Ingenic®

Magik Training Quantization Guide

Date: Feb. 2022

Ingenic®

Magik Training Quantization Guide

Release history

Date	Author	Revision	Change
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

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 (>=5.4.0)
- Nvidia Driver
- CUDA (>=9.0)
- CUDNN
- Python (>=3.5)
- Pytorch (>=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

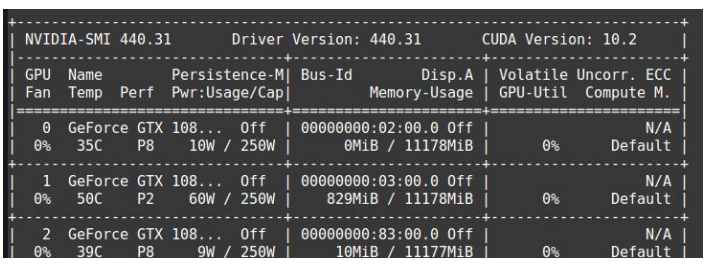
- Version：>= 440
- Specific Steps

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

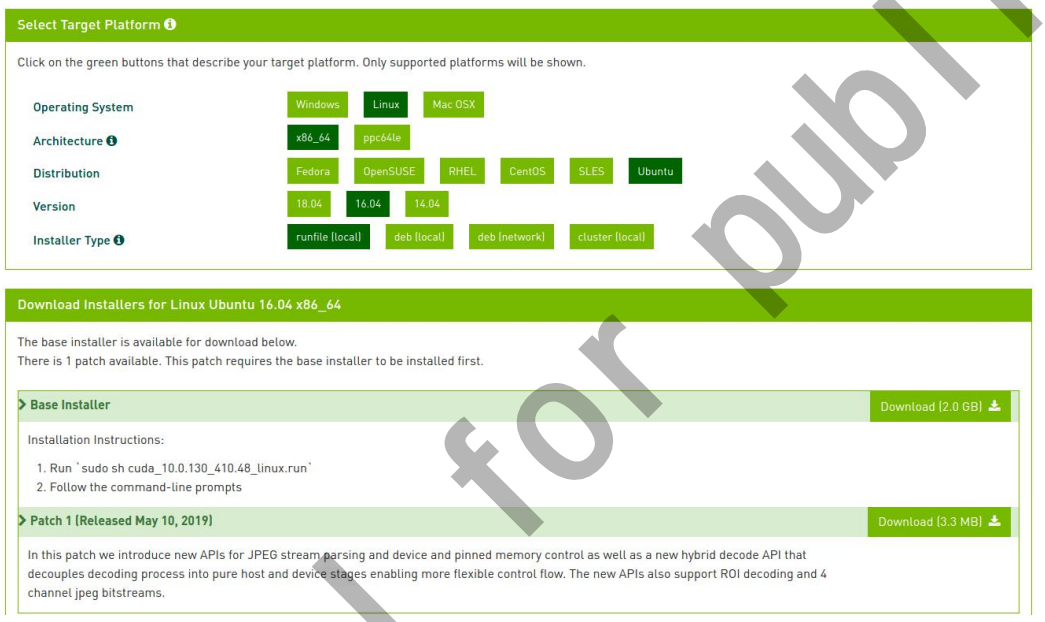
4．Check Version
$ reboot
$ nvidia-smi
If the required driver version appears, the installation is over.



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



  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__)

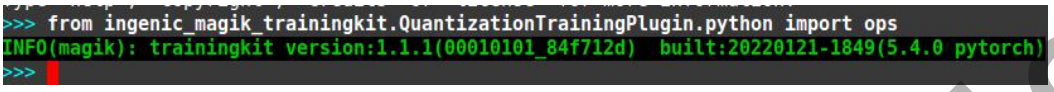
3.9 Magik Trainingkit

- 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) form our plug-in directory "magik-toolkit/TrainingKit/pytorch/magik_whl" according to the above enviroment 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



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 coda and model

1. Download link of yolov5(official):
   https://github.com/ultralytics/yolov5.git

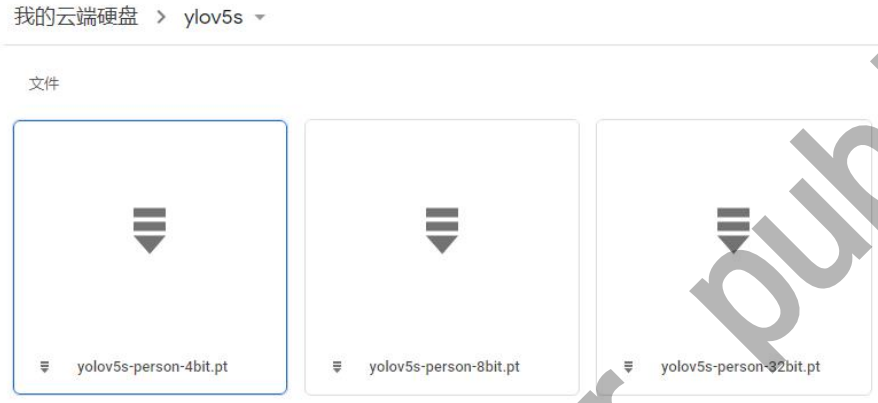
2. Our yolov5 code(modified based on the official)

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/T40/training/pytorch/yolov5s-person”, which the process of training, conversion and operation on board described below are completed in this directory.

3. Download address of the model

Link:https://drive.google.com/drive/folders/1J5KqIN91vpoM8g7xg6_7CIWbkQCXR1qe?usp=sharing

As shown in the figure below, these three models are 32bit, 8bit and 4bit models trained with 15000 person data of coco2017 step by step, and yolov5s-person-4bit.pt is used for later PC end testing and board end testing, please download and put it under "magik-toolkit/Models/T40/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/image_info_test2017.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

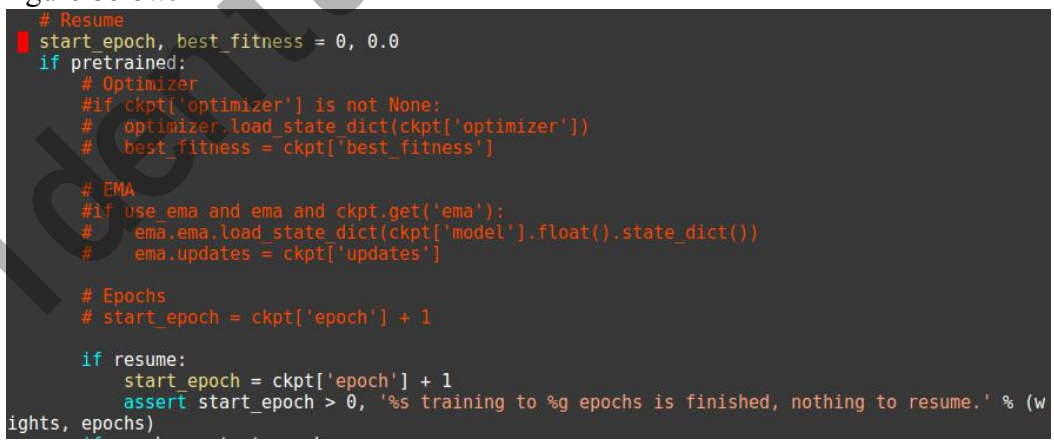
There is no pre training model in floating-point training, so -weights is ' ', 32bit model with sufficient accuracy is loaded in 8bit training, and 8bit trained is loaded in 4bit as the pre training model. In this way, the effect is better step by step.

(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.' % (w
ights, 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 fllows, 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.

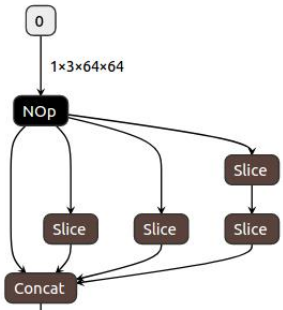
$ sh convert_onnx.sh(python convert_onnx.py \
--weights ./runs/train/yolov5s-person-4bit.pt)

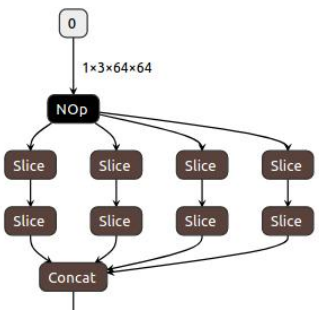
In this script, input is the path of pt -- the model we are looking forward to convert, and output is the onnx file that we want.

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] == 9223372036854775
    #     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 fllows.





before after

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-transform-tools”(which is under magik-toolkit/TransFormKit) and run.sh.

```
$ cd transform
$ sh run.sh
```

Specifically,

```
path=../../../../../../TransformKit

$ path/magik-transform-tools \
--framework onnx \
--target_device T40 \
--outputpath yolov5s-person-4bit.mk.h \
--inputpath ../runs/train/yolov5s-person-4bit.onnx \
--mean 0,0,0 \
--var 255,255,255 \
--img_width 416 \
--img_height 416 \
--img_channel 3
```

Parameter interpretation:

inputpath – onnx file we got before

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

target_device – set same as target_device during training

mean – mean value of preprocessing during training

var – variance of preprocessing during training

img_width – input’s width of running on-board

img_height – input’s height of running on-board

img_channel – input’s channels of running on-board

```
INFO(magik): magik-transform-tools version:1.0.0(00010000_bbb8835)  cuda version:9.0.176  built:20220214-1120_GPU
2022-02-15 14:53:37.819065: W clip_model_by_nop_optimizer.cc:48-collect_remove_nodes] Multiple output NOp exists on the current pathway!
2022-02-15 14:53:37.820572: W clip_model_by_nop_optimizer.cc:48-collect_remove_nodes] Multiple output NOp exists on the current pathway!
2022-02-15 14:53:37.820625: W clip_model_by_nop_optimizer.cc:48-collect_remove_nodes] Multiple output NOp exists on the current pathway!
2022-02-15 14:53:37.820665: W clip_model_by_nop_optimizer.cc:48-collect_remove_nodes] Multiple output NOp exists on the current pathway!
2022-02-15 14:54:00.861688: W quantize_operation.cc:305-search] Not found QuantizeOperation function for Unpool2D
2022-02-15 14:54:00.875195: W quantize_operation.cc:305-search] Not found QuantizeOperation function for Unpool2D
*******************************************************************************
*                      ^_^ Convert successfully, Enjoy it ^_^                 *
*******************************************************************************
```

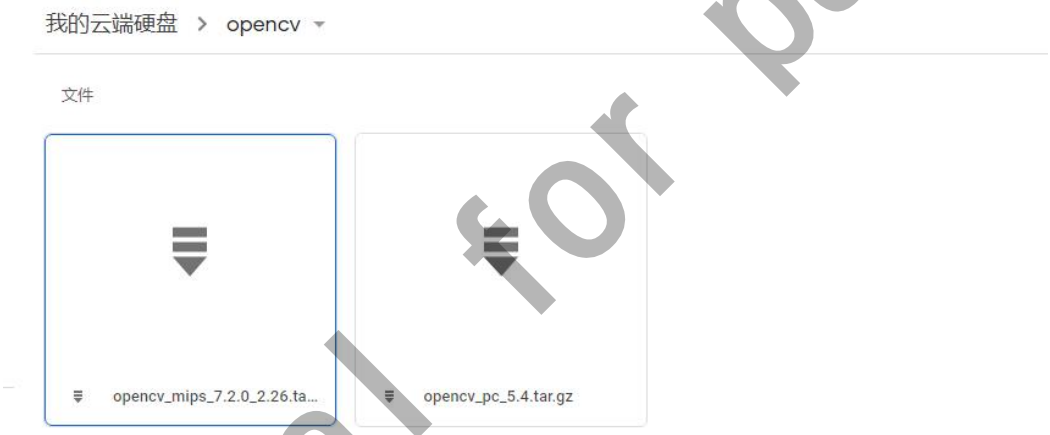
6. Run on-board

6.1 Code compiling

1. Preparation

First, download address of opencv library compiled by our mips

Link:https://drive.google.com/drive/folders/1Gdj4vF2TpaPzbPNT0H1hIEKDTXEj92Jg?usp=sharing



Please download and decompress it into magik-toolkit/ThirdParty.

$ tar -xvf opencv_mips_7.2.0_2.26.tar.gz

Secondly, we also provide inference.cpp, Makfile 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/T40/venus", 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 opencv library functions here to facilitate calling "imread()" and other functions, 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

```
oid generateBBox(std::vector<venus::Tensor> out_res, std::vector<magik::venus::ObjBbox_t>& candidate_boxes

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

opencv_path -- opencv library compiled by our mips

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.

6.2 Run

Opencv library:
magik-toolkit/ThirdParty/opencv_mips_7.2.0_2.26/uclibc/lib
Venus library:
magik-toolkit/InferenceKit/T40/venus/7.2.0/lib/uclibc

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, opencv library(which consists of libopencv_core.so.3.3, libopencv_highgui.so.3.3,libopencv_imgcodecs.so.3.3,libopencv_imgproc.so.3.3, libopencv_videoio.so.3.3), venus_yolov5s_bin_uclibc_release, yolov5s-person-4bit.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-person-4bit.bin bus.jpg". The operation results are shown in the figure below.

```
[root@Ingenic-uc1_1:doc_test]# ./venus_yolov5s_bin_uclibc_release   yolov5s-person-4bit.bin bus.jpg
Warning : The version number is not obtained. Please upgrade the soc-nna!
INFO(magik): venus version:0.9.0(00000900_86d1804)  built:20220125-1052(7.2.0 mips@aie)
w:810 h:1080
ori_image w,h: 810 ,1080
model-->416 ,416 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
pad_x:176 pad_y:0 scale:0.355556
box:   50 402 224 888 0.91
box:   222 401 343 857 0.89
box:   670 388 809 883 0.84
box:   0 598 74 873 0.56
```

2. debug(check data)

See step 7.2 for details. The input processing is a little different, and we do not need opencv library this step.

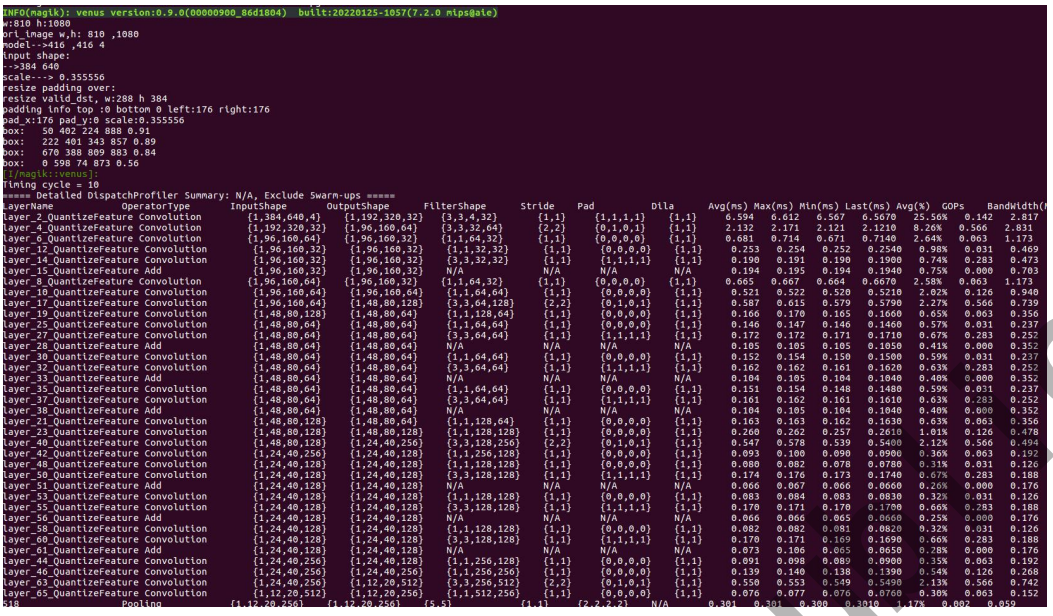
3. profile(network visualization)

$ 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, opencv library(which consists of libopencv_core.so.3.3, libopencv_highgui.so.3.3,libopencv_imgcodecs.so.3.3,libopencv_imgproc.so.3.3, libopencv_videoio.so.3.3), venus_yolov5s_bin_uclibc_profile, yolov5s-person-4bit.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_profile yolov5s-person-4bit.bin bus.jpg". The operation results are shown in the figure below.



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, opencv library(which consists of libopencv_core.so.3.3, libopencv_highgui.so.3.3,libopencv_imgcodecs.so.3.3,libopencv_imgproc.so.3.3, libopencv_videoio.so.3.3), venus_yolov5s_bin_uclibc_nmem, yolov5s-person-4bit.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-person-4bit.bin bus.jpg”. The operation results are shown in the figure below.

```
[root@Ingenic-uc1_1:doc_test]# ./venus_yolov5s_bin_uclibc_nmem  yolov5s-person-4bit.bin bus.jpg
Warning : The version number is not obtained. Please upgrade the soc-nna!
[INFO(magik): venus version:0.9.0(00000900_86d1804)  built:20220125-1045(7.2.0 mips@aie)
w:810 h:1080
ori_image w,h: 810 ,1080
model-->416 ,416 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
pad_x:176 pad_y:0 scale:0.355556
box:   50 402 224 888 0.91
box:   222 401 343 857 0.89
box:   670 388 809 883 0.84
box:   0 598 74 873 0.56
[root@Ingenic-uc1_1:doc_test]# cat /tmp/nmem_memory.txt
nmem total = 131072K
nmem used  = 13024K
nmem free  = 118048K
#### %9.94 ####
```

7. Check Data

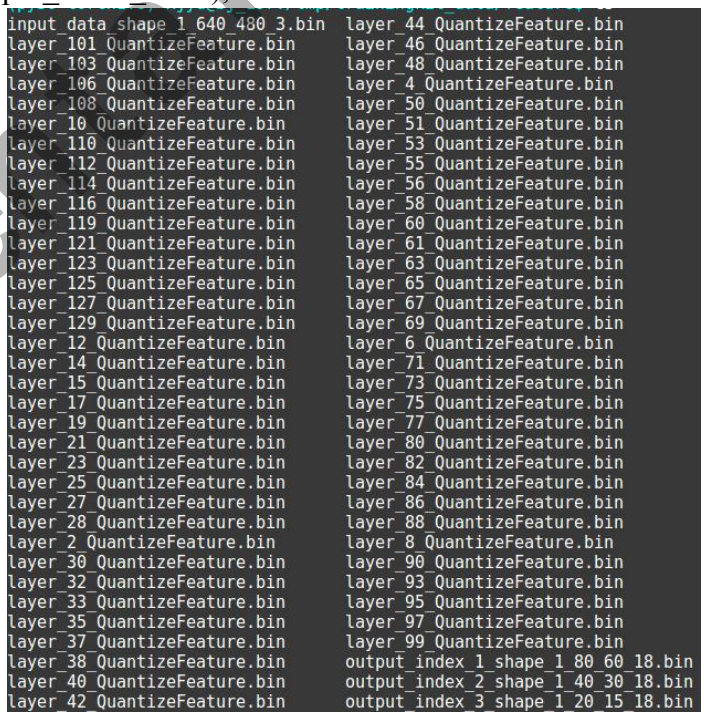
To verify whether the data of PC end and board end can be aligned, you can do the following:

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.



2. Board

The input data required at the board end is 4 channels. For the bus.jpg tested at the PC end above, we process input_data_shape_1_640_480_3.bin(which saved at run time) to header file(*.h), and then use it to test the board end to ensure that the inputs of the two are completely consistent. You can do offline processing first, you can get the processing script is in the "yolov5s-person/venus_sample_yolov5s/generate_img_input.py" .

Usage:

$ python generate_img_input.py bin_path w h c bin

($ python generate_img_input.py input_data_shape_1_640_480_3.bin \
480 640 3 bin)

bin_path -- the input bin file to be converted

w, h -- width and height of the bin file, you can get them from the name

bin -- flag, which indicates that the bin file turns to the header file, and another one is "img" means image to header file

You'd better clear the previously compiled mode by "make build_type=* clean" before compiling, otherwise, the link library of OpenCV may fail. 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-person-4bit.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-person-4bit.bin 480 640", among them, 480 is width, 640 is height. The operation results are shown in the figure below

```
layer_116_QuantizeFeature_bt.bin      layer_27_QuantizeFeature_out.bin      layer_53_QuantizeFeature_weight.bin
layer_116_QuantizeFeature_out.bin     layer_27_QuantizeFeature_weight.bin   layer_55_QuantizeFeature_bt.bin
layer_116_QuantizeFeature_weight.bin  layer_28_QuantizeFeature_out.bin      layer_55_QuantizeFeature_out.bin
layer_119_QuantizeFeature_bt.bin      layer_2_QuantizeFeature_bt.bin        layer_55_QuantizeFeature_weight.bin
layer_119_QuantizeFeature_out.bin     layer_2_QuantizeFeature_out.bin       layer_56_QuantizeFeature_out.bin
layer_119_QuantizeFeature_weight.bin  layer_2_QuantizeFeature_weight.bin    layer_58_QuantizeFeature_bt.bin
layer_121_QuantizeFeature_bt.bin      layer_30_QuantizeFeature_bt.bin       layer_58_QuantizeFeature_out.bin
layer_121_QuantizeFeature_out.bin     layer_30_QuantizeFeature_out.bin      layer_58_QuantizeFeature_weight.bin
layer_121_QuantizeFeature_weight.bin  layer_30_QuantizeFeature_weight.bin   layer_60_QuantizeFeature_bt.bin
layer_123_QuantizeFeature_bt.bin      layer_32_QuantizeFeature_bt.bin       layer_60_QuantizeFeature_out.bin
layer_123_QuantizeFeature_out.bin     layer_32_QuantizeFeature_out.bin      layer_60_QuantizeFeature_weight.bin
layer_123_QuantizeFeature_weight.bin  layer_32_QuantizeFeature_weight.bin   layer_61_QuantizeFeature_out.bin
layer_125_QuantizeFeature_bt.bin      layer_33_QuantizeFeature_out.bin      layer_63_QuantizeFeature_bt.bin
layer_125_QuantizeFeature_out.bin     layer_35_QuantizeFeature_bt.bin       layer_63_QuantizeFeature_out.bin
layer_125_QuantizeFeature_weight.bin  layer_35_QuantizeFeature_out.bin      layer_63_QuantizeFeature_weight.bin
layer_127_QuantizeFeature_bt.bin      layer_35_QuantizeFeature_weight.bin   layer_65_QuantizeFeature_bt.bin
layer_127_QuantizeFeature_out.bin     layer_37_QuantizeFeature_bt.bin       layer_65_QuantizeFeature_out.bin
layer_127_QuantizeFeature_weight.bin  layer_37_QuantizeFeature_out.bin      layer_65_QuantizeFeature_weight.bin
layer_129_QuantizeFeature_bt.bin      layer_37_QuantizeFeature_weight.bin   layer_67_QuantizeFeature_bt.bin
layer_129_QuantizeFeature_out.bin     layer_38_QuantizeFeature_out.bin      layer_67_QuantizeFeature_out.bin
layer_129_QuantizeFeature_weight.bin  layer_40_QuantizeFeature_bt.bin       layer_67_QuantizeFeature_weight.bin
layer_12_QuantizeFeature_bt.bin       layer_40_QuantizeFeature_out.bin      layer_69_QuantizeFeature_bt.bin
layer_12_QuantizeFeature_out.bin      layer_40_QuantizeFeature_weight.bin   layer_69_QuantizeFeature_out.bin
layer_12_QuantizeFeature_weight.bin   layer_42_QuantizeFeature_bt.bin       layer_69_QuantizeFeature_weight.bin
layer_14_QuantizeFeature_bt.bin       layer_42_QuantizeFeature_out.bin      layer_6_QuantizeFeature_bt.bin
layer_14_QuantizeFeature_out.bin      layer_42_QuantizeFeature_weight.bin   layer_6_QuantizeFeature_out.bin
layer_14_QuantizeFeature_weight.bin   layer_44_QuantizeFeature_bt.bin       layer_6_QuantizeFeature_weight.bin
layer_15_QuantizeFeature_out.bin      layer_44_QuantizeFeature_out.bin      layer_71_QuantizeFeature_bt.bin
layer_17_QuantizeFeature_bt.bin       layer_44_QuantizeFeature_weight.bin   layer_71_QuantizeFeature_out.bin
layer_17_QuantizeFeature_out.bin      layer_46_QuantizeFeature_bt.bin       layer_71_QuantizeFeature_weight.bin
layer_17_QuantizeFeature_weight.bin   layer_46_QuantizeFeature_out.bin      layer_73_QuantizeFeature_bt.bin
layer_19_QuantizeFeature_bt.bin       layer_46_QuantizeFeature_weight.bin   layer_73_QuantizeFeature_out.bin
layer_19_QuantizeFeature_out.bin      layer_48_QuantizeFeature_bt.bin       layer_73_QuantizeFeature_weight.bin
layer_19_QuantizeFeature_weight.bin   layer_48_QuantizeFeature_out.bin      layer_75_QuantizeFeature_bt.bin
layer_21_QuantizeFeature_bt.bin       layer_48_QuantizeFeature_weight.bin   layer_75_QuantizeFeature_out.bin
layer_21_QuantizeFeature_out.bin      layer_4_QuantizeFeature_bt.bin        layer_75_QuantizeFeature_weight.bin
layer_21_QuantizeFeature_weight.bin   layer_4_QuantizeFeature_out.bin       layer_77_QuantizeFeature_bt.bin
layer_23_QuantizeFeature_bt.bin       layer_4_QuantizeFeature_weight.bin    layer_77_QuantizeFeature_out.bin
layer_23_QuantizeFeature_out.bin      layer_50_QuantizeFeature_bt.bin       layer_77_QuantizeFeature_weight.bin
layer_23_QuantizeFeature_weight.bin   layer_50_QuantizeFeature_out.bin      layer_80_QuantizeFeature_bt.bin
layer_25_QuantizeFeature_bt.bin       layer_50_QuantizeFeature_weight.bin   layer_80_QuantizeFeature_out.bin
layer_25_QuantizeFeature_out.bin      layer_51_QuantizeFeature_out.bin      layer_80_QuantizeFeature_weight.bin
layer_25_QuantizeFeature_weight.bin   layer_53_QuantizeFeature_bt.bin       layer_82_QuantizeFeature_bt.bin
layer_27_QuantizeFeature_bt.bin       layer_53_QuantizeFeature_out.bin      layer_82_QuantizeFeature_out.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.

```
86859cb6efb8cfd932dbb57e32a260e3  /tmp/trainingkit_data/feature/layer_2_QuantizeFeature.bin
6e2c04cf0e560cbd3b76860a23132053  /tmp/trainingkit_data/feature/layer_129_QuantizeFeature.bin
```

```
[root@Ingenic-uc1_1:doc_test]# md5sum layer_2_QuantizeFeature_out.bin
86859cb6efb8cfd932dbb57e32a260e3  layer_2_QuantizeFeature_out.bin
[root@Ingenic-uc1_1:doc_test]# md5sum layer_129_QuantizeFeature_out.bin
6e2c04cf0e560cbd3b76860a23132053  layer_129_QuantizeFeature_out.bin
```

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_typeMetaDataInstanceIN3c108BFloat16EEEPKNS_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 occurres “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.