torch/lib/libtorch.so)
E frame #3: caffe2::serialize::PyTorchStreamReader::PyTorchStreamReader(std::string const&) + 0x60 (0x7f802b298800 in /home/chh/chenhao/anaconda3/envs/rknn_toolkit/lib/python3.6/site-packages/torch/lib/libtorch.so)
E frame #4: torch::jit::import_ir_module(std::shared_ptr<torch::jit::script::CompilationUnit>, std::string const&, c10::optional<c10::Device>, std::unordered_map<std::string, std::string>, std::hash<std::string>, std::equal_to<std::string>, std::allocator<std::string>, std::string const, std::string> > >) + 0x38 (0x7f802c377618 in /home/chh/chenhao/anaconda3/envs/rknn_toolkit/lib/python3.6/site-packages/torch/lib/libtorch.so)
```

这个错误是因为你要转换的模型只有权重，没有网络结构。

通常.pth 只有权重，并没有网络结构信息。正确的步骤应该是定义一个 net，然后再 net.load\_state\_dict 加载.pth 权重，最后再用 torch.jit.trace 将网络结构和权重固化成一个.pt 文件，然后再用 rknn.load\_pytorch 对这个.pt 文件进行转换，具体使用方法请参考 examples/pytorch/resnet18。只有一个.pth 文件通常是不可以转的。



## 5.3 转换时遇到“E KeyError: 'aten::xxx'”的错误

详细错误如下：

```
E Catch exception when loading pytorch model: resnet18.pt!  
E Traceback (most recent call last):  
E File "rknn/api/rknn_base.py", line 567, in rknn.api.rknn_base.RKNNBase.load_pytorch  
E File "rknn/base/RKNNlib/app/importer/import_pytorch.py", line 95, in rknn.base.RKNNlib.app.importer.import_pytorch.ImportPytorch.run  
E File "rknn/base/RKNNlib/converter/convert_pytorch.py", line 517, in rknn.base.RKNNlib.converter.convert_pytorch.convert_pytorch.__init__  
E File "rknn/base/RKNNlib/converter/convert_pytorch.py", line 601, in rknn.base.RKNNlib.converter.convert_pytorch.convert_pytorch.model_simplify  
E File "rknn/base/RKNNlib/converter/convert_pytorch.py", line 104, in rknn.base.RKNNlib.converter.convert_pytorch.torch_inference_engine.shape_pick  
E File "rknn/base/RKNNlib/converter/convert_pytorch.py", line 139, in rknn.base.RKNNlib.converter.convert_pytorch.torch_inference_engine.__ir_shape_inference  
E KeyError: 'aten::softmax'
```

这是因为某个 op 没有匹配到。我们在每次版本升级时都会修复这类 bug。请使用我们最新的 rknn\_toolkit。

## 5.4 转换时遇到“Syntax error in input! LexToken(xxx)”的错误

详细错误如下：

```
WARNING: Token 'COMMENT' defined, but not used  
WARNING: There is 1 unused token  
!!!!!! Illegal character ''  
Syntax error in input! LexToken(NAMED_IDENTIFIER,'fc',1,27)  
!!!!!! Illegal character ''  
D import clients finished  
2020-02-20 20:40:13.391282: I tensorflow/core/platform/cpu_feature_guard.cc:142]  
Your CPU supports instructions that this TensorFlow binary was not compiled to
```

这个错误的原因有很多种，请按照以下顺序排查：

1. 使用 pytorch 1.2.0。我们建议 pytorch 使用 1.2.0 版本，我们的测试都是基于 1.2.0 来做的，高版本有可能会有一些未知的错误。
2. 未继承 torch.nn.module 创建网络。请继承 torch.nn.module 这个基类来创建网络，然后再用 torch.jit.trace 生成 pt 文件。

## 6 深度神经网络模型设计建议

### 6.1 如何设计卷积神经网络，使其能在 RKNN 上实现最佳性能

以下是几点建议：

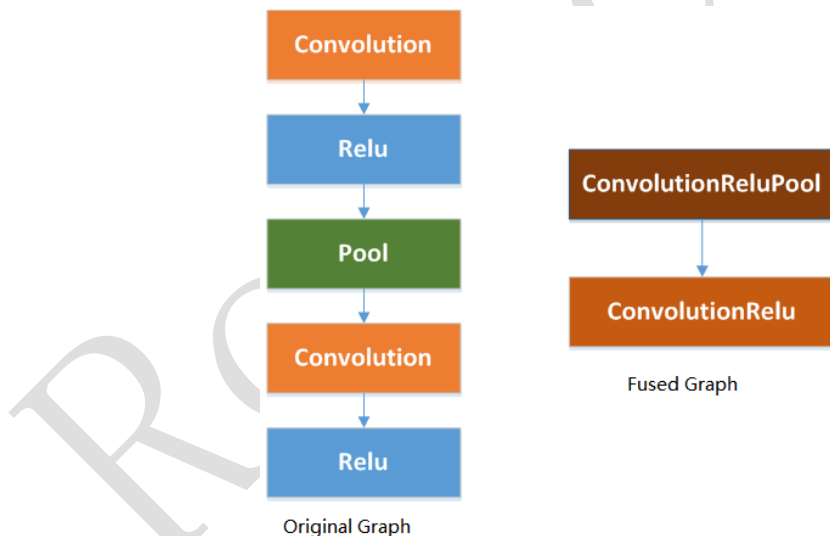
#### 1. 卷积核设置

推荐在设计的时候尽量使用 3x3 的卷积核，这样可以实现最高的 MAC 利用率，使得 NPU 的性能最佳。

NPU 也可以支持大范围的卷积核。支持的最小内核大小为[1]，最大值为 $[11 * stride - 1]$ 。同时 NPU 也支持非平方内核，不过会增加一些额外的计算开销。

#### 2. 融合结构设计

NPU 会对卷积后面的 ReLU 和 MAX Pooling 进行融合的优化操作，能在运行中减少计算和带宽开销。所以在搭建网络时，能针对这一特性，进行设计。



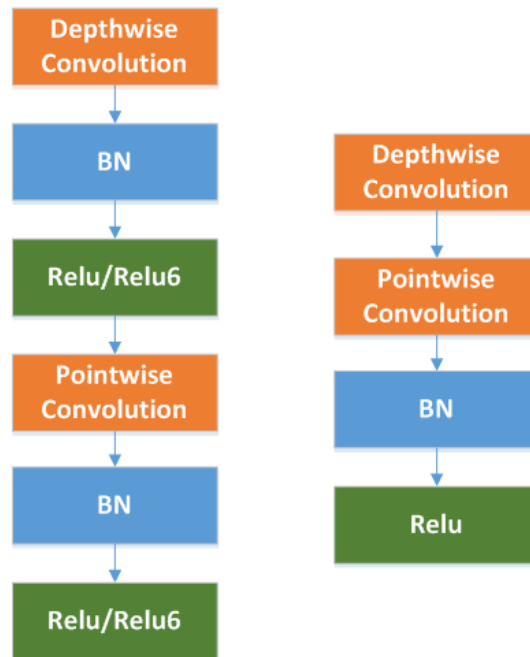
在设计网络时，卷积层后面的 ReLU 层将都会被融合。不过为了确保 MAX Pooling 层也能进行融合加速，需要尽量按照下面规则进行设计：

- pool size 必须是 2x2 或者 3x3，而步长 stride=2
- 2x2 池化的输入图片尺寸必须是偶数，而且不能有填充
- 3x3 池化的输入图片尺寸必须是非 1 的奇数，而且不能有填充
- 如果是 3x3 的池化，则水平输入大小必须小于 64（8-bit 模型）或 32（16-bit 模型）

### 3. 关于 2D 卷积和 Depthwise 卷积的使用

NPU 支持常规 2D 卷积和 Depthwise 卷积加速。由于 Depthwise 卷积特定的结构，使得它对于量化(int8)模型不太友好，而 2D 卷积的优化效果更好。所以设计网络时建议尽量使用 2D 卷积。

如果必须使用 Depthwise 卷积，建议按照下面的规则进行修改，能提高量化后模型的精度：



- 如果网络中的激活函数使用的是 ReLU6，建议将其都改为 ReLU。
- 在 Depthwise 卷积层的 BN 层和激活层，建议去除。
- 在训练时，针对 Depthwise 卷积层，对它的权重进行 L2 正则化。

### 4. 输出通道数设置

建议设定的卷积输出通道数是 NPU 中卷积核个数的倍数，以确保所有卷积核都被更好的利用，从而实现更高的硬件利用率。

### 5. 网络稀疏化

当前的神经网络存在过度参数化现象，并且在其设计时会存在很多冗余。NPU 针对稀疏矩阵，有进行跳零计算和内存提取方面的优化。所以建议在设计网络的时候，可以针对性的进行网络稀疏化设计，以利用该技术进一步提高网络性能。

## 6. 空洞卷积的使用

当使用 tensorflow 来创建带 dilations 参数的卷积时，请使用 `tf.nn.atrous_conv2d` 来创建卷积。目前 toolkit 不支持直接使用 `tf.nn.conv2d` 来创建带 dilations 参数的卷积，否则推理结果会出错。

另外，tensorflow 的版本需要高于 1.14.0，rknn-toolkit 的版本需要高于 v1.4.0。

Rockchip