

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

RKNN-Toolkit Visualization User Guide

(Technology Department, Graphic Computing Platform Center)

Mark: [<input type="checkbox"/>] Editing [<input checked="" type="checkbox"/>] Released	Version	V1.4.0
	Author	Chen Hao
	Completed Date	2020-08-13
	Reviewer	Randall
	Reviewed Date	2020-08-13

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd

(Copyright Reserved)

Revision History

Version	Modifier	Date	Modify description	Reviewer
V0.1.0	ChenHao	2019-11-25	Initial version	Randall
V1.3.2	Rao Hong	2020-04-15	Update version	Randall
V1.4.0	Rao Hong	2020-08-13	Adjust the document organization structure.	Randall

Table of Contents

1	OVERVIEW	4
2	REQUIREMENTS/DEPENDENCIES.....	6
3	INSTALLATION.....	7
3.1	INSTALLATION.....	7
3.2	STARTUP METHOD	7
4	INSTRUCTIONS	8
4.1	HOMEPAGE	8
4.2	MODEL CONVERSION.....	8
4.2.1	TensorFlow	9
4.2.2	TensorFlow Lite.....	14
4.2.3	MXNet	15
4.2.4	ONNX.....	16
4.2.5	Darknet	17
4.2.6	PyTorch.....	18
4.2.7	Caffe	19
4.3	USE RKNN MODEL.....	20
4.3.1	Model Visualization	20
4.3.2	Model Usage.....	22
5	REFERENCE	25

1 Overview

The visualization function provides a graphical human-computer interaction interface for RKNN Toolkit. Through the visualization function, users only need to fill in the form and click the corresponding function button to complete the model conversion/model inference/model evaluation functions. No more scripting. This greatly simplifies the user's use process and reduces the user's difficulty in use. The following features are supported now:

- 1) Model conversion: support to convert Caffe, Darknet, MXNet, ONNX, Pytorch, TensorFlow, TensorFlow Lite model to RKNN model.
- 2) Quantization: support to convert float model to quantization model. Currently supported quantization methods include asymmetric quantization (asymmetric_quantized-u8), dynamic fixed-point quantization (dynamic_fixed_point-8 and dynamic_fixed_point-16) and hybrid quantization.
- 3) Model inference: Ability to simulate NPU running model on PC (Linux x86_64) and obtain inference results. Users can also run the model on specified target (RK1806 / RK1808 / RK3399Pro / RV1109 / RV1126) and obtain inference results.
- 4) Performance evaluation: Ability to simulate NPU running model on PC and obtain the total time consumption and each layer's time consumption of the model. Users can also run the model on specified target (RK1806 / RK1808 / RK3399Pro / RV1109 / RV1126) to obtain the performance data of the model when it runs on the NPU device (total time and the time for each layer).
- 5) Memory evaluation: Get the memory usage of the model when it runs on the NPU. Through online debugging, run the RKNN model on the specified device and obtain the memory usage information of the model on the target device.
- 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.

Rockchip

2 Requirements/Dependencies

This tool only supports running on the Ubuntu、Windows and MacOS operating system. For system dependencies, please refer to the **Requirements/Dependencies** section of 《Rockchip_User_Guide_RKNN_Toolkit_V1.4.0_EN.pdf》.

Rockchip

3 Installation

3.1 Installation

The RKNN Toolkit on the Ubuntu/Windows/MacOS platform has its own visualization function, users only need to install the RKNN Toolkit. For the installation method, please refer to the **installation** section of <Rockchip_User_Guide_RKNN_Toolkit_V1.4.0_EN.pdf>.

3.2 Startup Method

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

```
python3 -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).

4 Instructions

4.1 Homepage

After opening the visualization program, the home page is shown in Figure 4-1-1.

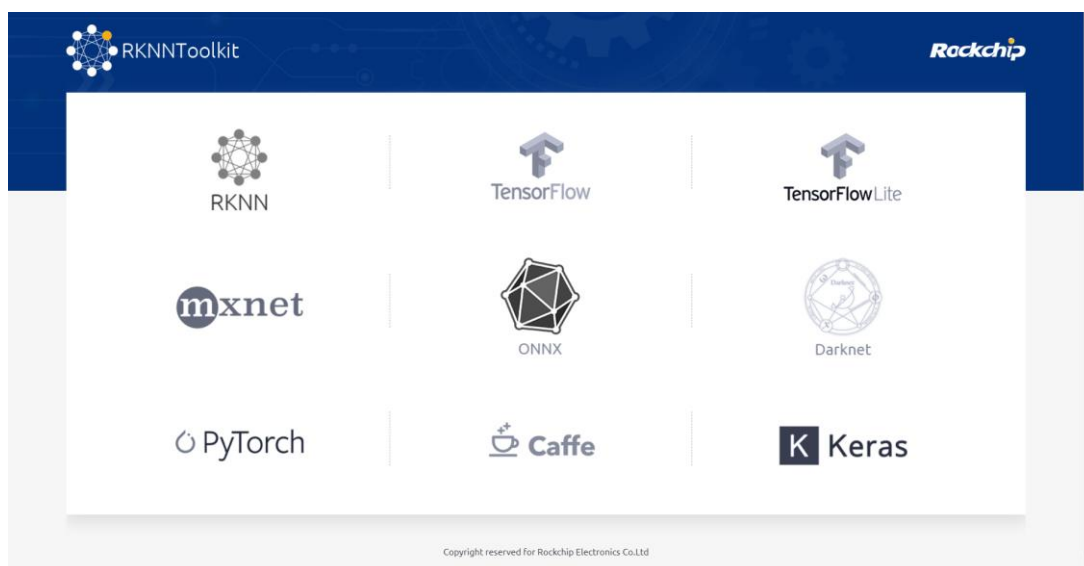


Figure 4-1-1 Visualization home page

Icons of different deep learning frameworks and RKNN models are arranged on the homepage. Click the icon of each deep learning framework (Keras does not currently support), it will enter the corresponding model conversion page. Click the RKNN model icon to enter the model inference / performance evaluation / memory evaluation page.

4.2 Model Conversion

RKNN Toolkit supports converting models trained by deep learning frameworks such as Caffe / Darknet / MXNet / ONNX / Pytorch / TensorFlow / TensorFlow Lite into RKNN models. Click on the icon of the corresponding frame on the homepage to enter the conversion page of the frame model. The following subsections will show how to use visualization programs to complete the conversion of each frame model through graphics and text.

4.2.1 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 parameter configuration is divided into two pages. The first page is the general configuration of the model. This part of the content needs to be filled in when the model of each frame is converted. The second page is the configuration of each frame.

The configuration items and descriptions of the model general configuration page are as follows:

- **Target Platform:**

Specify the chip platform on which the model can run: [RK1806, RK1808, RK3399PRO] or [RV1109, RV1126].

- **Mean Value:**

The channel mean and scaling factor are used when preprocessing the model input. Assuming an input with three channels, its mean value and scaling factor are (M0 M1 M2 S0), and the value of each channel is represented by (IN0, IN1, IN2). The preprocessing will be calculated according to the following formula: $OUT0 = (IN0 - M0) / S0$; $OUT1 = (IN1 - M1) / S0$; $OUT2 = (IN2 - M2) / S0$. (OUT0, OUT1, OUT2) are the outputs after preprocessing.

This parameter specifies the mean value of each channel during model input preprocessing. Each channel is separated by a space, if there are multiple inputs, each input is separated by "#".

For example: 127.5 127.5 127.5#127.5, which means that the model has two inputs, the first input has three channels, the mean values are both 127.5; the second input has only one channel, the mean value is 127.5.

- **Standard Value:**

This parameter specifies the scaling factor during model input preprocessing. If there are multiple inputs, the scaling factor of each input is separated by "#". For example, 127.5#127.5 means that the model has two inputs, and the scaling factors of the two inputs during

preprocessing are both 127.5.

- **Reorder Channel:**

Represent if need to adjust the order of the image channel or not, only works on 3 channels. '0 1 2' means inferring 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. For details of other quantification types, please refer to < Rockchip_User_Guide_RKNN_Toolkit_V1.4.0_EN.pdf >.

- **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.

The screenshot shows a web-based configuration form for RKNN conversion. It is divided into two main columns. The left column contains: 'Target Platform' (a dropdown menu with 'RK1806, RK1808, RK3399Pro' selected), 'Mean Value' (an empty text input), 'Quantized Dtype' (a dropdown menu with 'asymmetric_quantized-u8' selected), 'Batch Size' (a text input with '100'), 'Whether The Inception Series Model' (radio buttons for 'false' and 'true', with 'false' selected), and 'The Location To Save The Conversion Result' (a text input with a 'Select' button). The right column contains: 'Reorder Channel' (radio buttons for '0 1 2' and '2 1 0', with '0 1 2' selected), 'Standard Value' (a text input with '1'), 'Dataset' (a text input with a 'Select' button), 'Epochs' (a text input with '-1'), 'Whether To Enable Pre-Compile' (radio buttons for 'false' and 'true', with 'false' selected), and 'RKNN Model Filename' (an empty text input). At the bottom, there are two buttons: 'Previous Step' and 'Next Step'.

Figure 4-2-1-1 General Configuration

After filling in the general configuration, click Next to enter the TensorFlow model configuration page. Click Back to return to the home page..

The detailed description of each option on the TensorFlow configuration page 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.

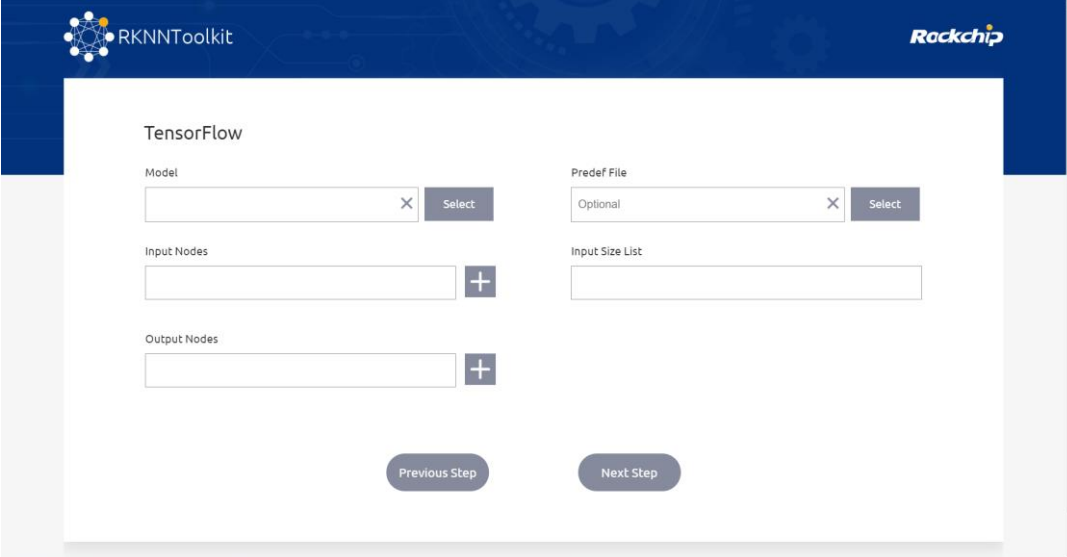
- **Input Size List:**

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

Such as 224, 224, 3.

- **Output Nodes:**

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



The screenshot shows the 'TensorFlow' configuration window in the RKNNToolkit. It features a dark blue header with the 'RKNNToolkit' logo on the left and the 'Rockchip' logo on the right. The main content area is white and contains several input fields: 'Model' and 'Predef File' (both with a 'Select' button), 'Input Nodes' (with a '+' button), and 'Output Nodes' (with a '+' button). There is also an 'Input Size List' field. At the bottom, there are two buttons: 'Previous Step' and 'Next Step'.

Figure 4-2-1-2 TensorFlow parameter configuration

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

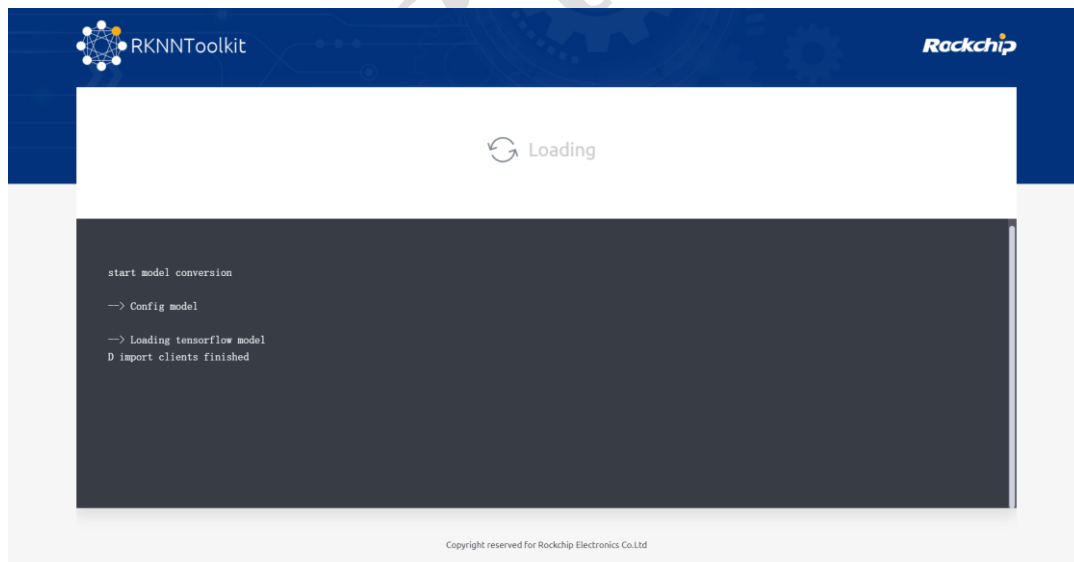


Figure 4-2-1-3 Model loading

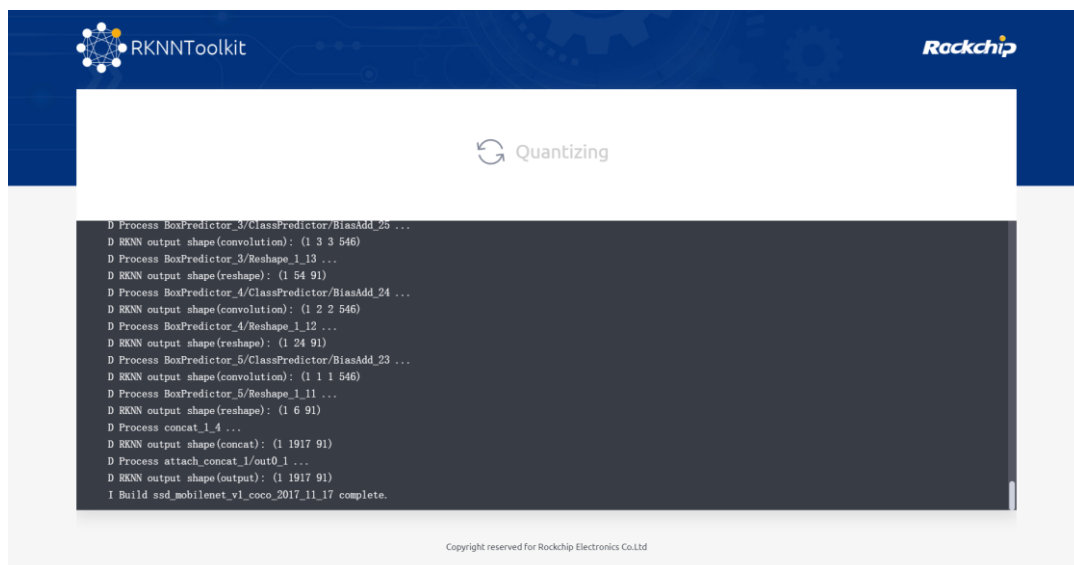


Figure 4-2-1-4 Model quantizing

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.

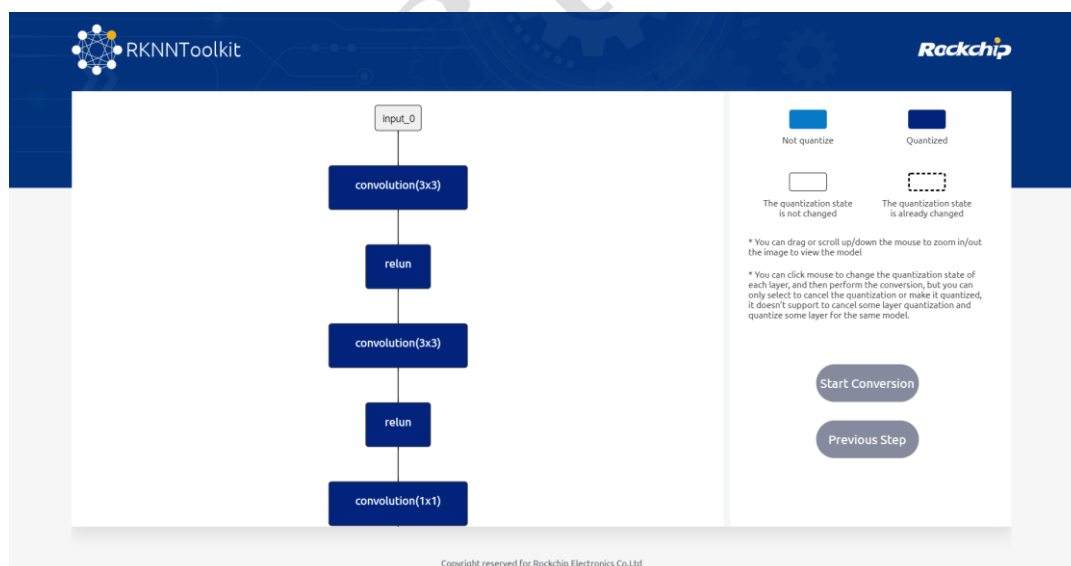


Figure 4-2-1-5 TensorFlow model visualization

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.

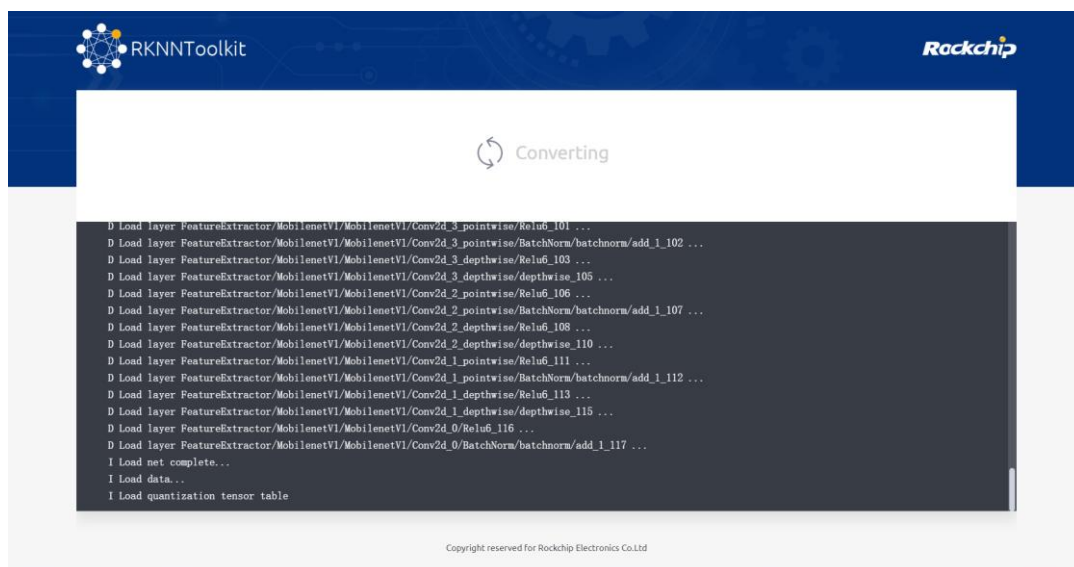


Figure 4-2-1-6 Export the RKNN model

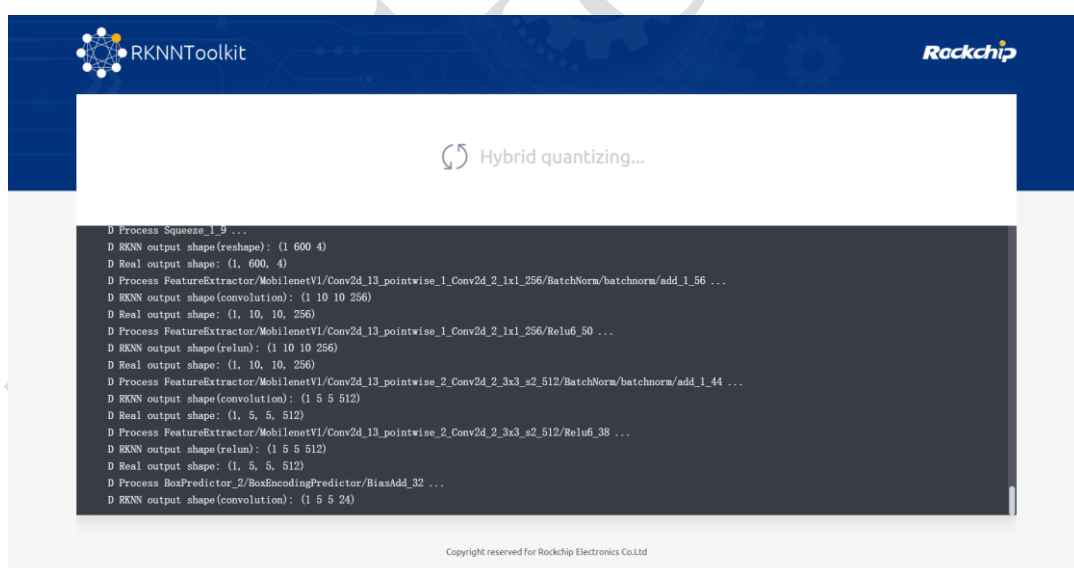


Figure 4-2-1-7 hybrid quantization

4.2.2 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 general model configuration, please refer to the detailed description in the **4.2.1 TensorFlow** section.

The detailed description of each option on the TensorFlow Lite configuration page is as follows:

- **Model:**

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

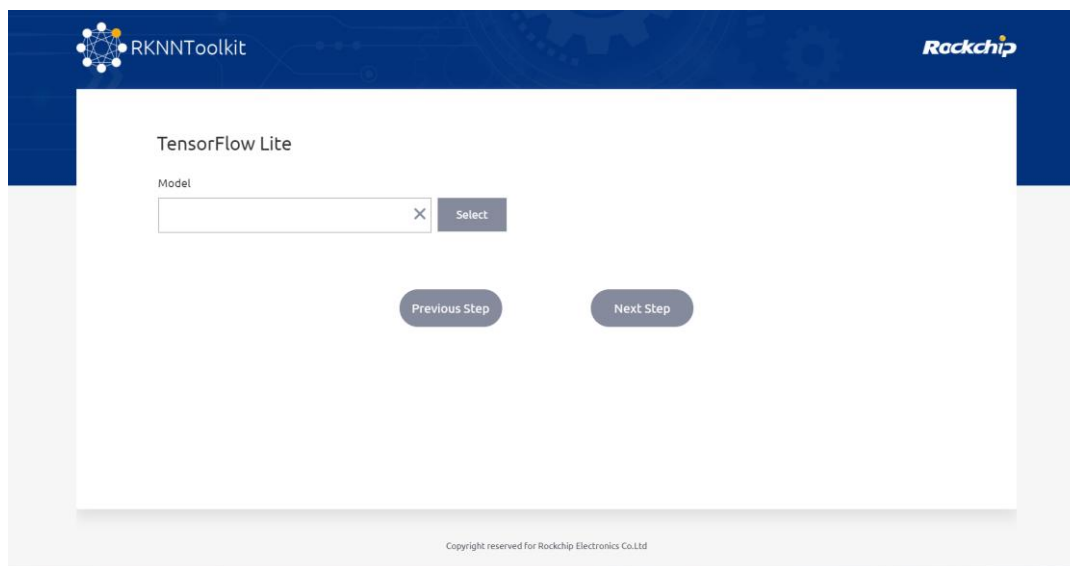


Figure 4-2-2-1 TensorFlow Lite parameter configuration

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **4.2.1 of TensorFlow**.

4.2.3 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 general model configuration, please refer to the detailed description in the **4.2.1 TensorFlow** section.

The detailed description of each option on the MXNet configuration page 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.

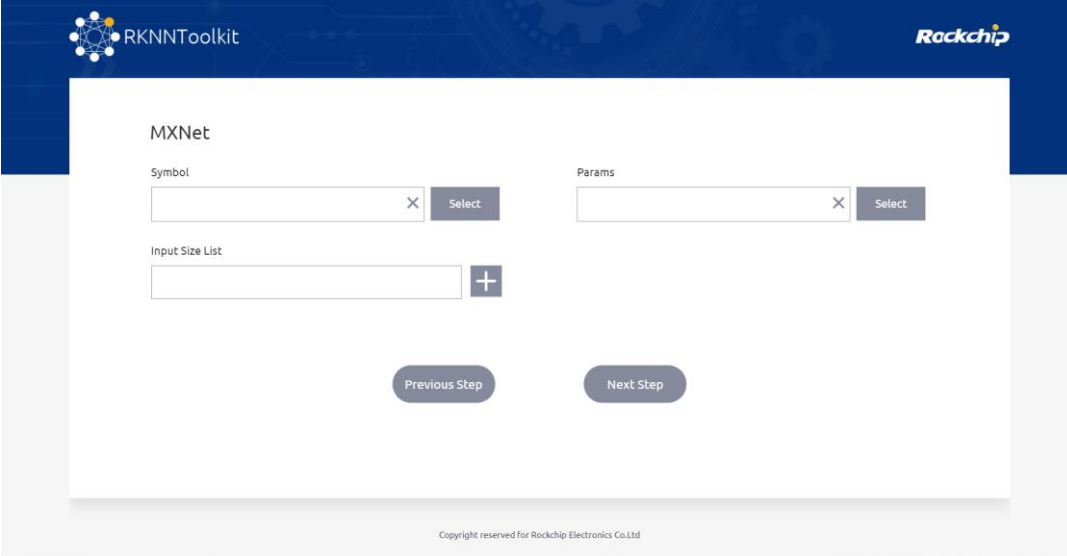


Figure 4-2-3-1 MXNet parameter configuration

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **4.2.1 of TensorFlow**.

4.2.4 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 general model configuration, please refer to the detailed description in the **4.2.1 TensorFlow** section.

The detailed description of each option on the ONNX configuration page is as follows:

- **Model:**

ONNX model file (.onnx suffix) located path.

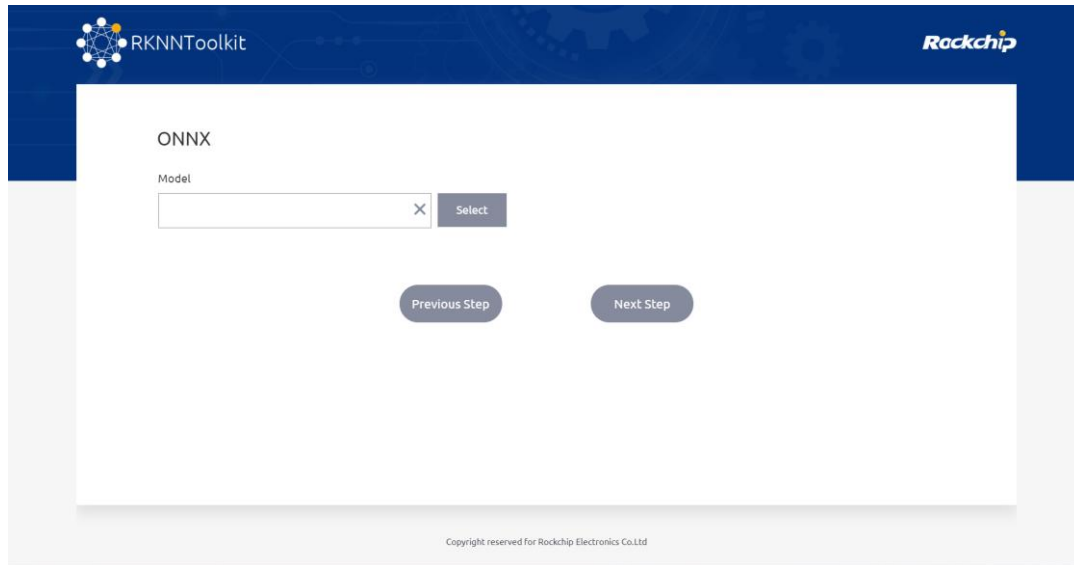


Figure 4-2-4-1 ONNX parameter configuration

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **4.2.1 of TensorFlow**.

4.2.5 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 general model configuration, please refer to the detailed description in the **4.2.1 TensorFlow** section.

The detailed description of each option on the Darknet configuration page is as follows:

- **Model:**

Darknet model file (.cfg suffix) located path.

- **Weight:**

Darknet weight file (.weights suffix) located path.

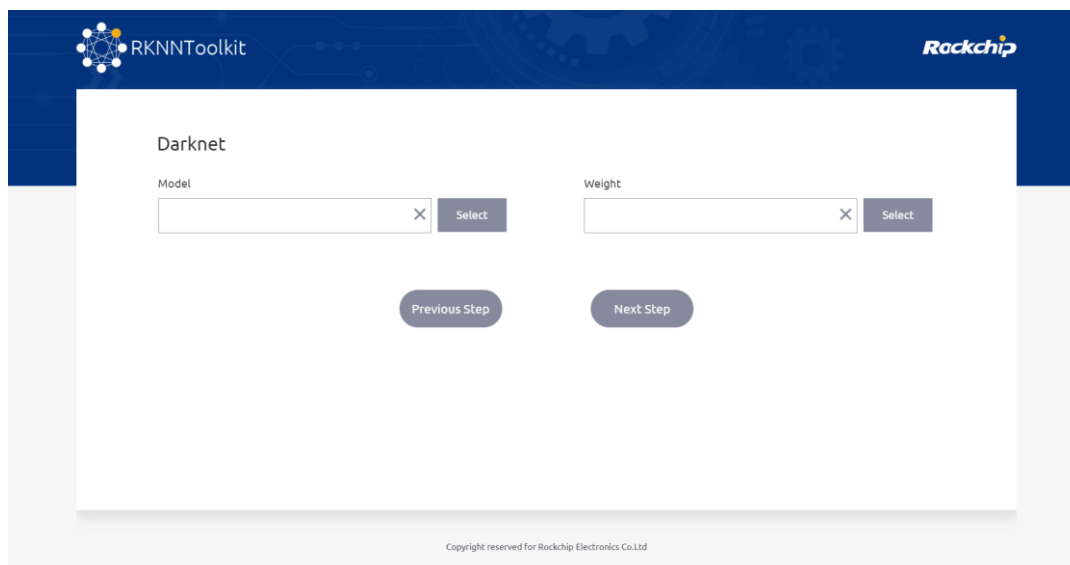


Figure 4-2-5-1 Darknet parameter configuration

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **4.2.1 of TensorFlow**.

4.2.6 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 general model configuration, please refer to the detailed description in the **4.2.1 TensorFlow** section.

The detailed description of each option on the PyTorch configuration page 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.

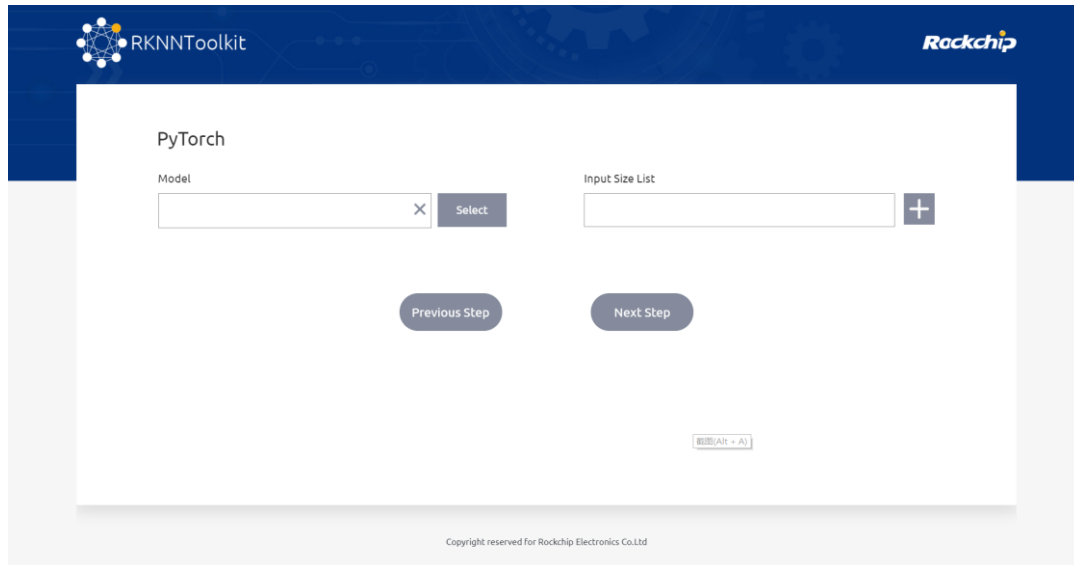


Figure 4-2-6-1 PyTorch parameter configuration

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **4.2.1 of TensorFlow**.

4.2.7 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 general model configuration, please refer to the detailed description in the **4.2.1 TensorFlow** section.

The detailed description of each option on the Caffe configuration page 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.

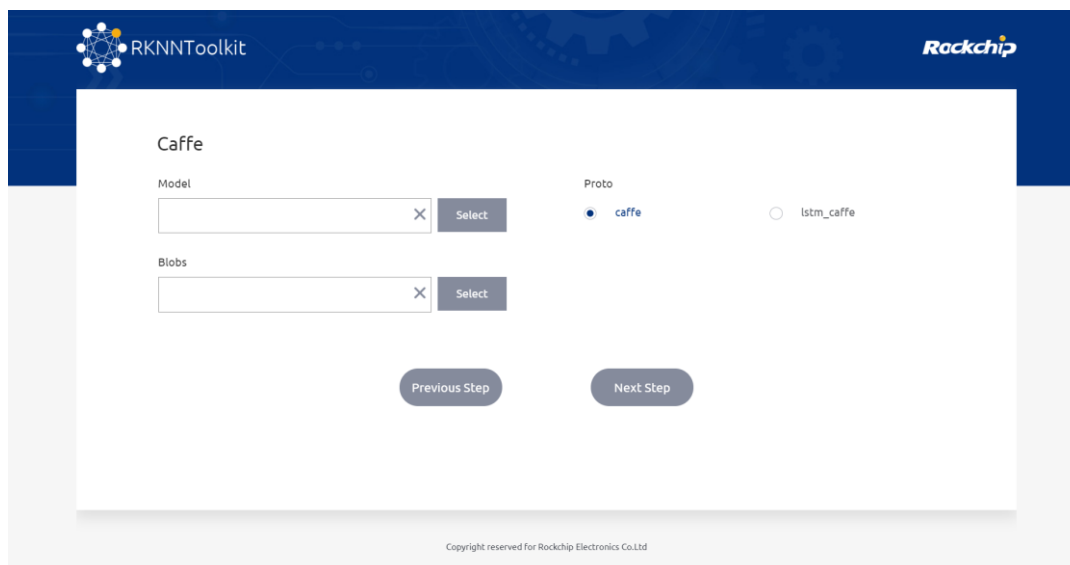


Figure 4-2-7-1 Caffe parameter configuration

For model loading, model quantizing, hybrid quantizing and model converting, please refer to section **4.2.1 of TensorFlow**.

4.3 Use RKNN model

Click the RKNN model icon on the homepage to enter the model usage page. This page provides functions such as model visualization, model inference, performance evaluation and memory usage evaluation. The following sections introduce the use of these functions in graphic form.

4.3.1 Model Visualization

After clicking the RKNN model icon on the homepage, a page for RKNN model selection will appear. Select the RKNN model we will use next on this page. The model selection page is shown in Figure 4-3-1-1.

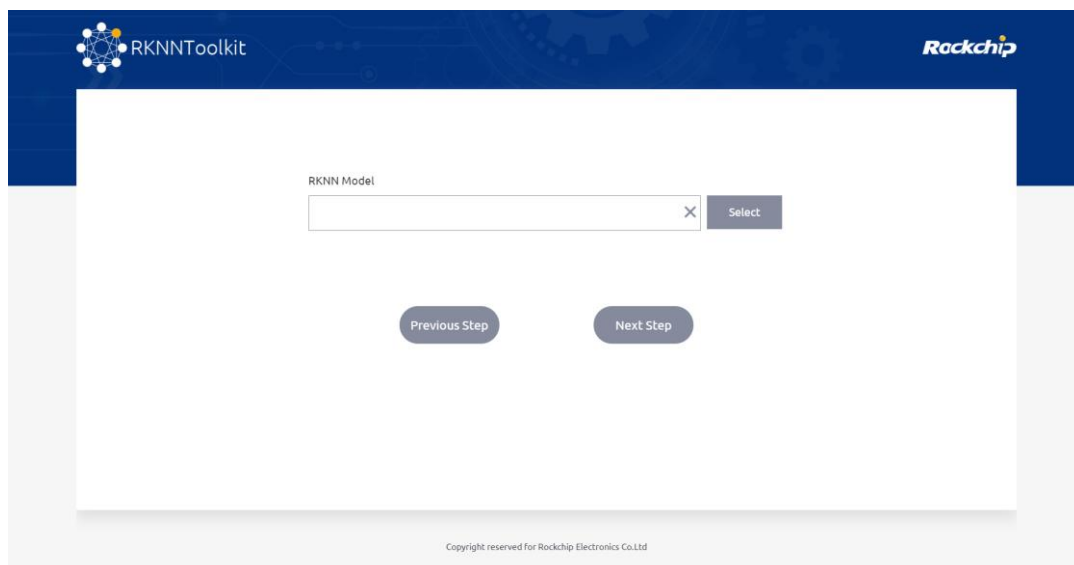


Figure 4-3-1-1 RKNN model selection

Click the select button to enter the RKNN model selection page. After selection, click the next button to enter the model visualization page.

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.

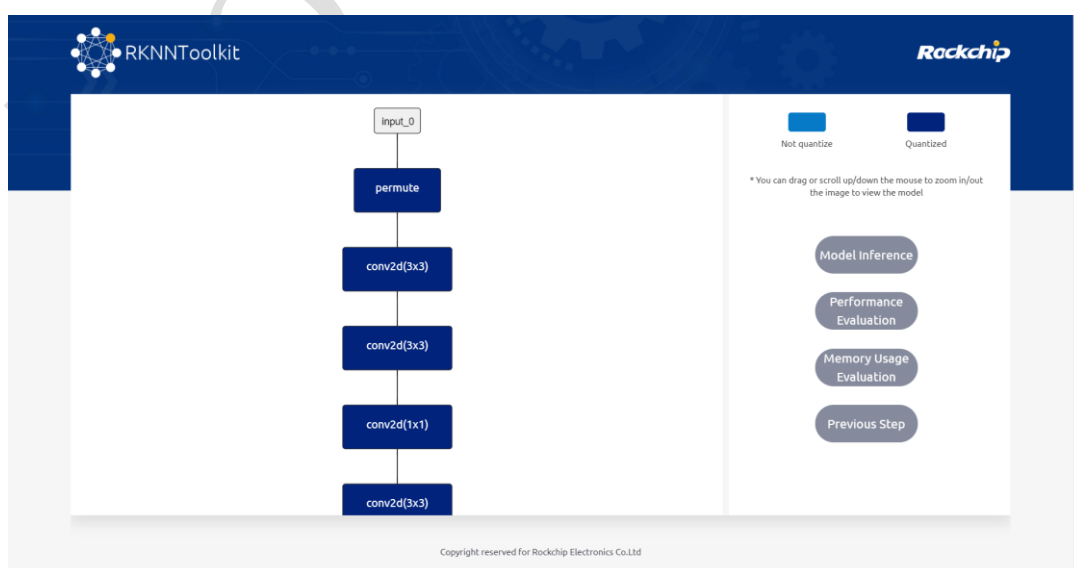


Figure 4-3-1-2 RKNN model visualization

4.3.2 Model Usage

The functions of this page are model inference, performance evaluation and memory usage evaluation. The description of each parameter is as follows:

- **Target:**

The platforms to be evaluated, supporting simulator (Only Linux x86_64 support), RK1806, RK1808, RK3399Pro, RV1109, RV1126.

- **Device ID:**

The device ID number of target, 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 thrown; 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 show the total running time of the model. This option doesn't take effect if the target is simulator.

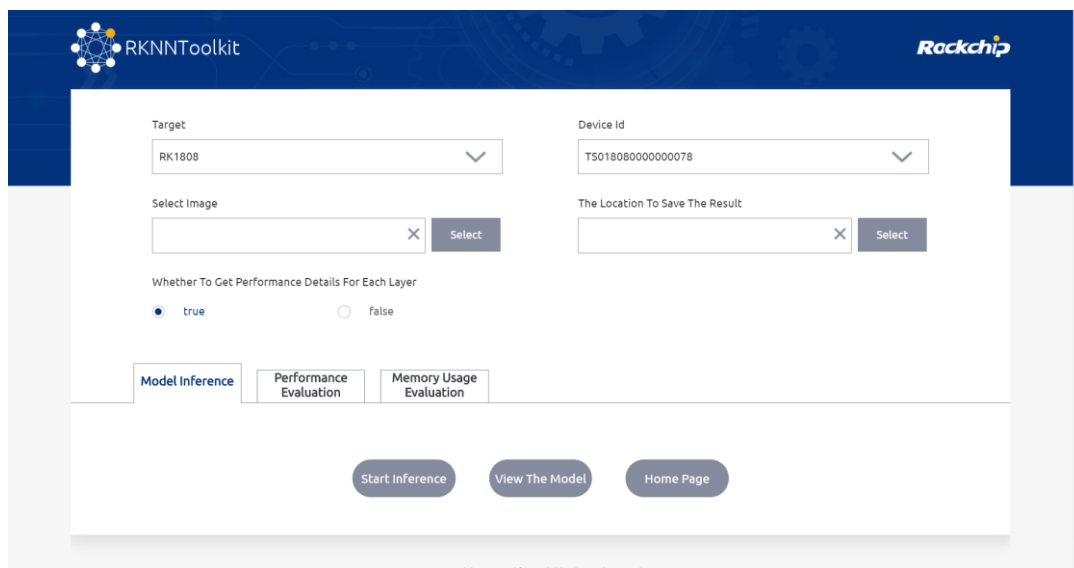


Figure 4-3-2-1 The page of RKNN model evaluation

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

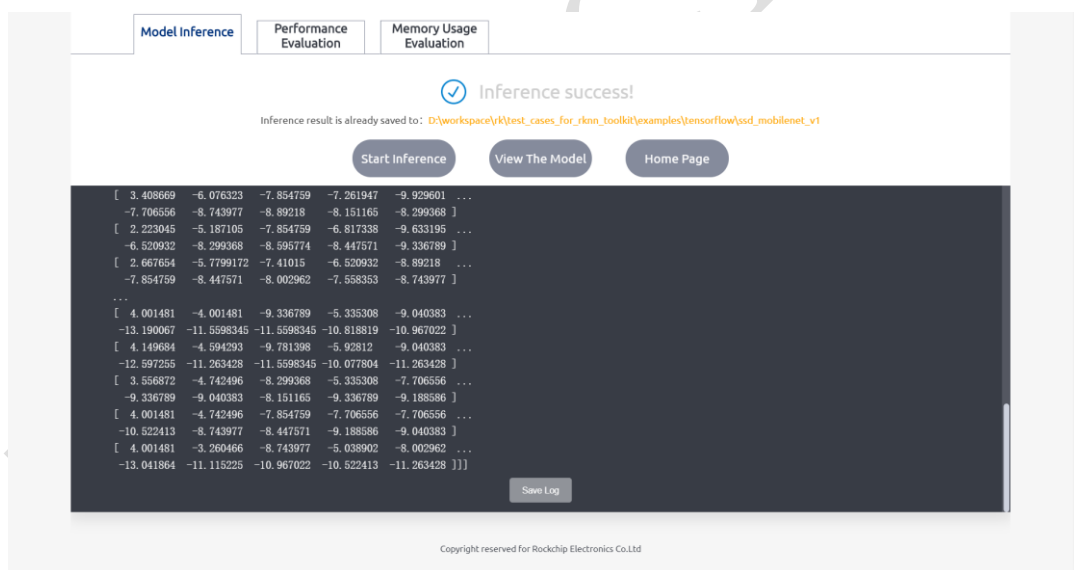


Figure 4-3-2-2 RKNN model inference

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.

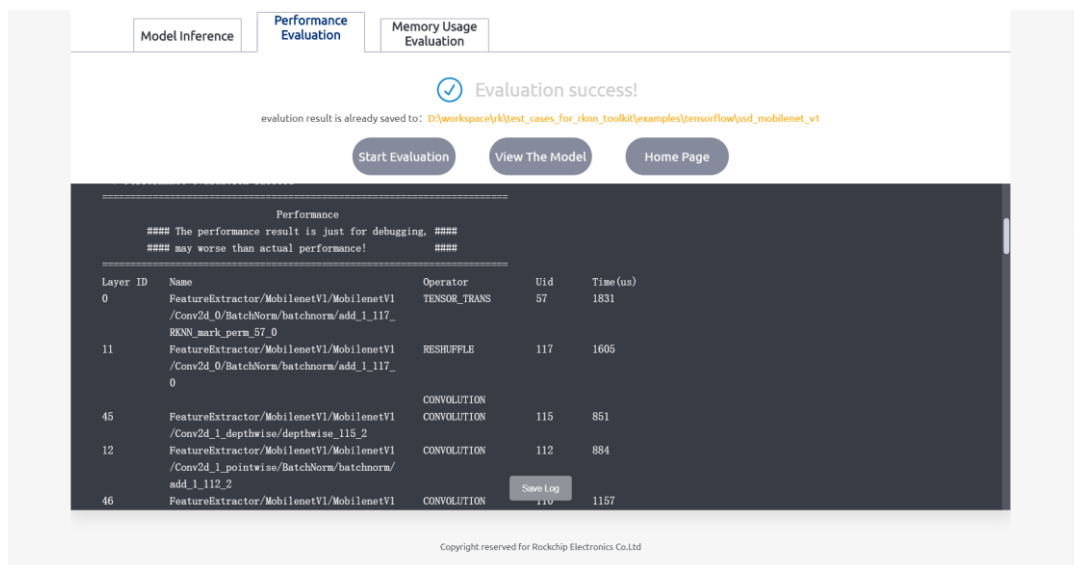


Figure 4-3-2-3 RKNN performance evaluation

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.

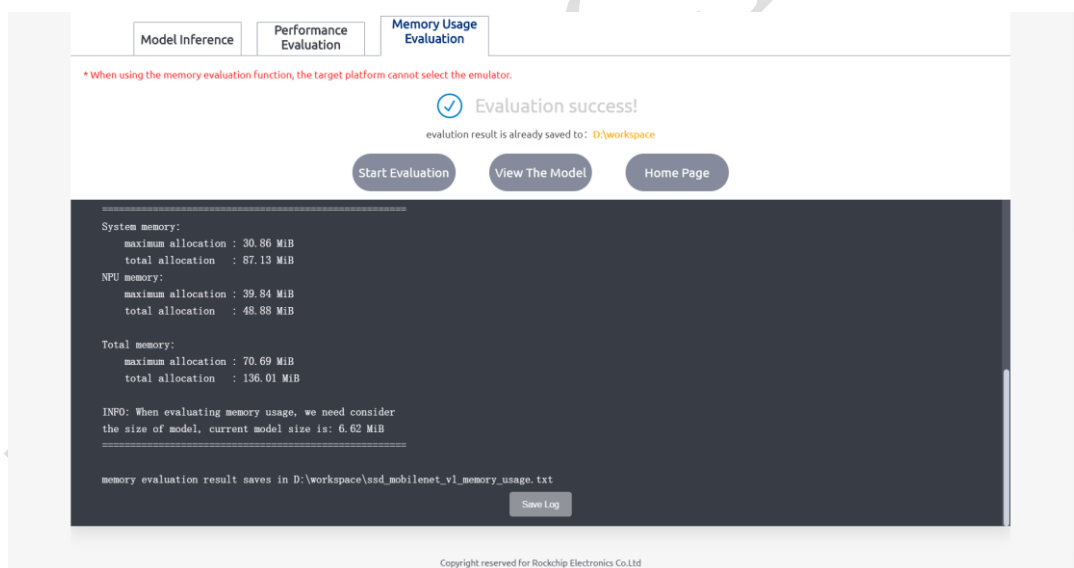


Figure 4-3-2-4 RKNN memory usage evaluation

5 Reference

Please refer to <Rockchip_User_Guide_RKNN_Toolkit_EN.pdf> to learn more about how to use RKNN Toolkit.

Please refer to <Rockchip_Quick_Start_RKNN_Toolkit_EN.pdf> for the installation method of RKNN Toolkit for each platform.

Rockchip