

Ingenic<sup>®</sup>

## Magik Training Quantization Guide

Date: Feb. 2022

**Ingenic®**

**Magik Training Quantization Guide**

Copyright© Ingenic Semiconductor Co. Ltd 2022. All rights reserved.

**Release history**

<b>Date</b>	<b>Author</b>	<b>Revision</b>	<b>Change</b>
Feb. 2022	Heidi	1.0.1	First release

**Disclaimer**

This documentation is provided for use with Ingenic products. No license to Ingenic property rights is granted. Ingenic assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement except as provided for by Ingenic Terms and Conditions of Sale.

Ingenic products are not designed for and should not be used in any medical or life sustaining or supporting equipment.

All information in this document should be treated as preliminary. Ingenic may make changes to this document without notice. Anyone relying on this documentation should contact Ingenic for the current documentation and errata.

**Ingenic Semiconductor Co., Ltd.**

**Ingenic Headquarters, East Bldg. 14, Courtyard #10**  
**Xibeiwang East Road, Haidian District, Beijing, China,**  
**Tel: 86-10-56345000**  
**Fax:86-10-56345001**  
**Http: //www.ingenic.com**

## Contents

1. Abstract .....	1
2. Environmental requirements .....	1
3. Environmental installation steps .....	1
3.1 Ubuntu .....	1
3.2 GCC .....	1
3.3 Nvidia Driver .....	2
3.4 CUDA .....	3
3.5 CUDNN .....	3
3.6 Python .....	4
3.7 Pytorch .....	5
3.8 Torchvision .....	5
3.9 Magik Trainingkit .....	5
3.10 Others .....	6
4. Yolov5s Training Process .....	6
4.1 Downloading of coda and model .....	6
4.2 Data Preparation .....	7
4.3 Network Training .....	7
4.4 Model Testing .....	10
5. Model Transformation .....	11
5.1 From pt to onnx .....	11
5.2 From onnx to bin .....	12
6. Run on-board .....	14
6.1 Code compling .....	14
6.2 Run .....	错误! 未定义书签。
7. Check data .....	17
1. PC .....	17
2. Board .....	18
8. Q & A .....	19

## 1. Abstract

This Guide is mainly for novices who use the magik platform of Ingenic processing core. Here, we take the yolov5s-person under the pytorch framework as an example, at the same time, the target device is set to “T40”. The whole process is introduced in detail from environment construction, data preparation, network training, model transformation to final board operation, which aims to guide users to be familiar with the use method of our magik platform and get the board operation process clearly.

## 2. Environmental Requirements

- Linux
- GCC ( $\geq 5.4.0$ )
- Nvidia Driver
- CUDA ( $\geq 9.0$ )
- CUDNN
- Python ( $\geq 3.5$ )
- Pytorch ( $\geq 1.3$ )
- Torchvision

If all the above environments are available (the version may not be subject to fixed requirements), you can directly go to step 3.9.

## 3. Environmental Installation Steps

### 3.1 Ubuntu

- system : Ubuntu16.04

### 3.2 GCC

- Version: 5.4.0
- Specific Steps
  1. Check Version  
\$ gcc -v
  2. Download Link  
<http://ftp.gnu.org/gnu/gcc>
  3. Compile and Install

```

$ tar -zxvf gcc-5.4.0.tar.bz2
$ cd gcc-5.4.0
$ ./contrib/download_prerequisites
$ cd ..
$ mkdir gcc-build-5.4.0
$ cd gcc-build-5.4.0
$ ../gcc-5.4.0/configure --enable-checking=release --enable-languages=
c,c++ --disable-multilib
$ sudo make
$ sudo make install
4. Check again
$ gcc -v

```

### 3.3 Nvidia Driver

- Version:  $\geq 440$
- Specific Steps
  1. Download link  
<http://www.nvidia.cn/Download/index.aspx?lang=cn>
  2. Disable Nouveau third party drivers  
 open edit profile:  

```
$ sudo gedit /etc/modprobe.d/blacklist.conf
```

 add to the last line: `blacklist nouveau`  

```
$ sudo update-initramfs -u
```

```
$ reboot
```
  3. Installation  

```
$ lsmod | grep nouveau
```

```
$ sudo /etc/init.d/lightdm stop (or: sudo service lightdm stop)
```

```
$ sudo chmod a+x NVIDIA-Linux-x86_64-440.64.run
```

```
$ sudo ./NVIDIA-Linux-x86_64-440.64.run --no-opengl-files
```

```
$ sudo /etc/init.d/lightdm start (or : sudo service lightdm start) --no-opengl-files --no-x-check --no-nouveau-check
```
  4. Check Version  

```
$ reboot
```

```
$ nvidia-smi
```

If the required driver version appears, the installation is over.

NVIDIA-SMI 440.31				Driver Version: 440.31				CUDA Version: 10.2			
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr. ECC					
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.					
0	GeForce GTX 108...	Off	00000000:02:00.0	Off	0%	N/A					
0%	35C	P8	10W / 250W	0MiB / 11178MiB		Default					
1	GeForce GTX 108...	Off	00000000:03:00.0	Off	0%	N/A					
0%	50C	P2	60W / 250W	829MiB / 11178MiB		Default					
2	GeForce GTX 108...	Off	00000000:83:00.0	Off	0%	N/A					
0%	39C	P8	9W / 250W	10MiB / 11177MiB		Default					

### 3.4 CUDA

- Version: 10.0
- Specific Steps
  1. Download link  
<https://developer.nvidia.com/cuda-10.0-download-archive?>  
Choose: linux—x86\_64—Ubuntu—16.04—runfile(local)  
Download: cuda\_10.0.130\_410.48\_linux.run

#### CUDA Toolkit 10.0 Archive

Select Target Platform ⓘ

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System: Windows, Linux, Mac OSX

Architecture ⓘ: x86\_64, ppc64le

Distribution: Fedora, OpenSUSE, RHEL, CentOS, SLES, Ubuntu

Version: 18.04, 16.04, 14.04

Installer Type ⓘ: runfile (local), deb (local), deb (network), cluster (local)

Download Installers for Linux Ubuntu 16.04 x86\_64

The base installer is available for download below.  
There is 1 patch available. This patch requires the base installer to be installed first.

Base Installer [Download (2.0 GB) ⬇]

Installation Instructions:

1. Run `sudo sh cuda_10.0.130_410.48_linux.run`
2. Follow the command-line prompts

Patch 1 (Released May 10, 2019) [Download (3.3 MB) ⬇]

In this patch we introduce new APIs for JPEG stream parsing and device and pinned memory control as well as a new hybrid decode API that decouples decoding process into pure host and device stages enabling more flexible control flow. The new APIs also support ROI decoding and 4 channel jpeg bitstreams.

2. Installation  
`$ sudo sh cuda_10.0.130_410.48_linux.run`
3. Check Version  
`$ cat /usr/local/cuda/version.txt`
4. CUDA Setting  
`$ export PATH=/usr/local/cuda/bin:$PATH`  
`$ export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH`

### 3.5 CUDNN

- Version: 7.6.5
- Specific Steps
  1. Download link  
<https://developer.nvidia.com/rdp/cudnn-download>  
cuDNN Library for Linux: cudnn-10.0-linux-x64-v7.6.5.32.tgz

2. Unzip
 

```
$ tar -zxvf cudnn-10.0-linux-x64-v7.6.5.32.tgz
cuda/include/cudnn.h
cuda/NVIDIA_SLA_cuDNN_Support.txt
cuda/lib64/libcudnn.so
cuda/lib64/libcudnn.so.7
cuda/lib64/libcudnn.so.7.6.5
cuda/lib64/libcudnn_static.a
```
3. Copy
 

```
$ sudo cp cuda/lib64/* /usr/local/cuda-10.0/lib64/
$ sudo cp cuda/include/* /usr/local/cuda-10.0/include/
```
4. Check version
 

```
$ cat /usr/local/cuda/include/cudnn.h | grep CUDNN_MAJOR -A 2
#define CUDNN_MAJOR 7
#define CUDNN_MINOR 6
#define CUDNN_PATCHLEVEL 5
#define CUDNN_VERSION (CUDNN_MAJOR * 1000 +
CUDNN_MINOR * 100 + CUDNN_PATCHLEVEL)
#include "driver_types.h"
```

### 3.6 Python

- Version: 3.7 (Please note that do not uninstall or delete the system comes with python2.7, so as not to affect the operation of the system.)
- Specific Steps
  1. Download Link
 

```
https://www.python.org/ftp/python/3.7.3/Python-3.7.3.tar.xz
```
  2. Compile & Install
 

```
$ tar -xvJf Python-3.7.3.tar.xz
$ cd Python-3.7.3
$ ./configure --prefix=/usr/local/bin/python3
$ sudo make
$ sudo make install
```
  3. Create Soft Link
 

```
$ ln -s /usr/local/bin/python3/bin/python3 /usr/bin/python3
$ ln -s /usr/local/bin/python3/bin/pip3 /usr/bin/pip3
```
  4. Check Version
 

```
$ python3 --version
```

### 3.7 Pytorch

- Version: 1.3.0
- Specific Steps
  1. Installation  
`$ pip3 install torch==1.3.0`
  2. Check Version  
`$ python3`  
`>> import torch`  
`>> print(torch.__version__)`

### 3.8 Torchvision

- Version: 0.4.2
- Specific Steps
  1. Installation  
`$ pip3 install torchvision==0.4.2`
  2. Check Version  
`$ python3`  
`>> import torchvision`  
`>> print(torchvision.__version__)`

### 3.9 Magik Trainingkit

- Version: 1.1.1
- Specific Steps
  1. Check Version  
`$ python`  
`>> import torch`  
`>> print(torch.__version__)` #version of torch  
`>> print(torch.version.cuda)` #version of cuda  
`>> print(torch.backends.cudnn.version())` #version of cudnn  
You can also set the other version of cuda and cudnn to meet your own demands.
  2. Get trainingkit  
You can get your trainingkit installation package(whl file) from our plugin directory “magik-toolkit/TrainingKit/pytorch/magik\_whl” according to the above environment and then you should install it by pip, for example, “pip3 install magik\_trainingkit\_torch\_130-1.1.1-py3-none-any.whl”.  
Please note that training installation package here is provided by us



according to the customer's environment. If there is any change and you cannot find the package in our directory, please inform the relevant technicians of the magik platform to compile the corresponding plug-ins which adapt to your new environment.

If you only update the previous installation package, you need to uninstall the old installation package first.

### 3. Check

```
>>> from ingenic_magik_trainingkit.QuantizationTrainingPlugin.python import ops
INFO(magik): trainingkit version:1.1.1(00010101_84f712d) built:20220121-1849(5.4.0 pytorch)
>>>
```

If the above figure appears, it indicates that the import is successful. The green font contains the version of the trainingkit, the commit number and the compilation date.

## 3.10 Others

Other packages:

pandas (\$ pip install pandas)

requests (\$ pip install requests)

cv2 (\$ pip install opencv-python)

yaml (\$ pip install pyyaml)

tqdm (\$ pip install tqdm)

matplotlib (\$ pip install matplotlib)

seaborn (\$ pip install seaborn)

If other installation packages are missing at runtime, please install them by “pip install” according to the prompt until normal operation.

## 4. YOLOv5s Training Process

### 4.1 Downloading of code and model

1. Download link of yolov5(official):

<https://github.com/ultralytics/yolov5.git>

2. Our yolov5 code(modified based on the official)

Since the training plug-in re encapsulates the relevant operators (such as convolution, depthwise-convolution and etc), we will provide a whole set of training code yolov5s-person based on the original yolov5. Our yolov5 code is in the directory “magik-toolkit/Models/training/pytorch/yolov5s-person/”, which the process of training, conversion and operation on board described below are completed in this directory.

yolov5s-person-4bit.pt is used for later PC end testing and board end testing, The model path is under in “magik-toolkit/Models/training/pytorch/yolov5s-person/runs/train”.

## 4.2 Data Preparation

Take coco2017 as an example, We need to extract the person (id:0) and convert it into the data format required for Yolo training.

Firstly, please download and decompress the coco2017 data set, the format of the annotation file is json. We are supposed to use wget to download the pictures and labels of coco2017 from the link below.

Pictures:

<http://images.cocodataset.org/zips/train2017.zip>

<http://images.cocodataset.org/zips/test2017.zip>

<http://images.cocodataset.org/zips/val2017.zip>

Labels:

[http://images.cocodataset.org/annotations/stuff\\_annotations\\_trainval2017.zip](http://images.cocodataset.org/annotations/stuff_annotations_trainval2017.zip)

[http://images.cocodataset.org/annotations/image\\_info\\_test2017.zip](http://images.cocodataset.org/annotations/image_info_test2017.zip)

[http://images.cocodataset.org/annotations/annotations\\_trainval2017.zip](http://images.cocodataset.org/annotations/annotations_trainval2017.zip)

Secondly, The script for generating data is in COCO\_forYOLO folder.

```
$ python batch_split_annotation_foryolo.py
```

Pay attention to modify the coco path by setting 'coco\_data\_dir=' in the file.

After running, three folders (person/data/images, person/data/ImageSets and person/data/labels) will be generated in the coco path, then please put train2017.txt, val2017.txt and test2017.txt under Imagesets into the folder “persondet/data/coco” and using absolute path of the pictures in the txt file.

## 4.3 Network Training

### 1. Training configuration

Please check the default parameters like is\_quantize, bitw, bita and etc in “models/commom.py” for reference.

```

bita = 32

if bita==32:
    bitw = 32
    is_quantize = 0
    clip_max_value = 6.0
    shortcut_clip_max_value = 2.0
elif bita==8:
    bitw = 8
    is_quantize = 1
    clip_max_value = 6.0
    shortcut_clip_max_value = 2.0
elif bita==4:
    bitw = 4
    is_quantize = 1
    clip_max_value = 4.0
    shortcut_clip_max_value = 1.5

weight_factor = 3.0
target_device = "T40"

```

When you want to train model of 32-bit, please set bita to 32. Similarly, you can set bita to 8 if you are going to train model of 8-bit. Of course, 4-bit is in the same way.

Parameter interpretation:

is\_quantize -- quantize or not, note that set it to 0 when bita is 32.

bitw - bitwidth of weight

bita - bitwidth of feature

clip\_max\_value - clip value of feature when quantization

shortut\_clip\_max\_value -- clip value of feature in shortcut when quantization

## 2. Training script

```
$ sh yolov5s-person/train.sh
```

```
export NCCL_IB_DISABLE=1
```

```
export NCCL_DEBUG=info
```

```
GPUS=6
```

```
python3 -m torch.distributed.launch --nproc_per_node=$GPUS
```

```
--master_port=60051 train.py \
```

```
--data data/coco-person.yaml \
```

```
--cfg models/yolov5s.yaml \
```

```
--weights '' \
```

```
--batch-size 132 \
```

```
--hyp data/hyp.scratch.yaml \
```

```
--project ./runs/train/yolov5s-person-32bit \
```

```
--epochs 300 \
```

```
--device 0,1,2,3,4,5
```

### (1) About pre training model

There is no pre training model in floating-point training, so -weights is ' ', 32bit model with sufficient accuracy is loaded in 8bit training, and 8bit trained is loaded in 4bit as the pre training model. In this way, the effect is better step by

step.

#### (2) About multi-GPU training

We added “torch.distributed.launch” during training to support multi-GPU training. The value of GPUs corresponds to the total number of devices below, which can be modified according to the actual situation. If it is single gpu training, directly use “python3 train.py” plus the following parameters. Batch size is the total batch number of all GPUs, which can be set according to the actual size of the GPUs.

#### (3) About learning rate

The super parameters are set in file data/hyp.scratch.yaml, lr0 is the initial learning rate, and the rest adopts the default value. We’ve removed the part about loading optimizer in file train.py, the low bit model re training based on 32-bit model instead of subsequent training, so that we can avoid affecting the training effect of low bit. We need to know that “ema” and “epoch” also have similar problems, so the corresponding modifications have been made as shown in the figure below.

```
# Resume
start_epoch, best_fitness = 0, 0.0
if pretrained:
    # Optimizer
    #if ckpt['optimizer'] is not None:
    #    optimizer.load_state_dict(ckpt['optimizer'])
    #    best_fitness = ckpt['best_fitness']

    # EMA
    #if use_ema and ema and ckpt.get('ema'):
    #    ema.ema.load_state_dict(ckpt['model'].float().state_dict())
    #    ema.updates = ckpt['updates']

    # Epochs
    # start_epoch = ckpt['epoch'] + 1

    if resume:
        start_epoch = ckpt['epoch'] + 1
        assert start_epoch > 0, '%s training to %g epochs is finished, nothing to resume.' % (weights, epochs)
```

#### (4) About saving model

We can use “--project” in train.py to set the saved directory about training model, there will be two models under runs/train/projec/weights: best.pt and last.pt, obviously, best.pt is the best training so far, and last.pt the latest training so far. The results of each epoch are saved in result txt, which can be viewed easily.

#### (5) Some experiences about training

About yov5s-person, our experiences at present are that 32-bit uses sgd and lr0.01, 8bit uses sgd and lr0.01 for pre training based on 32bit, 4bit uses adam and lr0.001 for pre training based on 8bit. Please see train.sh to get specific commands.

Since 32-bit has no pre training model, the convergence is slightly slow; 8bit

has pre training and can express, and the convergence is fast; Although 4bit has pre training, its expression ability is slightly weak and its convergence is slightly slow.

## 4.4 Model Testing

### 1. Test picture(s)

Through run “python detect.py -h” to view and select the required parameters. Set “--source” to detect pictures or videos or picture folders. And the detection results can be set to display (- - view-img) or save (- - save-img)

- Image: `--source file.jpg`
- Video: `--source file.mp4`
- Directory: `--source dir/`

The usage is as follows:

```
$ sh detect.sh(python detect.py --source data/images/bus.jpg \
--weights ./runs/train/yolov5s-person-4bit.pt \
--imgs 640 --device 0 --view-img)
```

Parameter interpretation:

- source - picture(s) or video to be tested
- weights - trained model for testing
- imgs - dest size when testing
- device - test with which GPU
- view-img - display image or not

Please make sure the detected model configurations are consistent with the training configuration.

### 2. Test accuracy

```
$ sh test.sh(python test.py --data data/coco-person.yaml \
--weights ./runs/train/yolov5s-person-4bit.pt \
--imgs 640 --device 0 --batch-size 40)
```

The model to be tested is specified by “--weights”, and the verification set is specified by val2017.txt in data/coco-person.yaml, and other parameters are given according to actual needs.

Our testing accuracy are as follows, with resolution is 640x640.

	Class	Images	Targets	P	R	mAP@0.5	mAP@.5:.95
32bit:	all	5000	11004	0.771	0.615	0.700	0.422
8bit:	all	5000	11004	0.751	0.638	0.706	0.430
4bit:	all	5000	11004	0.786	0.602	0.698	0.419

As you can see, we’ve also provided yolov5m.yaml and yolov5l.yaml under yolov5s-person/models, so that you can train your own yolov5m or yolov5s like the process of yolov5s we we introduced above to meet your needs.

## 5. Model Transformation

### 5.1 From pt to onnx

Please pay attention to that, the onnx file must be generated in the above training environment.

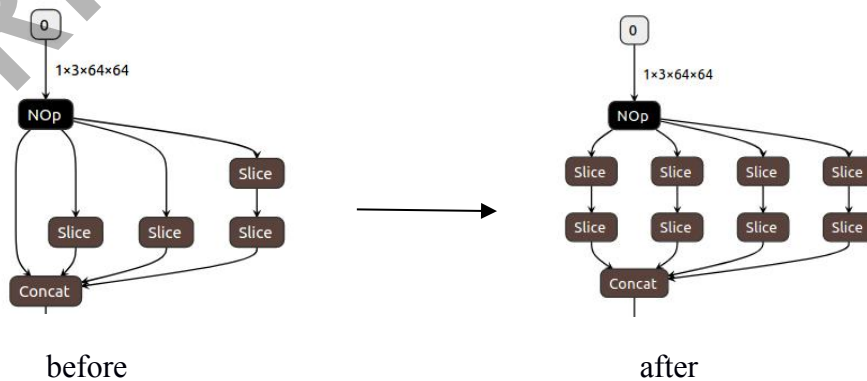
```
$ sh convert_onnx.sh(python convert_onnx.py \  
--weights ./runs/train/yolov5s-person-4bit.pt)
```

In this script, input is the path of pt -- the model we are looking forward to convert, and output is the onnx file that we want.

We are supposed to use opset9 when convert pt to onnx, so that we can use transform tool successfully later. If there are some errors occur when you set opset9, Please refer to the question answer in step 8. Different versions of torch will have some different problems.

```
def _slice(g, input, axes, starts, ends):  
    assert len(starts) == len(ends)  
    # if len(starts) == 1 and starts[0] == 0 and ends[0] == 9223372036854775807:  
    #     return input  
    return g.op("Slice", input, axes_i=axes, starts_i=starts, ends_i=ends)
```

In particular, for yolov5, we need to modify function \_slice() in torch/onnx/symbolic\_opset9.py like above picture before using the focus operator to onnx. Or we will get wrong focus operation, we can do visual view with netron as follows.



## 5.2 From onnx to bin

This step does not depend on python, torch, plug-ins or other environments, but only our conversion tool “magik\_transformer-\*-py3-none-any.whl” (which is under magik\_toolkit/TransFormKit) and run\_t40/t41/a1/x2500.sh.

After you get the “magik\_transformer-\*-py3-none-any.whl”, you can directly use “pip install magik\_transformer-\*-py3-none-any.whl” to install it. If you have installed the “magik\_transformer-\*-py3-none-any.whl” before, you need to use “pip uninstall magik\_transformer-\*-py3-none-any.whl” to uninstall the conversion tool first.

```
cd Models/fqat/pytorch/yolov5s-person/transform_sample
```

```
$ sh run_t40/t41/a1/x2500.sh
```

Different types of chips use different run\_ \* sh, take T40 chip as an example, Specifically:

```
run_t40.sh
```

```
cd ../
sed -i 's/(target_device = "[^"]*"|"[^"]*"|"/\1140/' models/common.py models/yolo.py
sh detect.sh
sh convert_onnx.sh
cd -
CUDA_VISIBLE_DEVICES=0 python transform.py --model_file ../runs/train/yolov5s-person-4bit.onnx --output_file ./venus_sample_yolov5s/yolov5s_t40_magik.mk.h --config_file cfg/
magik_t40.cfg
cd venus_sample_yolov5s
cp makefile_files/Makefile_t40 Makefile
```

First, replace the target\_device in the network with the corresponding chip model, use detect.py in the previous step to generate a 4bit onnx model file. Then execute the final conversion command transform.py, the specific conversion model parameters are as follows:

--model\_file input of transformation, corresponding to the onnx model file path obtained in the previous step.

--output\_file output of transformation, we will get yolov5s-person-4bit.bin in this directory after running

--config\_file configuration file at the time of conversion

Different chips have different config files, mainly due to different SOC devices, select different cfg files when using, take T40 as an example.

magik\_t40.cfg



```

"ARCH": {
  "SOC": "T40"
},
"MODEL": {
  "FORMAT": "onnx",
  "INPUT": [
    {
      "BATCH": 1,
      "WIDTH": 640,
      "HEIGHT": 640,
      "CHANNEL": 3,
      "COLOR": "RGB",
      "NORMAL": [255,255,255],
      "MEAN": [0,0,0]
    }
  ]
},
"EXTRA": {
  "BIZ_CODE": "Ingenic",
  "VERBOSE": "ON"
}
}

```

SOC -- set same as target\_device during training  
 FORMAT -- imported model format  
 INPUT -- input information of the model  
 BATCH -- number of inputs to the model  
 WIDTH -- input's width of running on-board  
 HEIGHT -- input's height of running on-board  
 CHANNEL -- input's channels of running on-board  
 COLOR -- input picture format("BGR"/"RGB"/"GRAY")  
 NORMAL -- variance of preprocessing during training  
 MEAN -- mean value of preprocessing during training

```

2022-07-21 12:22:38.484535: W clip_model_by_nop_optimizer.cc:48-collect_remove_nodes] Multiple output NOp exists on the current pathway!
2022-07-21 12:23:46.524015: W quantize_device.cc:55-get asynchronous quantize type] weight quantize devices is empty
2022-07-21 12:23:46.810610: W quantize_operation.cc:521-search] Not found QuantizeOperation function for Unpool2D
2022-07-21 12:23:46.826601: W quantize_operation.cc:521-search] Not found QuantizeOperation function for Unpool2D
*****
^ ^ Convert successfully, Enjoy it ^ ^
*****

```

Signs of success

The logo in the above figure will appear after the conversion is successful. The "yolov5s\_t40\_magik.bin" which is running on-board will be generated in "venus\_sample\_yolov5s/"

```

10_w1024_h714.nv12  magik_model_yolov5s_t40_magik.mk.h  std
bus.jpg             Makefile                                           yolov5s_t40_magik.bin
inference.cpp       makefile_files
inference_nv12.cpp  readme.md

```



## 6. Run on-board

### 6.1 Code compiling

#### 1. Preparation

we also provide inference.cpp, Makefile and test data, which are necessary when run model on board. In addition, we also need venus library and mips compilation tool. You may need to know that venus library is under the library “magik-toolkit/InferenceKit/”, and mips compilation tool is provided by our solutions’ colleagues.

#### 2. Input of network

Here, in order to all of you can realize the process quickly, we’ve added some functions under the library “venus\_sample\_yolov5s/stb”, so we can test jpg diagram directly on board.

#### 3. Model loading

The model in inference.cpp is passed in through parameters. So please pay attention to synchronously copying it to the corresponding directory on the board end and passing it in at run time.

#### 4. Setting of super parameters

```
void generateBBox(std::vector<venus::Tensor> out_res, std::vector<magik::venus::ObjBbox_t> & candidate_boxes, int img_w, int img_h)
{
    float person_threshold = 0.3;
    int classes = 1;
    float nms_threshold = 0.6;
    std::vector<float> strides = {8.0, 16.0, 32.0};
    int box_num = 3;
    std::vector<float> anchor = {10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326};
```

These settings should be consistent with the test code on the PC end.

#### 5. Compile

TOPDIR -- directory of venus

Libtype -- muclibc or other(mglibc)

build\_type -- release (Default setting.)

-- profile (Visualization of network structure at runtime and statistics of running time and GOPs at each layer of the network.)

-- debug (Save the results of quantization features of each layer.)

-- nmem (Count the memory usage of nmem when the model is running, and it's in /tmp/nmem\_memory.txt.)

Directly compile inference.cpp can generate venus\_yolov5s\_bin\_uclibc\_\*, and that is the executable file we need on the board.

We also provide the code inference\_nv12.cpp and input data 10\_w1024\_h714\_nv12 when you want to use nv12 type input. If necessary, just modify Makefile

for compilation and use test.

VENUS library corresponding to different SOC:

Soc	venus
T40	InferenceKit/nnal/mips720-glibc229/
T41	InferenceKit/nnal/mips720-glibc229/T41/
A1	InferenceKit/nnal/mips720-glibc229/A1/
X2500	InferenceKit/nnal/mips720-glibc229/

## 6.2 Run

Venus library:

magik-toolkit/InferenceKit/nnal/mips720-glibc229/

### 1. release(default)

\$ make build\_type=release

(use “make build\_type=release clean” if you want to clean)

We can get executable file `venus_yolov5s_bin_uclibc_release` after compile, then copy `libvenus.so`, `venus_yolov5s_bin_uclibc_release`, `yolov5s_t40_magik.bin` and `bus.jpg` to board.

When you enter the end of the board, please add library path first(`export LD_LIBRARY_PATH=$lib_path:$LD_LIBRARY_PATH`), after that , you just need run on board by command “`./venus_yolov5s_bin_uclibc_release yolov5s_t40_magik.bin bus.jpg`”. The operation results are shown in the figure below.

```
[root@ingenic-uc1_1:v5s-fx]# ./venus_yolov5s_bin_uclibc_release yolov5s_t40_magik.bin bus.jpg
The soc-nna version is 20220525
INFO(magik): venus memory map size: 0
INFO(magik): venus version:0.9.6.1.ALPHA(00000906_184a23e) built:20220715-1559(7.2.0 r5.1.3 glibc2.29 mips@NNA1)
INFO(magik): model version:0.9.6.NNA1_c2436c4
[I/magik:venus]: kv_size = 0
ori image w,h: 810 ,1080
model-->640 ,640 4
input shape:
-->384 640
scale--> 0.355556
resize padding over:
resize valid_dst, w:288 h 384
padding info top :0 bottom 0 left:176 right:176
test_net run time_ms:38.919000ms
pad_x:176 pad_y:0 scale:0.355556
post_net time_ms:1.046000ms
box: 51 408 239 904 0.90
box: 217 401 351 869 0.83
box: 669 409 811 891 0.67
[root@ingenic-uc1_1:v5s-fx]#
```

### 2.debug(check data)

See step 8.2 for details. The input processing is a little different.

### 3. profile(network visualization)

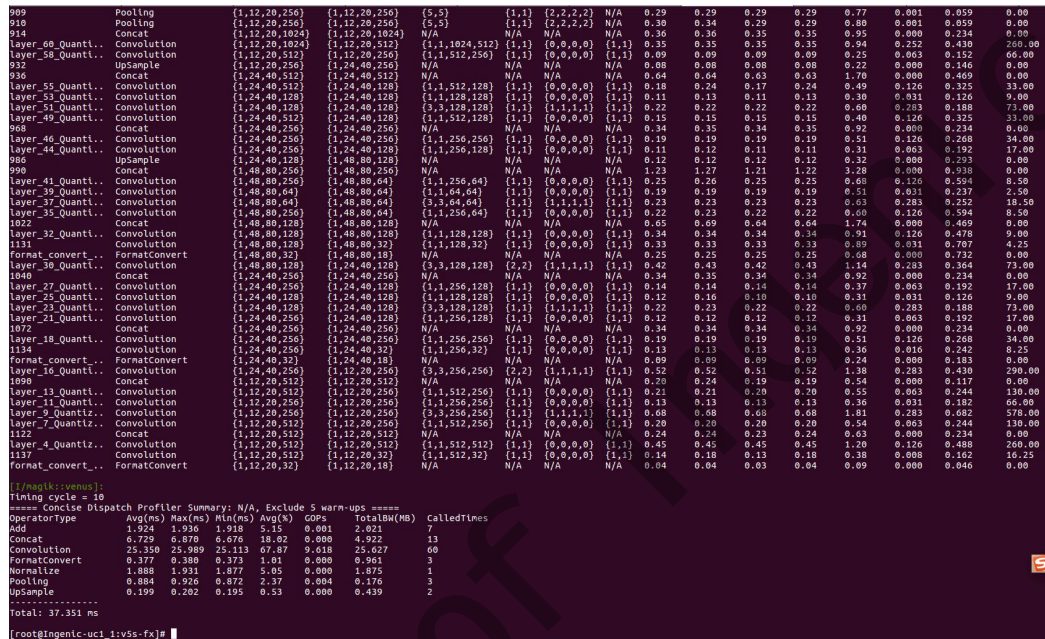
\$ make build\_type=profile

(use “make build\_type=profile clean” if you want to clean)

We can get executable file `venus_yolov5s_bin_uclibc_profile` after compile,

then copy libvenus.p.so, venus\_yolov5s\_bin\_uclibc\_profile, bus.jpg and yolov5s\_t40\_magik.bin to board.

When you enter the end of the board, please add library path first (export LD\_LIBRARY\_PATH=\$lib\_path:\$LD\_LIBRARY\_PATH), after that, you just need run on board by command “./venus\_yolov5s\_bin\_uclibc\_profile yolov5s\_t40\_magik.bin bus.jpg”. The operation results are shown in the figure below.



OperatorType	Avg(ms)	Max(ms)	Min(ms)	Avg(%)	GOps	TotalMB(MB)	CalledTimes
Add	1.924	1.936	1.918	5.15	0.001	2.021	7
Concat	6.720	6.876	6.676	18.02	0.000	0.922	13
Convolution	25.350	25.989	25.113	67.87	9.618	25.627	60
FormatConvert	0.377	0.380	0.373	1.01	0.000	0.961	3
Normalize	1.808	1.931	1.677	5.05	0.000	0.875	1
Pooling	0.884	0.926	0.872	2.37	0.004	0.176	3
Upsample	0.199	0.202	0.195	0.53	0.000	0.439	2

#### 4. nmem(nmem occupancy)

\$ make build\_type=nmem

(use “make build\_type=nmemclean” if you want to clean)

We can get executable file venus\_yolov5s\_bin\_uclibc\_release after compile, then copy libvenus.m.so, venus\_yolov5s\_bin\_uclibc\_nmem, yolov5s\_t40\_magik.bin and bus.jpg to board.

When you enter the end of the board, please add library path first (export LD\_LIBRARY\_PATH=\$lib\_path:\$LD\_LIBRARY\_PATH), after that, you just need run on board by command “./venus\_yolov5s\_bin\_uclibc\_nmem yolov5s\_t40\_magik.bin bus.jpg”. The operation results are shown in the figure below.

```
[root@ingenic-uc1_1:v5s-fx]# ./venus_yolov5s_bin_uclibc_nmem yolov5s_t40_magik.bin bus.jpg
The soc-nna version is 20220525
INFO(magik): venus memory map size: 0
INFO(magik): venus version:0.9.6.1.ALPHA(00000906_184a23e) built:20220715-1546(7.2.0 r5.1.3 glibc2.29 mips@NNA1)
INFO(magik): model version:0.9.6.NNA1_c2436c4
[! /magik::venus]: kv_size = 0
ori_image w,h: 810 ,1080
model-->640 ,640 4
input shape:
-->384 640
scale--> 0.355556
resize padding over:
resize valid_dst, w:288 h 384
padding info top :0 bottom 0 left:176 right:176
test_net run time_ms:375.395000ms
pad_x:176 pad_y:0 scale:0.355556
post_net time_ms:1.414000ms
box: 51 408 239 904 0.90
box: 217 401 351 869 0.83
box: 669 409 811 891 0.67
[root@ingenic-uc1_1:v5s-fx]#
```

## 7. Check Data

To verify whether the data of PC end and board end can be aligned, you can do the following:

### 1. PC

When we test single picture like Step 4.4-1, we can add the environment variable “MAGIK\_TRAININGKIT\_DUMP=1” to save the quantization results of each layer to “/tmp/trainingkit\_data/feature”, what’s more, you can also set the environment variable “MAGIK\_TRAININGKIT\_PATH” specify the directory to save. The specific running command is as below.

```
$ MAGIK_TRAININGKIT_DUMP=1
MAGIK_TRAININGKIT_PATH="/." python detect.py \
--source data/images/bus.jpg \
--weights ./runs/train/yolov5s-person-4bit.pt \
--imgs 640 --device 0
```

The original resolution of bus.jpg is 810x1080. During the test, the target size “--imgs” is set to 640. According to the scaling principle of yolov5 code (the long side is 640, the short side is scaled in equal proportion, and then filled to a multiple of 32), so the resolution of the final test is 480x640. The operation results of this network are finally saved under “./trainingkit\_data/feature” (see the figure below for details), it can be seen that the input layer is saved as input\_data\_shape\_1\_640\_480\_3.bin, among them, height is 640, width is 480, channel is 3. As you can see in the picture, we also save the last three output layers(out\_input\_index\_\*.bin), which can be used to check the network output.



```

input_data shape 1 640 480 3.bin
layer 101 QuantizeFeature.bin
layer 103 QuantizeFeature.bin
layer 106 QuantizeFeature.bin
layer 108 QuantizeFeature.bin
layer 10 QuantizeFeature.bin
layer 110 QuantizeFeature.bin
layer 112 QuantizeFeature.bin
layer 114 QuantizeFeature.bin
layer 116 QuantizeFeature.bin
layer 119 QuantizeFeature.bin
layer 121 QuantizeFeature.bin
layer 123 QuantizeFeature.bin
layer 125 QuantizeFeature.bin
layer 127 QuantizeFeature.bin
layer 129 QuantizeFeature.bin
layer 12 QuantizeFeature.bin
layer 14 QuantizeFeature.bin
layer 15 QuantizeFeature.bin
layer 17 QuantizeFeature.bin
layer 19 QuantizeFeature.bin
layer 21 QuantizeFeature.bin
layer 23 QuantizeFeature.bin
layer 25 QuantizeFeature.bin
layer 27 QuantizeFeature.bin
layer 28 QuantizeFeature.bin
layer 2 QuantizeFeature.bin
layer 30 QuantizeFeature.bin
layer 32 QuantizeFeature.bin
layer 33 QuantizeFeature.bin
layer 35 QuantizeFeature.bin
layer 37 QuantizeFeature.bin
layer 38 QuantizeFeature.bin
layer 40 QuantizeFeature.bin
layer 42 QuantizeFeature.bin
layer 44 QuantizeFeature.bin
layer 46 QuantizeFeature.bin
layer 48 QuantizeFeature.bin
layer 4 QuantizeFeature.bin
layer 50 QuantizeFeature.bin
layer 51 QuantizeFeature.bin
layer 53 QuantizeFeature.bin
layer 55 QuantizeFeature.bin
layer 56 QuantizeFeature.bin
layer 58 QuantizeFeature.bin
layer 60 QuantizeFeature.bin
layer 61 QuantizeFeature.bin
layer 63 QuantizeFeature.bin
layer 65 QuantizeFeature.bin
layer 67 QuantizeFeature.bin
layer 69 QuantizeFeature.bin
layer 6 QuantizeFeature.bin
layer 71 QuantizeFeature.bin
layer 73 QuantizeFeature.bin
layer 75 QuantizeFeature.bin
layer 77 QuantizeFeature.bin
layer 80 QuantizeFeature.bin
layer 82 QuantizeFeature.bin
layer 84 QuantizeFeature.bin
layer 86 QuantizeFeature.bin
layer 88 QuantizeFeature.bin
layer 8 QuantizeFeature.bin
layer 90 QuantizeFeature.bin
layer 93 QuantizeFeature.bin
layer 95 QuantizeFeature.bin
layer 97 QuantizeFeature.bin
layer 99 QuantizeFeature.bin
output_index 1 shape 1 80 60 18.bin
output_index 2 shape 1 40 30 18.bin
output_index 3 shape 1 20 15 18.bin

```

## 2.Board

You'd better clear the previously compiled mode by “make build\_type=clean” before compiling. Then compile it by command “make build\_type=debug”. We can get executable file venus\_yolov5s\_bin\_uclibc\_debug after compile, then copy libvenus.d.so, venus\_yolov5s\_bin\_uclibc\_debug, yolov5s\_t40\_magik.bin to board.

When you enter the end of the board, please add library path first (export LD\_LIBRARY\_PATH=\$lib\_path:\$LD\_LIBRARY\_PATH), after that, you just need run on board by command “./venus\_yolov5s\_bin\_uclibc\_debug yolov5s\_t40\_magik.bin magik\_input\_nhwc\_1\_640\_480\_3.bin”. The operation results are shown in the figure below

```

layer 103 QuantizeFeature_out.bin
layer 106 QuantizeFeature_bt.bin
layer 106 QuantizeFeature_out.bin
layer 106 QuantizeFeature_weight.bin
layer 109 QuantizeFeature_out.bin
layer 109 QuantizeFeature_weight.bin
layer 111 QuantizeFeature_out.bin
layer 111 QuantizeFeature_weight.bin
layer 114 QuantizeFeature_bt.bin
layer 114 QuantizeFeature_out.bin
layer 114 QuantizeFeature_weight.bin
layer 116 QuantizeFeature_out.bin
layer 116 QuantizeFeature_weight.bin
layer 118 QuantizeFeature_bt.bin
layer 118 QuantizeFeature_out.bin
layer 118 QuantizeFeature_weight.bin
layer 120 QuantizeFeature_bt.bin
layer 120 QuantizeFeature_out.bin
layer 120 QuantizeFeature_weight.bin
layer 122 QuantizeFeature_bt.bin
layer 122 QuantizeFeature_out.bin
layer 122 QuantizeFeature_weight.bin
layer 125 QuantizeFeature_bt.bin
layer 125 QuantizeFeature_out.bin
layer 125 QuantizeFeature_weight.bin
layer 136 QuantizeFeature_out.bin
layer 136 QuantizeFeature_weight.bin
layer 13 QuantizeFeature_bt.bin
layer 13 QuantizeFeature_out.bin
layer 13 QuantizeFeature_weight.bin
layer 16 QuantizeFeature_out.bin
layer 16 QuantizeFeature_weight.bin
layer 18 QuantizeFeature_bt.bin
layer 18 QuantizeFeature_out.bin
layer 18 QuantizeFeature_weight.bin
layer 21 QuantizeFeature_bt.bin
layer 21 QuantizeFeature_out.bin
layer 21 QuantizeFeature_weight.bin
layer 23 QuantizeFeature_bt.bin
layer 23 QuantizeFeature_out.bin
layer 23 QuantizeFeature_weight.bin
layer 25 QuantizeFeature_bt.bin
layer 25 QuantizeFeature_out.bin
layer 25 QuantizeFeature_weight.bin
layer 27 QuantizeFeature_bt.bin
layer 27 QuantizeFeature_out.bin
layer 27 QuantizeFeature_weight.bin
layer 30 QuantizeFeature_bt.bin
layer 30 QuantizeFeature_out.bin
layer 30 QuantizeFeature_weight.bin
layer 32 QuantizeFeature_bt.bin
layer 32 QuantizeFeature_out.bin
layer 32 QuantizeFeature_weight.bin
layer 35 QuantizeFeature_bt.bin
layer 35 QuantizeFeature_out.bin
layer 35 QuantizeFeature_weight.bin
layer 46 QuantizeFeature_out.bin
layer 46 QuantizeFeature_weight.bin
layer 49 QuantizeFeature_bt.bin
layer 49 QuantizeFeature_out.bin
layer 49 QuantizeFeature_weight.bin
layer 4 QuantizeFeature_bt.bin
layer 4 QuantizeFeature_out.bin
layer 4 QuantizeFeature_weight.bin
layer 51 QuantizeFeature_out.bin
layer 51 QuantizeFeature_weight.bin
layer 53 QuantizeFeature_out.bin
layer 53 QuantizeFeature_weight.bin
layer 55 QuantizeFeature_bt.bin
layer 55 QuantizeFeature_out.bin
layer 55 QuantizeFeature_weight.bin
layer 58 QuantizeFeature_bt.bin
layer 58 QuantizeFeature_out.bin
layer 58 QuantizeFeature_weight.bin
layer 60 QuantizeFeature_out.bin
layer 60 QuantizeFeature_weight.bin
layer 63 QuantizeFeature_bt.bin
layer 63 QuantizeFeature_out.bin
layer 63 QuantizeFeature_weight.bin
layer 65 QuantizeFeature_bt.bin
layer 65 QuantizeFeature_out.bin
layer 65 QuantizeFeature_weight.bin
layer 68 QuantizeFeature_bt.bin
layer 68 QuantizeFeature_out.bin
layer 68 QuantizeFeature_weight.bin
layer 79 QuantizeFeature_out.bin
layer 79 QuantizeFeature_weight.bin
layer 7 QuantizeFeature_bt.bin
layer 7 QuantizeFeature_out.bin
layer 7 QuantizeFeature_weight.bin
layer 82 QuantizeFeature_bt.bin
layer 82 QuantizeFeature_out.bin
layer 82 QuantizeFeature_weight.bin
layer 85 QuantizeFeature_out.bin
layer 85 QuantizeFeature_weight.bin
layer 87 QuantizeFeature_bt.bin
layer 87 QuantizeFeature_out.bin
layer 87 QuantizeFeature_weight.bin
layer 88 QuantizeFeature_bt.bin
layer 88 QuantizeFeature_out.bin
layer 88 QuantizeFeature_weight.bin
layer 90 QuantizeFeature_bt.bin
layer 90 QuantizeFeature_out.bin
layer 90 QuantizeFeature_weight.bin
layer 92 QuantizeFeature_bt.bin
layer 92 QuantizeFeature_out.bin
layer 92 QuantizeFeature_weight.bin
layer 95 QuantizeFeature_bt.bin
layer 95 QuantizeFeature_out.bin
layer 95 QuantizeFeature_weight.bin
layer 97 QuantizeFeature_bt.bin
layer 97 QuantizeFeature_out.bin
layer 97 QuantizeFeature_weight.bin
layer 99 QuantizeFeature_bt.bin
layer 99 QuantizeFeature_out.bin
layer 99 QuantizeFeature_weight.bin
layer 9 QuantizeFeature_bt.bin
layer 9 QuantizeFeature_out.bin
layer 9 QuantizeFeature_weight.bin

```

Where layer\_\*\_QuantizeFeature\_out.bin on board and layer\_\*\_QuantizeFeature.bin saved on PC, They correspond to each other one by one, You just need to check whether the md5 value of the two files is

consistent.

If the input is completely consistent, but there are two layers that do not correspond, please give feedback in time.

```
[root@Ingenic-uc1_1:v5s-fx]# md5sum layer_99_QuantizeFeature_out.bin
4b9cdd58b25ba5a6ae05b5235b9aa3d2 layer_99_QuantizeFeature_out.bin
[root@Ingenic-uc1_1:v5s-fx]#
```

```
(base) user@user-PR4768GW-Invalid-entry-length-16-Fixed-up-to-11:/tmp/trainingkit_data/features$ md5sum layer_99_QuantizeFeature.bin
4b9cdd58b25ba5a6ae05b5235b9aa3d2 layer_99_QuantizeFeature.bin
(base) user@user-PR4768GW-Invalid-entry-length-16-Fixed-up-to-11:/tmp/trainingkit_data/features$
```

In the debug mode, we only check the network results here, and there is no post-processing part, so there is not any results.

## 8. Q & A

1. Q: Why is the original resolution 810\*1080 of bus.jpg 640\*384 in release and 480\*640 in debug?

A: When we set the target resolution to 640, the scaling rule at the board end is to fill width to 640 after scaling in equal proportion (of course that can be modified), where 640 is width and 384 is height. But the rule in yolov5 code is to scale the long side by 640, and then fill the short side to a multiple of 32, so here 640 is height and 480 is width, and debug is consistent with the PC end. It can be set as needed in actual use.

2. Q: Similar errors occurred like “undefined symbol:

\_ZN6caffe28TypeMeta21\_typeMetaDataInstanceIN3c108BFloat16EEEEPKNS\_6detail12TypeMetaDataEv” while importing the magik package?

A: The version of torch in the current environment does not correspond to that of magik package.

3. Q: “ImportError: version ‘libcudart.so.10.1’ not found” ?

A: The version of cuda in the current environment does not correspond to that of magik package.

4. Q: Can the network with magik plug-ins load the native torch model for pre training?

A: The theory is OK as long as the parameters and their names can correspond one by one. However, the 32bit model based on the native float still needs to be fine tuned and trained. It is not recommended to directly load the native 8-bit model.

5. Q: It appears an error “RuntimeError: Address already in use” when training?

A: Specify an unused port such as “--master\_port=60053” after “python -m

`torch.distributed.launch`".

6. Q: When training, there is "expected scalar type Float but found Half" ?

A: You may use "amp.autocast" in torch to accelerates semi-precision training, but quantization does not support semi-precision training temporarily. Just comment out this. In addition, the original yolov5 saves the model according to `half()`. At present, it is found that the accuracy of the last saved model will be lost by quantifying, so it is recommended to remove it. In fact, we've done this in the code we provide.

7. Q: A error "step!=1 is currently not supported" when convert pt to onnx?

A: When we use Focus in yolov5, we may get this error after torch1.7, Please comment out the line corresponding to the error report(if + raise RuntimeError) in `torch/onnx/symbolic_opset9.py`.

8. Q: Error like "RuntimeError:input\_shape\_value==  
reshape\_value || input\_shape\_value==1 || reshape\_value == 1 INTERNAL  
ASSERT FAILED ....." when convert pt to onnx?

A: You may get this error after torch1.9 in yolov5, please modify the last parameter "onnx\_shape\_inference=True" in function `_export()` in file `torch/onnx/utils.py` to "onnx\_shape\_inference=False". Differnt versions my be on different lines, you just need to find out that.

9. Q: Warning like "Warning: Unsupported operator NOp. No schema registered for this operator." when convert pt to onnx after torch1.8?

A: It doesn't matter.

10. Q: An error occurs "After checking the input dimension, the input of concat cannot be concatenated!" when convert model from onnx to bin?

A: The problem about slice, please see step 5.1 to modify that (if+return).

11. Q: Errors like "error: 'constexpr bool std::isinf(double)' conflicts with a previous declaration" occurred while compiling C code?

A: The compiler of mipsconflicts with the built-in compiler of X86, please make sure the environment variable `CPLUS_INCLUDE_PATH` about "/usr/include/x86\_64-linux-gnu" is removed.

12. Q: "make: mips-linux-gnu-g++: Command not found" while compiling C code?

A: Mips compiler is not specified, please set it by export.

13. Q: What is the loss of board end accuracy and PC end accuracy?

A: Theoretically, the loss is very small(within one percentage point). If there

is a large gap, you can check whether the inputs are consistent, the training and verification data is BGR or RGB., the threshold setting is consistent or not. You can also compare the network output results to see whether the output is consistent. See Step 7 for details.

14. Q: Does the resolution of onnx, conversion tool and board end test need to be same?

A: It is better to keep consistent, especially the latter two. The reason why they are not consistent in this example is that they have no impact on this. For other non full convolution networks, it may affect the converted model. Therefore, when testing the accuracy of the verification set, it is recommended that the input data be processed offline to the same size, and then the PC end and board end are compared.