

RKNN-Toolkit Trouble Shooting

(Technology Department, Graphic Display Platform Center)

Mark: <input type="checkbox"/> Editing <input checked="" type="checkbox"/> Released	Version	1.3
	Author	HPC
	Completed Date	2019-12-25
	Auditor	Randall
	Reviewed Date	2019-12-25

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Electronics Co., Ltd

(All rights reserved)

Revision History

Version no.	Author	Revision Date	Revision description	Auditor
V0.9	HPC	2019-04-01	Initial version release	Randall
V1.0	HPC	2019-07-18	Add some questions.	Randall
V1.1	HPC	2019-08-22	Add some convolution acceleration tips	Randall
V1.2	HPC	2019-10-11	Add some questions	Randall
V1.3	HPC	2019-12-25	Add some questions, rearrange directory structure	Randall

Content

1. RKNN TOOLKIT USAGE RELATED QUESTIONS.....	4
2 QUESTIONS RELATED WITH QUANTIZATION ACCURACY	20
3 COMMON ISSUES OF CAFFE MODEL CONVERSION	22
4 COMMON ISSUES OF TENSORFLOW MODEL CONVERSION	25
5 COMMON ISSUES OF PYTORCH MODEL CONVERSION.....	26
6 RKNN CONVOLUTION ACCELERATION TIPS.....	27

ROCKCHIP

1. RKNN Toolkit usage related questions

1.1 RKNN Toolkit installation problem

1.1.1 “undefined symbol: PyFPE_jbuf” error occurs when install RKNN Toolkit

The reason of the error is Python environment is not clean, for example, numpy is installed in two different paths. You can re-build a clean Python environment and try again.

1.1.2 “Permission Denied” error occurs when install RKNN Toolkit on Toybrick

The reason is there is no root authority. Need to add ‘--user’ option for installation.

1.2 Model configuration problem

1.2.1 Why does channel_mean_value of rknn.config function have 4 values? If it is rgb image, does it still have 4 values?

channel-mean-value of rknn.config: used to set the preprocessing command line parameter. It includes four values (M0 M1 M2 S0). The first three values are mean value parameters and the last value is Scale parameter. If the input data have three channels (Cin0, Cin1, Cin2), the output data will be (Cout0,Cout1, Cout2) after preprocessing. The calculating process is as below:

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

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

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

For example, if need to formulate the input data into [-1, 1], you can set this parameter as (128 128 128 128);

If need to formulate the input data into [0, 1], you can set this parameter as (0 0 0 255).

1.2.2 When the input image is gray picture with single channel, how to set rknn.config interface?

Before version 1.3.0, Please refer to the answer of 1.2.1, when the input image is single channel, only "Cout0 = (Cin0 - M0)/S0" is used, so you can set as (M0, 0, 0, S0), while the values of M1 and M2 are not used.

Start from version 1.3.0, the channel_mean_value of single channel input should be set to (M0, S0).

1.2.3 How to set scale parameter of rknn.config function? That is to compress the input range into a certain scope, e.g. from (0-255) to (0-1).

Refer to the answer of 1.2.1

1.2.4 How to set "channel_mean_value" when input channel large than 3?

Before version 1.3.0, for example, when the input shape is 1x25x25x96(NHWC format), you don't need to set channel_mean_value or reorder_channel. The default value of mean will set to 0, scale will set to 1.

Start from version 1.3.0, for example, when the input shape is 1x25x25x4(NHWC format), the channel_mean_value should be set to (M0, M1, M2, M3, S0). When the input shape is 1x25x25x96(NHWC format, channel is large than 4), you don't need to set channel_mean_value or reorder_channel. The default value of mean will set to 0, scale will set to 1.

1.3 Model conversion problem

1.3.1 What deep learning framework does the RKNN Toolkit support? Whether to support all versions of these deep learning frameworks?

Deep learning frameworks supported by the RKNN Toolkit include TensorFlow, TensorFlow Lite, Caffe, ONNX and Darknet.

It corresponds to the version of each deep learning framework as follows:

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 1.3.0	Last commit: 810d7f7	Not Support	Not Support
1.1.0	>=1.10.0, <=1.13.2	Schema version = 3	1.0	Release version 1.3.0	Last commit: 810d7f7	Not Support	Not Support
1.2.0	>=1.10.0, <=1.13.2	Schema version = 3	1.0	Release version 1.4.1	Last commit: 810d7f7	Not Support	Not Support
1.2.1	>=1.10.0, <=1.13.2	Schema version = 3	1.0	Release version 1.4.1	Last commit: 810d7f7	Not Support	Not Support
1.3.0	>=1.10.0, <=1.13.2	Schema version = 3	1.0	Release version 1.4.1	Last commit: 810d7f7	>=1.0.0, <=1.2.0	>=1.4.0, <=1.5.1

Note:

1. In compliance with semver, SavedModels written with one version of TensorFlow can be loaded and evaluated with a later version of TensorFlow with the same major release. So in theory, the pb file generated by TensorFlow before version 1.14.0, RKNN Toolkit 1.0.0 and later are supported. For more information on TensorFlow version compatibility, please refer to the official link:
<https://www.tensorflow.org/guide/versions>
2. RKNN Toolkit uses the TF Lite schema commits in link:

<https://github.com/tensorflow/tensorflow/commits/master/tensorflow/lite/schema/schema.fb>

Commit hash: 0c4f5dfea4ceb3d7c0b46fc04828420a344f7598.

Because TF Lite schema may not compatible with each other, TF Lite models with older or newer schema may not be loaded successfully.

- There are two caffe protocols RKNN Toolkit uses, one based on the officially modified protocol of berkeley, and one based on the protocol containing the LSTM layer. The protocol based on the official revision of berkeley comes from this link: <https://github.com/BVLC/caffe/tree/master/src/caffe/proto>, commit hash is 21d0608. On this basis RKNN Toolkit have added some OPs. The protocol containing the LSTM layer refers to: <https://github.com/xmfbit/warpctc-caffe/tree/master/src/caffe/proto>, commit hash is bd6181b. These two protocols are specified by the proto parameter in the load_caffe interface.
- The relationship between ONNX release version and opset version, IR version refers to the official website description:

<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

- Darknet official Github link: <https://github.com/pjreddie/darknet>. Our current conversion rules are based on the latest commit of the master branch (commit value: 810d7f7).

1.3.2 Does RKNN Toolkit support model conversion with multiple inputs?

The RKNN Toolkit needs to be upgraded to version 1.2.0 or later.

1.3.3 When will it support to convert pytorch and mxnet model directly to rknn?

Please update RKNN Toolkit to version v1.3.0.

1.3.4 When I load model, the numpy module raises error: Object arrays cannot be loaded when allow pickle=False.

The error message is as follows:

```
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-tvenv/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-tvenv/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
```

This error is caused by the change in the default value of the `allow_pickle` parameter of the `load` file interface after `numpy` is upgraded to 1.16.3. There are two solutions: one is to reduce the `numpy` version to version 1.16.2 or lower; the other is to update RKNN Toolkit to version 1.0.0 or later.

1.3.5 Common troubleshooting steps when loading a model error.

First confirm whether the original deep learning framework can load the model and inference result is correct.

Secondly, if the model has a layer (or OP) that is not supported by the RKNN Toolkit, by opening the debug log switch, you can see in the log which layer or OP is not supported by the RKNN Toolkit. This type of error log usually contains "Try match xxx". Failed" or "Not match xxx" and so on.

1.4 Model quantification problem

1.4.1 RKNN Toolkit supported quantization method

RKNN supports two kinds of quantization mechanisms:

- **Quantization-aware training**

Refer to Tensorflow quantization-aware training

(<https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/quantize>), which requires user should have some re-training experience of fine tune. Use `rknn.build (do_quantization=False)` after the quantized model is loaded through RKNN Toolkit, and now RKNN Toolkit will use the own quantization parameter of the model, so there is no loss on the quantization accuracy.

Currently only imported quantified models of TensorFlow and TensorFlow Lite are supported.

And TensorFlow's quantized model is limited by TensorFlow functions on Windows and cannot be loaded.

- **Post training quantization**

When use this method, user loads the well-trained float point model, and RKNN Toolkit will do the quantization according to the dataset provided by user. Dataset should try to cover as many input type of model as possible. To make example simple, generally put only one picture. Suggest to put more.

Currently RKNN Toolkit supports three kinds of quantization methods:

- ✓ `asymmetric_quantized-u8` (default)

This is the quantization method supported by tensorflow, which is also recommended by Google. According to the description in the article of Quantizing deep convolutional networks for efficient inference: A whitepaper, the accuracy loss of this quantization method is the smallest for most networks.

Its calculation formula is as follows:

$$\begin{aligned} \text{quant} &= \text{round}\left(\frac{\text{float_num}}{\text{scale}}\right) + \text{zero_point} \\ \text{quant} &= \text{cast_to_bw} \end{aligned}$$

Where 'quant' represents the quantized number; 'float_num' represents float; data type of 'scale' is float32; data type of 'zero-points' is int32, it represents the corresponding quantized value when the real number is 0. Finally saturate 'quant' to `[range_min, range_max]`.

$$\begin{aligned} \text{range_max} &= 255 \\ \text{range_min} &= 0 \end{aligned}$$

Currently only supports the inverse quantization of u8, the calculation formula is as follows:

$$\text{float_num} = \text{scale}(\text{quant} - \text{zero_point})$$

✓ dynamic_fixed_point-8

For some models, the quantization accuracy of dynamic_fixed_point-8 is higher than asymmetric_quantized-u8.

Its calculation formula is as follows:

$$\begin{aligned} \text{quant} &= \text{round}(\text{float_num} * 2^{\text{fl}}) \\ \text{quant} &= \text{cast_to_bw} \end{aligned}$$

Where ‘quant’ represents the quantized number; ‘float_num’ represents float; ‘fl’ is the number of digits shifted to the left. Finally saturate ‘quant’ to [range_min, range_max].

$$\begin{aligned} \text{range_max} &= 2^{bw-1} - 1 \\ \text{range_min} &= -(2^{bw-1} - 1) \end{aligned}$$

If ‘bw’ equals 8, the range is [-127, 127].

✓ dynamic_fixed_point-16

The quantization formula of dynamic_fixed_point-16 is the same as dynamic_fixed_point-8, except bw=16. For RK3399pro/RK1808, there is 300Gops int16 computing unit inside NPU, for some quantized to 8 bit network with relatively high accuracy loss, you can consider to use this quantization method.

1.4.2 If do_quantization is False during model conversion, will it do quantization? What is the quantization accuracy? (because the model is nearly half the size after conversion)

There are two scenarios. When the loaded model is the quantized model, do_quantization=False will use the quantization parameter of the model, for more details please refer to the answer of 1.9. When the loaded model is the non-quantized model, do_quantization=False will not do quantization, but will convert the weight from float32 to float16, which will not cause accuracy loss.

1.4.3 When structure RKNN model(invoking build interface), set do_quantization=False can build successfully, but set True will fail to build

The error log is as below:

```

T Caused by op 'fifo_queue_DequeueMany', defined at:
T File "test.py", line 52, in <module>
T   ret = rknn.build(do_quantization=True, dataset='./dataset.txt')
T File "/home/ljqi/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/rknn/api/rknn.py", line 162, in build
T   ret = self.rknn_base.build(do_quantization=do_quantization, dataset=dataset, pack_vdata=pre_compile)
T File "/home/ljqi/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
T   return self.queue_task.queue.dequeue_many(batch_size)
T File "/home/ljqi/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
T   self.queue_ref, n=n, component_types=self.dtypes, name=name)
T File "/home/ljqi/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
T   component_types=component_types, timeout_ms=timeout_ms, name=name)
T File "/home/ljqi/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
T   op_def=op_def)
T File "/home/ljqi/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/tensorflow/python/util/deprecation.py", line 488, in new_func
T   return func(*args, **kwargs)
T File "/home/ljqi/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/tensorflow/python/framework/ops.py", line 3274, in create_op
T   op_def=op_def)
T File "/home/ljqi/work/rock3399pro/RKNPUTools/0.98/rknn-toolkit/venv/lib/python3.5/site-packages/tensorflow/python/framework/ops.py", line 1770, in _init__
T   self.traceback = tf_stack.extract_stack()
T OutOfRangeError (see above for traceback): FIFOQueue '_0_fifo_queue' is closed and has insufficient elements (requested 1, current size 0)
T [[node TfFifoQueue_DequeueMany (defined at /home/ljqi/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!

```

It is because there is no data in dataset.txt, or the data format is not supported. Recommend to use jpg or npy.

1.4.4 What is the role of dataset during RKNN quantization? Why does quantization need to relate to dataset?

During RKNN quantization, need to find appropriate quantization parameters, such as scale or zero point. These quantization parameters should be selected according to the inference of the actual input.

1.4.5 Upgraded to RKNN Toolkit 1.2.0, there are 200 pictures in dataset.txt, but quantitative correction is quickly completed. The accuracy of the rknn model is very low. Are these pictures used for quantitative correction?

RKNN Toolkit 1.2.0 adjusts the default value of `batch_size` in config interface. In this version, if you want to use multiple pictures for quantization correction, the value of this parameter should be set to the corresponding number of pictures. If this value is set too large, it may cause program exceptions due to exhaustion of system memory. In this situation, you need to upgrade to version 1.2.1 or later. In version 1.2.1, the default value of `batch_size` is restored to 100, and multiple quantization correction can be achieved with `epochs` parameter. The number of images used for quantization correction is the product of `batch_size` and `epochs`. For example, if there are 200 pictures in the dataset file, then `batch_size` is set to 100, `epochs` is set to 2, or `batch_size` is set to 200, and `epochs` is set to 1, all of which can achieve the quantization correction of 200 pictures. But the memory usage peak of the former is lower than that of the latter. If you only want to use 100 of them, you can set `batch_size` to 100 and `epochs` to 1.

1.4.6 The shape of numpy array in dataset is (4, 640, 480), but when building quantized rknn model, the log prompts shape (640, 480, 480), then build failure.

When using numpy array for quantization correction, if it is a three-dimensional array, it needs to be arranged in the order of 'whc'.

1.4.7 Is the size of the image used for quantization correction the same as the size of the model input?

Not required. RKNN Toolkit automatically scales images. However, because zooming can change the image information, it may have some impact on the accuracy, so it is better to use pictures of similar size.

1.5 Model inference problem

1.5.1 rknn.Inference() interface error or stuck happened after multiple invoke

If the error log is similar as below:

```
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.redirect_stdouter.redirect_stdout.func_wrapper
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/contextlib.py", line 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 624, in _exitfunc
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/weakref.py", line 548, 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 482, in rmtree
  File "/home/etest/.conda/envs/tensorflow_env/lib/python3.6/shutil.py", line 480, in rmtree
OSError: [Errno 24] Too many open files: '/tmp/tmp_d63w4jh'
```

Please update RKNN Toolkit to 0.9.9 or higher version.

1.5.2 rknn.inference() inferring speed slow issue

This issue has two kinds of phenomenon:

1) The speed of forward inferring test is slow, and some picture may take over 0.5s while testing mobilenet-ssd.

2) The time difference between model rknn.inference and rknn.eval_perf() is relatively big, such as:

Theoretical computing time(single picture)	1.79ms	8.23ms	7.485ms	30.55ms
Actual computing time(single picture)	21.37ms	39.82ms	33.12ms	76.13ms

There are two reasons for the issue of slow measured frame rate:

1. Using the method of pc + adb to upload picture is quite slow, as it has high frame rate requirement for network such as 1.79ms theoretically.
2. In the implementation of 0.9.8 and earlier, the inference included some extra time, and the 0.9.9 and later versions have been optimized.

For more real measured frame rate, you can directly use c/c++ api to test on the board.

1.5.3 The first inference of RKNN Toolkit 0.9.9 version is very slow

RKNN Toolkit 0.9.9 version postpones the model loading to the first inference, so the first inference is relatively slow. This issue has been resolved in versions 1.0.0 and later.

1.5.4 Returned outputs of YOLO forward test is [array1 , array2], the length is [10140 , 40560], what is the meaning of the returned value?

The outputs returned by `rknn.inference` is a list of numpy ndarray, the size and quantity of each model output data are different, users need to look up the corresponding output and analytic rule of models by themselves.

1.5.5 Does `rknn.inference()` support multiple pictures input at the same time? Or support batch input?

The RKNN Toolkit needs to be upgraded to version 1.2.0 or later. And you need to specify the number of input images when building the RKNN model. For detailed usage, refer to the description of the build interface in <Rockchip_User_Guide_RKNN_Toolkit_V1.2.1_CN.pdf>.

1.5.6 The same picture, why the inference result of `batch_size=1` is different from the result of `batch_size=4`?

Because some OPs are optimized when the batch size is 4, the accuracy of the operation is higher, so some model results look a little different.

1.5.7 How to correspond the inference result to the each input when the batch_size is large than 1?

If the inference result corresponding to the input image(shape is [224, 224, 3]) is output(shape is [1, 1001]). When the batch_size is 2, the input is the concat result of two images(shape is [448, 224, 3]), the shape of corresponding inference result is [2, 1001].

1.5.8 The RKNN Toolkit python interface inference results are slightly different from the RKNN API C interface inference. Why?

It is possible that cv2 decoding and jpeg decoding are inconsistent. The python side can use numpy's tofile() method to export the image data to binary, and the C API directly reads the binary image data as input, thus ensuring the comparison under the same input.

1.5.9 The input of the model is single-channel. After reading the graph with opencv, it is used as input to inference, and the result is wrong.

When use the default imread interface of opencv to read the image, it will return a 3-channel graph, so the result is incorrect. Use the following method to read the image and the result will be correct:

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

1.5.10 The RKNN Toolkit version is 1.1.0. When inferring, the sum of some models` softmax is not 1.

Please update to version 1.2.1 or later.

1.6 Precompilation issues

1.6.1 Fail to enable `pre_compile=true` when using RKNN Toolkit to convert model on the development board

Arm64 version RKNN Toolkit doesn't support `pre_compile` so far, if need to open `pre_compile`, suggest to use x86 version RKNN Toolkit to do the conversion.

1.6.2 Pre-compile model generated by RKNN Toolkit 0.9.9 can not run on RK3399Pro which NPU driver version is 0.9.6.

Pre-compiled model generated by RKNN Toolkit 1.0.0 can not run on device installed old driver (NPU driver version < 0.9.6), and pre-compiled model generated by old RKNN Toolkit (version < 1.0.0) can not run on device installed new NPU driver (NPU driver version == 0.9.6). The driver version number can be queried through the `get_sdk_version` interface.

1.6.3 When calling `rknn.build()` with `pre_compile=True`, it raises an error, it can be successful if it is not set.

The error message is as follows:

```
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'
```

Please confirm:

- 1) The system is equipped with the gcc compiler toolchain
- 2) The name of the model only contains “letters”, “numbers”, “_”. Or you can update RKNN Toolkit to version 1.3.0 or later, the RKNN Toolkit will automatically handle these special characters.

1.7 Initialize the runtime problem

1.7.1 When I call rknn_init(), it raises error: RKNN_ERR_MODEL_INVALID.

The error message is as follows:

```
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
```

There are two situations in which this error occurs:

1. The option to pre_compile=True was used when generating the rknn model. Different versions of the RKNN Toolkit and drivers have a corresponding relationship. It is recommended to upgrade the firmware of the RKNN Toolkit and the board to the latest version.
2. There is no option to use pre_compile=True when generating the rknn model. At this

time, the system firmware is too old. It is recommended to upgrade the firmware of the board to the latest version.

1.7.2 When I call rknn_init(), it raises error: RKNN_ERR_DEVICE_UNAVAILABLE.

The error message is as follows:

```
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
```

Please check it out as follows:

- 1) Make sure that the RKNN Toolkit and the firmware of devices have been upgraded to the latest version. The correspondence between each version of the RKNN Toolkit and the components of the system firmware is as follows:

RKNN Toolkit	rknn_server	NPU Driver	librknn_runtime
1.0.0	0.9.6/0.9.7	6.3.3.2, 22133405	0.9.8/0.9.9
1.1.0	0.9.8	6.3.3.2, 28225a	1.0.0
1.2.0	0.9.9	6.4.0, 67399c4	1.1.0
1.2.1	1.2.0	6.4.0, 5c7e2cbd	1.2.0
1.3.0	1.3.0	6.4.0, 227915	1.3.0

The version of these components is queried on RK1808 as follows:

```
# execute these commands on RK1808
dmesg | grep -i galcore # Query the NPU driver version
strings /usr/bin/rknn_server | grep build # Query the rknn_server version
strings /usr/lib/librknn_runtime.so | grep version # Query the librknn_runtime
```

version

The version information can also be queried through the `get_sdk_version` interface, where the DRV version corresponds to the version of `rknn_server`.

- 2) Make sure the “adb devices” command can get the device, and the target and device_id settings of `rknn.init_runtime()` are correct.
- 3) If you use RKNN Toolkit 1.1.0 and above, make sure `rknn.list_devices()` can get the devices list.
- 4) If you are using a compute stick or NTB mode for the RK1808 EVB version, make sure you have called `update_rk1808_usb_rule.sh` (contained in the RKNN Toolkit distribution) to get read and write access to the USB device.
- 5) If you are running the AARCH64 version of the RKNN Toolkit directly on the RK3399/RK3399Pro, make sure the system firmware has been upgraded to the latest version.

1.8 Log problems

1.8.1 When using the RKNN Toolkit, if the logging module is used in the program to output the log, it will report an error and exit.

Please upgrade to 1.2.1 or later.

1.8.2 Upgraded to RKNN Toolkit 1.2.0, after calling `load_xxx` interfaces, the program exits directly without any log.

The message is as follows:

```

_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.uint8, 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.uint16, 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
-> Loading model

```

New environment variables ("PYTHONLEGACYWINDOWSSTDIO") are required when using RKNN Toolkit 1.2.0 on Windows systems, and it should be set with value of 1.

It can also be upgraded to RKNN Toolkit 1.2.1 and later versions, which do not require manual setting of the environment variables.

2 Questions related with quantization accuracy

2.1 The accuracy doesn't match with original model after quantization, how to debug?

➤ **Firstly make sure the accuracy of float type is similar to test result of original platform:**

- (1) Make `rknn.build(do_quantization=False)` when the quantized model is loaded by RKNN Toolkit.
- (2) Refer to 1.1 to set **channel_mean_value** parameter, which should be same as the parameter used for training model.
- (3) Make sure the sequence of the input image channel must be R,G,B while testing. (Whatever the sequence of the image channel is used for training, it must be input by R,G,B while using RKNN to do testing)
- (4) Set **reorder_channel** parameter in `rknn.config` function, '0 1 2' stands for RGB, '2 1 0' stands for BGR, and it must be consistent with the sequence of the image channel used for training.

➤ **Accuracy test after quantization**

- (1) Use multiple pictures to do quantization, to ensure the stability of quantization accuracy.

Set `batch_size` parameter in `rknn.config` (recommend to set `batch_size = 200`) and provide more than 200 images path in `dataset.txt` for quantization.

If the display memory is not enough, you can set `batch_size = 1`, `epochs=200` instead of `batch_size = 200` for quantization.

- (2) Accuracy comparison, try to use relatively big data set to do testing. Compare the

accuracy of top-1, top-5 for classifying network, compare mAP, Recall of data set for checking network, and so on.

(3) If it is face recognition, the results of the float model and the results of the quantitative model cannot be used for feature comparison. For example, two pictures of A1 and A2, the results of using the float model are A1fp, A2fp, the results of running with the quantitative model are A1u8, A2u8. At this time, the Euclidean distance of A1fp and A2fp can be calculated to calculate the similarity of two pictures, and the Euclidean distance of A1u8 and A2u8 can also be calculated to calculate the similarity of two pictures. But can't calculate the Euclidean distance of A1fp and A2u8 to calculate the similarity of two pictures.

2.2 How to dump the output of each layer of network

Currently PC simulator supports to dump out data of each layer of network. Need to set an environment variable before executing inference script. The command is as below:

```
export NN_LAYER_DUMP=1  
python xxx.py
```

After execution, tensor data file of each layer of network will be generated in current directory, and then you can compare with data of other framework layer by layer.

Note, some layers may be combined, for example, conv+bn+scale may be combined into one conv, in this case, need to compare with output of scale layer of the original model.

2.3 Which frameworks` quantized model are currently supported by the RKNN Toolkit?

The RKNN Toolkit currently supports quantized models of the two frameworks TensorFlow and TensorFlow Lite.

3 Common issues of Caffe model conversion

3.1 “Deprecated caffe input usage” error occurs during model conversion

It means this model is old version of caffe mode. Need to change input layer into below format.

```
layer {
  name: "data"
  type: "Input"
  top: "data"
  input_param {
    shape {
      dim: 1
      dim: 3
      dim: 224
      dim: 224
    }
  }
}
```

3.2 “Message type "caffe.PoolingParameter" has no field named "round_mode"” error occurs during model conversion

round_mode field of Pool layer cannot be recognized, you can change it to ceil_model. For example, if originally it is round_mode: CEIL, then you can delete (ceil_mode is True by default) or change to ceil_mode:True.

3.3 “ValueError(“'%s' is not a valid scope name" % name)” error occurs during caffe or other model conversion

The detailed error log is as below:

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

In this case, it is because layer name '_plusxxx' is not allowed to use _ at the beginning. Need to follow the naming rule of tensorflow:

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

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

3.4 “Invalid tensor id(1), tensor(@mbox_conf_flatten_188:out0)” error occurs when Caffe version SSD conversion fails

Not support detectionoutput layer, you can delete and then change to CPU.

3.5 There should be three output tensor after Caffe version SSD model deletes detectionoutput, but actually only return two tensor by RKNN inference

The missing tensor is priori box. It is the same during training and inference stage, and for all inputs. In order to improve performance, RKNN Toolkit optimized the relative layer in the model. If want to get the tensor of priori box, you can save the tensor of priori box, or use Caffe to do inference once in training stage.

3.6 “ValueError: Invalid tensor id(1), tensor(@rpn_bbox_pred_18:out0)” error occurs during py-faster-rcnn model conversion

Comparing with official code, need to change 'proposal' layer of prototxt as below:

<pre> 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 } } </pre>	<pre> 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: 'ProposalLayer' param_str: "'feat_stride': 16" } } </pre>
--	---

```

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 “E Not supported caffe net model version(v0 layer or v1 layer)” error occurs during model conversion

```
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! 错误信息类似如下:
T Traceback (most recent call last):
T   File "rknn/api/rknn_base.py", line 288, in rknn.api.rknn_base.RKNNBase.load_caffe
T   File "rknn/base/RKNNlib/converter/caffeloader.py", line 997, in rknn.base.RKNNlib.converter.caffeloader.caffeLoader.load_blobs
T   File "rknn/base/RKNNlib/converter/caffeloader.py", line 893, in rknn.base.RKNNlib.converter.caffeloader.caffeLoader.parse_blobs
T   File "rknn/base/RKNNlib/RKNNLog.py", line 105, in rknn.base.RKNNlib.RKNNLog.RKNNLog.e
T ValueError: Not supported caffe net model version(v0 layer or v1 layer). Please use the newest model version.
Load vgg16 failed!
```

The main reason is that the version of the caffe model is too old and needs to be updated. The update method is as follows (take VGG16 as an example):

- 1) Download Caffe source code from <https://github.com/BVLC/caffe.git>
- 2) Compile Caffe
- 3) Convert the model to a new format

```
./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 Common issues of Tensorflow model conversion

4.1 “AttributeError: ‘NoneType’ object has no attribute op” error occurs during Google official ssd_mobilenet_v2 model conversion

One possible reason is that input node is not correct. You can modify as below:

```
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 “Cannot convert value dtype ([‘resource’, ‘u1’]) to a Tensorflow Dtype ” error occurs during SSD_Resnet50_v1_FPN_640x640 model conversion

Need to update RKNN Toolkit to version 0.9.8 or higher.

4.3 On RKNN Toolkit 1.0.0, is the output shape of RKNN model converted from TensorFlow changed?

Versions prior to 1.0.0 will convert output shape from "NHWC" to "NCHW". Starting from this version, the shape of the output will be consistent with the original model, and no longer convert from "NHWC" to "NCHW". Please pay attention to the location of the channel when performing post processing.

5 Common issues of Pytorch model conversion

Before version 1.3.0, RKNN Toolkit indirectly supports Pytorch through ONNX, so need to convert Pytorch to ONNX first. Start from version 1.3.0, RKNN Toolkit supports loading Pytorch model directly. If issue occurs during conversion, please update RKNN Toolkit to the latest version first.

5.1 “assert(tsr.op_type == 'Constant') ” error occurs during conversion

This issue is introduced after pytorch 0.4.5 version. In your model, if there is something like “x = x.view(x.size(0), -1)”, need to change to “x = x.view(int(x.size(0)), -1)” .

6 RKNN convolution acceleration tips

6.1 How to design a convolutional neural network to achieve optimal performance on RKNN

Here are some suggestions from us:

1. Optimal Kernel Size is 3x3

Convolution cores can support a large range of kernel sizes. The minimum supported kernel size is [1] and maximum is [11 * stride - 1].

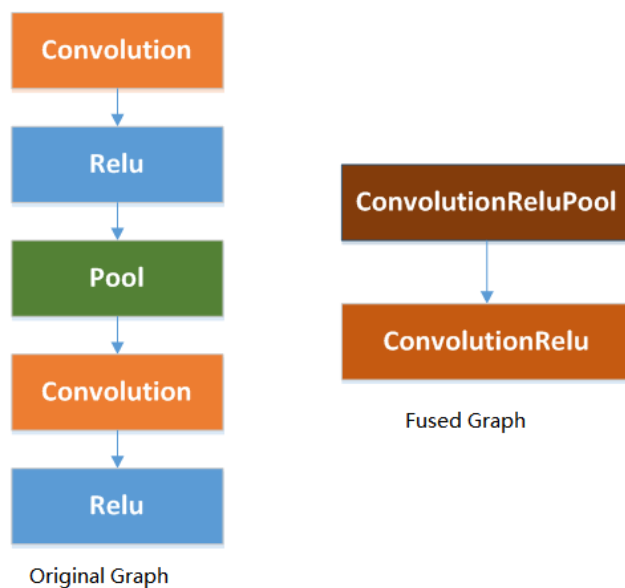
The NN Engine performs most optimally when the Convolution kernel size is 3x3, under which the highest MAC utilization can be achieved.

Non-square kernels are also supported, but with some computation overhead.

2. Fused Operations Reduce Overhead

The Convolution core can fuse ReLU and MAX Pooling operations on the fly to further reduce computation and bandwidth overhead. A ReLU layer following a Convolution layer will always be fused, while MAX pooling layer fusion has the following restrictions, Max pooling must

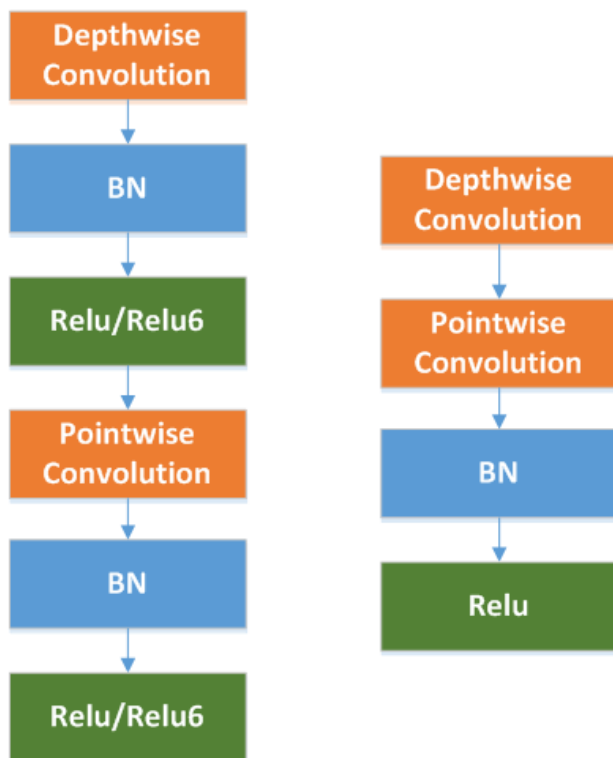
- have a pool size of 3x3 or 2x2, and stride of 2
- 2x2 pooling must have an even input size and no padding
- 3x3 pooling must have odd input size which is not one and no padding
- Horizontal input size must be less than 64 (8-bit mode) or 32 (16-bit mode) if pool size is 3x3



3. Depthwise Convolutions

Both regular 2D and Depthwise convolutions are supported, while 2D convolutions perform more optimally. Since Depthwise Convolution-specific structure makes it less friendly to quantized model. It's recommend to use 2D convolution whenever possible when designing your network.

If you must use a Depthwise convolution, it's recommend to follow the rules below that can improve the accuracy of the quantized model:



- Change the activation function RELU6 to RELU.
- Remove the BN layer and activation layer of the Depthwise convolution layer.
- In training, for the Depthwise convolutional layer, L2 regularization of its weight.

4. Output channel number setting

It's recommend to set the number of convolution output channels to be a multiple of the number of convolution kernels in the NPU to ensure that all convolution kernels are better utilized for higher hardware utilization.

5. Take advantage of Hardware's Sparse Matrix Support

Modern Neural-Networks are known to be over parameterized and have much redundancy in their design. Pruning a network to be sparse has been proven to reduce computation overhead while maintaining accuracy.

RKNN hardware is designed to support sparse matrix operations efficiently by skipping computations and memory fetches on zero values. The sparsity level can be fine grain down to individual weights. Designing a sparse network to take advantage of this technology could

further improve performance on RKNN.

ROCKCHIP