Classification Level: Top secret ( ) Secret ( ) Internal ( ) Public (√)

RKNN-Toolkit2 Trouble Shooting

(Technology Department, Graphic Computing Platform Center)

Mark: [ ] Editing [√] Released	Version	1.5.2
	Author	HPC
	Completed Date	2023-8-21
	Auditor	Vincent
	Reviewed Date	2023-8-21

Revision History

Version no.	Author	Revision Date	Revision description	Auditor
1.4.2	HPC	2023-2-10	Initial version release.	Vincent
1.5.0	HPC	2023-5-18	Modify the definition of whl package	Vincent
1.5.2	HPC	2023-8-21	Update version	Vincent

目录

1 Overview

The following documents are only applicable to RKNN-Toolkit2 unless otherwise specified. At this time, the model converted by the tool is suitable for RK3566, RK3568, RK3588, RK3588S, RV1103, RV1106 and RK3562 platforms.

When referring to this document, please confirm the version of RKNN-Toolkit2 in your environment, otherwise some content may not apply.

2 Relationship between RKNN-Toolkit2 and RKNPU2

RKNN-Toolkit2 is a model conversion tool based on Python language. It can convert models exported from other training frameworks to RKNN, and provides a limited inference interface to help users test the effect of model conversion. The project link is:

https://github.com/rockchip-linux/rknn-toolkit2

RKNPU2 is a board-side component that provides NPU driver and provides functions such as model loading and model inference based on C language. Compared with the RKNN-Toolkit2, the C API inference interface of RKNPU2 is more flexible and has better performance. The project link is:

https://github.com/rockchip-linux/rknpu2

RKNN-Toolkit2, RKNPU2 have consistent version. In order to avoid unnecessary trouble, users are recommended to use the same version of RKNN-Toolkit2 and RKNPU2; version updates usually include bug fixes and performance optimizations, and it's advised to use the latest version.

3 RKNN-Toolkit2 environment dependency problem

This chapter mainly introduces common tool installation problems.

3.1 The environment that RKNN-Toolkit2 depends on is too restrictive, making it impossible to install successfully

When all dependent libraries have been installed, but the versions and requirements of some libraries do not match, you can try adding the --no-deps parameter after the installation command to bypass the environment check during installation. Such as:

```
pip install rknn-toolkit2*.whl --no-deps
```

3.2 ONNX dependency description

The loading ONNX model function of RKNN-Toolkit2 depends on the ONNX version. Due to the good compatibility between versions of ONNX, no problems caused by versions have been found.

The recommended ONNX version is 1.10.0.

3.3 PyTorch dependency description

The function of loading PyTorch models of RKNN-Toolkit2 depends on the PyTorch version. PyTorch's models can be divided into floating-point models and quantized models (including QAT and PTQ quantized models). For models exported by PyTorch 1.6.0, it is recommended to downgrade the PyTorch version that RKNN-Toolkit2 depends on to 1.6.0 to avoid loading failures. For quantized models (QAT, PTQ), it`s recommend using PyTorch 1.10 to export the model, and upgrade the PyTorch version that RKNN-Toolkit2 depends on to 1.10. In addition, when loading the PyTorch model, it`s recommend exporting the PyTorch version of the original model, which should be consistent with the PyTorch version that RKNN-Toolkit2 depends on.

In general, it`s recommend using PyTorch 1.6.0, 1.9.0 or PyTorch 1.10.0.

3.4 TensorFlow dependency description

The TensorFlow model loading function of RKNN-Toolkit2 relies on TensorFlow. Due to the poor compatibility between versions of TensorFlow, other versions may cause exceptions or work normally. In addition, when loading the TensorFlow model, It is recommended that the TensorFlow version for exporting the original model should be the same as the TensorFlow version that the RKNN-Toolkit2 depends on.

For problems caused by the TensorFlow version, it is usually reflected in the load_tensorflow and build phases, and the error message will point to the tensorflow-lib path.

The recommended TensorFlow version is 2.6.2.

3.5 RKNN-Toolkit2 installation package naming rules

Taking the release of version 1.4.0 as an example, the RKNN-Toolkit2 wheel package command rules are as follows:

```
rknn_toolkit2-1.4.0+22dcfef4-cp36-cp36m-linux_x86_64.whl
```

- rknn_toolkit2: Tool name.
- 1.4.0: Tool version.
- 22dcfef4: Commit number.
- cp<xx>-cp<xx>m: Python version, such as cp36-cp36m indicates that the applicable Python version is 3.6.
- linux_x86_64: OS & CPU architecture.

Please install the corresponding RKNN-Toolkit2 according to your OS, CPU architecture and Python version, otherwise the installation will fail.

3.6 Is there an ARM version of RKNN-Toolkit2

RKNN-Toolkit2 does not have an ARM version. If you need to use the Python interface for inference on ARM, you can install RKNN-Toolkit-lite2, which has an ARM version.

3.7 The dependent library of bfloat16 cannot be installed

There is an error in the installation of the dependency library of bfloat16:

```
    bfloat16.cc:2013:57: note:   expected a type, got ‘bfloat16’
    bfloat16.cc:2013:57: error: type/value mismatch at argument 2 in templa
e, class Functor> struct greenwaves::{anonymous}::UnaryUFunc’
    bfloat16.cc:2013:57: note:   expected a type, got ‘bfloat16’
    bfloat16.cc:2015:67: error: ‘>>’ should be ‘> >’ within a nested templa
        RegisterUFunc<BinaryUFunc<bfloat16, bfloat16, ufuncs::NextAfter>>(
                                                                      ^
    bfloat16.cc:2015:58: error: type/value mismatch at argument 1 in templa
e, class Functor> struct greenwaves::{anonymous}::BinaryUFunc’
        RegisterUFunc<BinaryUFunc<bfloat16, bfloat16, ufuncs::NextAfter>>(
                                                                     ^
    bfloat16.cc:2015:58: note:   expected a type, got ‘bfloat16’
    bfloat16.cc:2015:58: error: type/value mismatch at argument 2 in templa
e, class Functor> struct greenwaves::{anonymous}::BinaryUFunc’
    bfloat16.cc:2015:58: note:   expected a type, got ‘bfloat16’
    error: command 'gcc' failed with exit status 1
```

Just replace the source of Python with Ali source.

4 User guide for setting parameters while model conversion.

This chapter mainly introduces the usage of common parameters in the model conversion stage.

4.1 Main parameters description

During model conversion, 'rknn.config' and 'rknn.build' will affect the conversion effect. 'rknn.load_onnx', 'rknn.load_tensorflow' specify input and output nodes, which will affect the conversion result. For 'rknn.load_pytorch', 'rknn.load_tensorflow', the size of the specified input will affect the conversion result.

Usually, you can refer to the following basic ideas for configuration:

1. Prepare data for quantization, provide dataset file.

2. Determine the platform to be used by the model, such as RK3566, RV1106, and determine the target_platform parameter in 'rknn.config'.

3. If the input of the model is a 3-channel image, and the quantized data is in image format (such as jpg, png format), you need to pay attention to whether the model input is in RGB or BGR format, and confirm the quant_img_RGB2BGR parameter in 'rknn.config'.

4. Confirm the normalization parameters during model training, and confirm the value of the mean_values/std_values parameter in 'rknn.config'.

5. Check the input size of the model, fill in the 'rknn.load' interface, such as the input_size_list parameters in 'rknn.load_pytorch'.

6. Confirm the quantized_dtype parameter in 'rknn.config' (It can be ignored when the model is not quantized).

7. Confirm the quantized_algorithm parameter in 'rknn.config' (It can be ignored when the model is not quantized).

8. Confirm whether to quantize the model and confirm the do_quantization parameter in 'rknn.build'. When quantizing, you need to fill in the dataset parameter in 'rknn.build' at the same time.

4.2 Compatibility issues on RK platform.

For the platform parameters set by target_platform, the compatibility relationship is as follows:

- The models used by the RK3566 and RK3568 platforms are compatible with each other.
- The models used by the RK3588 and RK3588S platforms are compatible with each other.
- The models used by the RV1103 and RV1106 platforms are compatible with each other.

4.3 Format and requirements of data for quantization

The format of the data is allowed to be in image (jpg, png) format, and RKNN-Toolkit2 relies on OpenCV for reading; it is also allowed to be in npy format, and numpy is called for reading at this time.

For models with non-RGB/BGR image input, we recommend using numpy's npy format to provide quantized data.

4.4 How to fill in dataset.txt file when multiple input

For dataset.txt file, one line is an example of one input; when there are multiple inputs, multiple inputs are written on the same line and separated by spaces.

For single input, such as:

```
sampleA.npy
sampleB.npy
```

For thriple inputs, such as:

```
sampleA_in0.npy sampleA_in1.npy sampleA_in2.npy
sampleB_in0.npy sampleB_in1.npy sampleB_in2.npy
```

4.5 How to confirm the quant_img_RGB2BGR parameter

When using pictures as data for quantization, you need to consider setting the

quant_img_RGB2BGR parameter.

When the model is trained with RGB images, the quant_img_RGB2BGR parameter is set to False. And when using rknn.inference, or RKNPU2 C API for inference, input RGB images.

When the model is trained with BGR images, the quant_img_RGB2BGR parameter is set to True. And when using rknn.inference, or RKNPU2 C API for inference, also need to input BGR images. (quant_img_RGB2BGR only affects images read in from the quantized dataset).

If the data for quantization is in numpy's npy format, it is recommended not to use the quant_img_RGB2BGR parameter to avoid confusion.

4.6 Calculation order of mean/std and quant_img_RGB2BGR

Because quant_img_RGB2BGR only controls whether to convert the channel when reading the correction set image during the quantization process, it will not affect other steps. For RKNN-Toolkit2 and RKNPU2 C API, the calculation sequence is as follows: first perform the mean subtraction (mean) operation, and then divide by the standard deviation (std). there is no channel order conversion operation.

4.7 The setting problem of mean/std when non-3-channel input (such as grayscale image) or multi-input

The setting format of mean, std is consistent. Here is an example of mean.

Assuming that the input has N channels, the value of mean_values is [[channel_1, channel_2, channel_3, channel_4, ..., channel_n]].

When there are multiple inputs, the value of mean_values is [[channel_1, channel_2, channel_3, channel_4, ..., channel_n], [channel_1, channel_2, channel_3, channel_4, ..., channel_n]].

4.8 Selection of quantization parameter correction algorithm and dataset size

quantized_algorithm has ‘normal’, ‘mmse’ and ‘kl_divergence’ to choose from. When this

parameter is not specified, ‘normal’ is used by default. The quantization algorithm of MMSE can achieve better quantization accuracy on some models, but the disadvantage is that quantization takes a long time. The KL_divergence quantization algorithm will take more time than Normal, but it will be much less than MMSE, and in some scenarios (when the feature distribution is uneven), a better quantization accuracy can be obtained.

It`s recommend to use the Normal method for quantization first. If the quantization effect is not good, you can try to use the MMSE or KL_divergence quantization algorithm.

When using Nonmal or KL_divergence quantization, it is recommended to give 50~200 sets of data for quantization. When using MMSE quantification, it is recommended to use 20~50 sets of data for quantification.

4.9 After the model is quantized, what will be change for the input and output during inference

When using the regular RKNPU2 C API operation (referring to calling C API without pass_through and zero_copy), the data type of the input data (such as uint8 data, float data) has nothing to do with model quantization. The data type of the output data can always be automatically processed into float32 format. According to the quantization type of the model, you can also choose to output the model directly. In this case, the data type is the data type output by the last node of the model. The Python inference interface will be slightly different, the specific relationship is as follows:

Table 4-1 Input and output relationship of quantized model and float model

After quantization	Python inference (rknn.inference)	C API inference (rknn.run)
Limit of input dtype	Unlimited. The input of rknn.inference is a numpy array with a data type attribute. the input data will be automatically converted into the data format required by the RKNN model.	Unlimited. The rknn_tensor_type parameter of rknn inputs can specify RKNN_TENSOR_FLOAT32, RKNN_TENSOR_FLOAT16, RKNN_TENSOR_INT8, RKNN_TENSOR_UIN8, RKNN_TENSOR_INT16 according to the actual input. After specifying, the input will be automatically converted into the data format required by the RKNN model.
Change of output dtype	No change. Regardless of whether the model is quantized or not, Python's rknn.inference interface always returns an output of type float. can not select other data type.	New output dtype allowed. For the rknn outputs of RKNPU2 C API , you can always set want_float=True or 1 to get the output of float type. After quantization, you can set want_float=False or 0, and you can output the original output type of the last node. For example, when i8 is quantized, output int8 data.

Change of input layout (NCHW, NHWC)	No change. Regardless of quantization, data_format can set NCHW or NHWC according to the input.	No change. Regardless of quantization, rknn_tensor_format of rknn inputs can always be set to NCHW or NHWC layout.

4.10 **Is there an online compile mode**

RKNN-Toolkit2 only supports exporting offline precompiled models, but does not support exporting online compile models (supported by RKNN-Toolkit1), therefore there is no mode selection for offline precompiled and online compile.

4.11 Can the RKNN exported by RKNN-Toolkit be used on the RK3566 platform?

Can't.

The RKNN model exported by RKNN-Toolkit is suitable for platforms such as RK1806 / RK1808 / RK3399Pro / RV1109 / RV1126; the RK3566 platform needs the RKNN model exported by RKNN-Toolkit2. The RKNN model exported by RKNN-Toolkit2 is suitable for such as RK3566 / RK3568 / RK3588 / RK3588S / RV1103 / RV1106 / RK3562.

Please refer to the following project for the usage of RKNN-Toolkit:

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

Please refer to the following project for the usage of RKNN-Toolkit2:

https://github.com/rockchip-linux/rknn-toolkit2

5 Problems when loading model

5.1 Framework support and version problem

Please refer to the chapter 1.4 of "Rockchip_User_Guide_RKNN_Toolkit2_EN" document.

5.2 OP support

It depends on the framework where the model comes from. For details, please refer to the "RKNNToolkit2_OP_Support" documents:

https://github.com/rockchip-linux/rknn-toolkit2/blob/master/doc/

5.3 Common issues of Caffe model conversion

5.4 Common issues of Darknet model conversion

5.5 Common issues of ONNX model conversion

5.5.1 Encountered the error that the IR version is too high during conversion

The error log is as follows:

```
Your model ir_version is higher than the checker`s
```

RKNN-Toolkit2 1.4.0 and earlier versions support models exported by ONNX 1.6.0~1.9.0, Newer ONNX exported models may report errors.

5.5.2 An error of 'Error parsing message' occurred when loading the model

When converting the 'examples/onnx/resnet50v2' model, it prompts that the loading failed:

```
E load_onnx: Catch exception when loading onnx model: /rknn_resnet_demo/resnet50v2.onnx!
E load_onnx: Traceback (most recent call last):
E load_onnx:   File "rknn/api/rknn_base.py", line 1094, in rknn.api.rknn_base.RKNNBase.load_onnx
E load_onnx:   File "/usr/local/lib/python3.6/dist-packages/onnx/__init__.py", line 115, in load_model
E load_onnx:     model = load_model_from_string(s, format=format)
E load_onnx:   File "/usr/local/lib/python3.6/dist-packages/onnx/__init__.py", line 152, in load_model_from_string
E load_onnx:     return _deserialize(s, ModelProto())
E load_onnx:   File "/usr/local/lib/python3.6/dist-packages/onnx/__init__.py", line 95, in _deserialize
E load_onnx:     decoded = cast(Optional[int], proto.ParseFromString(s))
E load_onnx: google.protobuf.message.DecodeError: Error parsing message
```

The reason may be that the resnet50v2.onnx model is damaged (if not all downloaded), you need to download the model again and make sure its MD5 value is correct, such as:

```
22ed6e6a8fb9192f0980acca0c941414  resnet50v2.onnx
```

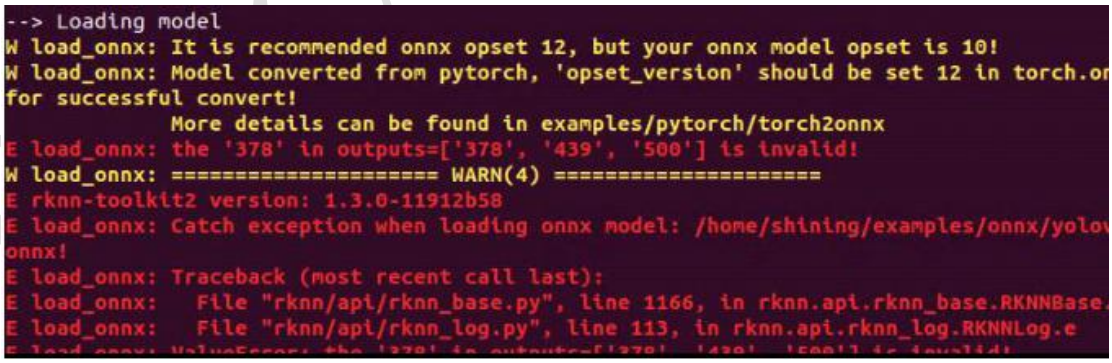
5.5.3 Is support dynamic input shape

RKNN-Toolkit2 version before 1.5.2 does not support dynamic input shape. For example, the input dimension of onnx is [-1, 3, -1, -1], which means that the batch, height and width dimensions are not fixed, which is not supported.

Versions 1.5.2 and later can simulate the dynamic input shape through the 'dynamic_input' parameter of 'rknn.config'. For details, see the introduction to the 'rknn.config' chapter of "Rockchip_User_Guide_RKNN_Toolkit2" and the usage example of 'examples/functions/dynamic_input'.

5.5.4 Error when custom output node

When load_onnx with 'outputs' parameter is passed in to trim the model, but the following error is reported:



The log indicates that the output node 378 is invalid, so the 'outputs' parameter needs to be set with the correct output node name.

5.6 Common issues of PyTorch model conversion

5.6.1 How to solve the problem of torch._C has no attribute _jit_pass_inline when loading the PyTorch model?

The error log is as follows:

```
'torch._C' has no attribute '_jit_pass_inline'
```

Please update version of PyTorch to 1.6.0 or later.

5.6.2 Save format of PyTorch model

Only models saved in torch.jit.trace can be used. The torch.save saves only the parameters of the model, and does not have a network structure. With only one network parameter, RKNN-Toolkit2 cannot construct the corresponding network.

5.6.3 The PytorchStreamReader failed error occurs during conversion

The error log is as follows:

```
E Catch exception when loading PyTorch model: ./mobilenet0.25_Final.pth!
E Traceback (most recent call last):
……
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: faild finding central directory frame #0 ……
```

This error is because the model you are converting contains only weights but no network structure.

Usually the pth file contains only weights and no network structure information. The correct step is to define a net, then load the pth weights with net.load_state_dict(), and finally use torch.jit.trace() to freeze the network structure and weights into a pt file, and then use rknn.load_pytorch() to convert this pt file.

5.6.4 KeyError occurs during conversion

The error log is as follows:

```
E Traceback (most recent call last):
……
E KeyError: ‘aten::softmax’
```

When an error message like KeyError: 'aten::xxx' appears, it means that the current version does not support the operation. RKNN-Toolkit2 will fix such bugs in each version upgrade, please use the latest version of RKNN-Toolkit2.

5.6.5 Syntax error in input error occurs during conversion

The error log is as follows:

```
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 ‘’’
```

There are many reasons for this error, please troubleshoot in the following order:

1) Torch.nn.module is not inherited when create a network. Please inherit torch.nn.module to create a network, and then use torch.jit.trace to generate a pt file.

2) RKNN-Toolkit2 1.4.0 and later version, it is recommended to use torch 1.6.0, 1.9.0 or 1.10.0.

5.7 Common issues of TensorFlow model conversion

5.7.1 Tensorflow1.x model error

Use the rknn.load_tensorflow to load the tensorflow1.x model if an error appears:

```
E load_tensorflow: Catch exception when loading tensorflow model: ./yolov3_mobilenetv2.pb!
E load_tensorflow: Traceback (most recent call last):
E load_tensorflow: File "/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/importer.py", line 427, in import_graph_def
```

```
E load_tensorflow: graph._c_graph, serialized, options) # pylint: disable=protected-access
E load_tensorflow: tensorflow.python.framework.errors_impl.InvalidArgumentError: Node 'MobilenetV2/expanded_conv/depthwise/BatchNorm/cond/Switch_1' expects to be colocated with unknown node 'MobilenetV2/expanded_conv/depthwise/BatchNorm/moving_mean'
E load_tensorflow: During handling of the above exception, another exception occurred:
E load_tensorflow: Traceback (most recent call last):
E load_tensorflow: File "rknn/api/rknn_base.py", line 990, in rknn.api.rknn_base.RKNNBase.load_tensorflow
E load_tensorflow: File "rknn/base/convertor/tensorflow2onnx/tf2onnx/convert.py", line 589, in rknn.base.convertor.tensorflow2onnx.tf2onnx.convert.from_graph_def
E load_tensorflow: File "rknn/base/convertor/tensorflow2onnx/tf2onnx/convert.py", line 590, in rknn.base.convertor.tensorflow2onnx.tf2onnx.convert.from_graph_def
E load_tensorflow: File "rknn/base/convertor/tensorflow2onnx/tf2onnx/convert.py", line 591, in rknn.base.convertor.tensorflow2onnx.tf2onnx.convert.from_graph_def
E load_tensorflow: File "rknn/base/convertor/tensorflow2onnx/tf2onnx/convert.py", line 592, in rknn.base.convertor.tensorflow2onnx.tf2onnx.convert.from_graph_def
E load_tensorflow: File "/usr/local/lib/python3.6/dist-packages/tensorflow/python/util/deprecation.py", line 507, in new_func
E load_tensorflow: return func(*args, **kwargs)
E load_tensorflow: File "/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/importer.py", line 431, in import_graph_def
E load_tensorflow: raise ValueError(str(e))
E load_tensorflow: ValueError: Node 'MobilenetV2/expanded_conv/depthwise/BatchNorm/cond/Switch_1' expects to be colocated with unknown node 'MobilenetV2/expanded_conv/depthwise/BatchNorm/moving_mean'
```

Suggest:

1) If the TensorFlow 1.x is installed currently, please try to install the TensorFlow 2.x.
2) Update RKNN-Toolkit2 / RKNPU2 to the latest version.

5.7.2 Erros like TransformGraph

An error is reported when the TensorFlow model is converted to RKNN:

```
--> Loading model
W load_tensorflow: inputs name should be a tensor name instead of node name
2022-04-14 12:59:42.674367: I tensorflow/tools/graph_transforms/transform_graph.cc:317] Applying strip_unused_nodes
2022-04-14 12:59:42.691060: I tensorflow/tools/graph_transforms/transform_graph.cc:317] Applying sort_by_execution_order
2022-04-14 12:59:42.700560: I tensorflow/tools/graph_transforms/transform_graph.cc:317] Applying fold_constants
2022-04-14 12:59:42.787221: I tensorflow/tools/graph_transforms/transform_graph.cc:317] Applying fold_batch_norms
2022-04-14 12:59:42.805569: I tensorflow/tools/graph_transforms/transform_graph.cc:317] Applying fold_old_batch_norms
Traceback (most recent call last):
  File "test.py", line 80, in <module>
    input_size_list=[[1, 368, 368, 3]])
  File "/usr/local/lib/python3.6/dist-packages/rknn/api/rknn.py", line 68, in load_tensorflow
    input_size_list=input_size_list, outputs=outputs)
  File "rknn/api/rknn_base.py", line 940, in rknn.api.rknn_base.RKNNBase.load_tensorflow
  File "/usr/local/lib/python3.6/dist-packages/tensorflow/tools/graph_transforms/__init__.py", line 51, in TransformGraph
    transforms_string, status)
  File "/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/errors_impl.py", line 548, in __exit__
    c_api.TF_GetCode(self.status.status))
tensorflow.python.framework.errors_impl.InvalidArgumentError: Beta input to batch norm has bad shape: [24]
```

Reason:

1) When the model directly calls by TensorFlow's native TransformGraph class for optimization, the above error will also be reported. (TransformGraph will also be called for optimization in RKNN-Toolkit2, so the same error will be reported).

2) The TensorFlow version that the model was generated is no longer compatible with the currently installed version.

Suggest:

Regenerate the model using TensorFlow version 1.14.0, or look for similar models from other frameworks.

5.7.3 Error "Shape must be rank 4 but is rank 0""

When loading the pb model:

```
rknn.load_tensorflow(tf_pb='./model.pb',
        inputs=["X","Y"],
        outputs=['generator/xs'],
        input_size_list=1, INPUT_SIZE, INPUT_SIZE, 3)
```

The error reported:

```
E load_tensorflow: Catch exception when loading tensorflow model: ./model.pb!
E load_tensorflow: Traceback (most recent call last):
E load_tensorflow: File "/usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/importer.py", line 427, in import_graph_def
E load_tensorflow: graph._c_graph, serialized, options) # pylint: disable=protected-access
E load_tensorflow: tensorflow.python.framework.errors_impl.InvalidArgumentError: Shape must be rank 4 but is
```

```
rank 0 for 'generator/conv2d_3/Conv2D' (op: 'Conv2D') with input shapes: [], [7,7,3,32].
```

The reason may be that this model is a multi-input model, and the input_size_list is not filled correctly. You can refer to the following usage in examples/functions/multi_input_test:

```
rknn.load_tensorflow(tf_pb='./conv_128.pb',
    inputs=['input1', 'input2', 'input3', 'input4'],
    outputs=['output'],
    input_size_list=1, 128, 128, 3], [1, 128, 128, 3], [1, 128, 128, 3], [1, 128, 128, 1)
```

5.8 Troubleshooting steps when loading model errors

First confirm whether the original deep learning framework can load the model and perform inference correctly.

Secondly, please upgrade RKNN-Toolkit2 to the latest version. If the model has a layer (or OP) that is not supported by RKNN-Toolkit2, by turning on the debug log switch, you can see which layer or OP is not supported by RKNN-Toolkit2 in the log.

If the first step fails, please check the original model for problems. If it still cannot be converted after upgrading to the latest version of RKNN-Toolkit2, or if some OPs does not supported, please report the version of the tool used and the detailed log when the conversion problem occurs to the Rockchip NPU development team.

5.9 Error “Please call rknn.config first”

Prompt when converting models:

```
E load_tflite: Please call rknn.config first!
W load_tflite: ======================= WARN(1) =======================
E rknn-toolkit2 version: 1.3.0-11912b58
E load_tflite: Catch exception when loading tflite model: point_history_classifier.tflite!
E load_tflite: Traceback (most recent call last):
E load_tflite:     File "rknn/api/rknn_base.py", line 1468, in rknn.api.rknn_base.RKNNBase.load_tflite
E load_tflite:     File "rknn/api/rknn_base.py", line 600, in rknn.api.rknn_base.RKNNBase._create_ir_and_inputs_meta
```

```
E load_tflite:      File "rknn/api/rknn_log.py", line 113, in rknn.api.rknn_log.RKNNLog.e
E load_tflite: ValueError: Please call rknn.config first!
```

The prompt “Please call rknn.config first” means that the rknn.config interface has not been called, just add a call to the rknn.config interface before loading model.

6 Model quantization problem

6.1 Model quantization overview

After the model is quantized, it will use a lower precision (such as int8/int16) to save the weight information of the model. After deployment, the memory usage of the model can be reduced and the model inference speed can be accelerated.

RKNN supports two kinds for model quantization:

- RKNN-Toolkit2 quantize the float model according to the calibration DataSet and generate the quantized RKNN model.
  - Support quantize type: int8, int16.
  - Quantize method: Post Training Static Quantization.
- Loading quantized model, which exported from the DL framework, RKNN-Toolkit2 can use the already exists quantization information to generate the quantized RKNN model.
  - Support framework: Pytoch(v1.9.0 or v1.10.0), TensorFlow, Tflite.
  - Support quantize type: int8, uint8.
  - Quantized method: Post Training Static Quantization, Quantization Aware Training(QAT).

6.2 QAT and PTQ quantization

- **Post training static quantization**

  Currently RKNN-Toolkit2 supported:

  - asymmetric_quantized-8（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:

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

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

Currently only supports the inverse quantization of i8, the calculation formula is as follows:

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

- **Quantization aware training(QAT)**

A model with quantized params/weights can be obtained through the Quantization-aware training. RKNN-Toolkit2 currently supports TensorFlow and PyTorch frameworks for QAT models. Please refer to the following link for the technical details of QAT:

TensorFlow:

https://www.tensorflow.org/model_optimization/guide/quantization/training

PyTorch:

https://pytorch.org/blog/introduction-to-quantization-on-pytorch/

This method requires to use the original framework to train (or fine tune) to obtain a quantized model, and then use RKNN-Toolkit2 to import the quantized model (you need to set do_quantization=False in the build setting during model conversion). At this time, RKNN-Toolkit2 will use the quantized parameters of the model, so theoretically there will be almost no loss of accuracy.

Note: RKNN-Toolkit2 also supports post-training quantization models of PyTorch, TensorFlow, and TensorFlow Lite. The model conversion method is the same as the quantization-aware training model. The do_quantization should be set to False when calling the build interface.

6.3 Quantization influence on model size

There are two scenarios. When the loaded model is the quantized model, do_quantization=False will use the quantization parameter of the model. 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.

6.4 What is the role of Dataset during RKNN quantization

In the quantization process of RKNN-Toolkit2, it is necessary to find the appropriate quantization parameters according to calibration Dataset data.

For this reason, the calibration Dataset as the representative subset from the training set or validation set is better than others (the subset only needs to be enough to cover the distribution of the data in the validation set). The recommended data number is between 50 and 200.

If the calibration Dataset is not selected properly, the quantized data distribution range will be much different from the validation set, and the quantification accuracy of the model will be poor.

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

Not required. RKNN-Toolkit2 automatically scales images. Howerver, 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.

6.6 Whether the Dataset needs to be modified according to the rknn_batch_size parameter

Unnecessary. The rknn_batch_size parameter of rknn.build only modifies the batch dimension of the last exported rknn model (changed from 1 to rknn_batch_size), and does not affect the process of the quantization stage, so the Datset can still be set in the way that batch is 1.

6.7 When the model is quantized, the program is killed or stuck after running for a period of time

In the process of model quantization, RKNN-Toolkit2 will apply for more system memory, which may cause the program to be killed or stuck.

Solution: Increase PC memory or Virtual memory.

7 Model conversion issues

7.1 Common conversion bugs

If encounter a conversion error similar to the following, it is likely due to a bug in the current version. You can try to update RKNN-Toolkit2 to the latest version.

7.1.1 "infer_shapes" similar error

```
(op_type:Mul, name:Where_2466_mul): Inferred elem type differs from existing elem type: (FLOAT) vs (INT64)
E build: Catch exception when building RKNN model!
E build: Traceback (most recent call last):
E build: File "rknn/api/rknn_base.py", line 1555, in rknn.api.rknn_base.RKNNBase.build
E build: File "rknn/api/graph_optimizer.py", line 5409, in rknn.api.graph_optimizer.GraphOptimizer.run
E build: File "rknn/api/graph_optimizer.py", line 5123, in rknn.api.graph_optimizer.GraphOptimizer._fuse_ops
E build: File "rknn/api/ir_graph.py", line 180, in rknn.api.ir_graph.IRGraph.rebuild
E build: File "rknn/api/ir_graph.py", line 140, in rknn.api.ir_graph.IRGraph._clean_model
E build: File "rknn/api/ir_graph.py", line 56, in rknn.api.ir_graph.IRGraph.infer_shapes
E build: File "/home/anaconda3/envs/rk2/lib/python3.6/site-packages/onnx/shape_inference.py", line 35, in infer_shapes
E build: inferred_model_str = C.infer_shapes(model_str, check_type)
E build: RuntimeError: Inferred elem type differs from existing elem type: (FLOAT) vs (INT64)
```

Or:

```
E build: Traceback (most recent call last):
E build: File "rknn/api/rknn_base.py", line 1643, in rknn.api.rknn_base.RKNNBase.build
E build: File "rknn/api/graph_optimizer.py", line 6256, in rknn.api.graph_optimizer.GraphOptimizer.fuse_ops
E build: File "rknn/api/ir_graph.py", line 285, in rknn.api.ir_graph.IRGraph.rebuild
E build: File "rknn/api/ir_graph.py", line 149, in rknn.api.ir_graph.IRGraph._clean_model
E build: File "rknn/api/ir_graph.py", line 62, in rknn.api.ir_graph.IRGraph.infer_shapes
E build: File "/usr/local/lib/python3.6/dist-packages/onnx/shape_inference.py", line 35, in infer_shapes
```

E build: inferred_model_str = C.infer_shapes(model_str, check_type)

E build: RuntimeError: Inferred shape and existing shape differ in rank: (0) vs (3)

Or:

```
(op_type:ReduceMax, name:ReduceMax_18): Inferred shape and existing shape differ in rank: (3) vs (0)
E build: Catch exception when building RKNN model!
E build: Traceback (most recent call last):
E build:   File "rknn/api/rknn_base.py", line 1638, in rknn.api.rknn_base.RKNNBase.build
E build:   File "rknn/api/graph_optimizer.py", line 5499, in rknn.api.graph_optimizer.GraphOptimizer.fold_constant
E build:   File "rknn/api/ir_graph.py", line 750, in rknn.api.ir_graph.IRGraph.make_model
E build:   File "rknn/api/ir_graph.py", line 62, in rknn.api.ir_graph.IRGraph.infer_shapes
E build:   File "/home/wenqiluo/anaconda3/envs/rknn/lib/python3.6/site-packages/onnx/shape_inference.py", line 35, in infer_shapes
E build:     inferred_model_str = C.infer_shapes(model_str, check_type)
E build: RuntimeError: Inferred shape and existing shape differ in rank: (3) vs (0)
```

7.1.2 “_p_fuse_two_mul” similar error

E build: Catch exception when building RKNN model!

E build: Traceback (most recent call last):

E build: File "rknn/api/rknn_base.py", line 1643, in rknn.api.rknn_base.RKNNBase.build

E build: File "rknn/api/graph_optimizer.py", line 6197, in rknn.api.graph_optimizer.GraphOptimizer.fuse_ops

E build: File "rknn/api/graph_optimizer.py", line 204, in rknn.api.graph_optimizer._p_fuse_two_mul

E build: ValueError: non-broadcastable output operand with shape () doesn't match the broadcast shape (3,2)

7.1.3 “Segmentation fault” similar error

Such as picodet model conversion error:

```
root@8d7572b2aa6b:/test/pytorch/picodet# python test_hpc203.py
W __init__: Verbose file path is invalid, debug info will not dump to file.
--> config model
pre-process config done
--> Loading model
Load pytorch model done
--> Building model
W build: The output shape [1, 6] of model is wrong, fix it to [11, 6]!
I _fold_constant remove nodes = ['Shape_0', 'Gather_4', 'Shape_1', 'Gather_6', 'Unsqueeze_0', 'Concat_8', 'Cast_3', 'ReduceMin_0', 'Unsqueez
Segmentation fault (core dumped)
```

7.1.4 “_p_fuse_mul_into_conv” similar error

```
I     remove_invalid_reshape: remove node = ['Mul_796_0_expand0', 'Mul_798_0_expand0']
I     fuse_two_reshape: remove node = ['Mul_798_0_expand1']
E build: Catch exception when building RKNN model!
E build: Traceback (most recent call last):
E build:   File "rknn/api/rknn_base.py", line 1643, in rknn.api.rknn_base.RKNNBase.build
E build:   File "rknn/api/graph_optimizer.py", line 6197, in rknn.api.graph_optimizer.GraphOptimizer.fuse_ops
E build:   File "rknn/api/graph_optimizer.py", line 344, in rknn.api.graph_optimizer._p_fuse_mul_into_conv
E build: ValueError: non-broadcastable output operand with shape (1,258,1,256) doesn't match the broadcast shape (80256,258,1,256)
```

7.2 How to judge whether the OP is supported in RKNN for each framework

Convert the model directly. If it is not supported, there will be relevant prompts.

7.3 It prompts that the Expand is not supported

Suggest:

1) The new version already supports Expand of CPU, you can try update RKNN-Toolkit2 / RKNPU2 to the latest version.

2) Modify the model and use the Repeat instead of the Expand.

7.4 Tips “Meet unsupported dims in reducesum”

Model conversion appears “Meet unsupported dims in reducesum, dims: 6”, as follows:

```
D RKNN: [14:54:19.434] >>>>>> start: N4rknn17RKNNInitCastConstE
D RKNN: [14:54:19.434] <<<<<<<< end: N4rknn17RKNNInitCastConstE
D RKNN: [14:54:19.434] >>>>>> start: N4rknn20RKNNMultiSurfacePassE
D RKNN: [14:54:19.434] <<<<<<<< end: N4rknn20RKNNMultiSurfacePassE
D RKNN: [14:54:19.434] >>>>>> start: N4rknn14RKNNTilingPassE
D RKNN: [14:54:19.434] <<<<<<<< end: N4rknn14RKNNTilingPassE
D RKNN: [14:54:19.434] >>>>>> start: N4rknn23RKNNProfileAnalysisPassE
D RKNN: [14:54:19.434] <<<<<<<< end: N4rknn23RKNNProfileAnalysisPassE
D RKNN: [14:54:19.434] >>>>>> start: OpEmit
E RKNN: [14:54:19.438] Meet unsupported dims in reducesum, dims: 6
Aborted (core dumped)
```

At present, RKNN does not support 6-dimensional OP. In most cases, it only supports 4-dimensional. Other dimensions will have some limitations.

7.5 Conversion error by NonMaxSuppression / TopK

1) Post-processing Ops such as NonMaxSuppression / TopK, RKNN currently does not support.

2) The post-processing subgraph part of the graph can be removed, such as:

rknn.load_onnx(model='picodet_xxx.onnx', outputs=['concat_4.tmp_0', 'tmp_16'])

3) The removed subgraphs are processed separately on the CPU side.

7.6 "invalid expand shape" similar error

For example, the following error occurs when rvm_mobilenetv3_fp32.onnx is converted:

```
2022-08-17 13:36:50.477462084 [E:onnxruntime:, sequential_executor.cc:333 Execute] Non-zero status code returned while running Expand node. Name:'Expand_294' Status Message: invalid expand shape
E build: Catch exception when building RKNN model!
E build: Traceback (most recent call last):
E build: File "rknn/api/rknn_base.py", line 1638, in rknn.api.rknn_base.RKNNBase.build
E build: File "rknn/api/graph_optimizer.py", line 5529, in rknn.api.graph_optimizer.GraphOptimizer.fold_constant
E build: File "rknn/api/session.py", line 69, in rknn.api.session.Session.run
E build: File "/home/cx/work/tools/Anaconda3/envs/rknn/lib/python3.8/site-packages/onnxruntime/capi/onnxruntime_inference_collection.py", line 124, in run
E build: return self._sess.run(output_names, input_feed, run_options)
E build: onnxruntime.capi.onnxruntime_pybind11_state.InvalidArgument: [ONNXRuntimeError] : 2 : INVALID_ARGUMENT : Non-zero status code returned while running Expand node. Name:'Expand_294' Status Message: invalid expand shape
```

Reason:

1) Because the input value of downsample_ratio will change the size of the feature in the middle of the model, so this kind of graph is essentially a dynamic graph.

2) RKNN does not currently support dynamic graphs.

Suggest:

downsample_ratio can be fixed as a constant (don't as a variable), so that RKNN can support it.

7.7 "output_tensor_type" similar error

The following error is reported during model conversion:

```
W __init__: rknn-toolkit2 version: v1.2.3-b4-ac69c24
--> Config model
Traceback (most recent call last):
  File "test.py", line 240, in <module>
    rknn.config(mean_values=[[0, 0, 0]], std_values=[[255, 255, 255]], output_te
nsor_type='int8')
TypeError: config() got an unexpected keyword argument 'output_tensor_type'
```

The reason is that the new version of RKNN-Toolkit2 no longer has the output_tensor_type parameter, so delete the output_tensor_type configuration parameter.

7.8 "mean_values" similar error

Set mean/std to:

rknn.config(mean_values=[128, 128, 128], std_values=[128, 128, 128])

An error is reported:

```
--> Loading model
transpose_input for input_1: shape must be rank 4, ignored
E load_tflite: The len of mean_values ([128, 128, 128]) for input 0 is wrong, expect 32!
```

The reason is that the input of the model may not be 3-channel image data (for example, the input shape is 1x32, not image data). At this time:

1) Need to set mean_values / mean_values according to the number of input channels.

2) If the model does not need to specify mean/std, rknn.config can not set mean_values / std_values (mean/std is generally only valid for image input).

7.9 An error is reported when the model has more than 4 dimensions (5 or 6 dimensions)

When there are more than 4-dimensional Ops in the model (such as 5-dimensional or 6-dimensional), the following error will be reported:

```
E build: Catch exception when building RKNN model!
E build: Traceback (most recent call last):
E build:   File "rknn/api/rknn_base.py", line 1580, in rknn.api.rknn_base.RKNNBa
se.build
E build:   File "rknn/api/rknn_base.py", line 341, in rknn.api.rknn_base.RKNNBas
e._generate_rknn
E build:   File "rknn/api/rknn_base.py", line 307, in rknn.api.rknn_base.RKNNBas
e._build_rknn
E build: IndexError: vector::_M_range_check: __n (which is 4) >= this->size() (w
hich is 4)
```

RKNN currently does not support OPs with dimensions above 4, and these nodes can be removed manually.

7.10 Does Toolkit2 support dynamic convolution?

RKNN-Toolkit2 and RKNPU2 do not currently support dynamic convolution.

7.11 “Not support input data type 'float16'” error

The weight type of the model trained by Pytorch is float16, and the following error occurs when converting to RKNN:

```
--> Building model
E build: Not support input data type 'float16'!
W build: ==================== WARN(3) ====================
E rknn-toolkit2 version: 1.3.0-11912b58
E build: Catch exception when building RKNN model!
E build: Traceback (most recent call last):
E build:   File "rknn/api/rknn_base.py", line 1638, in rknn.api.rknn_base.RKNNBase.build
E build:   File "rknn/api/graph_optimizer.py", line 5524, in rknn.api.graph_optimizer.Gra
E build:   File "rknn/api/load_checker.py", line 63, in rknn.api.load_checker.create_rand
E build:   File "rknn/api/rknn_log.py", line 113, in rknn.api.rknn_log.RKNNLog.e
E build: ValueError: Not support input data type 'float16'!
```

At present, RKNN-Toolkit2 does not support the Pytorch model of float16 weight type, and needs to be changed to float32.

7.12 Dynamic graph related error

When converting a model, if an error similar to the following occurs:

```
E build: ValueError: The Op of 'NonZero' is not support! it will cause the graph to be a dynamic graph!
```

It means that the OP will cause the model to be a dynamic graph, and the model needs to be manually modified, replaced with other OPs or removed.

8 Description of simulator inference, inference with board-connected, and on-board inference

8.1 Basic Instructions

Simulator inference: In the case that the PC side is not connected to the board, it is allowed to call the simulator inference model and obtain the inference results in the Linux x86_64 environment (the inference results may not be consistent with the board-connected or on-board end).

Inference with board-connected: When the board is connected to the PC [for connection, please refer to Section 3.22 of the Rockchip_User_Guide_RKNN_Toolkit2_CN document], use the RKNN-Toolkit2 Python API to inference model and obtain the inference results.

On-board inference: Refers to using the RKNPU2's C API interface to obtain inference results.

8.2 The inference result of the simulator is inconsistent with the inference result on board

When this happens, it may mean that the board-side results are incorrect.

Due to hardware and driver differences, the simulator is not guaranteed to get the exact same results as the board. But if the difference is too large, it is likely to be caused by a bug in the on-board driver, and the problem can be fed back to RK's NPU team for analysis and debugging.

8.3 The working method of inference with board-connected

When using inference with board-connected, RKNN-Toolkit2 will start a process of npu_transfer_proxy in the background. This process will communicate with the rknn-server on the board side. During communication, the model and model input will be transferred from the PC side to the board side, and then RKNPU2 C API will be called to performs model inference, and then sends the inference results back to the PC.

8.4 Differences between the results of inference with board-connected and inference on board

During the inference with board-connected, the C API interface of the RKNPU2 is actually called, so theoretically, when the inference result of board-connected is correct, the inference result of C API is also correct. If there is a difference between the two, please confirm whether the input preprocessing, data type, data layout format (NCHW, NHWC) and C API parameter settings are consistent.

It should be pointed out that if the difference is small, the difference is only in the 3 digits after the decimal point and the 4 digits after the decimal point, which is a normal phenomenon, and this difference may occur in the steps of data transmission and reception, data type conversion, etc.

8.5 Inference on-board is faster than inference with board-connected.

Due to the extra data copying and transmission process of board inference, the performance of inference with board-connected is not as good as the inference performance of RKNPU2 C API on board.

8.6 Getting more detail log

When connecting board debugging and connecting board inference, the initialization and inference of the model are mainly completed on the development board, and the log information is mainly generated on the board end.

In order to obtain specific board-side debugging information, you can enter the development board operating system through the serial port. Then execute the following two commands to set the environment variables for obtaining logs. Keep the serial port window not closed, and then perform board debugging. At this time, the error message on the board side will be displayed on the serial port window:

```
export RKNN_LOG_LEVEL=5
```

```
restart_rknn.sh
```

9 Common issue about model evaluation

9.1 Quantized model accuracy is not as good as expected

Refer to chapter 3.6 of the document "Rockchip_User_Guide_RKNN_Toolkit2_CN.pdf"

9.2 Which frameworks of quantized model are currently supported by the RKNN-Toolkit2

RKNN-Toolkit2 1.4 and later supports quantized models of TensorFlow, TensorFlow Lite and PyTorch.

9.3 Failed to connect to the device during inference with board-connected

The following error occurs during rknn.accuracy_analysis:

```
E accuracy_analysis: Connect to Device Failure (-1)
E accuracy_analysis: Catch exception when init runtime!
E accuracy_analysis: Traceback (most recent call last):
E accuracy_analysis: File "rknn/api/rknn_base.py", line 2001, in rknn.api.rknn_base.RKNNBase.init_runtime
E accuracy_analysis: File "rknn/api/rknn_runtime.py", line 194, in rknn.api.rknn_runtime.RKNNRuntime.__init__
E accuracy_analysis: File "rknn/api/rknn_platform.py", line 331, in rknn.api.rknn_platform.start_ntp_or_adb
```

Or the following error occurs when rknn.inference:

```
I target set by user is: rk3568
I Starting ntp or adb, target is RK3568
I Device [0c6a9900ef4871e1] not found in ntb device list.
I Start adb...
I Connect to Device success!
I NPUTransfer: Starting NPU Transfer Client, Transfer version 2.1.0 (b5861e7@2020-11-23T11:50:36)
D NPUTransfer: Transfer spec = local:transfer_proxy
D NPUTransfer: ERROR: socket read fd = 3, n = -1: Connection reset by peer
D NPUTransfer: Transfer client closed, fd = 3
E RKNNAPI: rknn_init,  server connect fail!  ret = -9(ERROR_PIPE)!
E init_runtime: Catch exception when init runtime!
E init_runtime: Traceback (most recent call last):
E init_runtime:   File "rknn/api/rknn_base.py", line 2011, in rknn.api.rknn_base.RKNNBase.init_runtime
E init_runtime:   File "rknn/api/rknn_runtime.py", line 361, in rknn.api.rknn_runtime.RKNNRuntime.build_graph
E init_runtime: Exception: RKNN init failed. error code: RKNN_ERR_DEVICE_UNAVAILABLE
```

The reason may be that the rknn_server is not enabled on the board side. You need to run the rknn_server according to the following instructions to connect the board normally:

https://github.com/rockchip-linux/rknpu2/blob/master/rknn_server_proxy.md

9.4 The rknn_init returns RKNN_ERROR_MODEL_INVALID

The error log is as follows:

```
E RKNNAPI: rknn_init,    msg_load_ack fail, ack = 1(ACK_FAIL), expect
0(ACK_SUCC)!
D NPUTransfer: Transfer client closed, fd = 4
E init_runtime: Catch exception when init runtime!
E init_runtime: Traceback (most recent call last):
E init_runtime:      File "rknn/api/rknn_base.py", line 2011, in
rknn.api.rknn_base.RKNNBase.init_runtime
E init_runtime:      File "rknn/api/rknn_runtime.py", line 361, in
rknn.api.rknn_runtime.RKNNRuntime.build_graph
E init_runtime: Exception: RKNN init failed. error code:
RKNN_ERR_MODEL_INVALID
```

There are some situations in which this error occurs:

1) Different versions of the RKNN-Toolkit2 and drivers have a corresponding relationship. It is recommended to upgrade the RKNN-Toolkit2 / RKNPU2 and the firmware of the board to the latest version.

2) The target_platform is not set correctly. For example, when the target_platform in the config interface is not set, the generated RKNN model can only be run on RK3566/RK3568, but not on RK3588/RV1103/RV1106/RK3562. If you want to run on RK3588/RV1103/RV1106/RK3562, you need to set target_platform when calling the config interface.

3) If the problem occurs when inferring in a Docker container, it may be because the npu_transfer_proxy process on the host machine is not terminated, resulting in abnormal communication. please exit the Docker container first, terminate the npu_transfer_proxy process on the host, and then enter the container to execute the inference script.

4) It may also be that the model transferred from RKNN-Toolkit2 has some problems. At this time, you can obtain the following information and feed it back to the Rockchip NPU team: If the PC is connected to the development board for debugging, or the model is running on the

development board, you can connect the serial port to the development board, and then set the environment variable RKNN_LOG_LEVEL=5, then execute restart_rknn.sh, and then re-run the program. Record the detailed log on the board.

9.5 The rknn_init returns RKNN_ERR_DEVICE_UNAVAILABLE

The error log is as follows:

```
E RKNNAPI: rknn_init,  msg_ioctl_ack fail, data_len = 104985, expect 103961!
D NPUTransfer: Transfer client closed, fd = 3
E init_runtime: Catch exception when init runtime!
E init_runtime: Traceback (most recent call last):
E init_runtime:   File "rknn/api/rknn_base.py", line 1916, in rknn.api.rknn_base.RKNNBase.init_runtime
E init_runtime:   File "rknn/api/rknn_runtime.py", line 360, in rknn.api.rknn_runtime.RKNNRuntime.build_graph
E init_runtime: Exception: RKNN init failed. error code: RKNN_ERR_DEVICE_UNAVAILABLE
```

Please check it out as follows:

Make sure that the RKNN-Toolkit2 / RKNPU2 and the firmware of devices have been upgraded to the latest version.

The version of these components is queried on RK3566 as follows:

```
# execute these commands on RK3566 (Android):
dmesg | grep rknpu       # query NPU driver version
strings /vendor/bin/rknn_server |grep build          # query rknn_server version
strings /vendor/lib64/librknnrt.so | grep version       # query librknnrt version

# execute these commands on RK3566 (Linux):
dmesg | grep rknpu       # query NPU driver version
strings /usr/bin/rknn_server |grep build       # query rknn_server version
strings /usr/lib/librknnrt.so | grep version       # query librknnrt version
```

The version information can also be queried through the get_sdk_version interface.

1) Make sure the “adb devices” command or rknn.list_devices() can get the device, and the target and device_id settings of rknn.init_runtime() are correct.

2) If you are using RV1103/RV1106, does not support board-connected.

9.6 “Invalid RKNN format” error in Runtime

The following error occurs on the Runtime:

```
rknn_init error ret=-1
root@firefly:/home/firefly/rknn_yolov5_demo/build/build_linux_aarch64# ./rknn_yolov5_demo yolov5s.rknn bus.jpg
post process config: box_conf_threshold = 0.50, nms_threshold = 0.60
Read bus.jpg ...
img width = 640, img height = 640
Loading mode...
E RKNN: [06:28:39.048] parseRKNN from buffer: Invalid RKNN format!
E RKNN: [06:28:39.049] rknn_init, load model failed!
rknn_init error ret=-1
root@firefly:/home/firefly/rknn_yolov5_demo/build/build_linux_aarch64#
```

Reason:

1) It may be that the target_platform during model conversion is set incorrect, or is not set (if not set, the default is rk3566).

2) Runtime version is not compatible with RKNN-Toolkit2.

Suggest:

1) Set the correct target_platform.

2) RKNN-Toolkit2 and Runtime should be updated to the same version.

9.7 rknn.inference is slower than rknn.eval_perf

Because rknn.inference uses PC & adb for inference with board-connected, there is some fixed data transmission overhead, which is inconsistent with the theoretical speed of rknn.eval_perf.

For a more realistic frame rate, it is recommended to use the RKNPU2 C API directly on the board for testing.

9.8 Does the rknn.inference support multi-batch inputs

The RKNN-Toolkit2 needs to be upgraded to version 1.4.0 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_Toolkit2_CN” document.

In addition, when rknn_batch_size is greater than 1 (e.g. equal 4), the inference code in python:

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

need modify to:

```
```

```
img = np.expand_dims(img, 0)
img = np.concatenate((img, img, img, img), axis=0)
outputs = rknn.inference(inputs=[img])
```

For a complete example, please refer to:

examples/functions/batch_size/

9.9 Run with multi RKNN models

When running two or more models, you need to create multiple RKNN objects. One RKNN object corresponds to a model, which is similar to a context. Each model initializes the model in its own context, makes inferences, and obtains inference results without interfering with each other. These models are inferred serially on the NPU.

9.10 Model inference takes a very long time, and the results obtained are wrong

If the inference takes more than 20s and the result is wrong, this is usually a NPU Hang bug in the NPU. If you encounter this problem, you can try to update the RKNN-Toolkit2 / RKNPU2 to the latest version.

9.11 The inference time of the model without quantization is much longer than the model with quantization. Is this normal?

It is normal. NPU floating-point computing power is relatively weak, and for the quantized model, NPU has been optimized. So the performance of the quantized model will be much better than that of the floating-point model. On NPU, it is recommended to use a quantized model.

9.12 Inference result of the board is wrong when input is 3-dimension

When the input of the model is 3-dimensional, if the inference result of the Simulator is correct, but the inference result of the board-connected is wrong. The reason may be that the input 3-dimensional support of the current NPU is not perfect, and the 3-dimensional support will be

improved later.

Suggest:

1) Change the input of the model to 4D.

2) Try to update the RKNN-Toolkit2 / RKNPU2 to the latest version.

9.13 Inference result of the board is wrong, and inconsistent every time

The results of the Simulator are correct, and the results are consistent every time. but the results are wrong and inconsistent every time when inference with board-connected. This kind of problem may be caused by a bug in the npu kernel driver on the board. In this case, the NPU kernel driver on the board needs to be updated, and the latest RKNN-Toolkit2 / RKNPU2 needs to be updated together.

9.14 When the model has more Resize OPs, the problem of accuracy decline occurs

When there are more Resize OPs in the ONNX model, the accuracy decreases after converting to RKNN. Possible reasons are:

1) The decrease in accuracy is because the NPU does not currently support hardware-level Resize (will be supported in the future), and the RKNN-Toolki2 will convert Reszie to ConvTranspose, which will cause a little loss of accuracy.

2) If the model has multiple Resizes in series, it may accumulate too much error and cause the accuracy to drop more.

Suggest:

1) At present, try to avoid the use of Resize (such as changing Resize to ConvTranspose and then re-training)

2) The parameter optimization_level=2 can be added to rknn.config. At this time, the Resize Op will use the CPU, and the precision will not drop, but it will cause performance degradation.

9.15 Inference results are all 'nan' when do_quantization = False

When do_quantization in the build interface is set to True, the inference results are not abnormal, but when it is set to False, the inference results become 'nan'. The reason may be that when do_quantization=False, the operation type of the RKNN model is fp16, but the output range of the intermediate layer (such as convolution) of the model may exceed the range of fp16 (65536) (such as -51597~75642).

Suggest:

When training, it is necessary to ensure that the output of the intermediate layer does not exceed the range of fp16 (generally normalized by adding a BN layer)

9.16 The results of QAT model and RKNN model are inconsistent

Use QAT to train a classification model under the Pytroch framework and converted it to an RKNN model. I used Pytorch and RKNN to infer the model respectively, and found that the results were different. The reason may be that the inference of Pytorch did not set engine='qnnpack', because The inference of RKNN is closer to qnnpack.

9.17 How to test the memory usage when the model is running

You can use the rknn.eval_memory interface, and there is a 'total' in the output log, which is the total occupied size.

9.18 The difference when perf_debug is turn on or turn off

When perf_debug is turned on, in order to collect the information of each layer, some debugging code will be added, and some parallel mechanisms may be disabled, so the time-consuming is more than when perf_debug=False.

The main reason for turning on perf_debug is to see if there are more time-consuming layers in the model, and to design an optimization scheme based on this.

9.19 After restarting the docker, the inference was stuck in the initialization environment stage?

This is because npu_transfer_proxy is similar to an abnormal exit state when docker restarts, so that the rknn_server on the board cannot detect that the host connection has been disconnected. At this time, it is necessary to restart the board and reset the connection state of rknn_server.

10 RKNN convolution acceleration tips

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

Here are some suggestions from us:

- Optimal Kernel Size is 3x3

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

In addition, in the case of group convolution, the value of group should not be set too large, and the input channels of weight should be aligned to 16 as much as possible.

- 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. The NPU 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.

- Remove redundant OP

When there is an invalid OP, the OP can be removed, such as Reshape with the same input and output shapes. When there are parallel OPs with consistent operations, one of them can be removed, such as two Relus from the same input, etc.

- OP fusion design

When there are consecutive OPs that can be fused, try to fuse them together (especially for quantized models of QAT/PTQ), such as:

1） When Mul+Gemm, Mul can be fused into Gemm.

2） When Gemm+Add, Add can be fused into Gemm.

3） When Conv+BN, BN can be fused into Conv.

4） When ConvTranspose+BN, BN can be fused into ConvTranspose.

5） When Conv+Add, Add can be fused into Conv.

6） When Conv+Mul, Mul can be fused into Conv.

7） When Mul+Conv, Mul can be fused into Conv.

8） When Add+Add, two Adds can be merged.

9） When Mul+Mul, two Muls can be merged.

10） When Mul+Add, Mul and Add can be converted to BN.

11） When MatMul+Add, MatMul and Add can be converted to Gemm.

- Resize improvements

If the model has Resize, because the NPU has weak support for Resize, it is recommended to convert Resize to ConvTranspose before training.

- Concat and Split alignment requirements

Try to ensure that the channel dimension of each input of Concat or the channel dimension of each output of Split is 8-aligned under RK3566/RK3568, and 16-aligned under RK3588/RV1103/RV1106/RK3562.

- 3D convolution support

At present, RKNN does not support 3D convolution, and it is necessary to replace the 3D convolution with the equivalent 2D convolution.

- Keepdims parameter

Try to ensure that the keepdims parameter of ReduceMean, ReduceMax, ReduceMin, ReduceSum, ArgMin, and ArgMax is 1.

- Non-4D OP support

At present, RKNN only supports 4-dimensional OP well, and other dimensions (such as 2/3/5 dimensions) are not fully supported, so it is necessary to try to change these Non-4D OP to 4D.

- OP Limit

Most OPs will have some restrictions.

For the support of OPs under each framework, please refer to:

https://github.com/rockchip-linux/rknn-toolkit2/blob/master/doc/RKNNToolKit2_OP_Support.md

For more detailed OP restrictions, you can refer to:

https://github.com/rockchip-linux/rknpu2/tree/master/doc/RKNN_Compiler_Support_Operat

or_List.pdf

11 Appendix

11.1 Reference documents

OP support list:

RKNNToolKit2_OP_Support.md

Quick start manual:

Rockchip_Quick_Start_RKNN_Toolkit2_EN.pdf

RKNN-Toolkit2 manual:

Rockchip_User_Guide_RKNN_Toolkit2_EN.pdf

Trouble shooting manual:

Rockchip_Trouble_Shooting_RKNN_Toolkit2_EN.pdf

The above documents can be found in the 'doc' directory. You can also visit the following link to view: https://github.com/rockchip-linux/rknn-toolkit2/tree/master/doc

11.2 Issue feedback

All the issue can be feedback via the follow ways:

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

Note: Redmine account can only be registered by an authorized salesperson. If your development board is from the third-party manufacturer, please contact them to report the issue.