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

RKNN-Toolkit Visualization User Guide

(Technology Department, Graphic Display Platform Center)

Mark:	Version	V0.1.0
[ ] Editing	Author	Chen Hao
[√] Released	Completed Date	2019-11-25
	Reviewer	Randall
	Reviewed Date	2019-11-25

福州瑞芯微电子股份有限公司

Fuzhou Rockchip Electronics Co., Ltd

Revision History

Version	Modifier	Date	Modify description	Reviewer
v0.1.0	ChenHao	2019-11-25	Initial version	Randall

Table of Contents

1 Overview

This function presents various functions of RKNN-Toolkit in the form of a graphical interface, simplifying the user's operation steps. Users can complete model conversion and inference by filling out forms and clicking function buttons, and no need to write scripts manually. The following features are supported now:

1） Model conversion: support to convert TensorFlow、TensorFlow Lite、MXNet、ONNX、Darknet、Pytorch、Caffe、Keras model to RKNN model(Keras is not supported now), support RKNN model import/export, which can be used on hardware platform later. Multiple input models are not supported now.

2） Quantization: support to convert float model to quantization model, currently support quantized methods including asymmetric quantization (asymmetric_quantized-u8) and dynamic fixed point quantization (dynamic_fixed_point-8 and dynamic_fixed_point-16) and hybrid quantization.

3） Model inference: able to simulate running model on PC and obtain the inference results. Also able to run model on specific hardware platform RK3399Pro (or RK3399Pro Linux development board), RK1808, TB-RK1808 AI Compute Stick and obtain the inference results.

4） Performance evaluation: able to simulate running on PC and obtain the total time consumption and each layer's time consumption of the model. Also able to run model with on-line debugging method on specific hardware platform RK3399Pro, RK1808, TB-RK1808 AI Compute Stick or directly run on RK3399Pro Linux development board to obtain the total time consumption and each layer's time consumption when the model runs completely once on the hardware.

5） Memory evaluation: Evaluate system and NPU memory consumption at runtime of the model. It can obtain the memory usage through on-line debugging method when the model is running on specific hardware platform such as RK3399Pro, RK1808, TB-RK1808 AI Compute Stick or RK3399Pro Linux development board.

6） Model pre-compilation: with pre-compilation techniques, model loading time can be reduced, and for some models, model size can also be reduced. However, the pre-compiled RKNN model can only be run on a hardware platform with an NPU, and this feature is currently only supported by the x86_64 Ubuntu platform.

2 Requirements/Dependencies

This tool only supports running on the Ubuntu、Windows and MacOS operating system. For system dependencies, please refer to the **2 Requirements/Dependencies** section of 《Rockchip_User_Guide_RKNN_Toolkit_EN.pdf》.

2.1 Installation

For the installation method, please refer to the **3.1 installation** section of 《Rockchip_User_Guide_RKNN_Toolkit_EN.pdf》.

2.2 Startup method

1) You can open a window by entering the following command in the environment where the whl package is installed.

   python -m rknn.bin.visualization

2) If you want to open a new window again, please open a new terminal and type python -m rknn.bin.visualization in the new terminal. (You need to wait until the first window is initialized before open the second, third or more window).

2.3 Instructions

2.3.1 Home

After starting the visualization, the home page is shown in Figure 1. The function is as follows:

The function of icons of TensorFlow, TensorFlow Lite, MXNet, ONNX, Darknet, Pytorch, Caffe is to convert the original model to the RKNN model, which can be used on hardware platform later. (Keras is not supported now)

The function of RKNN icon is RKNN model evaluation, supporting model visualization, model inference, performance evaluation and memory usage evaluation.

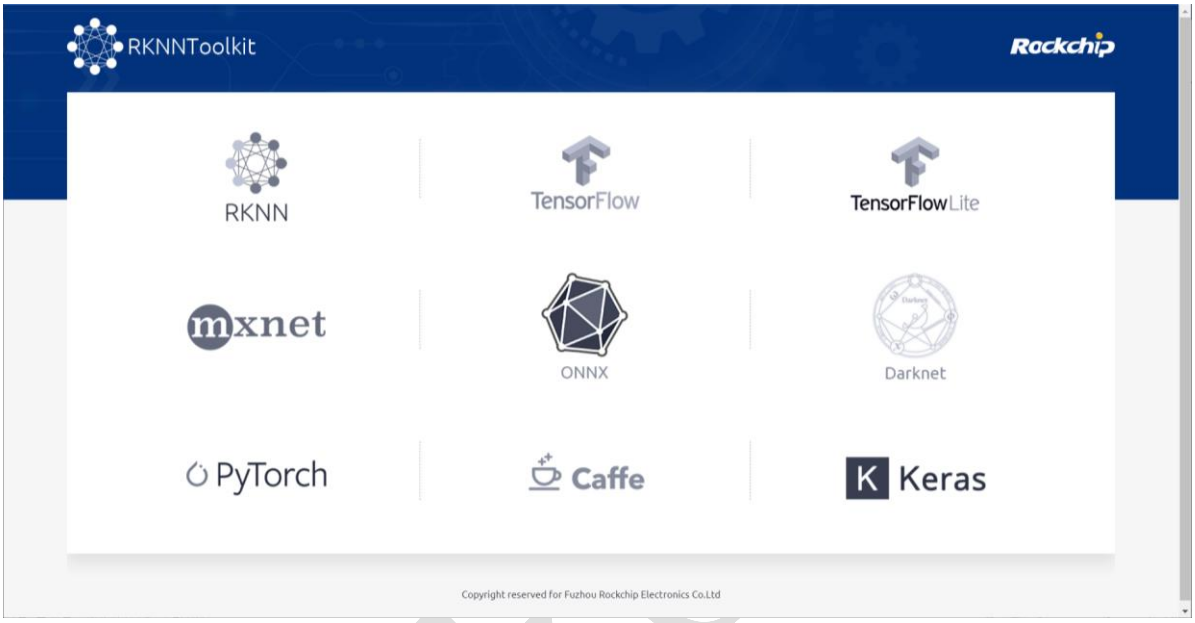

Figure 1 Visualization home page

2.3.2 RKNN

The function of RKNN icon is RKNN model evaluation, supporting model visualization, model inference, performance evaluation and memory usage evaluation.

First select the RKNN model which you want to evaluate, and click the Next Step icon to go to the RKNN model visualization page.

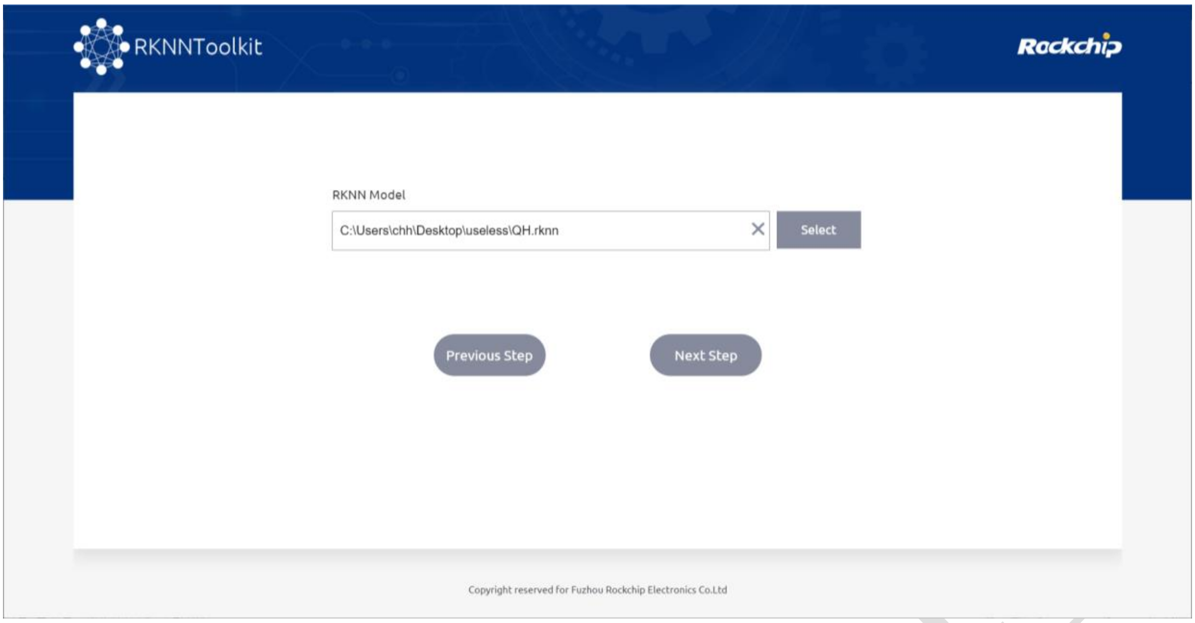

Figure 2 RKNN model selection

The visualization page shows the details of each layer of the RKNN model (including layer names and parameters). If the current window only displays partial information of the model, you can drag the model or scroll up/down the mouse to zoom in/out the image to see the other parts of the model. Dark blue is the quantized layer and light blue is the unquantized layer. After viewing the model, you can choose model inference, performance evaluation or memory usage evaluation to go to the next page.

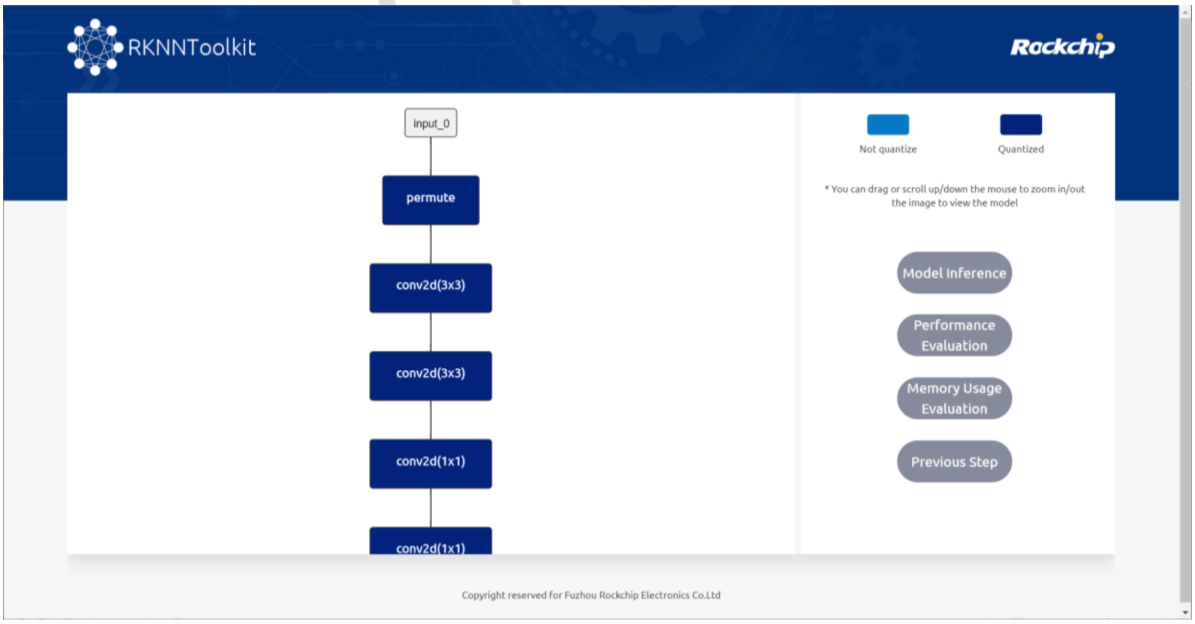

Figure 3 RKNN model visualization

The function of this page is model inference, performance evaluation and memory usage evaluation.

The function of each option is as follows:

- **Target：**

  The platforms to be evaluated, supporting simulator, RK1808 and RK3399PRO.

- **Device ID：**

  The device ID number of RK1808 and RK3399PRO, or none if the device is not found. This option is automatically hidden when the target is simulator.

- **Select Image：**

  Select the image which you want to evaluate. If the selected image size is smaller than the model input size, an exception will be throwed; if the selected image size is larger than the model input size, it will be cropped from the upper left corner according to the model input size before evaluated.

- **The Directory To Save The Result：**

  The results of model inference, performance evaluation and memory usage evaluation will be saved in this directory. The result of model inference will be saved as a npy file, and the results of performance evaluation and memory usage evaluation will be saved as a txt file.

- **Whether To Get Performance Details For Each Layer：**

  If set as True, it will print the performance information of each layer, otherwise it will only get the total running time of the model. This option doesn't take effect if the target is simulator.

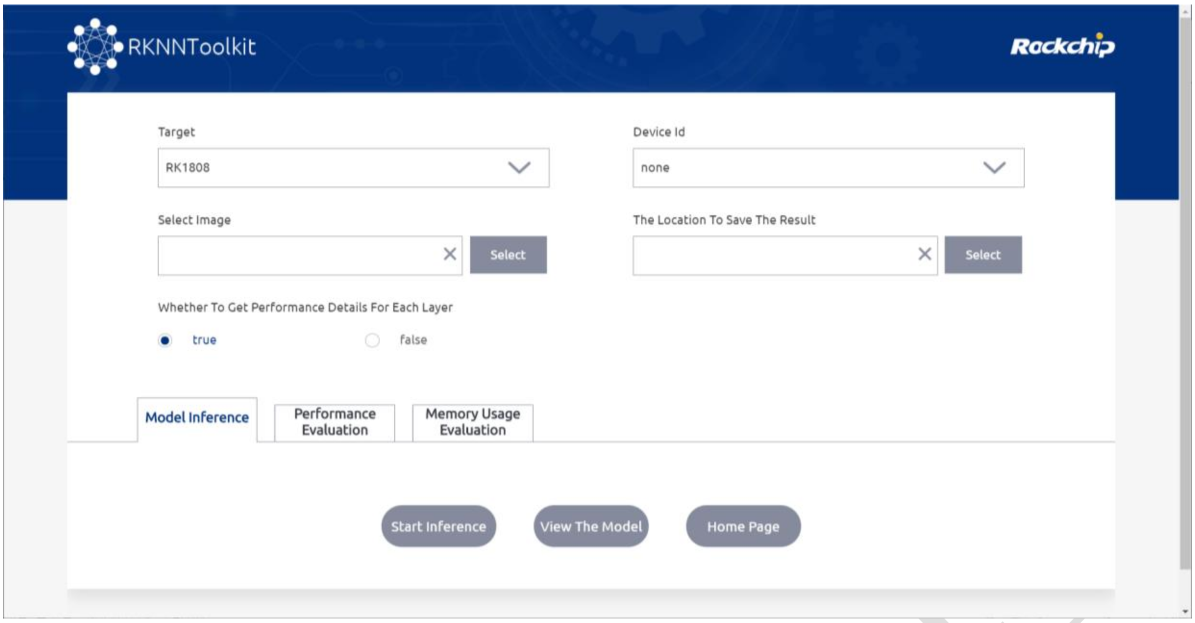

Figure 4 The page of RKNN model evaluation

Select the model inference function, the model will infer the image and save the inference result as an npy file.

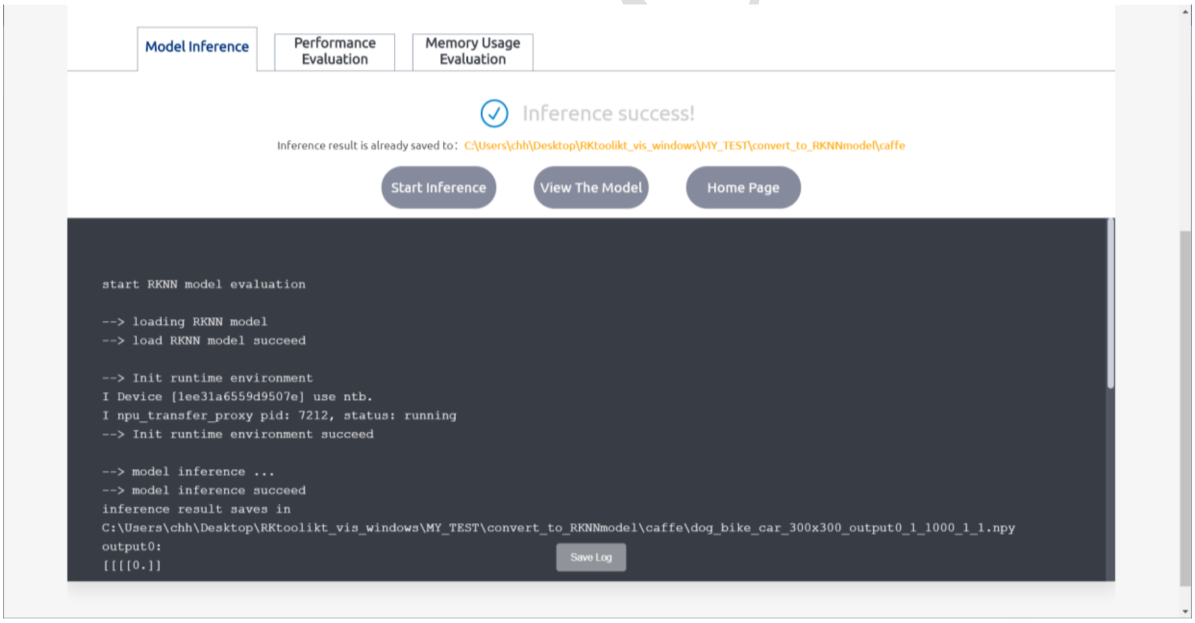

Figure 5 RKNN model inference

Select the performance evaluation function, it will print the detail performance data of the model and save the performance evaluation results as a txt file.

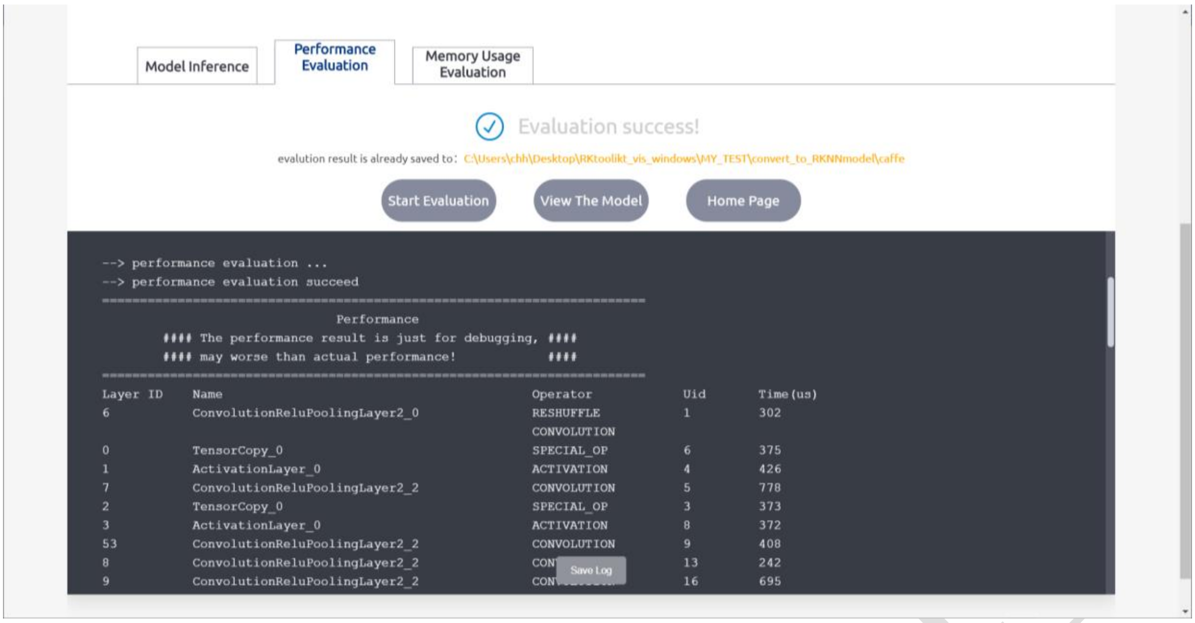

Figure 6 RKNN performance evaluation

Select the memory usage evaluation function, it will print the memory usage of the model and save the memory usage results as a txt file.

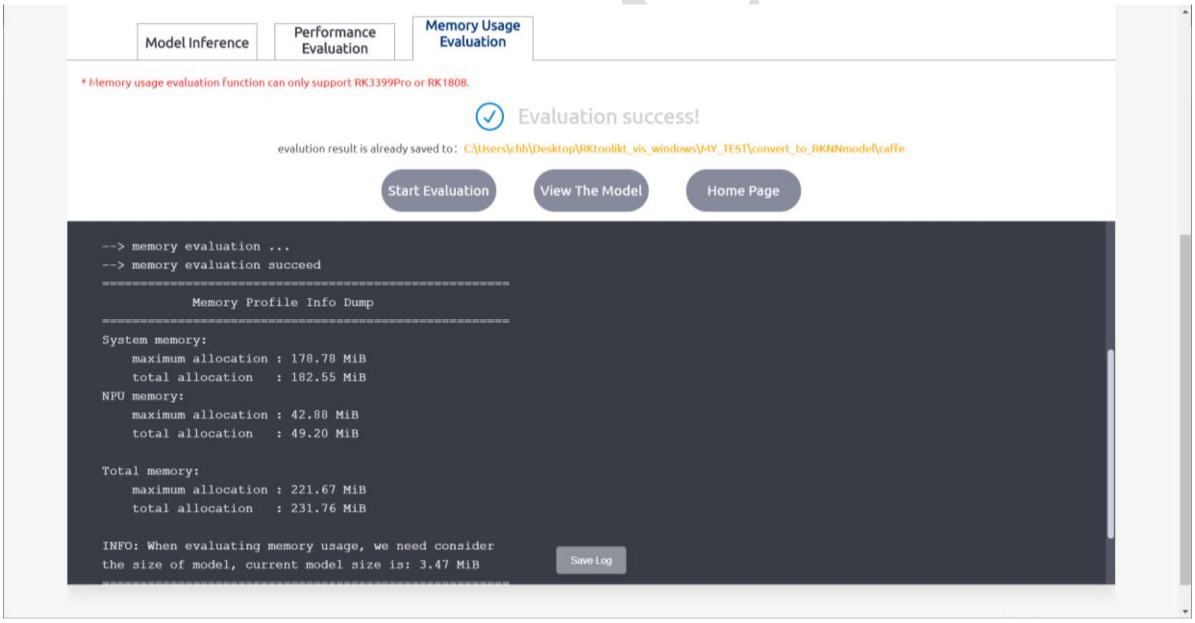

Figure 7 RKNN memory usage evaluation

2.3.3 TensorFlow

Click the TensorFlow icon to go to the TensorFlow page. Parameter configuration is required before converting TensorFlow model to RKNN model. The function of each option of the parameter configuration page is described below.

The function of each option of the parameter configuration 1 is as follows:

- **Channel Mean Value：**

  Including four values (M0 M1 M2 S0), the first three values are mean, and the last one is scale. Assuming the input is (Cin0, Cin1, Cin2) and the output is (Cout0, Cout1, Cout2), the calculation formula is: Cout0 = (Cin0 - M0) / S0, Cout1 = (Cin1 - M1) / S0, Cout2 = (Cin2 - M2)/S0.

- **Reorder Channel:**

  Represent if need to adjust the order of the image channel or not, only works on 3 channels. '0 1 2' means infering according to the input channel order. For example, when the input picture is RGB order, the inference will be in RGB order; '2 1 0' means doing channel conversion to the input channel. For example, when the input picture is RGB order, the inference will be in BGR order.

- **Dataset:**

  The calibration dataset used for quantization. Currently support text file format, users can put the calibration image (jpg or png format)or npy file path to a .txt file. Each line in the text file means a path.

- **Batch Size:**

  The size of each batch data used for quantization.

- **Epochs:**

  The iteration number during quantization. For each iteration, it will select the images per the number specified by batch size to perform the quantization and calibration. If the value is -1, it will be automatically calculated according to the total number of datasets and batch size.

- **Quantized Type:**

  If the quantization type is none, no quantization is performed, floating point type is used, and the hybrid quantization function cannot be used.

- **Whether The Inception Series Model:**

  If the model is inception v1/v3/v4, turning this option on will improve performance.

- **Whether To Enable Pre-Compile:**

  If pre-compiling is enabled, it can reduce the first loading time of the model running on the hardware device. But after this switch is enabled, the converted model can only be run on the hardware platform.

- **The Location To Save The Conversion Result:**

  The directory to save the converted RKNN model.

- **RKNN Model Filename:**

  The converted RKNN model is saved by the name, the file suffix is rknn.

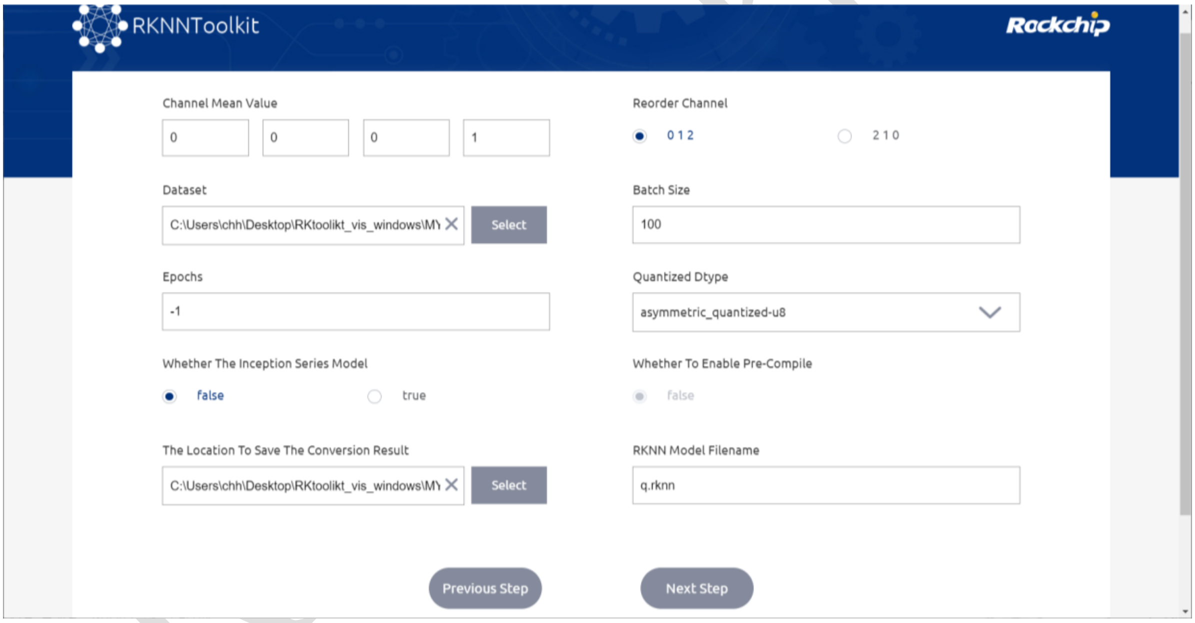

Figure 8 TensorFlow parameter configuration 1

After completing parameter configuration 1, click Next Step to enter the page of parameter configuration 2. The function of each option of the parameter configuration 2 is as follows:

- **Model:**

  The path of Pb model

- **Predef File:**

  In order to support some control logics, need to provide a pre-defined file with npz format. Can be empty.

- **Input Nodes:**

  The input nodes of the model.

- **Input Size List:**

  The size and channel of the image corresponding to each input node, separated by comma. Such as 224, 224, 3.

- **Mean Values:**

  The average input value. This parameter needs to be set only when the loaded model is already quantized and the average values of all three input channels of the model are the same. It is recommended that the quantization type of the previous step be set to none, otherwise the model will be re-quantized.

- **Scale:**

  The input scale value. This parameter needs to be set only when the loaded model is quantized. It is recommended that the quantization type of the previous step be set to none, otherwise the model will be re-quantized.

- **Output Nodes:**

  The output node of the model, support multiple output nodes.

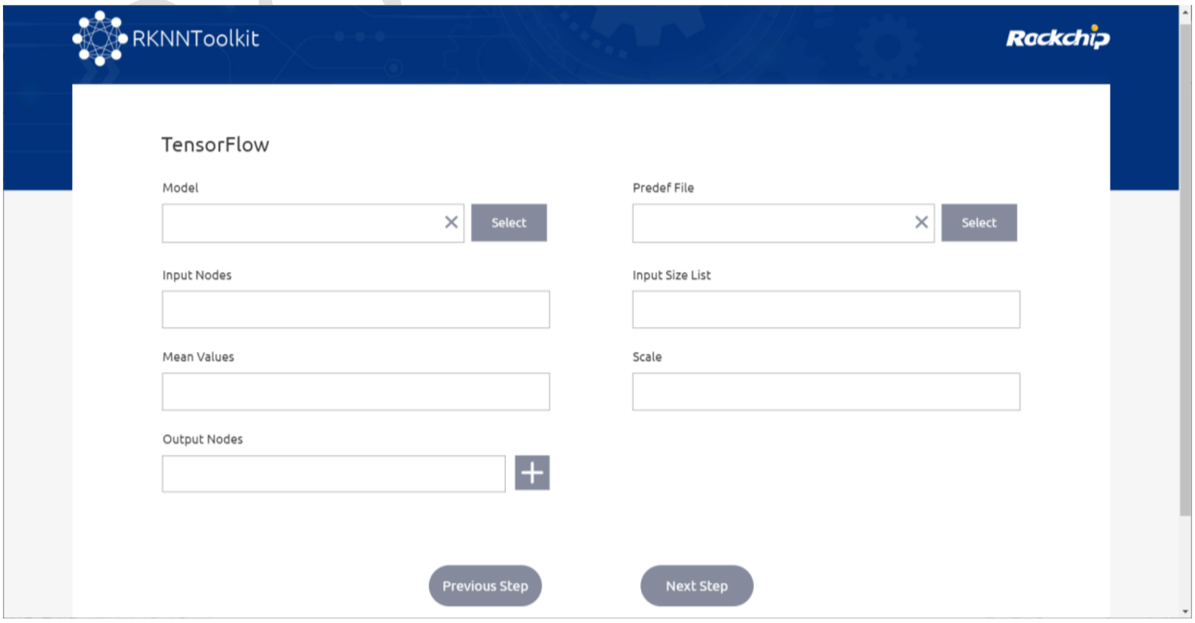

Figure 9 TensorFlow parameter configuration 2

If all the parameters are configured, click Next Step to start loading the model and quantizing the model.

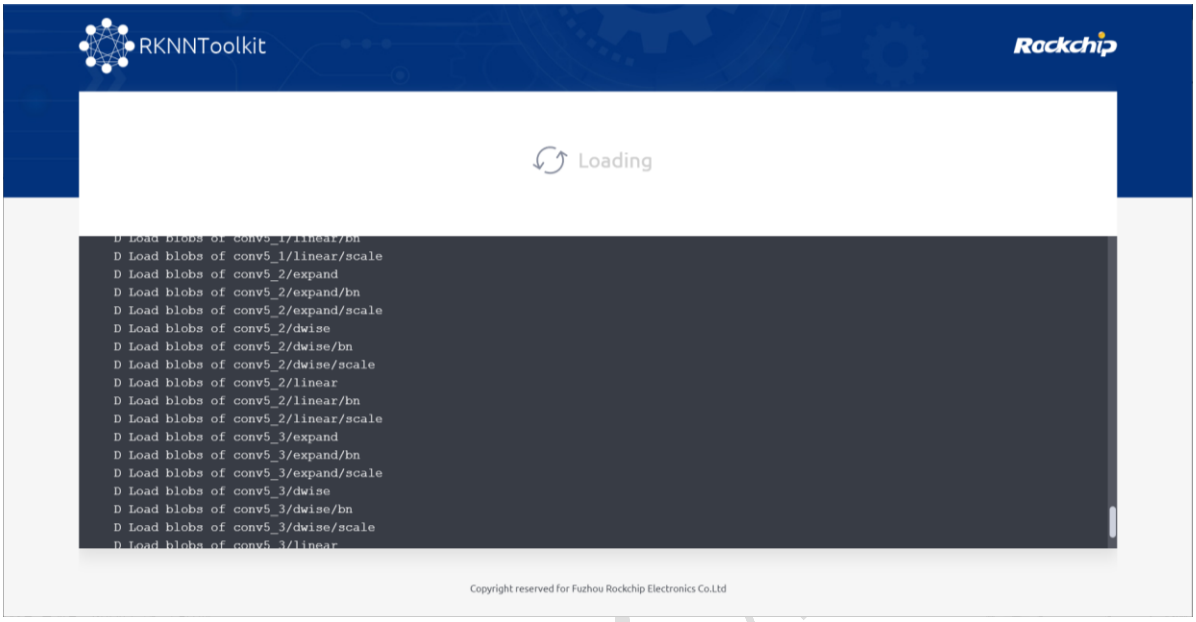

Figure 10 Model loading

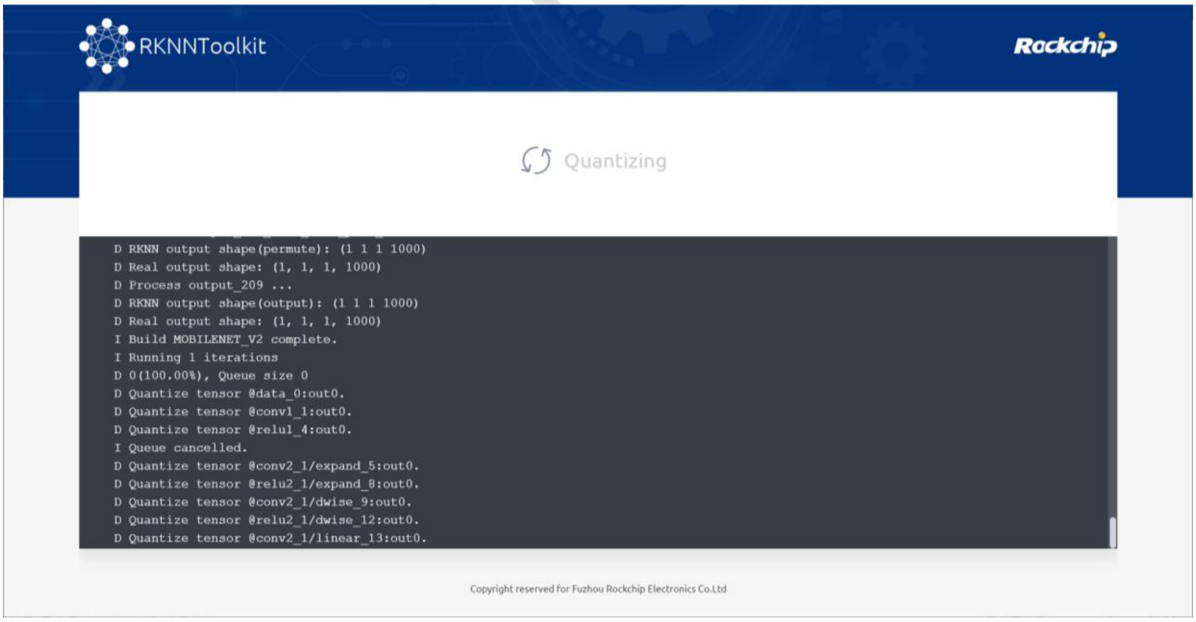

Figure 11 Model quantizing

You will go to the model visualization page after loading and quantizing model is finished. The visualization page shows the details of each layer of the TensorFlow model (including layer names and parameters). If the current window only displays partial information of the model, you can drag the model or scroll up/down the mouse to zoom in/out the image to see the other parts of the model. Dark blue is the quantized layer and light blue is the unquantized layer.

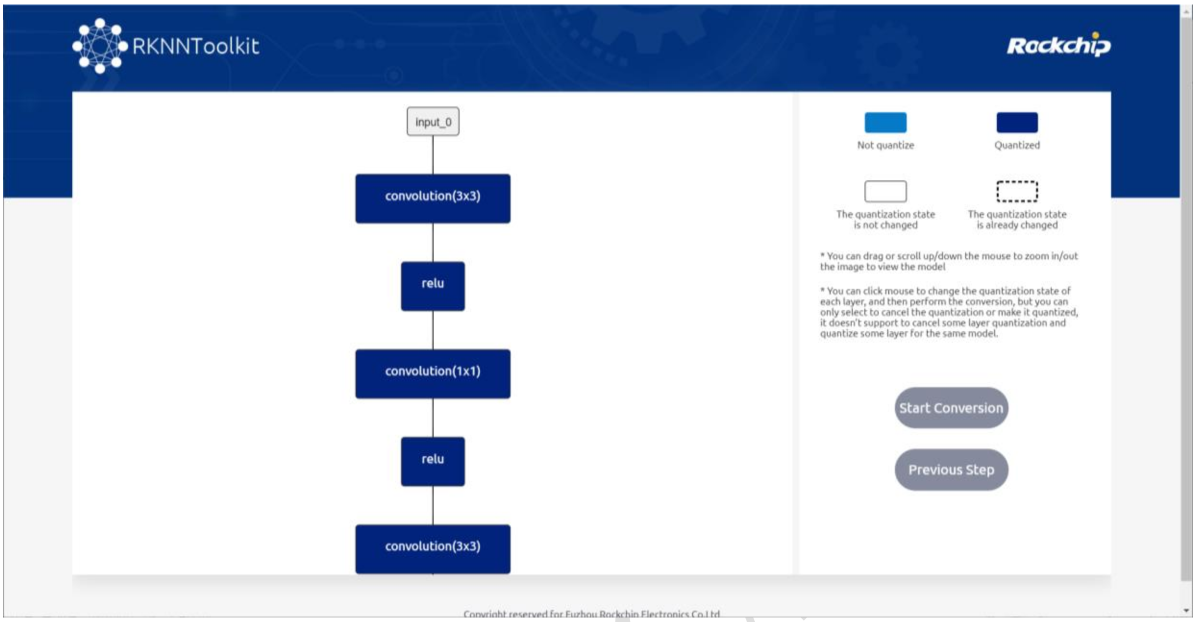

Figure 12 TensorFlow model visualization

You can change the quantization state of each layer by clicking each layer, such as changing the quantized layer to unquantized layer, or changing the unquantized layer to quantized layer. If the original quantization state has not been changed, click Start Conversion to directly export the RKNN model; if the original quantization state is changed, click Start Conversion will do hybrid quantization first then export the RKNN model.

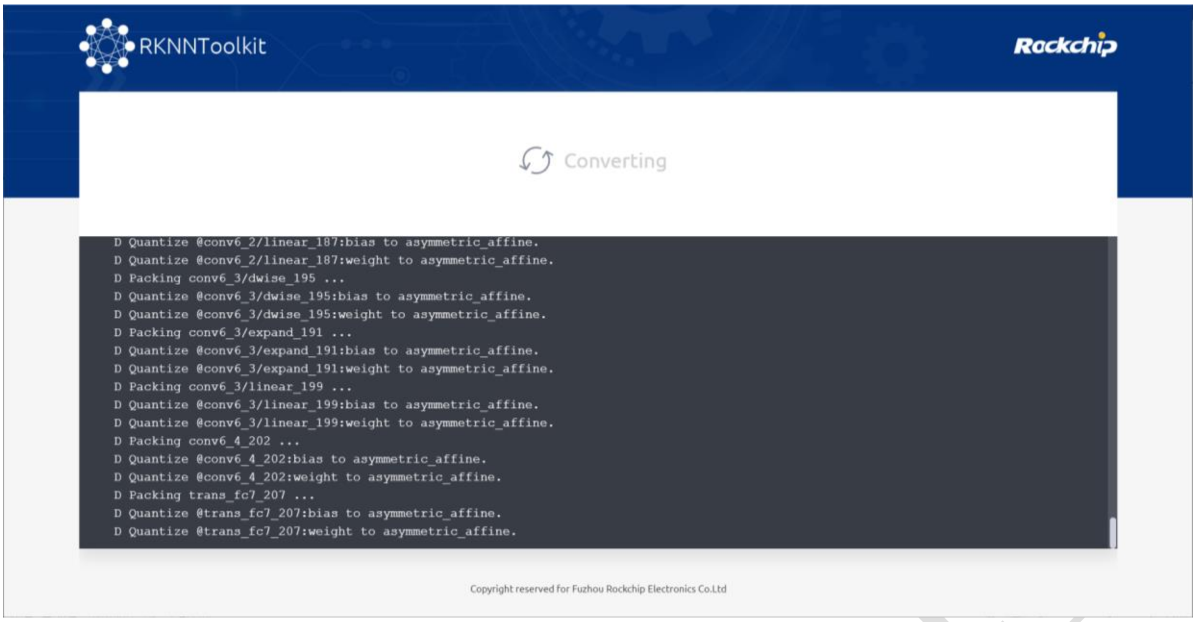

Figure 13 Export the RKNN model

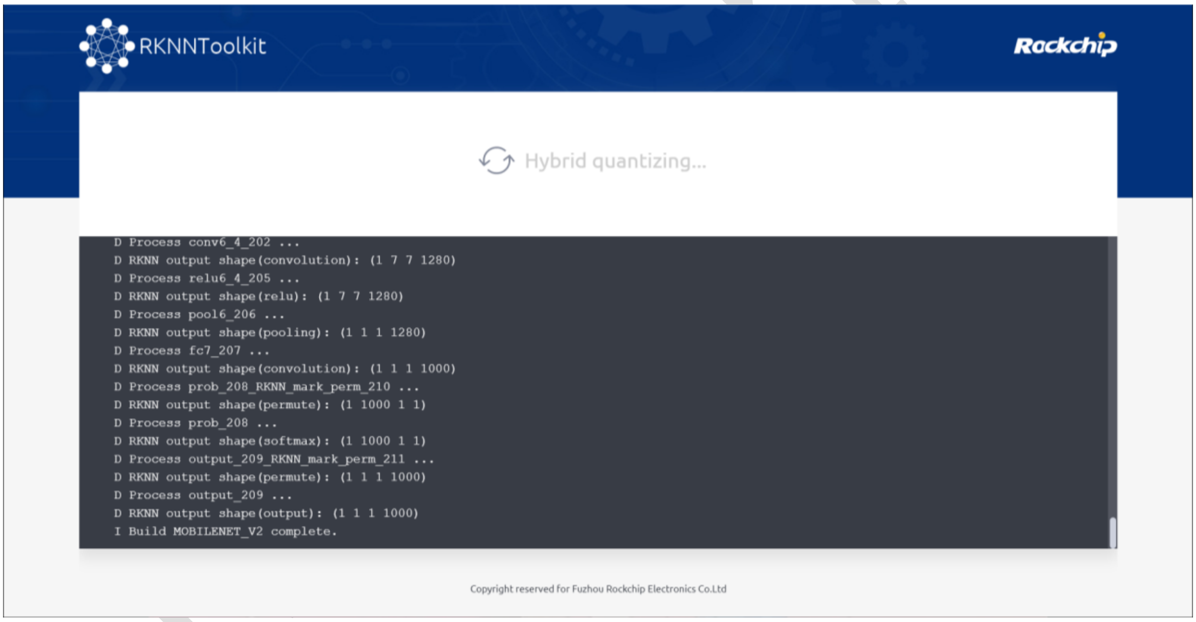

Figure 14 hybrid quantization

2.3.4 TensorFlow Lite

Click the TensorFlow Lite icon to go to the TensorFlow Lite page. You also need to configure the parameters before converting TensorFlow Lite model to RKNN model.

For parameter configuration 1, please refer to parameter configuration 1 in section **2.3.3 of TensorFlow**, which will not be repeated here.

Parameter configuration 2 is as follows:

- **Model:**

  TensorFlow Lite model file (.tflite suffix) located path.

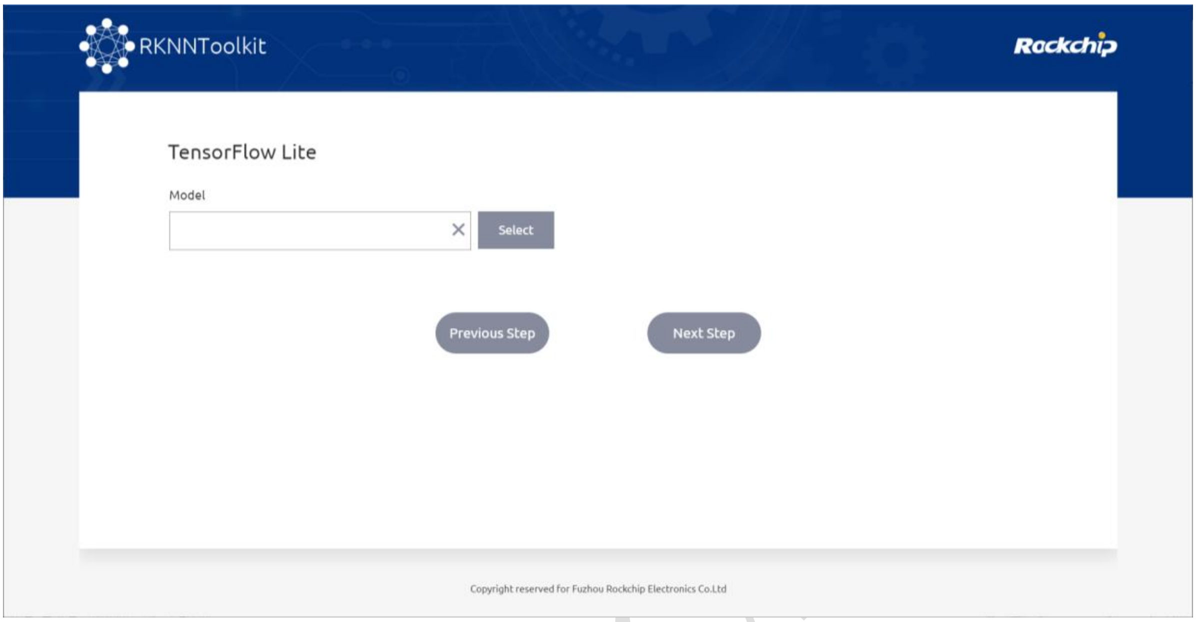

Figure 15 TensorFlow Lite parameter configuration 2

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **2.3.3 of TensorFlow**, which will not be repeated here.

2.3.5 MXNet

Click the MXNet icon to go to the MXNet page. You also need to configure the parameters before converting MXNet model to RKNN model.

For parameter configuration 1, please refer to parameter configuration 1 in section **2.3.3 of TensorFlow**, which will not be repeated here.

Parameter configuration 2 is as follows:

- **Symbol:**

  The path of the MXNet model file (.json suffix).

- **Params:**

  The path of the MXNet weight file (.params suffix).

- **Input Size List:**

  The size and channel of the image corresponding to each input node, separated by comma. Such as 3, 224, 224.

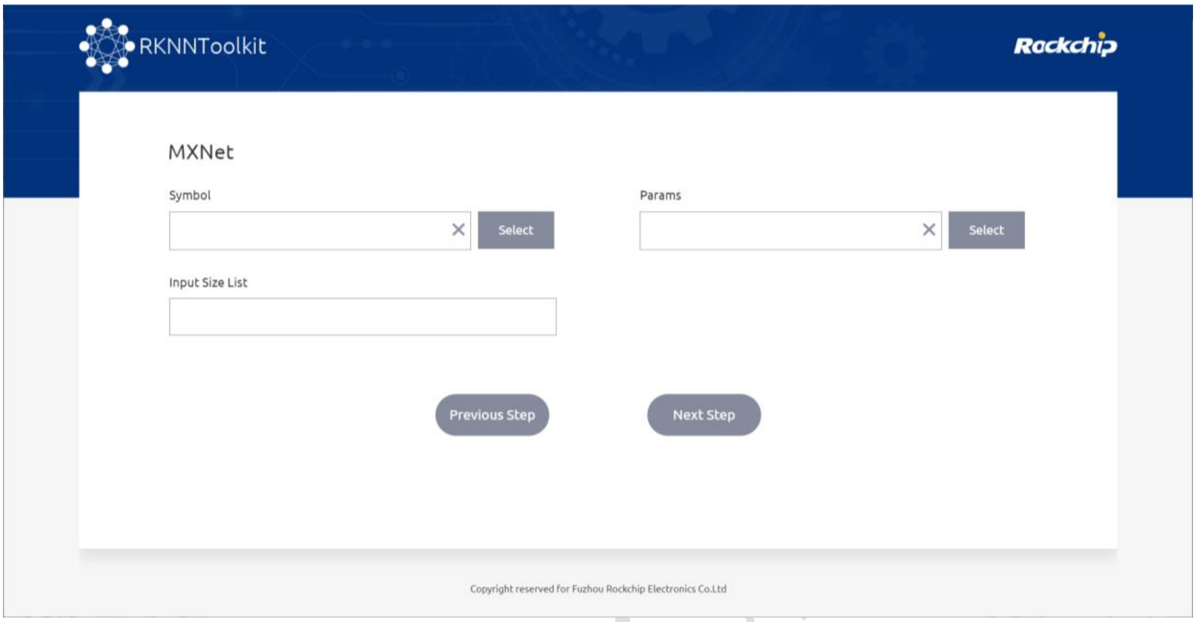

Figure 16 MXNet parameter configuration 2

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **2.3.3 of TensorFlow**, which will not be repeated here.

2.3.6 ONNX

Click the ONNX icon to go to the ONNX page. You also need to configure the parameters before converting ONNX model to RKNN model.

For parameter configuration 1, please refer to parameter configuration 1 in section **2.3.3 of TensorFlow**, which will not be repeated here.

Parameter configuration 2 is as follows:

- **Model:**

  ONNX model file (.onnx suffix) located path.

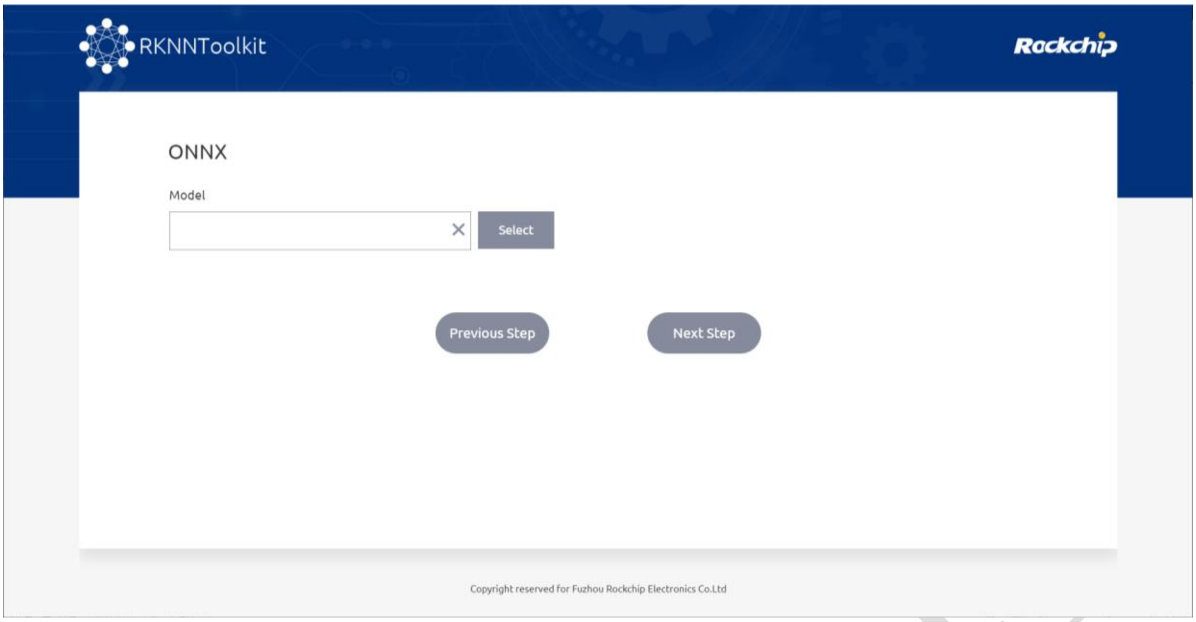

Figure 17 ONNX parameter configuration 2

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **2.3.3 of TensorFlow**, which will not be repeated here.

2.3.7 Darknet

Click the Darknet icon to go to the Darknet page. You also need to configure the parameters before converting Darknet model to RKNN model.

For parameter configuration 1, please refer to parameter configuration 1 in section **2.3.3 of TensorFlow**, which will not be repeated here.

Parameter configuration 2 is as follows:

- **Model:**

  Darknet model file (.cfg suffix) located path.

- **Weight:**

  Darknet weight file (.weights suffix) located path.

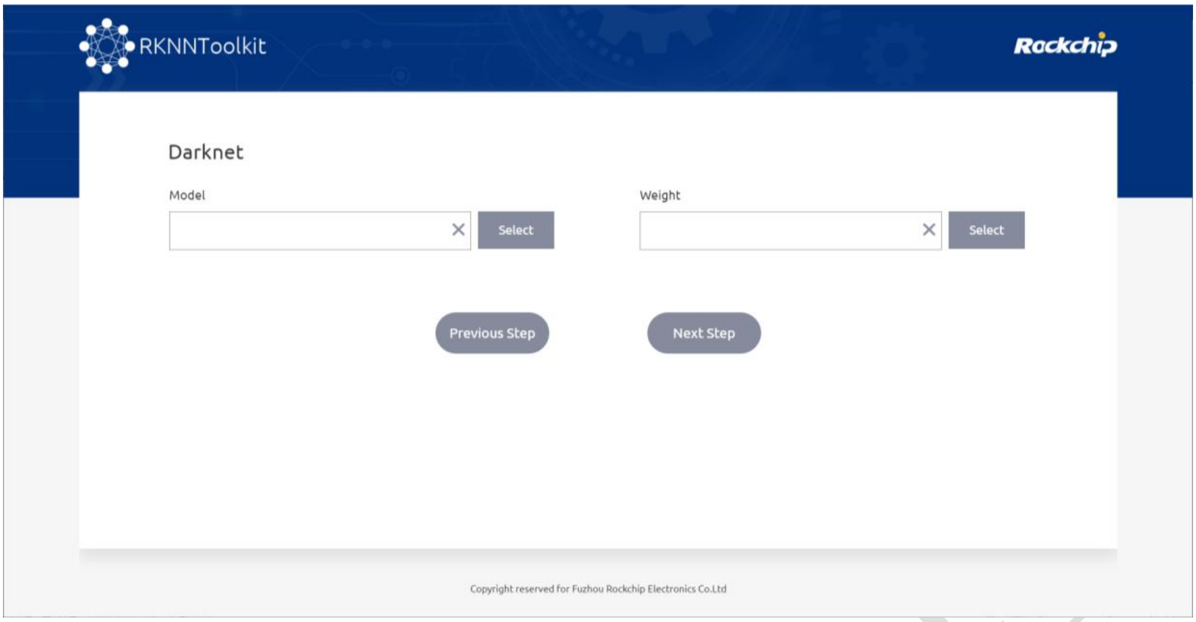

Figure 18 Darknet parameter configuration 2

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **2.3.3 of TensorFlow**, which will not be repeated here.

2.3.8 PyTorch

Click the PyTorch icon to go to the PyTorch page. You also need to configure the parameters before converting PyTorch model to RKNN model.

For parameter configuration 1, please refer to parameter configuration 1 in section **2.3.3 of TensorFlow**, which will not be repeated here.

Parameter configuration 2 is as follows:

- **Model:**

  PyTorch model file (.pt suffix) located path. The pth model usually only contains weights, not contains model structure. Before conversion, you need to call some functions (such as torch.jit.trace) to convert the pth model into torchscript (.pt suffix) model which contains both weights and network structure.

- **Input Size List:**

  The channel and size of the image corresponding to each input node, separated by comma.

Such as 3, 224, 224.

- **Input Nodes:**

  Optional, default query from model.

- **Output Nodes:**

  Optional, default query from model.

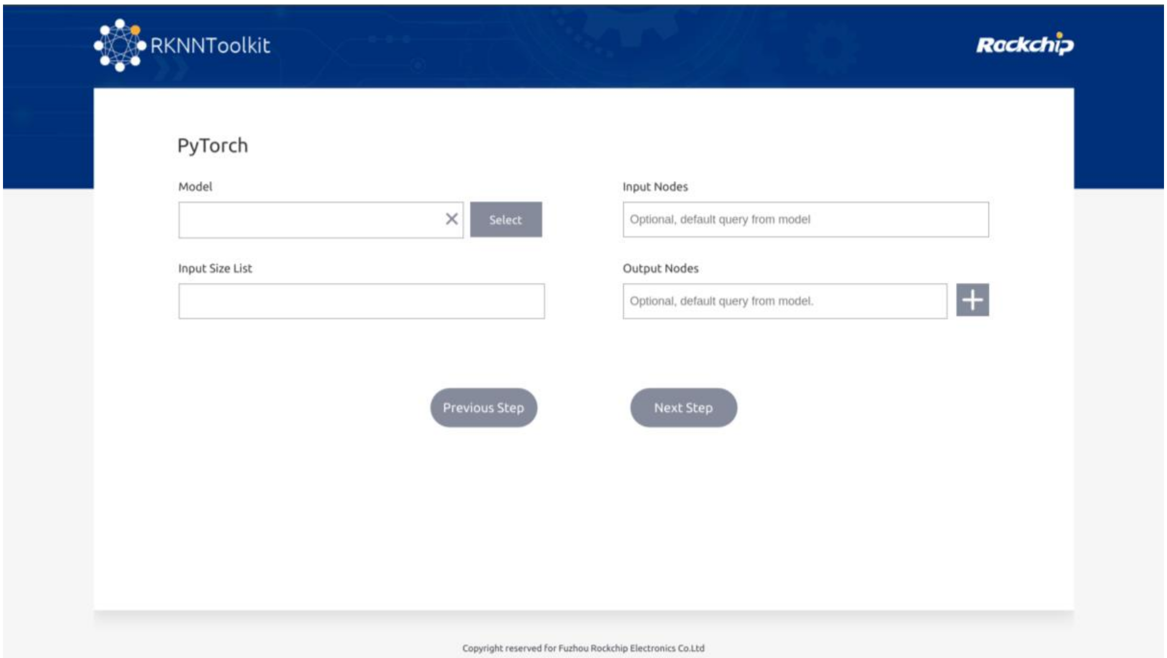

Figure 19 PyTorch parameter configuration 2

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **2.3.3 of TensorFlow**, which will not be repeated here.

2.3.9 Caffe

Click the Caffe icon to go to the Caffe page. You also need to configure the parameters before converting Caffe model to RKNN model.

For parameter configuration 1, please refer to parameter configuration 1 in section **2.3.3 of TensorFlow**, which will not be repeated here.

Parameter configuration 2 is as follows:

- **Model:**

  Caffe model file (.prototxt suffix file) located path.

- **Proto:**

  Caffe model format.

- **Blobs:**

  Caffe model binary data file (.caffemodel suffix file) located path.

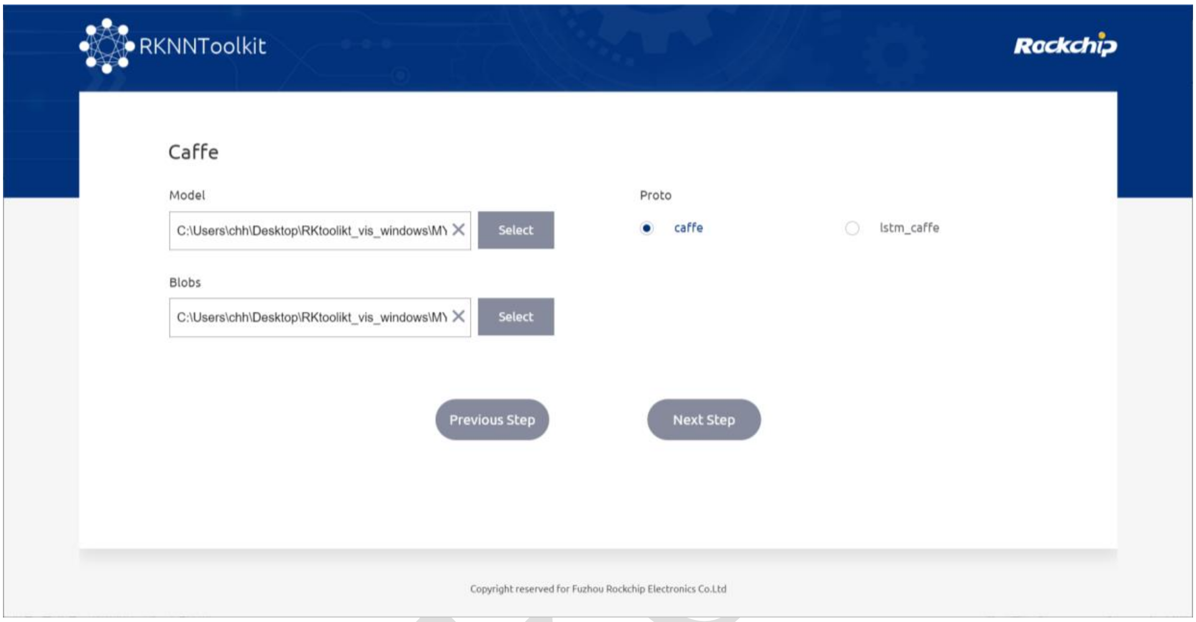

Figure 20 Caffe parameter configuration 2

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **2.3.3 of TensorFlow**, which will not be repeated here.