

AI NR 环境搭建和使用手册

文档版本 V1.5

发布日期 2025-12-18

前言

概述

本文档主要说明 AI_NR 在模型训练阶段时候的具体安装、数据采集、训练使用以及部署和调试的方法。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2024-12-21	V1.0	DI 版本发布。
2025-03-07	V1.1	修改和完善 DI 版本中相关描述问题。
		第 3 章节描述 添加相关 yml 参数详细说明。
		第 2.3 章节描述 修改 finetune 阶段实际训练的.pth 模型文件，图片文件，log 文件位置与文档描述不符的地方。
		第 2 章描述 补充 AI ISP 文档 2.2 指 2.5 章节各阶段缺少必要步骤描述。
		第 2.6 章描述 补充在 quant_deploy 阶段缺少 txt 文件生成的操作说明。
2025-06-03	V1.2	添加 ai_isp_toolchain 工具链的使用，添加数据采集的格式和要求，修改部分 yml 字段。
2025-08-20	V1.3	添加更详细的数据采集步骤和 ai_isp.bin 的使用操作，以及相关文档排版问题，添加 5AINR 调优过程及 Q&A。
		添加采集场景示意图，以及 aitoolchain 时候完整编译文件示意图。
2025-09-29	V1.4	添加一些实际操作和检查的相关步骤。 更新 gpu 相关支持说明。
2025-12-18	V1.5	添加一些异常调试图片和修改 bnr 部分说明。

目 录

1 环境搭建	1
1.1 硬件环境搭建	1
1.2 安装 miniconda	2
1.3 运行 ai_nr	4
2 环境实际运行流程	5
2.2 数据准备	6
2.2.1 采集前注意事项	7
2.2.2 采集蒸馏 Raw 数据	15
2.2.3 采集 Benchmark Raw 数据	20
2.3 preprocess	20
2.4 float_finetune	22
2.5 float_inference	24
2.6 quant_finetune	25
2.6.1 1 阶段	26
2.6.2 2 阶段	27
2.7 quant_inference	28
2.8 quant_deploy	29
3 相关参数说明	31
3.1 工具配置说明	31
3.2 quant 字段	31
3.3 data 字段	32
3.4 preprocess 字段	33
3.5 augmentation 字段	33
3.6 log 字段	34

3.7 model 字段	35
3.8 train 字段	35
3.9 infer 字段	36
3.10 deploy 字段	37
3.11 loss 字段	38
4 模型编译说明	39
4.1 环境配置	39
4.1.1 安装 docker	39
4.1.2 创建容器环境	39
4.1.3 测试容器环境	40
4.2 模型编译	41
4.2.1 编译前确认	41
4.2.2 编译过程	42
5 AINR 调优过程及 Q&A	46
5.1 固定参数检查	46
5.1.1 固定参数验证	46
5.1.2 proc 信息核对	46
5.1.3 模型阶段回灌数据仿真	47
5.1.4 模型效果正确性	50
5.1.5 一些常见的命令	53
5.2 B3D 和 AINR 的耦合过程	53
5.3 AINR 调优过程	55
5.3.1 Sensor 噪声模型标定	55
5.3.2 AI 模型训练	56
5.3.3 基础传统通路调试	57
5.3.4 AI 通路调试	57

5.4 Q & A:	57
------------------	----

仅限深圳市安远威视科技有限公司使用

图 1-1 查看硬件信息 2

图 1-2 conda -version 2

图 1-3 修改 Python 解释器位置..... 3

图 1-4 查看当前环境路径 3

图 1-5 conda init 3

图 2-1 工具逻辑架构 6

图 2-2 常用的对焦卡 7

图 2-3 一些缺少鲜艳色彩可能带来的问题，例如红色格子 8

图 2-4 sample_vio 1 全离线模式启动图片 9

图 2-5 Cat /proc/umap/vi 查看模式（仅离线模式可以抓图） 9

图 2-6 启动 pqtool..... 9

图 2-7 工具连接成功 9

图 2-8 采集数据要求 10

图 2-9 pqtool 采集界面..... 10

图 2-10 采集 raw 图像信息.....11

图 2-11 确认 vb 数量 12

图 2-12 awb/ccm 信息 12

图 2-13 nr 标定界面 13

图 2-14 泊松高斯噪声和导出的 nr k/b 值 13

图 2-15 实际标定出的高斯泊松噪声值 13

图 2-16 nr 参数在线标定验证..... 14

图 2-17 nr 参数标定异常对比..... 15

图 2-18 采集场景示意图..... 16

图 2-19 采集场景实际布置	16
图 2-20 过曝场景	17
图 2-21 旋转角度示意图	17
图 2-22 iso1w 场景的实际 raw 图	18
图 2-23 iso8w 场景的实际 raw 图	18
图 2-24 iso25w 场景的实际 raw 图	18
图 2-25 较低环境亮度下两种模式下采集的图像	19
图 2-26 一组蒸馏数据格式示意	19
图 2-27 训练大体流程图	20
图 2-28 input-label	21
图 2-29 float_finetime 输出 pth	23
图 2-30 打印中读取 psnr 信息	23
图 2-31 浮点训练结果(input-label-predict)	24
图 2-32 float_inference 生成结果	25
图 2-33 float_inference 效果展示	25
图 2-34 quant_finetime 生成文件	26
图 2-35 quant_finetime 生成图像	26
图 2-36 量化训练结果(input-label-predict)	27
图 2-37 quant infer 生成文件	28
图 2-38 quant infer 效果展示	28
图 4-1 7206 版本模型 deploy 编译提供文件	41
图 4-2 7606 版本模型 deploy 编译提供文件	41
图 4-3 7606 模型训练文件	42
图 4-4 7606 模型训练完整文件	42
图 4-5 7206 模型训练文件	43
图 4-6 7206 模型训练完整文件	44
图 4-7 ai isp.bin 生成方法	44

图 4-8 7206 yamI	45
图 5-1 cat /proc/umap/npu 信息.....	46
图 5-2 7206 sc485sl b k curve	46
图 5-3 weight proc 信息	47
图 5-4 采集 check raw 流程图.....	48
图 5-5 npu 在 pipeline 中的位置	48
图 5-6 常规采集界面	49
图 5-7 打开 npu 采集开关	49
图 5-8 显示 npu 采集界面	49
图 5-9 Artifact.....	50
图 5-10 2688*1520 的 4M 场景 deploy bin	51
图 5-11 5M 场景 deploy bin	52
图 5-12 7206_mis40h1_10_bit 编译生成的 Bin 文件.....	53
图 5-13 7206_mis40h1_10_bit deploy 生成的 Bin 文件	53
图 5-14 AI 生效的位置的示意图	54
图 5-15 AI 不同组合下回叠噪声生效的示意图	54
图 5-16 sensor.c 驱动参数.....	56
图 5-17 sensor.ex.h 驱动参数.....	56

1 环境搭建

1.1 硬件环境搭建

硬件要求

Linux 开发环境

GPU 环境

显卡：至少 6G 显存以上,能运行对应的 cuda 版本

Cuda：推荐 10.2 版本

Python 和 torch 环境

软件包和环境管理：linux 推荐安装 miniforge/miniconda

Python 环境依赖：工具附属环境包

当前工具环境参数：

python:3.8.13

pytorch :1.10.1

pytorch_cuda:11.3

须知

目前工具暂不支持 NVIDIA **Blackwell** 等 RTX50 系列架构及更新版本的 GPU，会有 sm_120 架构支持问题；支持 RTX40 系列和 RTX30 系列等同系列架构 cpu。

硬件环境搭建好后，输入 nvidia-smi 查看显卡信息，nvidia 驱动信息和最高支持的 cuda 版本。

图1-1 查看硬件信息

```
AISWserver08:~> nvidia-smi
Tue Nov 19 19:21:14 2024
```

NVIDIA-SMI 555.52.04			Driver Version: 555.52.04			CUDA Version: 12.5		
GPU Fan	Name Temp	Perf	Persistence-M Pwr:Usage/Cap	Bus-Id	Disp.A Memory-Usage	Volatile GPU-Util	Uncorr. Compute	ECC M. MIG M.
0 30%	NVIDIA GeForce RTX 4090 42C	P0	Off 59W / 450W	00000000:31:00:0	Off 1MiB / 24564MiB	0%	Default	Off N/A
1 30%	NVIDIA GeForce RTX 4090 41C	P0	Off 52W / 450W	00000000:98:00:0	Off 1MiB / 24564MiB	0%	Default	Off N/A

```
Processes:
GPU  GI  CI  PID  Type  Process name  GPU Memory Usage
ID   ID   ID
=====
No running processes found
```

1.2 安装 miniconda

1. 安装 miniconda

图1-2 conda --version

```
AISWserver08:~> conda --version
conda 24.9.2
```

检查 miniconda 是否安装成功

2. 激活 miniconda 环境

复制我司 pytorch AIBNRtools_env_xx (xx 为当前 env 发布时间) 环境到 miniconda/envs/路径下解压并查看 conda 环境, 输入 conda info --envs, 显示 AIBNRtools_env_xx;

```
# conda environments:
```

```
#
```

```
base /path_to/miniconda3
```

```
AIBNRtools_env_xx /path_to/miniconda3/envs/AIBNRtools_env_xx
```

进入 envs/AIBNRtools_env_xx /bin 目录, 修改依赖路径, 将第一行 #! /xxx, 改为自己的 AIBNRtools_env_xx 下 bin 环境路径。

图1-3 修改 Python 解释器位置

```
AISWserver08:~/miniconda3/envs/mmdet_master_v2/bin> cat pip
#!/home/yangmaolin/miniconda3/envs/mmdet_master_v2/bin/python3.8

# -*- coding: utf-8 -*-
import re
import sys

from pip._internal.cli.main import main

if __name__ == '__main__':
    sys.argv[0] = re.sub(r'(-script\.pyw?|\.\exe)?$', '', sys.argv[0])
    sys.exit(main())
```

检查环境变量，检查有没有 miniconda 环境变量，添加环境变量。

不同 shell 环境可能激活 miniconda 环境方式不同，以 tcsh 为例，输入 echo \$0，可以查看当前 shell 环境，输入 printenv 查看当前环境变量。

图1-4 查看当前环境路径

```
AISWserver08:~> printenv
USER=yangmaolin
LOGNAME=yangmaolin
HOME=/home/yangmaolin
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
MAIL=/var/mail/yangmaolin
SHELL=/bin/tcsh
SSH_CLIENT=192.168.110.110 57513 22
SSH_CONNECTION=192.168.110.110 57513 10.20.129.8 22
SSH_TTY=/dev/pts/0
TERM=xterm
DISPLAY=localhost:10.0
LANG=en_US.UTF-8
LC_NUMERIC=zh_CN.UTF-8
LC_TIME=zh_CN.UTF-8
LC_MONETARY=zh_CN.UTF-8
LC_PAPER=zh_CN.UTF-8
LC_NAME=zh_CN.UTF-8
LC_ADDRESS=zh_CN.UTF-8
LC_TELEPHONE=zh_CN.UTF-8
LC_MEASUREMENT=zh_CN.UTF-8
LC_IDENTIFICATION=zh_CN.UTF-8
HOSTTYPE=x86_64-linux
VENDOR=unknown
OSTYPE=linux
MACHTYPE=x86_64
SHLVL=1
PWD=/home/yangmaolin
GROUP=yangmaolin
HOST=AISWserver08
REMOTEHOST=192.168.110.110
```

添加 miniconda 环境变量；setenv PATH

"/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin :/path_to/miniconda3/bin"

输入 Conda init，可在窗口查看是否激活环境，需要重启窗口

图1-5 conda init

```
AISWserver08:~/miniconda3/envs> conda init tcsh
no change      /home/yangmaolin/miniconda3/condabin/conda
no change      /home/yangmaolin/miniconda3/bin/conda
no change      /home/yangmaolin/miniconda3/bin/conda-env
no change      /home/yangmaolin/miniconda3/bin/activate
no change      /home/yangmaolin/miniconda3/bin/deactivate
no change      /home/yangmaolin/miniconda3/etc/profile.d/conda.sh
no change      /home/yangmaolin/miniconda3/etc/fish/conf.d/conda.fish
no change      /home/yangmaolin/miniconda3/shell/condabin/Conda.psm1
no change      /home/yangmaolin/miniconda3/shell/condabin/conda-hook.ps1
no change      /home/yangmaolin/miniconda3/lib/python3.12/site-packages/xontrib/conda.xsh
no change      /home/yangmaolin/miniconda3/etc/profile.d/conda.csh
modified       /home/yangmaolin/.tcshrc

==> For changes to take effect, close and re-open your current shell. <==
```

1.3 运行 ai_nr

切换 conda 环境，切换成功后前面会有括号显示当前路径 AIBNRtools_env_xx

输入 `conda activate /path_to/miniconda3/envs/ AIBNRtools_env_xx`

```
AISAserver03:~/miniconda3/envs> conda activate AIBNRtools_env_xx  
(AIBNRtools_env_xx) AISAserver03:~/miniconda3/envs>
```

切换成功后可以检查 cudnn 环境是否存在

```
import torch
```

```
print(torch.backends.cudnn.version())
```

输出 8200

新建目录，复制相关对应版本 AIBNRtools_build_xx 文件到目录下，cd build 主目录下，执行

```
python3 main.py --yaml-path /path_to/AIBNRtools_build_xx/options/4M/04a10/  
config_04a10_template.yml
```

即可运行程序（其中 yml 中包含很多参数，可以选择使用不同参数，详见文档第三部分）。

2 环境实际运行流程

工具使用完整流程：

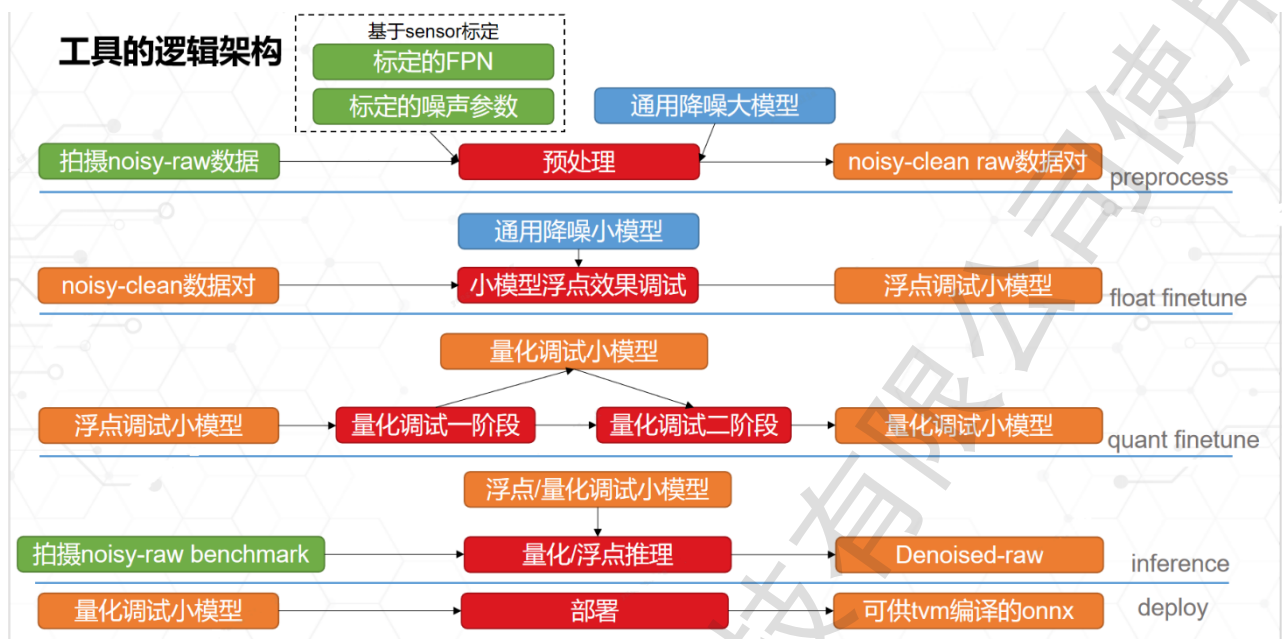
1. 确认当前需调优 sensor 的基础信息，包括：数据位宽、bayer pattern、分辨率、最大 Again 对应的 iso 值。
2. 采集当前 sensor 的噪声参数、fpn。
3. 使用当前 sensor 采集多组训练数据，同时需要提供 awb 及 ccm 信息。
4. 在 config.yml 中添加 sensor id 以及对应的噪声参数。
5. 根据 sensor 基本信息修改 config.yml 中 data 及 preprocess 相关字段信息。
6. 调整 preprocess, augmentation 相关配置进行数据处理后，得到以.h5 为后缀的训练数据。
7. 调整 model、log、loss、train 等字段进行两阶段的 float 模型 finetune 训练操作。
8. 以训练好的 float 模型为基准进行两阶段量化训练操作。
9. 训练后调整 infer 字段，确认当前模型效果。
10. 调整 deploy 字段得到模型部署文件。

共计六个阶段

1. preprocess
2. float_finetrain
3. float_inference
4. quant_finetrain
5. quant_inference
6. quant_deploy

工具的逻辑架构如下：

图2-1 工具逻辑架构



2.2 数据准备

生成训练集+生成验证集

数据需求如下，工具使用前须知

在工具使用之前，用户需首先准备一些必要信息，其中包括：

1. 确认需微调的 sensor 的噪声参数（共五个）。
2. 确认需微调的 sensor 的 fpn（需包含黑电平信息）。
3. 确认需微调的 sensor 的 Bayer pattern、Again 最大值。
4. 确认需微调的 sensor 的分辨率。
5. 提前准备适配该 sensor 分辨率的对应预训练浮点模型。
6. 提前采集该 sensor 的真实数据，包括 raw 序列，对应的 meta 信息，awb 以及 ccm 信息。数据存放格式为每组数据存放一个文件夹，文件夹以 iso 值命名，文件夹内需存放。
 - 1) `***.raw`
 - 2) `***.txt`(前缀名和 raw 数据需保持一致)
 - 3) `awb.txt`

4) ccm.txt

2.2.1 采集前注意事项

2.2.1.1 RAW 采集前检查

步骤 1 检查镜头清晰度。

- 使用镜片布轻微擦拭镜头。
- 对镜头进行调焦，直到镜头最清晰为止。

须知

每对焦时在光线明亮的环境，确认镜头无脏污使用专业对焦卡或其他可分辨清晰度的场景，务必保持最佳清晰度，同时镜头无暗区死角，下面为使用的对焦卡图片。

图2-2 常用的对焦卡



步骤 2 检查 AE 配置。

- 检查需采集的最大 ISO 能否达到，环境光调到接近 0 Lux 后，check ISO 能否达到最大的采集 ISO；若 ISO 有被限制，需检查 ae rout&exposure auto 里面的 sensor A&D gain&ISP d gain 的参数范围。
- 检查 sensor 的 D gain；需保持 1024，在 exposure auto 下把 sensor D gain，设置 max=min=1024。

步骤 3 检查采集场景。

- 一般需要有高饱和度场景，如红色的对联，红色的中国结。
- 确认采集画面是否有出现过曝的地方，若有需调整拍摄画面或 ae compensation 设定。
- 采集时色温尽量保持在 5000k~6000k 色温之间。

图2-3 一些缺少鲜艳色彩可能带来的问题，例如红色格子



步骤 4 检查 BLC 设定。

确认是否为标准的 blc 值，如果不确认可以先标定，设置标定后的 BLC 值后，可通过 sensor_ex.h 将 BLC op_type 设 1，为 manual mode。

步骤 5 检查 sensor 采集位宽及是否带 ir cut。

- 一般需采集 12bit，若本身 sensor 为 10bit，使用 isptool 启动时需将 sensor.ini 的 bit_width 由 1 改为 2，sample 则对应修改代码中 bit 位。
- 若没有带 ir cut，需要添加。

须知

采集时一直保持 ir cut 打开，要过滤掉红外光，不然会有干扰，白天的话颜色会异常；夜间模式也保持和白天一样，都过滤掉红外光，保持一致。

步骤 6 提前做好 AWB,CCM 标定，在采集的时候会使用到这些参数。

相关部分可参考《图像质量调试工具使用指南》中关于 awb, ccm 的标定模块。

步骤 7 采集前先 bypass ae，保证采集途中 ae 不发生变化。

2.2.1.2 采集 sample

可以使用 sample_vio 1 或者 isptool 工具启动的全离线模式进行采集。

图2-4 sample_vio 1 全离线模式启动图片

```
***** Error: There's something wrong, please check! *****
SENSOR[Func]:mis40h1_set_slave_addr [Line]:163 [Info]:Sensor dev [0] set slave addr 0x60 success.
SENSOR[Func]:mis40h1_i2c_init [Line]:190 [Info]:===== i2c(5) init success!=====
SENSOR[Func]:mis40h1_set_wdr_mode [Line]:406 [Info]:linear mode
sample_comm_isp_sensor_init:269: sensor i2c_id = 5, clk_id = 2
SENSOR[Func]:mis40h1_get_se_common_default [Line]:646 [Info]:man_ratio_enable: 0,
SENSOR[Func]:mis40h1_master_calc_fps [Line]:112 [Info]:mis40h1 set fps = 30.000000
mcl software version 7fad057 # [17:24:31] Jun 24 2025
ISP[Func]:isp_run [Line]:827 [Info]:isp run
ISP[Func]:isp_init [Line]:909 [Info]:isp init success
ISP[Func]:isp_start [Line]:475 [Info]:isp start.
SENSOR[Func]:mis40h1_2lane_linear_2688x1520_init [Line]:121 [Info]:===== MIS40H1_24MInput_MIPI_2lane_10bit_792M/bps_2688x1520_30fps init success!=====
SENSOR[Func]:mis40h1_2lane_linear_2688x1520_init [Line]:122 [Info]:===== MIS40H1_24MInput_MIPI_2lane_10bit_792M/bps_2688x1520_30fps init success!=====
SENSOR[Func]:mis40h1_2lane_linear_2688x1520_init [Line]:123 [Info]:===== MIS40H1_24MInput_MIPI_2lane_10bit_792M/bps_2688x1520_30fps init success!=====

tsp://10.4.65.76:554/livestream/0
tsp://10.4.65.76:554/livestream/1
-----Press any key to exit!-----
media_venc_request_idr(chn0) successful!
```

图2-5 Cat /proc/umap/vi 查看模式（仅离线模式可以抓图）

```
#
#
# cat /proc/umap/vi

ersion:[SPC021 ], [VI V1.0 ], Release, Build Time[Jun 24 2025 17:24:28]
-----VI-GDC-VPSS MODE-----
pipe_id| VICAP-VIPROC MODE| VIPROC-VPSS MODE| GDC-VPSS MODE|
0| offline| offline| offline|
```

使用 sample_vio 1 的时候，同时使用 isptool- -c 启动连接 pqtool，可以实时查看当前的曝光,awb.ccm 等信息。

图2-6 启动 pqtool

```
/mnt/4_evb_out_20250624/4_pqboard # ./IspTool.sh -c
/mnt/4_evb_out_20250624/4_pqboard # pqboard v1.3.8: ittb_control begin to run

<PQT_CreatServerSocket>(1592)bind success

<XMEDIA_PQT_CommuInit>(1884)Not support uart communication!

<XMEDIA_PQT_CommuInit>(1891)ittb_control run success, Waiting for connection from client
```

图2-7 工具连接成功

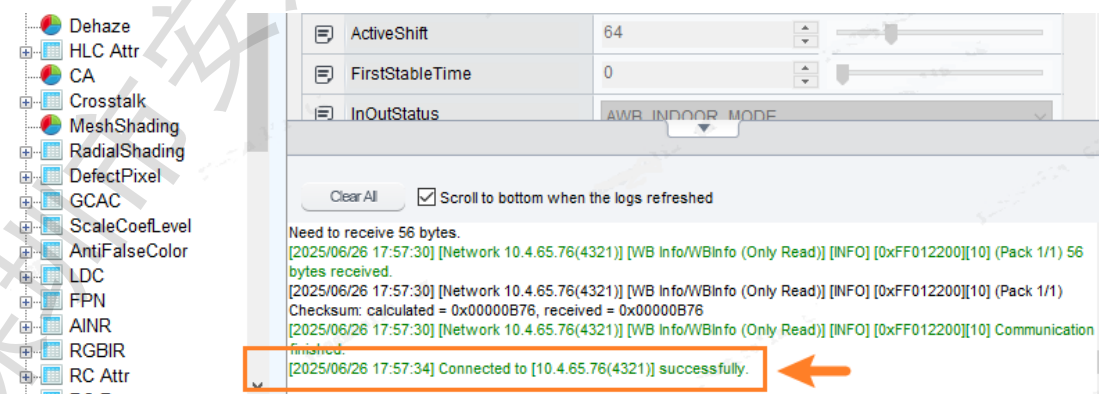


图2-8 采集数据要求

□ Noisy-Raw的输入数据

➤ 数据要求

- 数量：至少30-40组室内、室外场景。
- 场景：最好包含运动物体，且至少10帧以上。
- ISO：采集数据在sensor again和ISP dgain范围的ISO上需要覆盖采样(建议随ISO进行密度下降采样)。
- AWB/CCM：提前预标定。

➤ 数据组织形式

父目录	一级子目录	二级子目录	文件
root(命名规则：分辨率(8M, 4M or 2M)_sensor(200ai, 04a10)_distill_data_日期)	indoor	ISO_数值	*.raw
			*.txt
	outdoor	ISO_数值	*.raw
			*.txt

□ Noisy-Raw的benchmark

数据要求

- 数量：至少10-20组室内、室外场景。
- 场景：包含运动物体，且至少200帧。
- ISO：采集数据在sensor again和ISP dgain范围的ISO上需要覆盖采样(建议随ISO进行密度下降采样)。
- AWB/CCM：提前预标定。

➤ 数据组织形式

父目录	一级子目录	二级子目录	文件
root(命名规则：分辨率(8M, 4M or 2M)_sensor(200ai, 04a10)_benchmark_日期)	indoor	ISO_数值	*.raw
			*.txt
	outdoor	ISO_数值	*.raw
			*.txt

□ 标定数据之泊松高斯参数

➤ 数据要求

- 获取形式：参考传统参数标定(带BL)。
- 泊松参数：KA, KB(0-1浮点格式下的)
- 高斯参数：BA, BB, BC(0-1浮点格式下的)

➤ 数据组织形式 (追加config的参数词条)

```
04a10:
KA : 0.000131794734
KB : -0.000000019901
BA : 0.000000030969
BB : 0.000000000008
BC : -0.000000472919
```

□ 标定数据之FPN

➤ 数据要求

- 获取形式：参考传统FPN标定(带BL)。
- 自身格式：带BL存储成无符号位的高bit位数据。

➤ 数据组织形式

父目录	文件
fpn	*.raw
	*.txt

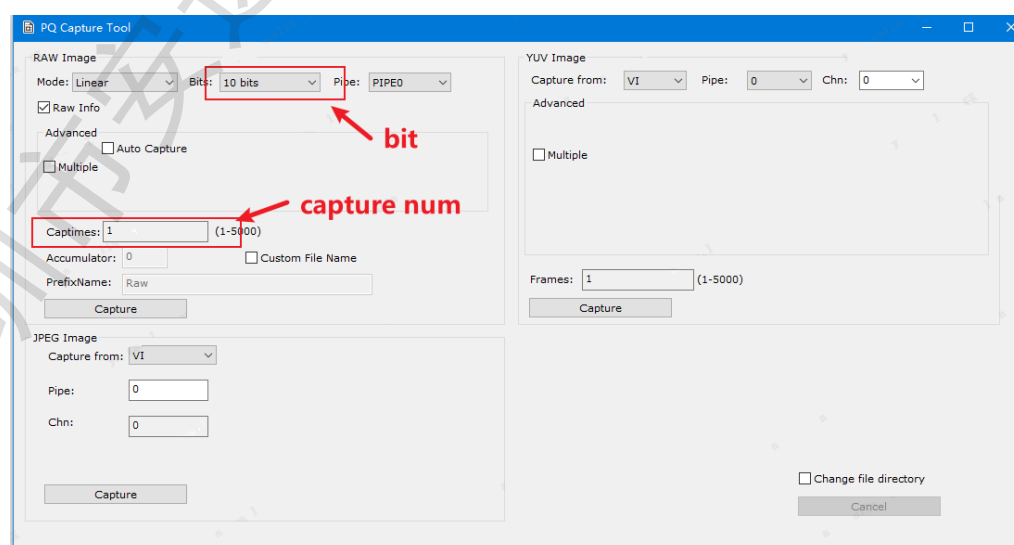
须知

采集时候使用我司大内存芯片进行采集，例如 7606，如果使用 7206 小内存单板，采集上限 4M 分辨率可能只能采集 3 帧左右，采集数量不够。

1. Benchmark 和 distill_data

采集数据可以直接使用我司 pqtool 工具。

图2-9 pqtool 采集界面



勾选 raw_info，同时会采集出实时的 info，包含曝光相关信息，如下图所示。

图2-10 采集 raw 图像信息

```
[Common]
width=2688
height=1520
bits=12
bayer_format=BGGR
sensor_id=0
sensor_mode=0
blackLevel_r=256
blackLevel_gr=256
blackLevel_gb=256
blackLevel_b=256
wdr_mode=0

[Frame0]
iso=343941
exposure_time=33143
exposure_bias_value=0
exposure_program=0
exposure_mode=0
exposure_ratio[0]=64
exposure_ratio[1]=0
exposure_ratio[2]=0
isp_dgain=32788
isp_nr_strength=0
sensor_hmax=0
sensor_vmax=0
sensor_again=109993
sensor_dgain=1024
f_number=0
max_aperture_value=0
white_balance_mode=0
vc_num=0
```

sensor信息

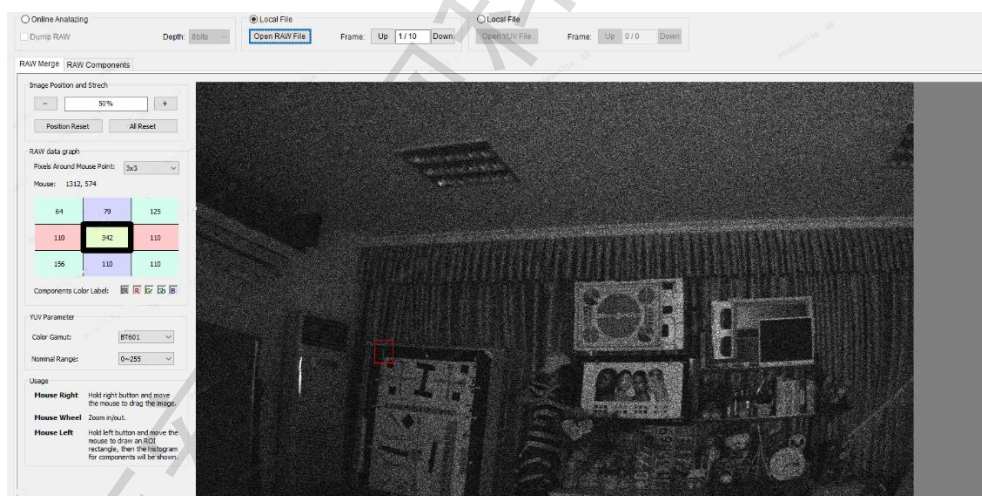
黑电平

iso

同时生成info.txt

RAW_2688x1520_12bits_BGGR_Linear_20250429171029.raw
RAW_2688x1520_12bits_BGGR_Linear_20250429171029.txt

可以在 pqtool 中的 raw yuv analyzer 工具中查看我们抓拍的 raw 图。



可预先采集一帧画面观察角度，清晰度，曝光等是否符合要求。

- 采集一帧 raw 数据，确认 raw 数据的画面是否有异常，是否有出现镜像或倒置；若有需调整，查看 sensor 数据手册下对应的镜像和倒置控制寄存器；
如 sc485sl 反向，控制寄存器设置为 0x66；输入 i2c_write 0x5 0x60 0x3221 0x66 2 1
- 抓取 capture raw 时，若 pq tool 提示内存容量不够，使用 isptool 可设 sensor.ini vb 的 pool 的 block_cnt = 100，sample 则修改对应的 vb 大小位置，采集 100 帧需要 100 个 vb 以上

输入 Cat /proc/umap/vb 可以查看当前 vb 池使用情况

```

~ # cat /proc/umap/vb
Version:[SPC021 ], [vb V1.0 ], Release, Build Time[Jun 24 2025 17:24:23]

-----VB CONFIG-----
destroy_flag:0

-----PROCESS 1 VB CONFIG-----
max_pool_cnt:512
tgid:330      , process_num:0
busy:N

-----POOL INFO-----
pool_cnt blk_cnt comm_pool_cnt comm_blk_cnt pri_pool_cnt pri_blk_cnt
11      128      3      120      8      8

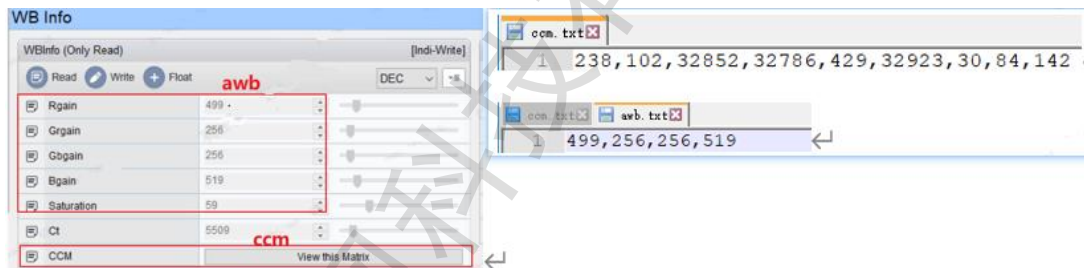
-----POOL DETAILS-----
pool_id  phy_addr      vir_addr      is_comm  owner      blk_size      blk_cnt free      min_free
0x5081a000 0x0      0x0      1      -1      5107200      110      109      108
blk VII  ISP  VO  VGS  VENC  VDEC  H265E  H264E  JPEGE  H264D  JPEGD  WPS  U15  USER  AI  AENC  RC  GDC  AVS  MCF  phy_addr      vir_addr
32  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0x5a3f6000  0x0
sum  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

```

人 pqttool 上读取出来。

采集数据时，当前环境 Awb/ccm 信息可以从 pqttool 上读取出来。

1 238,102,32852,32786



Awb 和 ccm 的具体格式信息。

Awb 和 ccm 的具体格式信息如上图。长短帧和 benchmark、蒸馏数据记录 awb、ccm 信息，也可以通过回灌 raw 再板端读取。

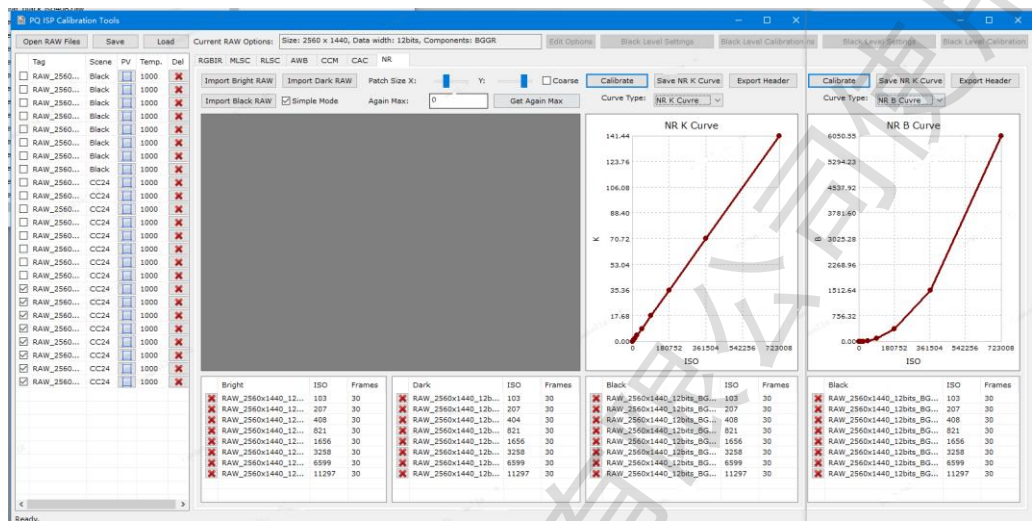
须知

每一个对应的 iso 下，都有一个对应 awb 和 ccm 的文件，用于指导当前 iso 的色彩。

泊松高斯噪声参数

其中标定泊松高斯噪声参数可参考我司文档《图像质量调试工具使用指南》中 BayerNR 标定部分，通过 pq tool 中的 isp calibration tool 生成 alg_nr_calib_info.h 和 alg_nr_k_curve_info.h 泊松高斯噪声参数，新版工具的 nr/k 和 nr/b 曲线都支持验证。

图2-13 nr 标定界面



须知

上图中的 iso 显示和实际采集时候 iso 有关，并非为标准 100, 200 类似，可能会有微小浮动，注意和采集时候的信息一致即可，NR K 和 NR B curve 应为单调递增趋势。

图2-14 泊松高斯噪声和导出的 nr k/b 值

```
1: #ifndef __ALG_NR_CALIB_INFO_XXX_H__
2: #define __ALG_NR_CALIB_INFO_XXX_H__
3:
4: #define MAX_CALIBRATE_NOISE_LUT_LEN 16
5:
6: static isp_cmos_noise_calibration_coef_calib_g_coef_noise_calibration = {
7:     { 6400.00000000f, 0.000648168f, 0.0479460172f, 0.0000000038f, 0.0000623386f, -0.0139708603f,
8:       0.0000677343f, 0.0000000041f, -0.0000000004f, 0.0000114405f, 0.0, 0.0, 0.0, 0.0 }
9: };
10:
11: #endif // __ALG_NR_CALIB_INFO_XXX_H__
```

图2-15 实际标定出的高斯泊松噪声值

```
float coef_calib[BAYERNR_CALIB_CARVE_NUM_XXX][3] = {
    { 100.000000f, 0.019558f, 0.073206f },
    { 200.000000f, 0.037421f, 0.084028f },
    { 400.000000f, 0.072578f, 0.078628f },
    { 600.000000f, 0.142178f, 0.093486f },
    { 1400.000000f, 0.291575f, 0.144753f },
    { 3200.000000f, 0.556676f, 0.302800f },
    { 6400.000000f, 1.137536f, 0.750553f },
    { 12800.000000f, 1.932915f, 1.738753f },
    { 25600.000000f, 3.865830f, 6.955011f },
    { 51200.000000f, 7.731600f, 27.820044f },
    { 102400.000000f, 15.463321f, 111.280174f },
    { 204800.000000f, 30.926641f, 445.120697f },
    { 409600.000000f, 61.853283f, 1780.402788f },
    { 819200.000000f, 123.706566f, 7121.931152f },
    { 1638400.000000f, 247.413132f, 28487.724609f },
    { 3276800.000000f, 494.826263f, 113950.898438f }
};
```

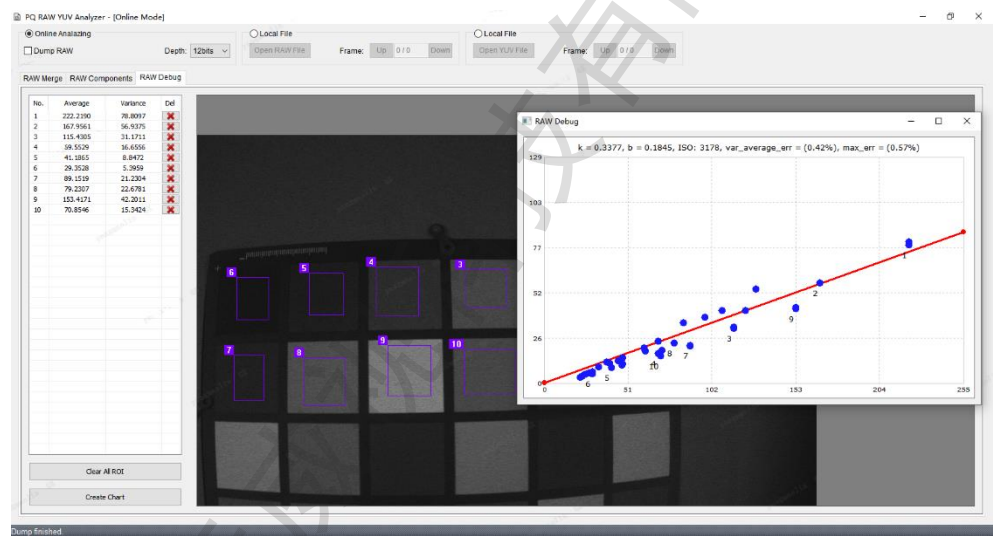
须知

NR 标定参数分两段拟合显示 以 iso6400 为分界，上面部分是小于 iso6400 的拟合参数，下面部分是大于 iso6400 的拟合参数，在 ainr 的 yaml 中一般使用下面的拟合参数。

采集真实场景，框出 ROI 判断亮度方差是否正常。

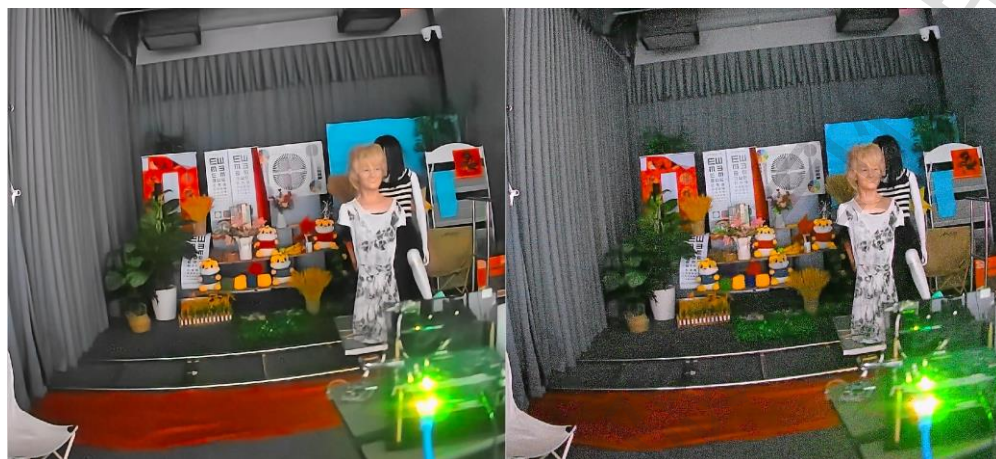
可参考文档《图像质量调试工具使用指南》BayerNR 标定部分关于标定区域验证的操作指南。

图2-16 nr 参数在线标定验证



上板测试当前噪声模型下传统 BNR 效果，左边为正常 nr 参数，右边为异常 nr 参数，异常参数会有明显噪声形态的问题。

图2-17 nr 参数标定异常对比



4. Fpn 数据采集

直接采集黑帧，采集 100~200 帧黑帧（带 bl），采集 fpn 的时候保持镜头全黑，不能有一点光透入；fpn.raw 就是 again 打满，ispgian=1,sensor dgian=1 倍的黑帧。

须知

FPN 采集黑帧的时候，曝光时间采用和实际最暗场景时使用的曝光时间一致即可，例如最暗 12fps，曝光时间使用 83ms 即可，曝光时间越短，fpn 上所能暴露的坏点越少，label 所学习的去坏点能力也越弱。

FPN 是和 sensor 强绑定数据，如果更换采集的 sensor，需要一并更换对应的 fpn 数据。推理的时候也同样需要使用对于的 FPN

部分 sensor 出厂时带有 dpc 矫正功能，依据实际情况来决定是否打开，有些 sensor 打开 dpc 之后对图像清晰度模糊影响较大，建议关闭。

2.2.2 采集蒸馏 Raw 数据

步骤 1 设置当前室内室外 iso 下需要采集 Raw 数据的 fps，一般为当前 iso 下对应的 fps 即可；

步骤 2 调整环境光到对应的 ISO 后，设置 AE bypass enable；采集的时候，前方采集视场区域不要用灯光,将灯光布置在采集设备之后；固定 ae 的目的是，采集的时候，轻微转动角度，ae 可能会有轻微变化，但是这个变化不太大，所以可以固定 ae 去采集；如果转动角度之后，ae 变化很大，过曝或者过暗，就需要重新调整环境布置。下图是一

个比较理想的灯光布置图，可以使采集设备转动角度的时候，前方采集场景灯光较为均匀；

图2-18 采集场景示意图



下面是我司其中一个实际采集场景的灯光配置，采集实验室的长宽为 7m*7m 的一个方形实验室。

图2-19 采集场景实际布置



图2-20 过曝场景



上图是一个不太合适的左边场景过曝的场景示意图，可能出现偏粉问题，训练时过曝光区域也是无效数据。

步骤 3 采集 10 帧，图像中须有人物运动；作为一个采集视角；

步骤 4 每抓完 10 帧，需调整角度继续采，一组数据需 10 个视角，水平 5 个，降低视角 5 个，可刚好覆盖当前采集场景即可；采集完当前 iso 下的 10 个角度之后，再调整环境亮度，继续下一个 iso 的采集，iso 越高，raw 图越黑，如下图所示。

图2-21 旋转角度示意图



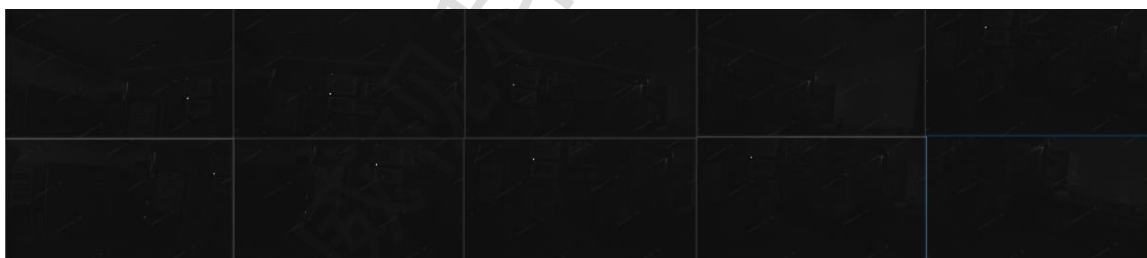
图2-22 iso1w 场景的实际 raw 图



图2-23 iso8w 场景的实际 raw 图



图2-24 iso25w 场景的实际 raw 图



须知

在采集数据时有视场角变化但是 ae 是固定的，尽量保持前方视场角灯光均匀。亮区多了（例如前方或者侧面有灯光）就有可能存在过曝的可能，过曝区域会影响训练数据质量，导致 NAN 等问题。

步骤 5 获取当前 awb&ccm 参数并保存，一般需覆盖 iso_1w-iso_max，中间 iso 均匀过渡；

步骤 6 在 pqtool 中打开采集的 raw 图，观察图像亮度是否正常，采集结点是否正确，要求非 AINR 模式下采集，如果为 ai 模式下采集的图，图像会明显偏亮；下图是左边是较高 iso 下，非 ai 模式下采集的图像，右边是 ai 模式下采集的图像。

图2-25 较低环境亮度下两种模式下采集的图像

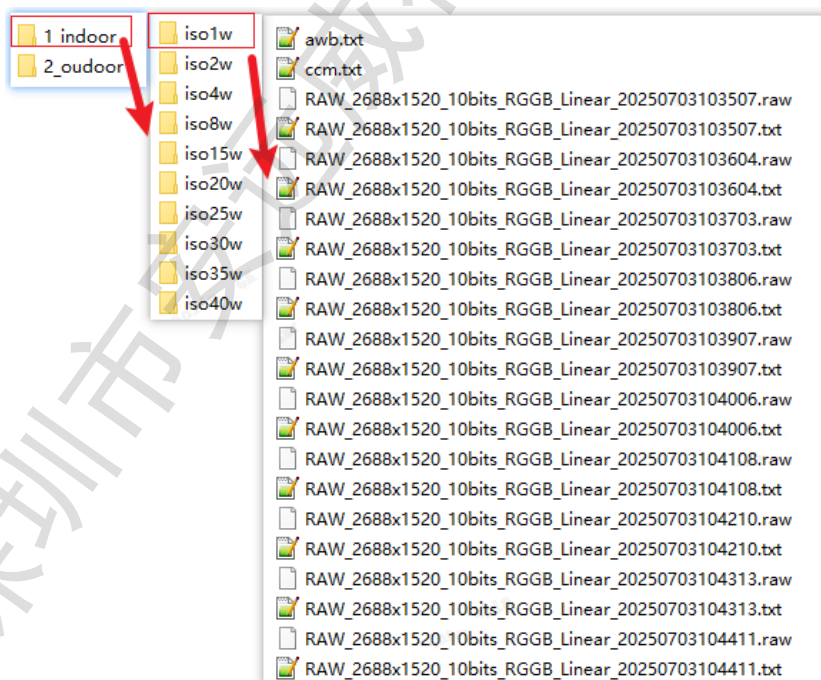


须知

训练使用的蒸馏数据是 again 打满以后采集，nr 标定数据是采集到 againmax 下的，因为 Againmax 后面的增益都是数字增益，噪声参数可以拟合出来的；蒸馏数据是训 ai 的，ai 处理的场景都是 againmax 以上的场景。

采集的蒸馏 Raw 数据低 ISO 段 ISO 分布密度高，高 ISO 段 ISO 分布密度低，比如 iso1w,iso3w,iso5w,iso8w,iso11w,iso15w,iso20w，如下图所示是一个 again_max 128 倍和 ispdgain_16 倍的 sensor 的一个采集例子。

图2-26 一组蒸馏数据格式示意

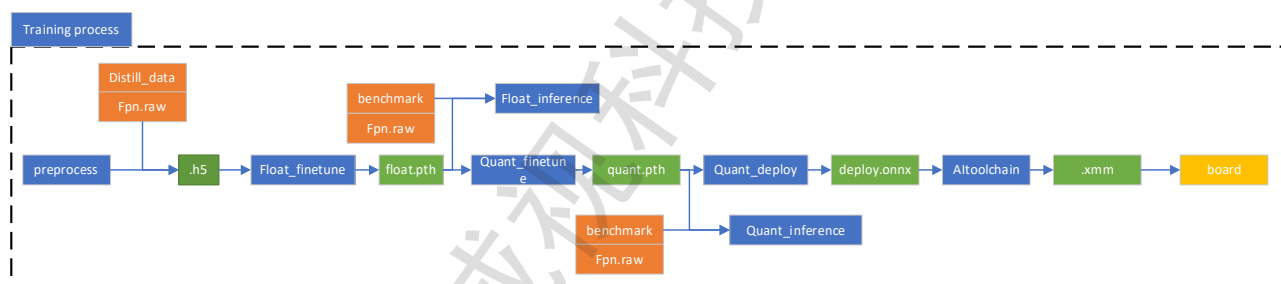


2.2.3 采集 Benchmark Raw 数据

Benchmark Raw 主要是再 float infer 和 quant infer 阶段推理验证模型效果使用。

- 步骤 1 设置当前需要采集 Raw 数据的 fps，一般为当前 iso 下对应的 fps 即可；
- 步骤 2 调整环境光到对应的 ISO 后，设置 AE bypass enable；
- 步骤 3 采集 100 帧，图像中须有人物运动；人物先运动再点 pq tool capture 采集；
- 步骤 4 获取当前 awb&ccm 参数并保存；
- 步骤 5 继续调整环境光到合适的 ISO 采集，一般需覆盖 iso 1w-iso_max，中间 iso 均匀过渡。
- 步骤 6 采集完蒸馏数据，benchmark 和 fpn 之后，就可以开始训练，下图是训练大致步骤。

图2-27 训练大体流程图



2.3 preprocess

preprocess 是通过大模型生成当前蒸馏数据的 noise-clear raw 数据对。

- 步骤 1 配置 yml 文件。

state 配成：preprocess，

substate 配成：preprocess_sm

- 步骤 2 在 data 字段填写 bit,max_dgain,max_iso,bayer_pattern 等参数。

填写当前 sensor id，以 04a10 为例

sensor_id : "04a10"

- 步骤 3 同时再 yml 末尾填写 sensor 的高斯泊松标定参数。

04a10:

KA : 0.0001316532

KB : 0.0195275507
BA : 0.0000000310
BB : -0.0000002260
BC : 0.0101759057

步骤 4 填写 fpn 路径和 bit。

fpn_dir : "/path_to/fpn.raw"
fpn_bit : 12

步骤 5 填写 distill_data 数据路径。

data_dir : "/path_to/distill_data"

填写 dump 路径，这里回生成一个 time_sensor_preprocess.h5 文件用于保存 preprocess 的结果

dump_pre_root : './preprocess'

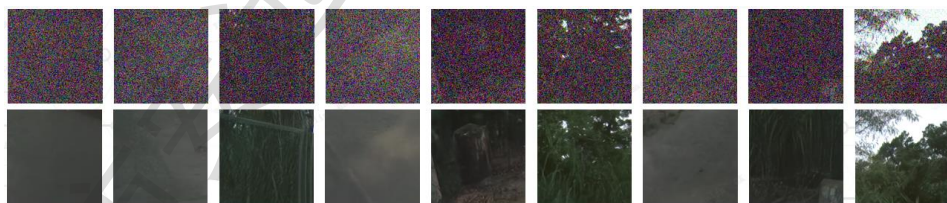
步骤 6 在 preprocess 字段填写 teacher 模型的路径。

teacher : "./checkpoints/pretrained_model/teacher_ect.pth"

height 和 width 即是当前 sensor 宽高

步骤 7 运行程序，运行结束后，在 dump_pre_root 路径下，会生成当前数据集的可视化结果 (input-label)，可以用于判断 preprocess 过程是否正常,当前目录也会生成相关打印 log，下面是生成的 noise-clear raw 数据对。

图2-28 input-label



须知

sensor 驱动中 ispdgain_sep_thd 和 yml 文件中的 max_dgain 保持一致，AINR 模式下，ispdgain 在 FE 和 BE 进行分别配置的阈值，当 ispdgain 小于等于 ispdgain_sep_thd 时，生效点在 FE；当 ispdgain 大于等于 ispdgain_sep_thd 时，不超过阈值部分在 FE 生效，其余部分在 BE 生效。

我司的 ai 是在 again_max 启动之后才会运行；again_max 越小，启动 ai 的 iso 越小；a/b 参数小数点较多，极易填错，请仔细核对。

2.4 float_finetune

float_finetune 是用于训练生成浮点调试小模型。

步骤 1 输入 state 和 substate 如下设置。

state: "float"

substate: "float_finetune"

步骤 2 填写训练数据 h5 路径，即上一步 preprocess 生成的 h5。

train_root : "/path_to_h5/"

train_name : ["xxxxx.h5"]

valid_name : "xxxx.h5"

rain_name 是训练集，valid_name 是验证集，(验证集数量可以少一些)

须知

在原有数据的基础上添加新数据的方法。

1. 将采集的新数据训练生成新的 new1_h5。
2. 将原来的 train_name : ["old1.h5", "old2.h5"] 添加上新的 h5，即，

train_name : ["old1.h5", "old2.h5", "new1_h5"]

这样新的数据就会和原来的数据一起在 floatfine 阶段训练。

步骤 3 在 train 字段设置 stage 1, 和 epoch 的数量。

stage : 1 # [1, 2], 一阶段二阶段分别写 1 , 2

start_epoch : 0

total_epoch : 60

步骤 4 在 model 字段下，填写 save_root 和 save_freq, save_root 是当前步骤保存 pth, log, 以及中间 train_dump 小图的路径。

save_root : "./checkpoints/"

save_freq : 1000

步骤 5 默认初始化 float_finetone 模型位置，使用对应分辨率的.pth, 网格 net_name 选为对应的大小，例如 2688*1520 分辨率选择 4M。

net_name : "4M_12g"

best_float_model : "./checkpoints/pretrained_model/4M_12g_float_ect.pth"

训练中途会输出一些 pth 模型文件，可以选择 PSNR 较高的，用于下一步量化使用。

图2-29 float_finetune 输出 pth

```

AISMCserver08:~/work/temp/AIBNTools_build20250210/checkpoints/lm/2_sc501ai_float_finetune_20250212_2058_sc501ai_float_finetune> ls
2_sc501ai_float_finetune_202502121952.yml.log  model_12000.pth  model_16000.pth  model_18500.pth  model_22500.pth  model_318500.pth  model_4000.pth  model_5500.pth  model_72000.pth  train_dump
model_0.pth  model_105000.pth  model_16500.pth  model_19000.pth  model_23000.pth  model_319500.pth  model_5000.pth  model_6000.pth  model_82500.pth
model_100000.pth  model_1500.pth  model_169500.pth  model_19500.pth  model_275000.pth  model_33000.pth  model_500.pth  model_61000.pth  model_90000.pth
model_1000.pth  model_153500.pth  model_170500.pth  model_211500.pth  model_290000.pth  model_3500.pth  model_50500.pth  model_6500.pth  model_9000.pth
model_100000.pth  model_15500.pth  model_183000.pth  model_217000.pth  model_313500.pth  model_36000.pth  model_52500.pth  model_65500.pth  model_9500.pth
    
```

在打印中包含了当前训练的 psnr 信息。

图2-30 打印中读取 psnr 信息

```

50 INFO: init AIBNR valid dataset loader success
52 INFO: eval average psnr | 34.657
40 INFO: Iter:00779/Epoch:0011 | loss:0.0224 | time:0.1542
48 INFO: Iter:00799/Epoch:0011 | loss:0.0270 | time:0.1580
90 INFO: Iter:00819/Epoch:0011 | loss:0.0303 | time:0.1526
60 INFO: Iter:00839/Epoch:0011 | loss:0.0207 | time:0.1533
49 INFO: Iter:00859/Epoch:0011 | loss:0.0234 | time:0.1479
18 INFO: Iter:00879/Epoch:0011 | loss:0.0358 | time:0.1761
73 INFO: Iter:00899/Epoch:0011 | loss:0.0218 | time:0.1779
55 INFO: Iter:00919/Epoch:0011 | loss:0.0243 | time:0.1777
18 INFO: Iter:00939/Epoch:0011 | loss:0.0216 | time:0.1724
78 INFO: Iter:00959/Epoch:0011 | loss:0.0237 | time:0.1769
25 INFO: Iter:00979/Epoch:0011 | loss:0.0236 | time:0.1787
93 INFO: Iter:00999/Epoch:0011 | loss:0.0204 | time:0.1777
62 INFO: save iter:14507 model success
35 INFO: Iter:01019/Epoch:0011 | loss:0.0228 | time:0.1787
17 INFO: Iter:01039/Epoch:0011 | loss:0.0191 | time:0.1785
78 INFO: Iter:01059/Epoch:0011 | loss:0.0223 | time:0.1782
57 INFO: Iter:01079/Epoch:0011 | loss:0.0238 | time:0.1779
25 INFO: Iter:01099/Epoch:0011 | loss:0.0217 | time:0.1785
57 INFO: Iter:01119/Epoch:0011 | loss:0.0314 | time:0.1798
94 INFO: Iter:01139/Epoch:0011 | loss:0.0193 | time:0.1824
38 INFO: Iter:01159/Epoch:0011 | loss:0.0268 | time:0.1775
42 INFO: Iter:01179/Epoch:0011 | loss:0.0301 | time:0.1824
11 INFO: Iter:01199/Epoch:0011 | loss:0.0285 | time:0.1781
15 INFO: Iter:01219/Epoch:0011 | loss:0.0192 | time:0.1750
93 INFO: Iter:00019/Epoch:0012 | loss:0.0302 | time:0.1886
56 INFO: Iter:00039/Epoch:0012 | loss:0.0344 | time:0.1822
92 INFO: init AIBNR valid dataset loader success
62 INFO: eval average psnr | 34.674
62 INFO: Iter:00059/Epoch:0012 | loss:0.0206 | time:0.1518
91 INFO: Iter:00079/Epoch:0012 | loss:0.0292 | time:0.1530
    
```

一些中途转换出来的小图，在当前 pth 路径下 train_dump 保存了对应的训练小图和 debug 信息。

```

AISMCserver08:~/work/temp/AIBNTools_build20250210/checkpoints/lm/2_sc501ai_float_finetune_20250212_2058_sc501ai_float_finetune> ls
2_sc501ai_float_finetune_202502121952.yml.log  model_12000.pth  model_16000.pth  model_18500.pth  model_22500.pth  model_318500.pth  model_4000.pth  model_5500.pth  model_72000.pth  train_dump
model_0.pth  model_105000.pth  model_16500.pth  model_19000.pth  model_23000.pth  model_319500.pth  model_5000.pth  model_6000.pth  model_82500.pth
model_100000.pth  model_1500.pth  model_169500.pth  model_19500.pth  model_275000.pth  model_33000.pth  model_500.pth  model_61000.pth  model_90000.pth
model_1000.pth  model_153500.pth  model_170500.pth  model_211500.pth  model_290000.pth  model_3500.pth  model_50500.pth  model_6500.pth  model_9000.pth
model_100000.pth  model_15500.pth  model_183000.pth  model_217000.pth  model_313500.pth  model_36000.pth  model_52500.pth  model_65500.pth  model_9500.pth

AISMCserver08:~/work/temp/AIBNTools_build20250210/checkpoints/lm/2_sc501ai_float_finetune_20250212_2058_sc501ai_float_finetune/train_dump> ls
Epoch_0000_Iter_00199_input.png  Epoch_0070_Iter_00399_input.png  Epoch_0140_Iter_00599_input.png  Epoch_0210_Iter_00799_input.png  Epoch_0281_Iter_00199_input.png  Epoch_0351_Iter_00399_input.png
Epoch_0000_Iter_00199_label.png  Epoch_0070_Iter_00399_label.png  Epoch_0140_Iter_00599_label.png  Epoch_0210_Iter_00799_label.png  Epoch_0281_Iter_00199_label.png  Epoch_0351_Iter_00399_label.png
Epoch_0000_Iter_00199_output.png  Epoch_0070_Iter_00399_output.png  Epoch_0140_Iter_00599_output.png  Epoch_0210_Iter_00799_output.png  Epoch_0281_Iter_00199_output.png  Epoch_0351_Iter_00399_output.png
Epoch_0000_Iter_00399_input.png  Epoch_0070_Iter_00599_input.png  Epoch_0140_Iter_00799_input.png  Epoch_0211_Iter_00199_input.png  Epoch_0281_Iter_00399_input.png  Epoch_0351_Iter_00599_input.png
Epoch_0000_Iter_00399_label.png  Epoch_0070_Iter_00599_label.png  Epoch_0140_Iter_00799_label.png  Epoch_0211_Iter_00199_label.png  Epoch_0281_Iter_00399_label.png  Epoch_0351_Iter_00599_label.png
Epoch_0000_Iter_00399_output.png  Epoch_0070_Iter_00599_output.png  Epoch_0140_Iter_00799_output.png  Epoch_0211_Iter_00199_output.png  Epoch_0281_Iter_00399_output.png  Epoch_0351_Iter_00599_output.png
Epoch_0000_Iter_00599_input.png  Epoch_0070_Iter_00799_input.png  Epoch_0141_Iter_00199_input.png  Epoch_0211_Iter_00399_input.png  Epoch_0281_Iter_00599_input.png  Epoch_0351_Iter_00799_input.png
Epoch_0000_Iter_00599_label.png  Epoch_0070_Iter_00799_label.png  Epoch_0141_Iter_00199_label.png  Epoch_0211_Iter_00399_label.png  Epoch_0281_Iter_00599_label.png  Epoch_0351_Iter_00799_label.png
    
```

在 train_dump 路径下生成 input, label, predict 相关图像。

图2-31 浮点训练结果(input-label-predict)



须知

finetune 时概率可能损失值为 nan，首先看下 train_dump 的小图有没有异常，若数据有问题的时候，可能 nan；检查下 yml 里面的 fpn bit sensor bit 是否正确，黑帧最好有个 100 帧。

同时可以先检查下 input 和 label 的内容能不能对应上，检查下颜色亮度是否也一致，如果所有数据都是正常的，尝试调小一下学习率，batch size 可以试下调大一点，例如 24，32，同时 raw loss 可以去掉尝试。

建议 float_finetune 训练到 60epoch 以上。

2.5 float_inference

float_inference 可以直观看到 float_finetune 训练的结果。

步骤 1 输入 state 和 substate 如下设置。

```
state: "float"          # ["preprocess", "float", "quant"]
substate: "float_inference"
```

步骤 2 float_inference 使用的是 float_finetune 训练出来的 pth 模型,使用的是 psnr 值较高时对应的 pth 模型。

```
best_float_model : "/path_to/best_float_model.pth"
```

步骤 3 同时在 data 字段填写 benchmark 的路径,用于推理。

```
infer_dir : "/path_to/benchmark/"
infer_result : './results/aibnr'
```

步骤 4 infer 字段下的 select_iso 用于选择只推理当前 iso 下的 benchmark。

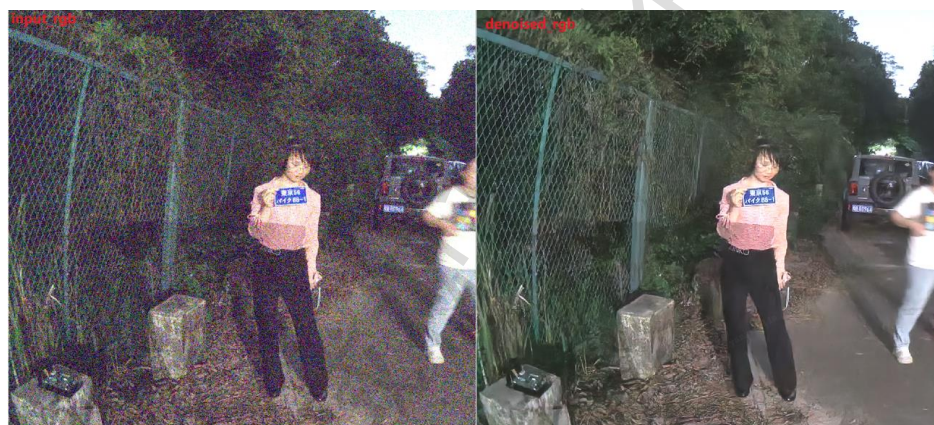
select_iso : False
select_range : [[6000, 20000]]

步骤 5 跑完 float_inference 之后查看输出结果，在 result 文件对应的目录下，其中对应的 debug 信息也在此目录下。

图2-32 float_inference 生成结果

```
AISWearVer08\c:\work\temp\AISWearTools\build\20250210\results\abmr\9_sc50a1_floatinfer_202502131010\20250213_1010_sc50a1_float_inference\ts
2_sc50a1a_floatinfer_202502121052_yyl\log
RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso11939_BGR to RGB input_rgb_000.png RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso11939_BGR to RGB nrr_out_000.png
RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso11939_BGR to RGB nrr_out_000.png RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso11939_BGR to RGB output_rgb_000.png
RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso15887_BGR to RGB input_rgb_000.png RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso15887_BGR to RGB nrr_out_000.png
RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso15887_BGR to RGB nrr_out_000.png RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso15887_BGR to RGB output_rgb_000.png
RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso1648_BGR to RGB input_rgb_000.png RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso1648_BGR to RGB nrr_out_000.png
RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso1648_BGR to RGB nrr_out_000.png RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso1648_BGR to RGB output_rgb_000.png
RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso24207_BGR to RGB input_rgb_000.png RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso24207_BGR to RGB nrr_out_000.png
RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso24207_BGR to RGB nrr_out_000.png RAW_2880x1620_12bits_BGR_L_infer_sc50a1a_benchmark_indoor_iso24207_BGR to RGB output_rgb_000.png
```

图2-33 float_inference 效果展示



须知

如果 float_inference 出效果的部分区域涂抹感较严重，并且调整纹理相关的参数，如果这部分区域效果也不大，可以调下 motion loss；motion loss 的含义是，图像高频信息可以体现出来运动，5 通道主要是用来做运动检测的，所以对应的 loss 就叫 motion loss，实际上 5 通道就是输出的图像边缘。

2.6 quant_finetime

quant_finetime 是用于将浮点调试小模型转化成量化调试小模型。

quant_finetime 分为两个阶段

每个阶段 60 轮，调整到 20~30 轮也可，在 train 字段设置不同的 stage

stage : 1 # [1, 2], 一阶段二阶段分别写 1 , 2

```
start_epoch      : 0
total_epoch      : 60
```

2.6.1 1 阶段

步骤 1 输入 state 和 substate 如下设置。

```
state: "quant"
```

```
substate: "quant_finetune"
```

train_root, train_name, valid_name 和上阶段 floatfinetune 一致就可以

步骤 2 在 model 选择上阶段 float_finetune 上阶段选出来的.pth 用于量化。

步骤 3 best_float_model : "./path_to/best_float_model.pth"。

在 model 字段下, 填写 save_root 和 save_freq, save_root 是当前步骤保存 quant_finetune_stage1.pth,log,以及中间 train_dump 小图的路径

```
save_root        : "./checkpoints/"
```

```
save_freq        : 1000
```

步骤 4 运行后, 在当前 save_root 路径下会生成量化后.pth, log, 以及 dump 的训练图像。

图2-34 quant_finetune 生成文件

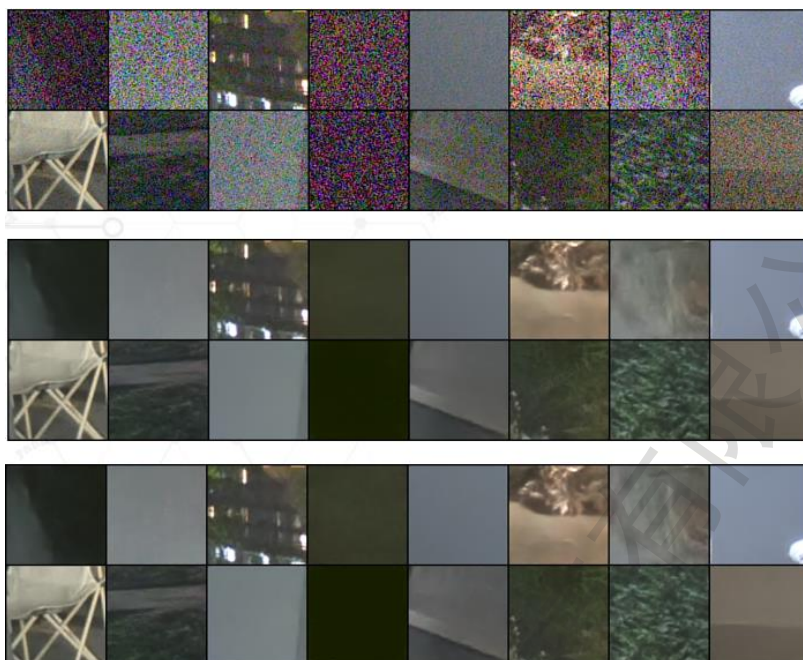
```
AIWServer08:/work/temp/AIBNtools_build20250210/checkpoints/ln/4_sc501ai_quant_finetune_stage1_202502131145/20250213_1144_sc501ai_quant_finetune> ls
2_sc501ai_float_finetune_202502121952_yal_log
model_0.pth
model_101513.pth
model_101246.pth
model_104079.pth
model_105008.pth
model_109712.pth
model_108445.pth
model_110178.pth
model_111011.pth
model_113644.pth
model_113079.pth
model_115377.pth
model_117118.pth
model_118043.pth
model_120576.pth
model_122399.pth
model_124042.pth
model_125775.pth
model_126508.pth
model_127508.pth
model_129241.pth
model_130594.pth
model_131330.pth
model_131500.pth
model_132797.pth
model_134440.pth
model_136372.pth
model_137906.pth
model_139639.pth
model_141372.pth
model_141105.pth
model_144838.pth
model_146571.pth
model_148304.pth
model_146633.pth
model_150037.pth
model_15099.pth
model_151776.pth
model_153503.pth
model_155236.pth
model_156969.pth
model_158702.pth
model_160435.pth
model_162168.pth
model_163903.pth
model_165634.pth
model_165999.pth
model_168628.pth
model_168313.pth
model_167367.pth
model_168062.pth
model_169190.pth
model_170831.pth
model_172566.pth
model_174299.pth
model_176032.pth
model_177765.pth
model_179498.pth
model_179500.pth
model_181233.pth
model_182964.pth
model_183229.pth
model_184697.pth
model_186430.pth
model_188163.pth
model_189896.pth
model_191629.pth
model_193362.pth
model_195100.pth
model_196828.pth
model_198561.pth
model_198926.pth
model_200594.pth
model_200862.pth
model_202027.pth
model_203760.pth
model_205493.pth
model_207228.pth
model_208959.pth
model_210692.pth
model_212424.pth
model_214159.pth
model_215893.pth
model_217624.pth
model_219357.pth
model_221090.pth
model_222823.pth
model_224556.pth
model_226289.pth
model_228022.pth
model_229755.pth
model_231488.pth
model_233221.pth
model_234954.pth
model_236687.pth
model_238420.pth
model_240153.pth
model_241886.pth
model_243619.pth
model_245352.pth
model_247085.pth
model_248818.pth
model_250551.pth
model_252284.pth
model_254017.pth
model_255750.pth
model_257483.pth
model_259216.pth
model_260949.pth
model_262682.pth
model_264415.pth
model_266148.pth
model_267881.pth
model_269614.pth
model_271347.pth
model_273080.pth
model_274813.pth
model_276546.pth
model_278279.pth
model_280012.pth
model_281745.pth
model_283478.pth
model_285211.pth
model_286944.pth
model_288677.pth
model_290410.pth
model_292143.pth
model_293876.pth
model_295609.pth
model_297342.pth
model_299075.pth
model_300808.pth
model_302541.pth
model_304274.pth
model_306007.pth
model_307740.pth
model_309473.pth
model_311206.pth
model_312939.pth
model_314672.pth
model_316405.pth
model_318138.pth
model_319871.pth
model_321604.pth
model_323337.pth
model_325070.pth
model_326803.pth
model_328536.pth
model_330269.pth
model_332002.pth
model_333735.pth
model_335468.pth
model_337201.pth
model_338934.pth
model_340667.pth
model_342400.pth
model_344133.pth
model_345866.pth
model_347599.pth
model_349332.pth
model_351065.pth
model_352798.pth
model_354531.pth
model_356264.pth
model_358007.pth
model_359740.pth
model_361473.pth
model_363206.pth
model_364939.pth
model_366672.pth
model_368405.pth
model_370138.pth
model_371871.pth
model_373604.pth
model_375337.pth
model_377070.pth
model_378803.pth
model_380536.pth
model_382269.pth
model_384002.pth
model_385735.pth
model_387468.pth
model_389201.pth
model_390934.pth
model_392667.pth
model_394399.pth
model_396132.pth
model_397865.pth
model_399598.pth
model_401331.pth
model_403064.pth
model_404797.pth
model_406530.pth
model_408263.pth
model_410007.pth
model_411740.pth
model_413473.pth
model_415206.pth
model_416939.pth
model_418672.pth
model_420405.pth
model_422138.pth
model_423871.pth
model_425604.pth
model_427337.pth
model_429070.pth
model_430803.pth
model_432536.pth
model_434269.pth
model_436002.pth
model_437735.pth
model_439468.pth
model_441201.pth
model_442934.pth
model_444667.pth
model_446399.pth
model_448132.pth
model_449865.pth
model_451598.pth
model_453331.pth
model_455064.pth
model_456797.pth
model_458530.pth
model_460263.pth
model_461996.pth
model_463729.pth
model_465462.pth
model_467195.pth
model_468928.pth
model_470661.pth
model_472394.pth
model_474127.pth
model_475860.pth
model_477593.pth
model_479326.pth
model_481059.pth
model_482792.pth
model_484525.pth
model_486258.pth
model_487991.pth
model_489724.pth
model_491457.pth
model_493190.pth
model_494923.pth
model_496656.pth
model_498389.pth
model_500122.pth
model_501855.pth
model_503588.pth
model_505321.pth
model_507054.pth
model_508787.pth
model_510520.pth
model_512253.pth
model_513986.pth
model_515719.pth
model_517452.pth
model_519185.pth
model_520918.pth
model_522651.pth
model_524384.pth
model_526117.pth
model_527850.pth
model_529583.pth
model_531316.pth
model_533049.pth
model_534782.pth
model_536515.pth
model_538248.pth
model_539981.pth
model_541714.pth
model_543447.pth
model_545180.pth
model_546913.pth
model_548646.pth
model_550379.pth
model_552112.pth
model_553845.pth
model_555578.pth
model_557311.pth
model_559044.pth
model_560777.pth
model_562510.pth
model_564243.pth
model_565976.pth
model_567709.pth
model_569442.pth
model_571175.pth
model_572908.pth
model_574641.pth
model_576374.pth
model_578107.pth
model_579840.pth
model_581573.pth
model_583306.pth
model_585039.pth
model_586772.pth
model_588505.pth
model_590238.pth
model_591971.pth
model_593704.pth
model_595437.pth
model_597170.pth
model_598903.pth
model_600636.pth
model_602369.pth
model_604102.pth
model_605835.pth
model_607568.pth
model_609301.pth
model_611034.pth
model_612767.pth
model_614500.pth
model_616233.pth
model_617966.pth
model_619699.pth
model_621432.pth
model_623165.pth
model_624898.pth
model_626631.pth
model_628364.pth
model_630097.pth
model_631830.pth
model_633563.pth
model_635296.pth
model_637029.pth
model_638762.pth
model_640495.pth
model_642228.pth
model_643961.pth
model_645694.pth
model_647427.pth
model_649160.pth
model_650893.pth
model_652626.pth
model_654359.pth
model_656092.pth
model_657825.pth
model_659558.pth
model_661291.pth
model_663024.pth
model_664757.pth
model_666489.pth
model_668222.pth
model_669955.pth
model_671688.pth
model_673421.pth
model_675154.pth
model_676887.pth
model_678620.pth
model_680353.pth
model_682086.pth
model_683819.pth
model_685552.pth
model_687285.pth
model_689018.pth
model_690751.pth
model_692484.pth
model_694217.pth
model_695950.pth
model_697683.pth
model_699416.pth
model_701149.pth
model_702882.pth
model_704615.pth
model_706348.pth
model_708081.pth
model_709814.pth
model_711547.pth
model_713280.pth
model_715013.pth
model_716746.pth
model_718479.pth
model_720212.pth
model_721945.pth
model_723678.pth
model_725411.pth
model_727144.pth
model_728877.pth
model_730610.pth
model_732343.pth
model_734076.pth
model_735809.pth
model_737542.pth
model_739275.pth
model_741008.pth
model_742741.pth
model_744474.pth
model_746207.pth
model_747940.pth
model_749673.pth
model_751406.pth
model_753139.pth
model_754872.pth
model_756605.pth
model_758338.pth
model_760071.pth
model_761804.pth
model_763537.pth
model_765270.pth
model_767003.pth
model_768736.pth
model_770469.pth
model_772202.pth
model_773935.pth
model_775668.pth
model_777401.pth
model_779134.pth
model_780867.pth
model_782600.pth
model_784333.pth
model_786066.pth
model_787799.pth
model_789532.pth
model_791265.pth
model_793007.pth
model_794740.pth
model_796473.pth
model_798206.pth
model_799939.pth
model_801672.pth
model_803405.pth
model_805138.pth
model_806871.pth
model_808604.pth
model_810337.pth
model_812070.pth
model_813803.pth
model_815536.pth
model_817269.pth
model_819002.pth
model_820735.pth
model_822468.pth
model_824201.pth
model_825934.pth
model_827667.pth
model_829400.pth
model_831133.pth
model_832866.pth
model_834599.pth
model_836332.pth
model_838065.pth
model_839798.pth
model_841531.pth
model_843264.pth
model_845007.pth
model_846740.pth
model_848473.pth
model_850206.pth
model_851939.pth
model_853672.pth
model_855405.pth
model_857138.pth
model_858871.pth
model_860604.pth
model_862337.pth
model_864070.pth
model_865803.pth
model_867536.pth
model_869269.pth
model_871002.pth
model_872735.pth
model_874468.pth
model_876201.pth
model_877934.pth
model_879667.pth
model_881400.pth
model_883133.pth
model_884866.pth
model_886599.pth
model_888332.pth
model_890065.pth
model_891798.pth
model_893531.pth
model_895264.pth
model_897007.pth
model_898740.pth
model_900473.pth
model_902206.pth
model_903939.pth
model_905672.pth
model_907405.pth
model_909138.pth
model_910871.pth
model_912604.pth
model_914337.pth
model_916070.pth
model_917803.pth
model_919536.pth
model_921269.pth
model_923002.pth
model_924735.pth
model_926468.pth
model_928201.pth
model_929934.pth
model_931667.pth
model_933400.pth
model_935133.pth
model_936866.pth
model_938599.pth
model_940332.pth
model_942065.pth
model_943798.pth
model_945531.pth
model_947264.pth
model_949007.pth
model_950740.pth
model_952473.pth
model_954206.pth
model_955939.pth
model_957672.pth
model_959405.pth
model_961138.pth
model_962871.pth
model_964604.pth
model_966337.pth
model_968069.pth
model_969802.pth
model_971535.pth
model_973268.pth
model_975001.pth
model_976734.pth
model_978467.pth
model_980200.pth
model_981933.pth
model_983666.pth
model_985399.pth
model_987132.pth
model_988865.pth
model_990598.pth
model_992331.pth
model_994064.pth
model_995797.pth
model_997530.pth
model_999263.pth
model_1000000.pth
```

在 train_dump 路径会生成当前训练的 input, label, predict 图像。

图2-35 quant_finetune 生成图像

```
AIWServer08:/work/temp/AIBNtools_build20250210/checkpoints/ln/4_sc501ai_quant_finetune_stage1_202502131145/20250213_1144_sc501ai_quant_finetune/train_dump> ls
Epoch_0000_Iter_00199_input.png
Epoch_0000_Iter_00199_label.png
Epoch_0000_Iter_00199_output.png
Epoch_0000_Iter_00399_input.png
Epoch_0000_Iter_00399_label.png
Epoch_0000_Iter_00399_output.png
Epoch_0000_Iter_00599_input.png
Epoch_0000_Iter_00599_label.png
Epoch_0000_Iter_00599_output.png
Epoch_0000_Iter_00799_input.png
Epoch_0000_Iter_00799_label.png
Epoch_0000_Iter_00799_output.png
Epoch_0000_Iter_00999_input.png
Epoch_0000_Iter_00999_label.png
Epoch_0000_Iter_00999_output.png
Epoch_0000_Iter_01199_input.png
Epoch_0000_Iter_01199_label.png
Epoch_0000_Iter_01199_output.png
Epoch_0060_Iter_01199_input.png
Epoch_0060_Iter_01199_label.png
Epoch_0060_Iter_01199_output.png
Epoch_0100_Iter_01199_input.png
Epoch_0100_Iter_01199_label.png
Epoch_0100_Iter_01199_output.png
Epoch_0140_Iter_01199_input.png
Epoch_0140_Iter_01199_label.png
Epoch_0140_Iter_01199_output.png
Epoch_0180_Iter_01199_input.png
Epoch_0180_Iter_01199_label.png
Epoch_0180_Iter_01199_output.png
Epoch_0220_Iter_01199_input.png
Epoch_0220_Iter_01199_label.png
Epoch_0220_Iter_01199_output.png
Epoch_0260_Iter_01199_input.png
Epoch_0260_Iter_01199_label.png
Epoch_0260_Iter_01199_output.png
Epoch_0300_Iter_01199_input.png
Epoch_0300_Iter_01199_label.png
Epoch_0300_Iter_01199_output.png
Epoch_0340_Iter_01199_input.png
Epoch_0340_Iter_01199_label.png
Epoch_0340_Iter_01199_output.png
Epoch_0380_Iter_01199_input.png
Epoch_0380_Iter_01199_label.png
Epoch_0380_Iter_01199_output.png
Epoch_0420_Iter_01199_input.png
Epoch_0420_Iter_01199_label.png
Epoch_0420_Iter_01199_output.png
Epoch_0460_Iter_01199_input.png
Epoch_0460_Iter_01199_label.png
Epoch_0460_Iter_01199_output.png
Epoch_0500_Iter_01199_input.png
Epoch_0500_Iter_01199_label.png
Epoch_0500_Iter_01199_output.png
Epoch_0540_Iter_01199_input.png
Epoch_0540_Iter_01199_label.png
Epoch_0540_Iter_01199_output.png
Epoch_0580_Iter_01199_input.png
Epoch_0580_Iter_01199_label.png
Epoch_0580_Iter_01199_output.png
Epoch_0620_Iter_01199_input.png
Epoch_0620_Iter_01199_label.png
Epoch_0620_Iter_01199_output.png
Epoch_0660_Iter_01199_input.png
Epoch_0660_Iter_01199_label.png
Epoch_0660_Iter_01199_output.png
Epoch_0700_Iter_01199_input.png
Epoch_0700_Iter_01199_label.png
Epoch_0700_Iter_01199_output.png
Epoch_0740_Iter_01199_input.png
Epoch_0740_Iter_01199_label.png
Epoch_0740_Iter_01199_output.png
Epoch_0780_Iter_01199_input.png
Epoch_0780_Iter_01199_label.png
Epoch_0780_Iter_01199_output.png
Epoch_0820_Iter_01199_input.png
Epoch_0820_Iter_01199_label.png
Epoch_0820_Iter_01199_output.png
Epoch_0860_Iter_01199_input.png
Epoch_0860_Iter_01199_label.png
Epoch_0860_Iter_01199_output.png
Epoch_0900_Iter_01199_input.png
Epoch_0900_Iter_01199_label.png
Epoch_0900_Iter_01199_output.png
Epoch_0940_Iter_01199_input.png
Epoch_0940_Iter_01199_label.png
Epoch_0940_Iter_01199_output.png
Epoch_0980_Iter_01199_input.png
Epoch_0980_Iter_01199_label.png
Epoch_0980_Iter_01199_output.png
Epoch_1020_Iter_01199_input.png
Epoch_1020_Iter_01199_label.png
Epoch_1020_Iter_01199_output.png
Epoch_1060_Iter_01199_input.png
Epoch_1060_Iter_01199_label.png
Epoch_1060_Iter_01199_output.png
Epoch_1100_Iter_01199_input.png
Epoch_1100_Iter_01199_label.png
Epoch_1100_Iter_01199_output.png
Epoch_1140_Iter_01199_input.png
Epoch_1140_Iter_01199_label.png
Epoch_1140_Iter_01199_output.png
Epoch_1180_Iter_01199_input.png
Epoch_1180_Iter_01199_label.png
Epoch_1180_Iter_01199_output.png
Epoch_1220_Iter_01199_input.png
Epoch_1220_Iter_01199_label.png
Epoch_1220_Iter_01199_output.png
Epoch_1260_Iter_01199_input.png
Epoch_1260_Iter_01199_label.png
Epoch_1260_Iter_01199_output.png
Epoch_1300_Iter_01199_input.png
Epoch_1300_Iter_01199_label.png
Epoch_1300_Iter_01199_output.png
Epoch_1340_Iter_01199_input.png
Epoch_1340_Iter_01199_label.png
Epoch_1340_Iter_01199_output.png
Epoch_1380_Iter_01199_input.png
Epoch_1380_Iter_01199_label.png
Epoch_1380_Iter_01199_output.png
Epoch_1420_Iter_01199_input.png
Epoch_1420_Iter_01199_label.png
Epoch_1420_Iter_01199_output.png
Epoch_1460_Iter_01199_input.png
Epoch_1460_Iter_01199_label.png
Epoch_1460_Iter_01199_output.png
Epoch_1500_Iter_01199_input.png
Epoch_1500_Iter_01199_label.png
Epoch_1500_Iter_01199_output.png
Epoch_1540_Iter_01199_input.png
Epoch_1540_Iter_01199_label.png
Epoch_1540_Iter_01199_output.png
Epoch_1580_Iter_01199_input.png
Epoch_1580_Iter_01199_label.png
Epoch_1580_Iter_01199_output.png
Epoch_1620_Iter_01199_input.png
Epoch_1620_Iter_01199_label.png
Epoch_1620_Iter_01199_output.png
Epoch_1660_Iter_01199_input.png
Epoch_1660_Iter_01199_label.png
Epoch_1660_Iter_01199_output.png
Epoch_1700_Iter_01199_input.png
Epoch_1700_Iter_01199_label.png
Epoch_1700_Iter_01199_output.png
Epoch_1740_Iter_01199_input.png
Epoch_1740_Iter_01199_label.png
Epoch_1740_Iter_01199_output.png
Epoch_1780_Iter_01199_input.png
Epoch_1780_Iter_01199_label.png
Epoch_1780_Iter_01199_output.png
Epoch_1820_Iter_01199_input.png
Epoch_1820_Iter_01199_label.png
Epoch_1820_Iter_01199_output.png
Epoch_1860_Iter_01199_input.png
Epoch_1860_Iter_01199_label.png
Epoch_1860_Iter_01199_output.png
Epoch_1900_Iter_01199_input.png
Epoch_1900_Iter_01199_label.png
Epoch_1900_Iter_01199_output.png
Epoch_1940_Iter_01199_input.png
Epoch_1940_Iter_01199_label.png
Epoch_1940_Iter_01199_output.png
Epoch_1980_Iter_01199_input.png
Epoch_1980_Iter_01199_label.png
Epoch_1980_Iter_01199_output.png
Epoch_2020_Iter_01199_input.png
Epoch_2020_Iter_01199_label.png
Epoch_2020_Iter_01199_output.png
Epoch_2060_Iter_01199_input.png
Epoch_2060_Iter_01199_label.png
Epoch_2060_Iter_01199_output.png
Epoch_2100_Iter_01199_input.png
Epoch_2100_Iter_01199_label.png
Epoch_2100_Iter_01199_output.png
Epoch_2140_Iter_01199_input.png
Epoch_2140_Iter_01199_label.png
Epoch_2140_Iter_01199_output.png
Epoch_2180_Iter_01199_input.png
Epoch_2180_Iter_01199_label.png
Epoch_2180_Iter_01199_output.png
Epoch_2220_Iter_01199_input.png
Epoch_2220_Iter_01199_label.png
Epoch_2220_Iter_01199_output.png
Epoch_2260_Iter_01199_input.png
Epoch_2260_Iter_01199_label.png
Epoch_2260_Iter_01199_output.png
Epoch_2300_Iter_01199_input.png
Epoch_2300_Iter_01199_label.png
Epoch_2300_Iter_01199_output.png
Epoch_2340_Iter_01199_input.png
Epoch_2340_Iter_01199_label.png
Epoch_2340_Iter_01199_output.png
Epoch_2380_Iter_01199_input.png
Epoch_2380_Iter_01199_label.png
Epoch_2380_Iter_01199_output.png
Epoch_2420_Iter_01199_input.png
Epoch_2420_Iter_01199_label.png
Epoch_2420_Iter_01199_output.png
Epoch_2460_Iter_01199_input.png
Epoch_2460_Iter_01199_label.png
Epoch_2460_Iter_01199_output.png
Epoch_2500_Iter_01199_input.png
Epoch_2500_Iter_01199_label.png
Epoch_2500_Iter_01199_output.png
Epoch_2540_Iter_01199_input.png
Epoch_2540_Iter_01199_label.png
Epoch_2540_Iter_01199_output.png
Epoch_2580_Iter_01199_input.png
Epoch_2580_Iter_01199_label.png
Epoch_2580_Iter_01199_output.png
Epoch_2620_Iter_01199_input.png
Epoch_2620_Iter_01199_label.png
Epoch_2620_Iter_01199_output.png
Epoch_2660_Iter_01199_input.png
Epoch_2660_Iter_01199_label.png
Epoch_2660_Iter_01199_output.png
Epoch_2700_Iter_01199_input.png
Epoch_2700_Iter_01199_label.png
Epoch_2700_Iter_01199_output.png
Epoch_2740_Iter_01199_input.png
Epoch_2740_Iter_01199_label.png
Epoch_2740_Iter_01199_output.png
Epoch_2780_Iter_01199_input.png
Epoch_2780_Iter_01199_label.png
Epoch_2780_Iter_01199_output.png
Epoch_2820_Iter_01199_input.png
Epoch_2820_Iter_01199_label.png
Epoch_2820_Iter_01199_output.png
Epoch_2860_Iter_01199_input.png
Epoch_2860_Iter_01199_label.png
Epoch_2860_Iter_01199_output.png
Epoch_2900_Iter_01199_input.png
Epoch_2900_Iter_01199_label.png
Epoch_2900_Iter_01199_output.png
Epoch_2940_Iter_01199_input.png
Epoch_2940_Iter_01199_label.png
Epoch_2940_Iter_01199_output.png
Epoch_2980_Iter_01199_input.png
Epoch_2980_Iter_01199_label.png
Epoch_2980_Iter_01199_output.png
Epoch_3020_Iter_01199_input.png
Epoch_3020_Iter_01199_label.png
Epoch_3020_Iter_01199_output.png
Epoch_3060_Iter_01199_input.png
Epoch_3060_Iter_01199_label.png
Epoch_3060_Iter_01199_output.png
Epoch_3100_Iter_01199_input.png
Epoch_3100_Iter_01199_label.png
Epoch_3100_Iter_01199_output.png
Epoch_3140_Iter_01199_input.png
Epoch_3140_Iter_01199_label.png
Epoch_3140_Iter_01199_output.png
Epoch_3180_Iter_01199_input.png
Epoch_3180_Iter_01199_label.png
Epoch_3180_Iter_01199_output.png
Epoch_3220_Iter_01199_input.png
Epoch_3220_Iter_01199_label.png
Epoch_3220_Iter_01199_output.png
Epoch_3260_Iter_01199_input.png
Epoch_3260_Iter_01199_label.png
Epoch_3260_Iter_01199_output.png
Epoch_3300_Iter_01199_input.png
Epoch_3300_Iter_01199_label.png
Epoch_3300_Iter_01199_output.png
Epoch_3340_Iter_01199_input.png
Epoch_3340_Iter_01199_label.png
Epoch_3340_Iter_01199_output.png
Epoch_3380_Iter_01199_input.png
Epoch_3380_Iter_01199_label.png
Epoch_3380_Iter_01199_output.png
Epoch_3420_Iter_01199_input.png
Epoch_3420_Iter_01199_label.png
Epoch_3420_Iter_01199_output.png
Epoch_3460_Iter_01199_input.png
Epoch_3460_Iter_01199_label.png
Epoch_3460_Iter_01199_output.png
Epoch_3500_Iter_01199_input.png
Epoch_3500_Iter_01199_label.png
Epoch_3500_Iter_01199_output.png
Epoch_3540_Iter_01199_input.png
Epoch_3540_Iter_01199_label.png
Epoch_3540_Iter_01199_output.png
Epoch_3580_Iter_01199_input.png
Epoch_3580_Iter_01199_label.png
Epoch_3580_Iter_01199_output.png
Epoch_3620_Iter_01199_input.png
Epoch_3620_Iter_01199_label.png
Epoch_3620_Iter_01199_output.png
Epoch_3660_Iter_01199_input.png
Epoch_3660_Iter_01199_label.png
Epoch_3660_Iter_01199_output.png
Epoch_3700_Iter_01199_input.png
Epoch_3700_Iter_01199_label.png
Epoch_3700_Iter_01199_output.png
Epoch_3740_Iter_01199_input.png
Epoch_3740_Iter_01199_label.png
Epoch_3740_Iter_01199_output.png
Epoch_3780_Iter_01199_input.png
Epoch_3780_Iter_01199_label.png
Epoch_3780_Iter_01199_output.png
Epoch_3820_Iter_01199_input.png
Epoch_3820_Iter_01199_label.png
Epoch_3820_Iter_01199_output.png
Epoch_3860_Iter_01199_input.png
Epoch_3860_Iter_01199_label.png
Epoch_3860_Iter_01199_output.png
Epoch_3900_Iter_01199_input.png
Epoch_3900_Iter_01199_label.png
Epoch_3900_Iter_01199_output.png
Epoch_3940_Iter_01199_input.png
Epoch_3940_Iter_01199_label.png
Epoch_3940_Iter_01199_output.png
Epoch_3980_Iter_01199_input.png
Epoch_3980_Iter_01199_label.png
Epoch_3980_Iter_01199_output.png
Epoch_4020_Iter_01199_input.png
Epoch_4020_Iter_01199_label.png
Epoch_4020_Iter_01199_output.png
Epoch_4060_Iter_01199_input.png
Epoch_4060_Iter_01199_label.png
Epoch_4060_Iter_01199_output.png
Epoch_4100_Iter_01199_input.png
Epoch_4100_Iter_01199_label.png
Epoch_4100_Iter_01199_output.png
Epoch_4140_Iter_01199_input.png
Epoch_4140_Iter_01199_label.png
Epoch_4140_Iter_01199_output.png
Epoch_4180_Iter_01199_input.png
Epoch_4180_Iter_01199_label.png
Epoch_4180_Iter_01199_output.png
Epoch_4220_Iter_01199_input.png
Epoch_4220_Iter_01199_label.png
Epoch_4220_Iter_01199_output.png
Epoch_4260_Iter_01199_input.png
Epoch_4260_Iter_01199_label.png
Epoch_4260_Iter_01199_output.png
Epoch_4300_Iter_01199_input.png
Epoch_4300_Iter_01199_label.png
Epoch_4300_Iter_01199_output.png
Epoch_4340_Iter_01199_input.png
Epoch_4340_Iter_01199_label.png
Epoch_4340_Iter_01199_output.png
Epoch_4380_Iter_01199_input.png
Epoch_4380_Iter_01199_label.png
Epoch_4380_Iter_01199_output.png
Epoch_4420_Iter_01199_input.png
Epoch_4420_Iter_01199_label.png
Epoch_4420_Iter_01199_output.png
Epoch_4460_Iter_01199_input.png
Epoch_4460_Iter_01199_label.png
Epoch_4460_Iter_01199_output.png
Epoch_4500_Iter_01199_input.png
Epoch_4500_Iter_01199_label.png
Epoch_4500_Iter_01199_output.png
Epoch_4540_Iter_01199_input.png
Epoch_4540_Iter_01199_label.png
Epoch_4
```

图2-36 量化训练结果(input-label-predict)



2.6.2 2 阶段

步骤 1 Stage 2 阶段的设置基本于 stage 1 一致，需要修改 `best_quant_model`，训练上阶段的量化模型。

`best_quant_model` : `"/path_to/quant_finetune_stage1.pth"`

其中 `best_quant_model` 是上个 stage1 阶段训练的 `quant_finetune_stage1.pth`

步骤 2 同样在 `save_root` 下会保存 `quant_finetune_stage2.pth`,log,以及中间 `train_dump` 小图的路径，生成的 `pth` 会用于 `quant_inference` 和下一步的 `depoly`。

须知

建议 `quant_finetune` 训练到 10 epoch 以上。

在 `quant` 训练两个阶段的时候，在原有数据的基础上添加新数据的时候，同样需要将 `quant` 阶段的 `h5` 同步修改。

1. 将采集的新数据训练生成新的 `new1_h5`。

2. 将原来的 `train_name` : `["old1.h5", "old2.h5"]` 添加上新的 `h5`，即，

`train_name` : `["old1.h5", "old2.h5", "new1_h5"]`

这样新的数据就会和原来的数据一起在 `quantfinetune` 阶段训练。

2.7 quant_inference

quant_inference 可以直观看到 quant_finetime 训练的结果。

步骤 1 输入 state 和 substate 如下设置。

```
state: "quant" # ["preprocess", "float", "quant"]
substate: "quant_inference"
```

步骤 2 quant_inference 使用的是 quant_finetime_stage2 二阶段量化训练出来的 pth 模型。

```
"best_quant_model": "/path_to/quant_finetime_stage2.pth"
```

步骤 3 同时在 data 字段填写 benchmark 的路径,用于推理,推理的数据与上次 float_infer 使用的推理数据一致。

```
infer_dir: "/path_to/benchmark/"
infer_result: "/results/aibnr"
```

步骤 4 infer 字段下的 select_iso 用于选择只推理当前 iso 下的 benchmark。

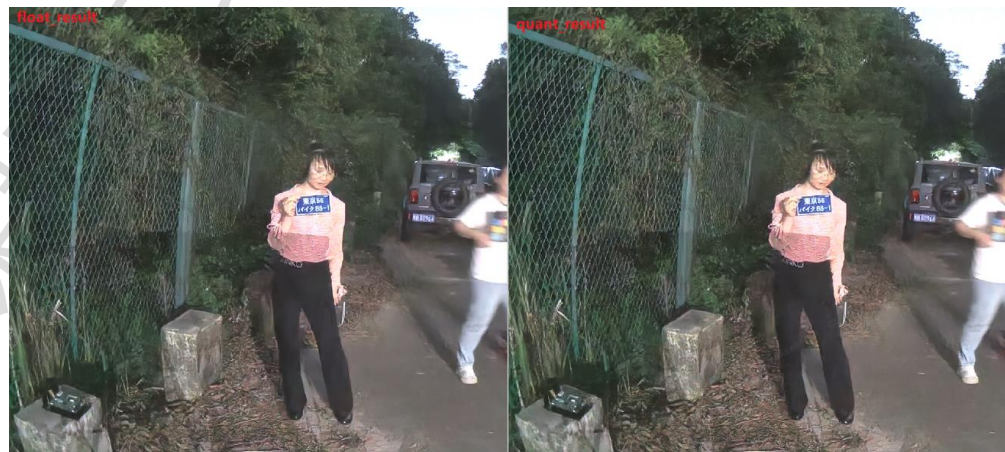
```
select_iso: False
select_range: [[6000, 20000]]
```

步骤 5 跑完 quant_inference 之后查看输出结果,在 result 文件对应的目录下,其中对应的 log 信息也在此目录下。

图2-37 quant infer 生成文件

```
AIHWserver@work1/tmp/AIHWtools_built20250210/results/aibnr/3_sc50la1_floatinfer_202502131010/20250213_1010_sc50la1_float_inference: cd ../3_sc50la1_float_inference/
ls -l
-rw-rw-r-- 1 ai_hwr 202502121952 rgb_100
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso11939_BGGF_to_RGGF_input_rgb_000_iso011939.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso11939_BGGF_to_RGGF_nrr_out_000_iso011939.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso11939_BGGF_to_RGGF_output_rgb_000_iso011939.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso11939_BGGF_to_RGGF_nrr_out_000_iso011939.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso15887_BGGF_to_RGGF_input_rgb_000_iso015887.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso15887_BGGF_to_RGGF_nrr_out_000_iso015887.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso15887_BGGF_to_RGGF_output_rgb_000_iso015887.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso15887_BGGF_to_RGGF_nrr_out_000_iso015887.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso1648_BGGF_to_RGGF_input_rgb_000_iso01648.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso1648_BGGF_to_RGGF_nrr_out_000_iso01648.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso1648_BGGF_to_RGGF_output_rgb_000_iso01648.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso1648_BGGF_to_RGGF_nrr_out_000_iso01648.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso24207_BGGF_to_RGGF_input_rgb_000_iso024207.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso24207_BGGF_to_RGGF_nrr_out_000_iso024207.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso24207_BGGF_to_RGGF_output_rgb_000_iso024207.png
RAM_2880x1620_12bits_BGGF_Linear_sc500a1_benchmark_indoor_iso24207_BGGF_to_RGGF_nrr_out_000_iso024207.png
```

图2-38 quant infer 效果展示



2.8 quant_deploy

步骤 1 在 RAW/YUV 分析工具的主界面上，点击左边的“Local File”单选框。输入 state 和 substate：

```
state: "quant"          # ["preprocess", "float", "quant"]
substate: "quant_deploy"
```

步骤 2 其中使用的是 quant_finetune_stage2 二阶段量化训练出来的 pth 模型。

```
best_quant_model : "/path_to/quant_finetune_stage2.pth"
```

步骤 3 padding_h, padding_w, input_shape 三个参数需要随分辨率改变,padding_w 固定为 32, padding_h 计算公式：

$$\text{padding_h} = \frac{\left(\left\lfloor \frac{h}{(\text{row_num} * 32)} \right\rfloor + ps_h \right) \times 16 - \frac{h}{(\text{row_num} * 2)}}{2}$$

参数如上设置，one step deploy，打开一次性跑完三个阶段的 deploy

```
one_step_deploy : True
```

步骤 4 设置 deploy_iso，在部署的时候同时推理此 iso 下的图像。

```
deploy_iso      : [12000, 13000]
deploy_path     : './deploy'
```

步骤 5 运行后，输出相关_tvm.encrypted 模型，对接下一步 tvn 编译使用，以及对应的权重 txt,log,对应 iso 的 infer 图像也在此文件夹下面。

其中对应的 bin 文件里面包含了图像信息，可以与下一步工具链对齐效果使用

针对 7206 版本的部署，需要额外做一个操作，即使用 toolchains/ai_toolchain 工具中的 aiisp_tool 指令：

```
./aiisp_tool 2688 1520 12 BGGR
```

其中“2688”为 sensor 图像的宽，“1520”为 sensor 图像的高，“12”为要部署的模型为 12bit，若部署 10bit 模型则需改成“10”，“BGGR”为 sensor 图像的 Bayer 格式，和训练时的设定保持一致，会生成一个 aiisp.bin，在下一步使用。

须知

quant deploy 最后阶段可以会报“The model has overflow!!!!”的报错，这个时候可以先定位是 qaunt1 阶段的溢出还是 quant2 阶段的溢出;使用 quant 1 阶段的结果去 deploy，如果没有溢出，则是 quant2 阶段训练导致的溢出。

发生溢出的时候，可以先使用前面几个 epoch 训练的 pth 查看是否溢出，如果溢出可以采用调小学习率，调大 batch size 的方式重新训练 quant 阶段，如果还是溢出的话，可以打开 overflow_protect，overflow_val 值越大，保护越强，但是精度会降低。

overflow_val 理解，假设原来是 $1e-8$ ； $F_{\max} - F_{\min} = 2e-6$ ，新设置的 $1e-5$ 。原来 $S = 1e-8$ ， $\text{effective_range} = \max(2e-6, 1e-8) = 2e-6$ ，原来刻度 $= (2e-6) / 255$ ，改大后 $S=1e-5$ ， $\text{effective_range} = \max(2e-6, 1e-5) = 1e-5$ ，现在刻度 $= (1e-5) / 255$ ，导致量化间隔 $(2e-6) / 255 \rightarrow (1e-5) / 255$ 变大，原本细微的数值差异（在 $2e-6$ 范围内）现在会被映射到更粗糙的刻度，导致量化噪声增大，精度下降。

3 相关参数说明

3.1 工具配置说明

工具使用是通过修改 config.yml 文件来实现，每个参数的使用说明如下：

参数名称	取值范围	描述
state	"preprocess" "float" "quant"	控制工具功能状态，其中“preprocess”表示对数据进行处理，“float”表示浮点模型的训练，推理等操作，“quant”表示量化模型的训练，推理，部署等操作。状态为“preprocess”时，substate 参数无效。
substate	"float_finetune" "float_inference" "quant_finetune" "quant_inference" "quant_deploy" "preprocess_sm"	"float_finetune": 浮点模型微调训练 "float_inference": 浮点模型推理配置浮点模型操作时，state 需设为“float” "quant_finetune": 量化模型训练 "quant_inference": 量化模型推理 "quant_deploy": 量化模型部署配置量化模型操作时，state 需设为“quant” "preprocess_sm": 小模型训练数据生成

3.2 quant 字段

控制模型量化过程中的一些参数配置，一般采用默认值不做修改。

参数名称	描述
a_qconfig	激活值的量化配置
w_qconfig	权重的量化配置
quantizer	量化器选择
observer	观察器选择
bit	量化位宽
symmetric	是否对称量化

per_channel	是否逐通道量化
backend	量化模型部署类型
overflow_protect	量化溢出保护开关（一般不建议打开，除非 deploy 部署报错的时候再打开）
overflow_val	量化溢出保护阈值

3.3 data 字段

所有训练数据以及推理数据相关的基础配置。

参数名称	描述	取值范围
bit	当前 sensor 的数据位宽	10/12/16
max_dgain	当前 sensor 的最大 isp dgain 值，该值可调试，一般不超过 32 倍，max_iso 较大时，可设置为 16，max_iso 较小时，可设置为 32	1-64
max_iso	当前 sensor Again 最大，所有 Dgain 为 1 时对应的 ISO 值	100-200000
bayer_pattern	当前 sensor 的 Bayer 格式，其中 0 代表 R，1 代表 G，2 代表 B，RGGB 表示为[0,1,1,2]，BGGR 表示为[2,1,1,0]，其他依此类推	[0,1,1,2], [2,1,1,0]等组合，目前支持 RGGB,BGGR,GRBG,GBRG 四种格式
sensor_id	当前 sensor 的标识，用于确定噪声数据	用户自定义字符串
fpn_dir	当前 sensor fpn 的路径，值为~ 时代表不做 fpn 操作，若指定 fpn 路径，则默认做 fpn，支持包含 bl 的 fpn 和不包含 bl 的 fpn	~: 表示空 或用户自定义的字符串
fpn_bit	fpn 文件的 bit 位宽，不包含 bl 的 fpn，该值需设成 8，否则 fpn_bit 应和 sensor 数据位宽保持一致	8/10/12/16
data_dir	用于训练采集的真实数据存放路径，这些数据会经过预处理以及数据增广过程而生成训练数据。	用户自定义的字符串
train_root	训练数据的存放路径	用户自定义的字符串
train_name	训练数据的名称，以.h5 为后缀	用户自定义字符串，以.h5 为

		后缀
valid_name	验证数据的名称，以.h5 为后缀， 训练数据和验证数据在同一目录下	用户自定义字符串，以.h5 为 后缀
dump_pre_root	数据预处理中间结果存放位置	用户自定义字符串
infer_dir	推理数据的存放路径	用户自定义字符串
infer_result	推理结果的存放路径	用户自定义字符串

3.4 preprocess 字段

数据预处理的基础配置。

参数名称	描述	取值范围
teacher	大模型的存放位置	用户自定义字符串
height	当前 sensor 的图像垂直方向像素 个数	2-10000
width	当前 sensor 的图像水平方向像素 个数	2-10000
pch_size	预处理分块的图像大小，一般为 默认值	128
overlap	预处理取块时 overlap 值，一般为 默认值	2
num_frames	一个场景的 raw 数据选取帧数， 训练数据 10 帧以上，验证数据 2 帧以上	2-10
awb_exists	是否存在 awb 和 ccm 参数	True/False
save_img_log	是否保存中间预处理图像	用户自定义字符串
save_img_freq	保存图像的频率	50/100/200/500
device	生成数据用的 GPU 编号	取决于卡的数量： 0-num-1

3.5 augmention 字段

数据增广的基础配置。

参数名称	描述	取值范围
diversity_on	是否对数据进行多样性增广，默认 打开	True/False

smooth_on	是否对 label 进行平滑操作	True/False
smooth_model	平滑模型地址，采用默认值	用户自定义字符串
smooth_ratio	平滑程度，0 为不做平滑，1 为强度最强	0~1.0
smooth_range	需要对 label 进行平滑操作的 iso 范围，可设定多个范围，smooth_ratio 也需添加对应平滑强度。	[0-9999999]
enhance_on	是否对 label 纹理进行增强	True/False
enhance_ratio	纹理增强的程度，值越大增强幅度越大，一般范围为 0~2	0~2.0
enhance_range	需要纹理增强的 iso 范围，可设定多个范围，enhance_ratio 也需添加对应增强程度	[0-9999999]
saturate_on	是否进行 label 饱和度调整	True/False
saturate_ratio	饱和度的调整程度，值越大，饱和度增强程度越大，值大于 1.0 时饱和度增强，值为 1.0 时饱和度不做变化，值小于 1.0 时饱和度衰弱，一般范围设定在 0.5~1.5	0.5~1.5
saturate_range	需要做饱和度调整的 iso 范围，可设定多个范围，saturate_ratio 也需添加对应饱和度调整程度	[0-9999999]
crop_size	数据增广的 crop 尺寸，一般采用默认值	64
data_size	数据增广输入数据的尺寸，值和 preprocess 字段下 pch_size 一致，一般采用默认值	128
Bpc_on	去坏点开关	True/False
Bpc_value	去坏点强度	0.98 #建议 0.9~0.99 之间，越小强度越高

3.6 log 字段

训练过程中打印 log 的基础配置。

参数名称	描述	取值范围
debug	是否对训练中间结果进行保存	True/False
log_file	运行过程 log 文件名称	用户自定义字符串

3.7 model 字段

模型的基础配置。

参数名称	描述	取值范围
save_root	模型保留路径	用户自定义字符串
save_freq	模型保存频率，控制多少次迭代后保存一次模型文件	500/1000/1500
net_name	选择模型对应的分辨率，可选项为 “2M_3g” “2M_5g” “4M_8g” “4M_12g” “8M_19g”，在选择分辨率时，可以根据就近原则选择，如 sensor 数据为 3M 则选择 “4M” 模型，数据为 5M，同样选择 “4M” 模型，数据为 6M，则选择 “8M” 模型。	“2M_3g” “2M_5g” “4M_8g” “4M_12g” “8M_19g”
best_float_model	预训练浮点模型地址，注意该模型文件需要适配设定的分辨率。量化时，浮点模型地址也是存放在这里。	用户自定义字符串
best_quant_model	预训练量化模型地址，该模型为量化二阶段训练的初始模型，量化部署时，量化模型地址也是存放在这里。	用户自定义字符串

3.8 train 字段

训练过程的基础配置。

参数名称	描述	取值范围
------	----	------

stage	取值为 1/2，代表浮点或量化模型训练的阶段	1/2
start_epoch	训练起始 epoch 数	0~500
total_epoch	训练终止 epoch 数	0~500
learning_rate	训练学习率	0.000001~0.01
batch_size	训练 batch_size	1~16
inp_channel	输入通道数，默认为 6，不做调整	6
out_channel	输出通道数，默认为 5，不做调整	5
valid_step	每隔多少次迭代跑一次验证集	0~1000
dump_freq	每隔多少次迭代 dump 一次中间结果	0~1000
device	训练用 GPU 编号	取决于卡的数量： 0-num-1

3.9 infer 字段

模型推理的基础配置。

参数名称	描述	取值范围
select_iso	是否设定 iso 范围进行推理	True/False
select_range	设定需要推理的 iso 范围	[0-9999999]
device	推理用 GPU 编号	取决于卡的数量： 0-num-1
infer_num	每个场景推理的帧数	1~frame_num (frame_num 为 raw 数据的帧数)
awb_exists	是否存在 awb 和 ccm 参数，若存在，则推理直出结果使用提供的 awb 和 ccm 参数进行色彩调整，否则使用默认参数进行色彩调整，建议使用真实 awb 和 ccm 参数	True/False
row_num	分块推理时，垂直方向分块个数（一般不做调整）	1
col_num	分块推理时，水平方向分块个数（一般不做调整）	4
sharp_on	推理输出 RGB 图时是否需要做锐化，默认做锐化	True/False

sharp_ratio	锐化的程度，一般设定为 1.0~2.0	1.0~2.0
Integration_bit	推理结果输出数据位宽，一般和 sensor 数据位宽一致	10/12/16

3.10 deploy 字段

模型部署的基础配置。

参数名称	描述	取值范围
one_step_deploy	量化模型部署开关，默认打开	True/False
padding_h	垂直方向 pad 数，如果 $(Height/2)\%16=0$, padding_h = 8 否则 padding_h = $((height/2/16 + 1)*16 - height/2) / 2$	1-8
padding_w	水平方向 pad 数，默认 32	32,7206_8M 部署时为 16
input_shape	[1,4,height,width], height 为图像高，width 为图像宽	[1,4,height,width]
deploy_iso	部署模型时的 iso 设定，iso 范围包含 infer_dir 中的任意一组数据 iso 即可	[0-99999999]
deploy_path	部署文件的存储路径	用户自定义字符串
layer_bit	量化 bit 位宽，12bit 或 10bit	10/12
deploy_channel	模型部署的输出通道数，一般不做修改	5
need_dump	部署过程中是否需要保存一组真实数据结果，一般不做修改	True/False
need_ab_fusion	是否需要做 ab_fusion，一般不做修改	True/False
dump_channel	保存输出时的通道数默认 5，一般不做修改	5/4
deploy_version	部署的版本，可选 "7206" 和 "7606"	"7206" "7606"

3.11 loss 字段

训练过程中 loss 的基础配置，float 和 quant 都包含两个阶段不同的 loss 配置。

stage1: 第一阶段

参数名称	描述	取值范围
float_finetune	["RGB_Loss", "RGB_Texture_Loss", "Motion_Loss", "RAW_Loss"] 分别指 RGB 域的图像 loss, RGB 域的纹理 loss, 运动 loss, raw 域 loss, 字段不可修改	["RGB_Loss", "RGB_Texture_Loss", "Motion_Loss", "RAW_Loss"]
float_weight	分别对应上述每个 loss 的权重	0~2.0
quant_finetune	["RAW_Loss", "RGB_Loss", "RGB_Texture_Loss", "Motion_Loss"]	["RGB_Loss", "RGB_Texture_Loss", "Motion_Loss", "RAW_Loss"]
quant_weight	分别对应上述每个 loss 的权重	0~2.0

Stage2: 第二阶段

参数名称	描述	取值范围
float_finetune	["RGB_Loss", "RGB_Texture_Loss", "Motion_Loss"]	["RGB_Loss", "RGB_Texture_Loss", "Motion_Loss", "RAW_Loss"]
float_weight	分别对应上述每个 loss 的权重	0~2.0
quant_finetune	["RAW_Loss", "RGB_Loss", "Motion_Loss"]	["RGB_Loss", "RGB_Texture_Loss", "Motion_Loss", "RAW_Loss"]
quant_weight	分别对应上述每个 loss 的权重	0~2.0
texture_ratio	用于提升量化第二阶段的纹理，默认为 0，值越大纹理越多（texture ratio 可以不怎么调，除非发现 量化 2 阶段解析力下降过多，再增加）	0~1.0

4 模型编译说明

当使用 AINR 训练工具完成模型训练，并进行 quant deploy 后，下一步就是对训练好的模型进行编译得到 xmm 文件后才可以在板端验证在线效果。模型编译和训练工具没有集成在一起，因此需要单独配置编译工具环境。

4.1 环境配置

4.1.1 安装 docker

硬件环境：在之前同样训练的硬件环境上即可

输入 `docker --version` 打印当前 docker 版本

-- docker 版本建议在 24.0.2 及以上

说明

对应各个模块的版本：

工具链版本 tag: `AI_TOOLCHAIN_V020_V3010_124_xx` (xx 是当前版本时间)

量化工具版本 tag: `MULTI_INPUT_MQB_FOR_AIBNR_V39_xx`

opensource_model_zoo 版本: XM7206 分支

4.1.2 创建容器环境

因开源测试样例包 `opensource_model_zoo` 太大，故做了分割处理，共有 4 个连续的分割文件：

`opensource_model_zoo.v3010.tar.gz.0`

`opensource_model_zoo.v3010.tar.gz.1`

`opensource_model_zoo.v3010.tar.gz.2`

`opensource_model_zoo.v3010.tar.gz.3`

使用如下命令将其合并：

```
$ cat opensource_model_zoo.v3010.tar.gz.* > opensource_model_zoo.v3010.tar.gz
```

同样，因工具链完整镜像包太大，也做了分割处理，共有 5 个连续的分割文件：

```
ubuntu18.04_xmtvm_AI_TOOLCHAIN_V020_V3010_124_xx.tar.0
```

```
ubuntu18.04_xmtvm_AI_TOOLCHAIN_V020_V3010_124_xx.tar.1
```

```
ubuntu18.04_xmtvm_AI_TOOLCHAIN_V020_V3010_124_xx.tar.2
```

```
ubuntu18.04_xmtvm_AI_TOOLCHAIN_V020_V3010_124_xx.tar.3
```

```
ubuntu18.04_xmtvm_AI_TOOLCHAIN_V020_V3010_124_xx.tar.4
```

使用如下命令将其合并：

```
$ cat ubuntu18.04_xmtvm_AI_TOOLCHAIN_V020_V3010_124_xx.tar.* >
ubuntu18.04_xmtvm_AI_TOOLCHAIN_V020_V3010_124_xx.tar
```

1. 加载工具链镜像：

```
$ docker load -i ubuntu18.04_xmtvm_AI_TOOLCHAIN_V020_V3010_124_
xx.tar
```

2. 创建及启动容器：

```
$ docker run -dit --name #### --net=host --shm-size 32G
ubuntu18.04_xmtvm:AI_TOOLCHAIN_V020_V3010_124_xx /bin/bash
```

-- 这里的 #### 指代容器名字，自行命名

3. 将测试样例包 opensource_model_zoo.v3010.tar.gz 拷贝进容器中：

```
$ docker cp opensource_model_zoo.v3010.tar.gz ####:/root/xxxx/
```

-- 这里的 #### 指代第 3 步创建的容器名字

4. 进入容器：

```
$ docker exec -it #### bash
```

-- 这里的 #### 指代第 3 步创建的容器名字

可输入 `docker ps -a` 查看当前所有容器的状态列表

5. 添加环境变量

```
Export XMTVM_MODEL_ALIGN_BYTES=16
```

4.1.3 测试容器环境

1. 解压缩测试样例包：

```
~/xxxx# tar xf opensource_model_zoo.v3010.tar.gz
```

2. 样例测试:

```
~/xxx# cd opensource_model_zoo/object_detection/yolov5
```

```
~/xxx# python build_coco.py
```

等待测试完成即可。

4.2 模型编译

4.2.1 编译前确认

在模型部署阶段需确认部署生成文件包含下述内容。

图4-1 7206 版本模型 deploy 编译提供文件

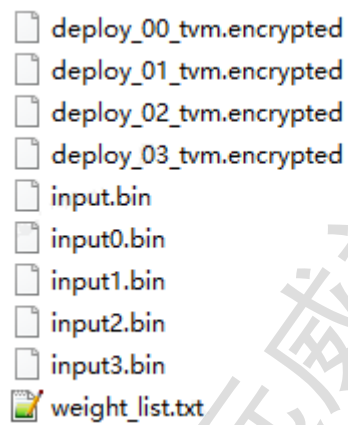
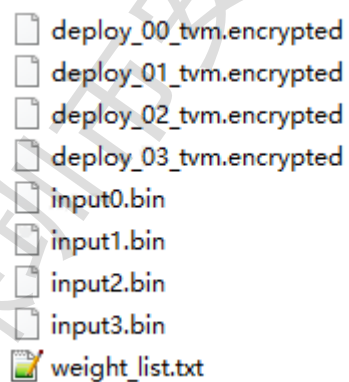


图4-2 7606 版本模型 deploy 编译提供文件



4.2.2 编译过程

编译时需要区分 7206 和 7606 版本，然后按照 4.2.1 章节的方法，将 deploy 生成的内容存放在一个文件夹内。

7606

步骤 1 对于 7606 的模型编译来说，需要确认文件夹中包含下图文件。

图4-3 7606 模型训练文件

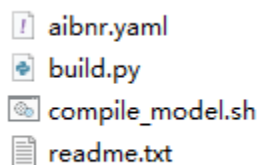
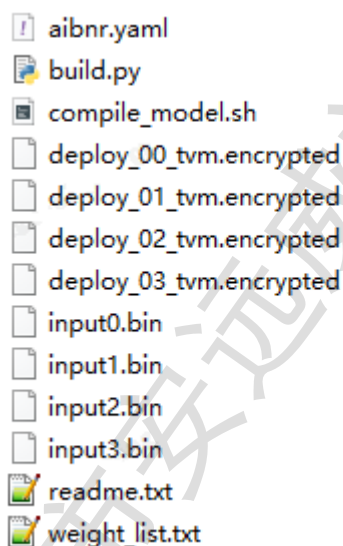


图4-4 7606 模型训练完整文件



步骤 2 通常在编译前需要修改 aibnr.yaml 中的 model_aibnr_value_list 字段，这部分内容在部署文件中的 weight_list.txt 中。将 weight_list.txt 中的 down1_sequential_weight 中的内容和 down1_conv_weight 中的内容合并成一列填入 model_aibnr_value_list 字段即可。

步骤 3 修改后执行 compile_model.sh 即可。

步骤 4 7606 模型编译结果为四个 zip 文件，选取 _00.zip, _01.zip 和 _03.zip 三个文件解压后得到 3 个 xmm 文件为编译后结果。

须知

7606 编译 8M19G 模型时需要额外添加:

group_partition_8M19G:True

```
compile:
  model_mode: high
  modify_io_dtype: uint12
  group_partition_8M19G: True
```

7206

步骤 1 对于 7206 的模型编译来说, 需要确认文件夹中包含下图文件

图4-5 7206 模型训练文件

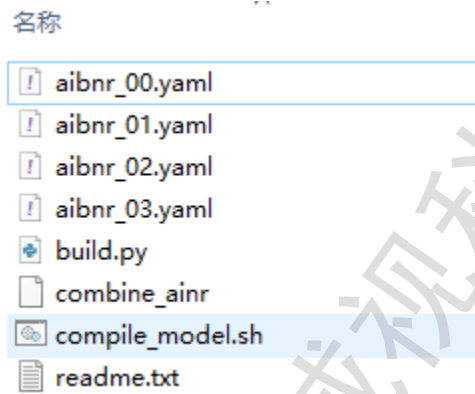
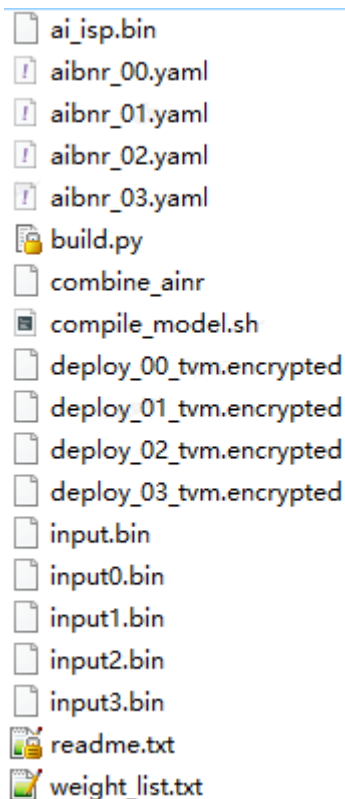


图4-6 7206 模型训练完整文件



其中 aiisp.bin 通过执行 ./aiisp_tool 来生成，仅 7206 需要使用 aiisp.bin 需要设定相关的分辨率 bit 位宽 bayer 格式等信息，如下图

图4-7 ai isp.bin 生成方法

```
Usage:
./aiisp_tool width height bit_width bayer_pattern
Option:
  bayer_pattern: RGGB GRBG GBRG BGGR
Example:
./aiisp_tool 1920 1080 12 RGGB
```

步骤 2 aibnr_0x.yaml 中的 model_aibnr_value_list 字段也是由 weight_list 生成，和 7606 是一致的。在编译 7206 前同样需要修改 aibnr_0x.yaml 中的 model_aibnr_value_list 字段，这部分内容在部署文件中的 weight_list.txt 中。将 weight_list.txt 中的 down1_sequential_weight 中的内容和 down1_conv_weight 中的内容合并成一列填入 model_aibnr_value_list 字段即可。

步骤 3 在编译前需要修改所有的 aibnr_00.yaml, aibnr_01.yaml, aibnr_02.yaml, aibnr_03.yaml 文件，修改规则为，若部署 12bit 模型，则将 modify_io_type 设为 uint12，且 model_aibnr_num_and_stride 的设定为：[x, width*1.5, width*1.5, width*1.5/2]，其中 x 对应 aibnr_0x.yaml 文件中的“x”，若 aibnr_00.yaml，则 x 为 0。例如 2688*1520 12bit 分辨率，则 aibnr_03yaml 设置为[3,4032,4032,2016]。

7206 也支持 8M 分辨率模型部署，例如 8M_19G_12bit,

model_aibnr_num_and_stride 的设定为: [x, 5760, 5760, 2880], 由于 7206npu 性能限制, 8M 分辨率下的帧率约为 10 帧。

步骤 4 若部署 10bit 模型, 则将 modify_io_type 设为 uint10, 且

model_aibnr_num_and_stride 的设定为: [x, width*1.25, width*1.25, width*1.25/2],

其中 x 对应 aibnr_0x.yaml 文件中的“ x” , 若 aibnr_00.yaml, 则 x 为 0。例如

2688*1520 10bit 分辨率, 则 aibnr_03yaml 设置为[3,3360,3360,1680], NPU 版本中的 005 指代芯片 7205, 020 指代芯片 7606, 3010 指代芯片 7206。

图4-8 7206 yaml

```
version:
  npu: "3010"

input_format: RAW
input_dtype: uint16

compile:
  model_mode: high
  modify_io_dtype: uint12

runner:
  model_aibnr_num_and_stride: [0, 2880, 2880, 1440]
  aibnr_model_input_data: ./input.bin
  model_aibnr_value_list:
  [
  ]
```

步骤 5 修改后执行 compile_model.sh 即可。

7206 模型编译结果为 1 个 neuron_network.xmm 文件。

7606 模型编译结果为 3 个 neuron_network.xmm 文件。

5 AINR 调优过程及 Q&A

5.1 固定参数检查

5.1.1 固定参数验证

驱动设定的 noise profile 是否和训练一致

驱动中 sep_thd 参数是否与算法训练时的 isp dgain max 一致

其中 sep_thd 取值在 16~24 比较合理，如果值太小，可能导致总的曝光量不够，如果过大，可能会造成训练结果模糊

5.1.2 proc 信息核对

AINR 模式下，pos_gain 和 gau_gain 设为 64 情况下，将 iso 调至 AINR 开启阈值以上，通过 cat /proc/umap/npu 查看 weight_a/b 是否和部署时的 weight_list.txt 一致，计算当前 iso 下 a,b 值是否符合预期，计算逻辑是 $k/255$ ， $b/255/255$ ，然后乘以 4095 (12bit 模型) 或乘 1023(10bit 模型)；以 sc485sl 为例,在 iso82354 场景下显示 iso-a-b[82354][247][2]

图5-1 cat /proc/umap/npu 信息

```
run cycle ----- 3764684
***job node info start***
user job-----c7dbd693|0|3
kernel job-----5bd598c9|3275607096|3
***job node info end***
***ainr info begin***
  hw_cycle|  hw_cost_tm|  max_hw_cost_tm|  isr_interval|  max_isr_interval
3764684|  7170|  7170|  7669|  7669
model id[0]---layer num[2]---iso-a-b[82354][247][2]
abfusion param group[0]:
weight_a [32] : -321 286 355 -133 -961 -192 -224 -130 -333 -204
32 758 274 87 -570 111 130 -285 124 558 319 -424 546
-538 -116 414 436 38 46 688 58 385 868 318 318
```

图5-2 7206 sc485sl b k curve

ka	kb	ba	bb	bc
0.000187246	8.293E-07	6.4E-09	0.000000003	-7.10802E-05

$$K=(ka*82354+kb)/255*4095=247$$

$$B=(ba*82354*82354+bb*82354+bc)/255/255*4096=2$$

计算 weight_list 是否符合预期

图5-3 weight proc 信息

```
/usr/pq_board_0714/pq_board # cat /proc/umap/npu
npu info :
Version[SPC020 ], [Npu V1.0 ], Release, Build Time[Jul 9 2025 10:12:32]
usage rate ----- 49%
total finish num ----- 0
cnt rpt ----- 0
run cycle ----- 4073439
***job node info start***
***job node info end***
***ainr info begin***
hw_cycle| hw_cost_tm| max_hw_cost_tm| isr_interval| max_isr_interval
4073439| 7758| 7758| 8293| 8293
model id[0]---layer num[2]---iso-a-b[204918][134][5]
abufusion param group[0]:
weight_a [32] : -188 470 444 -78 -906 -145 -93 -130 -251 -121 551 -203 296 3
80 722 354 101 -386 147 109 -340 136 620 412 -488 511 -226 -16 705
-325 -95 365
weight_b [32] : 231 265 81 135 296 197 819 -265 -310 516 124 2 516 -
950 54 127 -402 -138 -195 -1126 -915 -141 834 -143 -269 11 -174 -254 86
45 290 -239 -3014
init bias [32] : 4793 30504 6483 13420 -20365 17201 7474 49453 -2418 10117 -10870 -585 -3709 3166
040 11454 3978
3 4822 -18441 -1432
real bias [32] : -19244 61291 56930 -23920 -129125 -13714 -30898 -158769 -31534 -13576 73277 -66521 41601 5
2210 108472 50779 42028 -45931 32143 -11389 -32934 24993 136703 52075 -56620 57659 -31739 -7123 1265
63 -38503 -29721 46283
abufusion param group[1]:
weight_a [32] : -26 -115 -114 -114 -84 -104 47 -90 103 -103 108 127 -113 1
25 100 -96 0 -16 18 -3 -96 -82 -122 -128 122 -124 -93 -88 -76
40 -119 -59
weight_b [32] : -32 -128 -110 -128 -22 117 35 -68 -19 -70 85 44 -101 -
128 -34 -128 -67 105 -29 -12 6 -37 -58 -125 50 -128 44 36 -64
47 -58 -128
init bias [32] : -11990 -8007 3747 11150 13048 1745 20704 237 -7648 8589 -34533 -108637 1614 -
29412 -25931 -4981 -12127 -10964 -44257 -35873 -68762 -25204 153 -1955 -14485 -4849 8967 -11407 -601
58 4935 -9279 -28369
real bias [32] : -15634 -24057 -12079 -4766 1682 -11606 27177 -12163 6059 -5563 -19636 -91399 -14033 -
13302 -12701 -18485 -12462 -12583 -41990 -36335 -81596 -36377 -16485 -19732 2113 -22105 -3275 -23019 -706
62 10530 -25515 -36915
-----
***ainr info end***
```

weight_a 和 weight_b 与 deploy 阶段的 weight_list.txt 中的一致

其中: $\text{real bias} = \text{init bias} + \text{weight_a} * a + \text{weight_b} * b$

5.1.3 模型阶段回灌数据仿真

如果遇到无法对齐的问题, 可以采集实际场景的 raw, 使用 infer 推理功能推理出模型仿真效果, 步骤如下

抓取 10 帧以上 (注意 fpn 状态, 如果是打开 fpn, 在采集的时候也打开 fpn, 并同步提供 fpn.bin), ai model 是开启 ai 时候的通路, linear model 就是普通线性模式, 两种模式采集数据的时候, 外界环境需要完全一致。

1. 原始输入 (AINR off), 下图 A raw

2. NPU 输入 (AINR on), 下图 B raw

3. NPU 输出 (AINR on), 下图 C raw

仿真和伪量化结果对比

可以 1.B/C RAW 对比直观感受 npu 的处理结果; 2 使用 A raw 进行 quant infer 查看效果是否有异常; 3 使用 A raw 进行 quant deploy 得到 bin 文件, 用于 aitoolchain 进行编译, 查看 alttoolchain 的结果是否正确

图5-4 采集 check raw 流程图

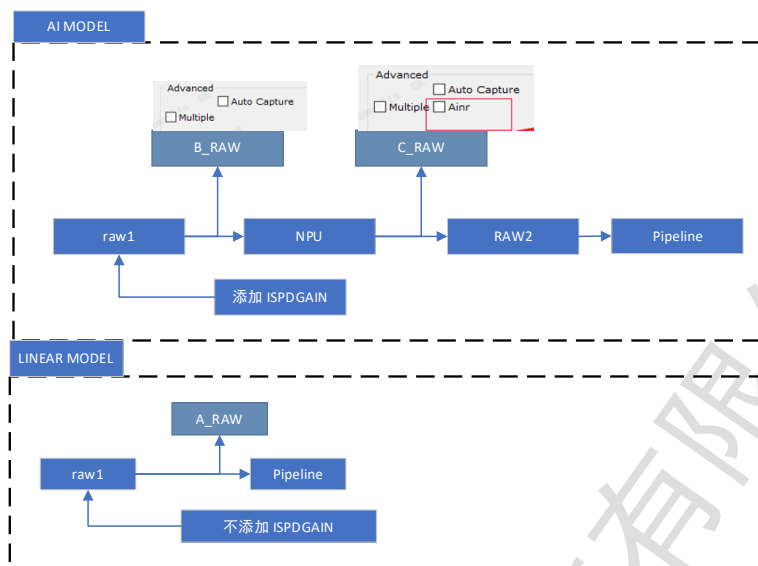


图5-5 npu 在 pipeline 中的位置

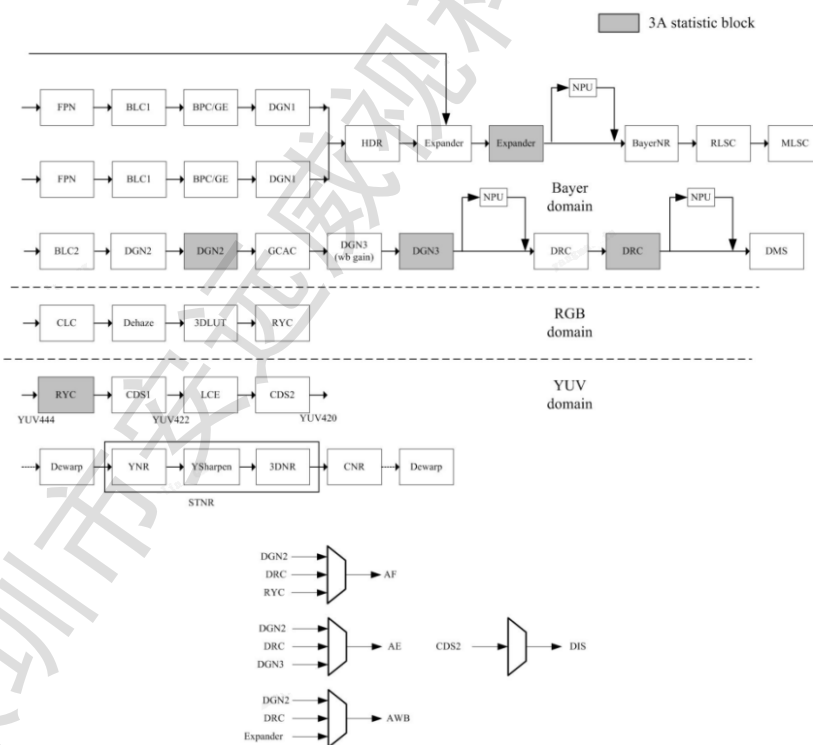
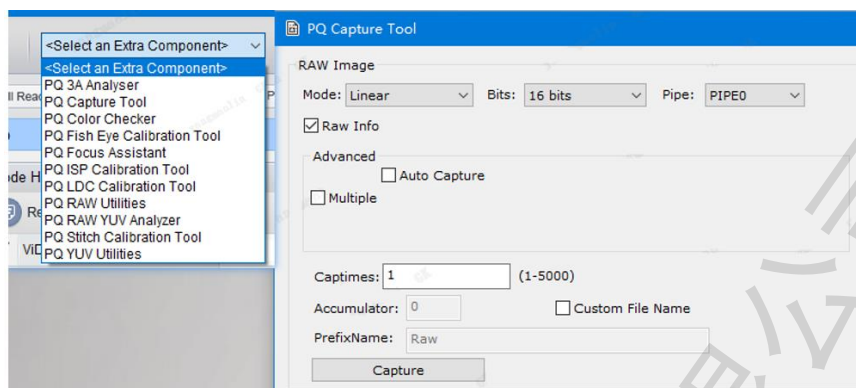


图5-6 常规采集界面



将 pqtool<MACRO ID="AINR_CAP_ENABLE" VALUE="0"/>修改为:

<MACRO ID="AINR_CAP_ENABLE" VALUE="1"/>, 打开 npu 采集 raw 的开关。

图5-7 打开 npu 采集开关

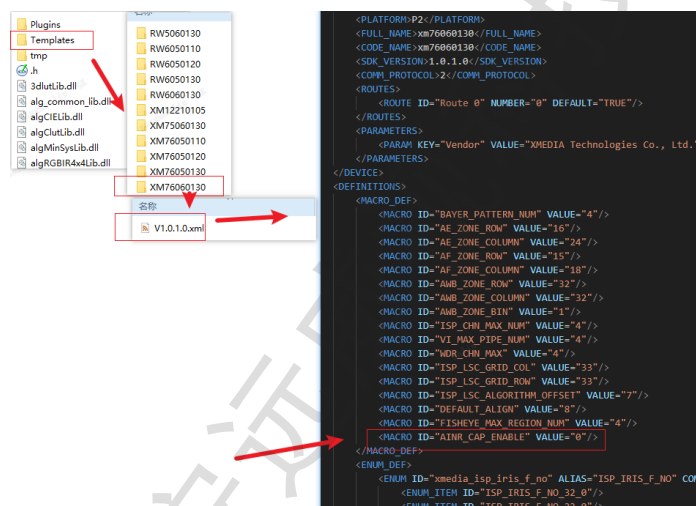
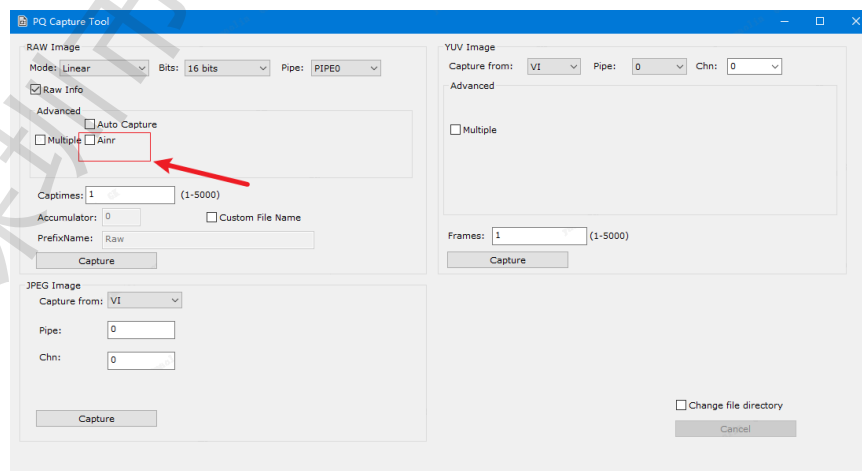


图5-8 显示 npu 采集界面



须知

模型训练的第五通道，输出的是灰度图，由两帧帧差得到，第五通道主要是边缘信息，一些时阈噪声在高频区域跳动，相减更容易得到，npu 输出的时候，只有四通道数据。

5.1.4 模型效果正确性

大模型结果在蒸馏数据或者 benchmark 上没有明显 artifact，偏色或运动虚影等异常现象,可以在 preprocess 阶段，只放置一组蒸馏数据，然后将 save_img_freq: 1，这样就能连续获得大模型推理的 dump 小图结果，方便对比；注意输入数据在 10 帧以上，同时注意 fpn 数据的准确性（黑帧在 100 帧以上），左边为 artifact 异常图像，右边是正常图像。

图5-9 Artifact



使用 float_inference 推理出 benchmark 的图片

- 1.若存在竞品，则和竞品做对比，要求模型输出在噪声，清晰度，色彩，artifact 等维度不差于竞品
- 2.若没有竞品，则和输入进行对比，要求模型输出在噪声，清晰度，色彩，artifact 等维度表现正常，也可以大模型的结果为目标；将同样的 raw 通过 float_inference 和 preprocess 进行处理的 dump 小图结果对比；一般来说，ISO 越高，小模型相比大模型差异主要在清晰度和弱纹理维度可能会出现下降。

量化阶段

- 1.以浮点模型结果为基准，量化模型输出和浮点模型输出没有肉眼可见的差距，不能出现明显解析力差异或颜色差异

2.量化模型不能出现溢出，可用部署过程中的溢出校验来判定，直接将跑出来的模型用 deploy 部署测试一下即可，如果正确生成 onnx 模型则代表模型没有溢出。

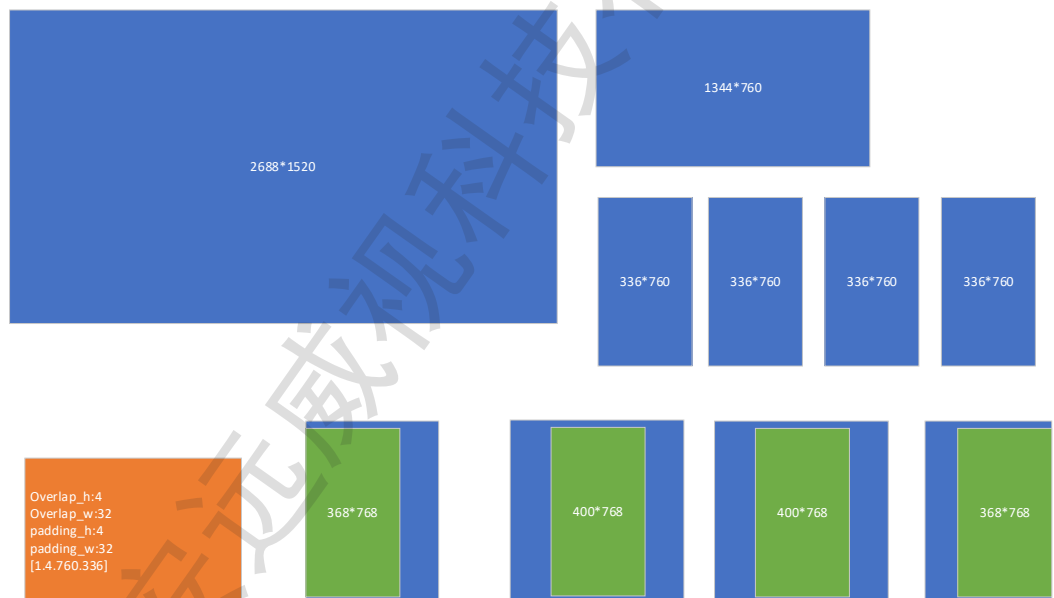
部署阶段

1.确认部署时设定参数是否正确：

- 1) 分辨率
- 2) padding 设定
- 3) bin 文件分析

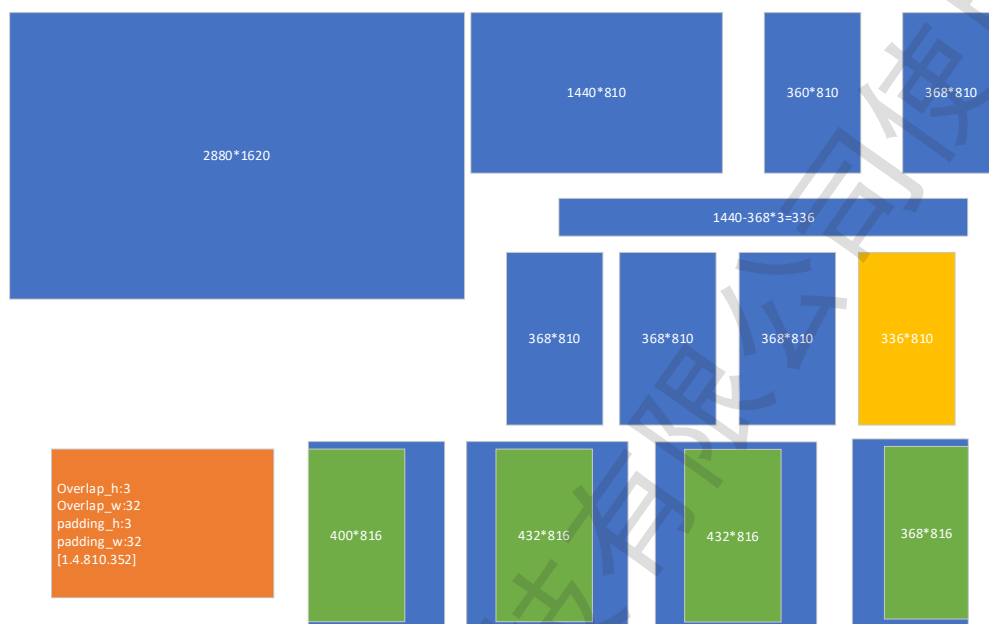
以 2688*1520 的 4M 场景为例，deploy 阶段产生的 bin 文件可以用于我们分析。

图5-10 2688*1520 的 4M 场景 deploy bin



一些 5M 场景可能出现最后一块 slice 的尺寸可能和前面几块不一样，下面为 2880*1620 场景

图5-11 5M 场景 deploy bin



4) bit 位宽

5) 平台 (7206/7606)

2.确认部署生成的文件完备性: onnx,weight_list.txt, input 和 slice 数据

3.确认 aiisp.bin 的正确性

编译阶段

1.确认编译设定参数的正确性

2.确认编译过程是否成功生成 xmm 文件, 7606 需要 3 个, 7206 需要 1 个, 是否生成对应的 bin 文件, 下图为 7206+mis40h1 在 depoly 阶段生成的 input_bin 和 output_bin,启动 output.bin 包含五通道数据, input; 比较时采用 deploy1 阶段的模型的相关模型数据在工具链比较, 具体参数设置

0).one_step_deploy = False

1).Need abfusion = true

2).Need dump = true

3).Deploy channel = 4

图5-12 7206_mis40h1_10_bit 编译生成的 Bin 文件



3.可以使用 sensorstudio 等软件解析上述 bin 文件，确认编译过程中得到的 tvn 仿真结果与伪量化结果没有肉眼可见的差距，可解析出结果用于对比

5.1.5 一些常见的命令

DDR 总带宽查看：xmddr 指令；

OS 系统内存：free 指令，或者 cat /proc/meminfo；

SDK 媒体内存：cat /proc/umap/media_mem；

cat /proc/umap/npu -- 里面可以查看 npu 的利用率。

5.2 B3D 和 AINR 的耦合过程

AI 生效和回叠的位置在工具中可以修改，如下图所示，注意 bnr 工具中 AInrPosSwitch 和 CoringPosSwitch 位置的选择，一般两个都是 post

图5-13 7206_mis40h1_10_bit deploy 生成的 Bin 文件

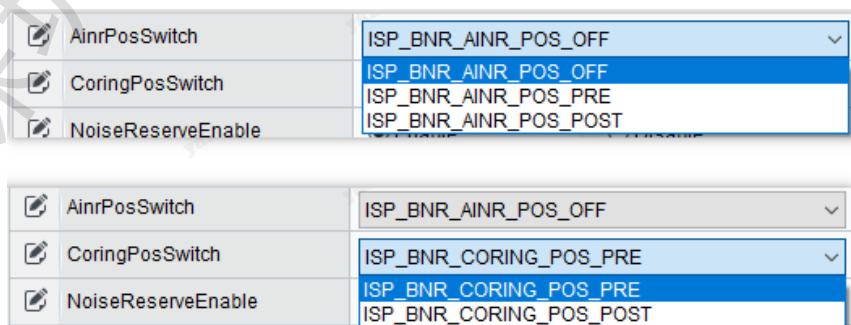


图5-14 AI 生效的位置的示意图

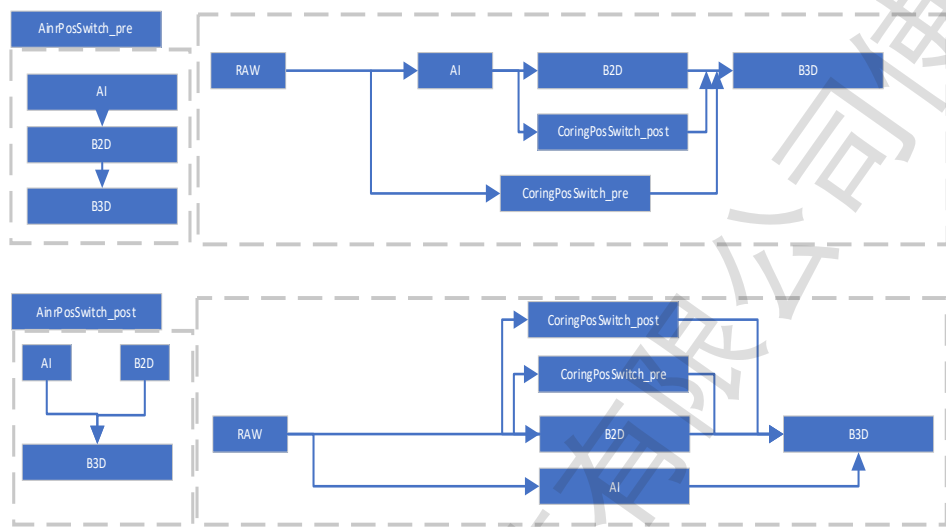
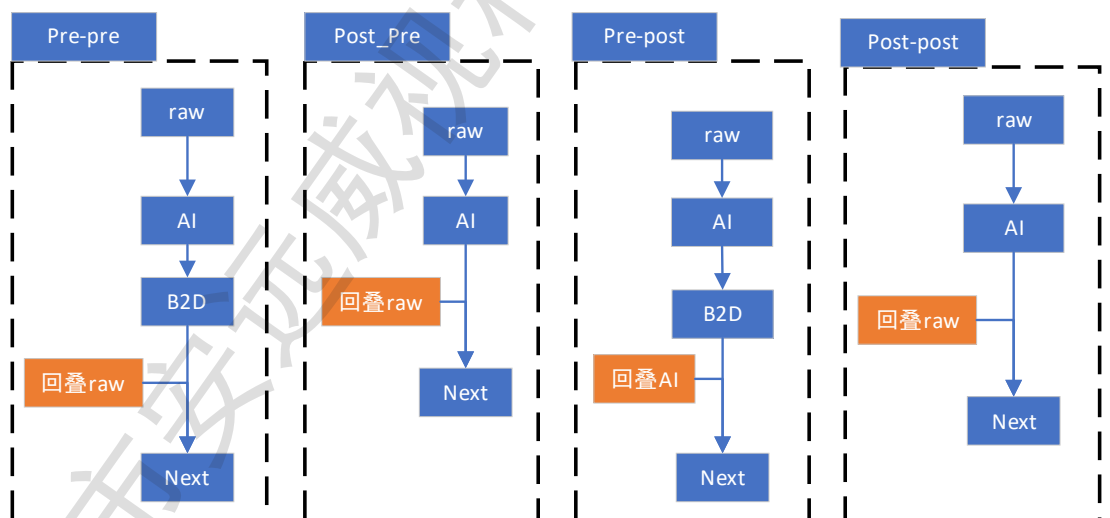


图5-15 AI 不同组合下回叠噪声生效的示意图



ai2d 和 B3D 强耦合：先 2d 再 3d 过程如下

在我们 bnr 模块有三个选择，具体要看使用场景，

传统模式 先 b2d ----> b3d; 传统 2d 的前后帧做差异；

pre:ai nr-->b2d ;

post :ai nr-->b3d 代替了传统 bayer2d，作为数据进来，主要参考的是 ai 的效果；post 的情况，输入是 ai 的结果作为前后帧去使用

B2D 的噪声回叠，是模块 NoiseReserveEnable，SnrCoringWgt 以及 NoiseReserveLut。它回叠的是 B2D 的输入

>当传统模式时，B2Dinput 是原始的 raw，即回叠的原始的未降噪的 raw，特点是会把噪声和细节都回叠上去。

>当 AI 开启，且位置为 pre，这个时候二选一选择器选 AI 的结果作为 B2Dinput，那么回叠的是 AI 的结果。如果 raw 先过一遍 AI 再过一遍 B2D，两次降噪会非常糊了。pre 模式一般用于 debug 查看 AI 的结果比较多，因为只有把 B2D 的降噪强度设为 1(最弱)，B3D 关掉，那么这个时候就只有 AI 的结果了。

>当 AI 开启，且位置为 post，这个时候 NoiseReserve 还是回叠的原始的 raw

B3D 回叠，回叠的是算法处理过后的细节，已经去过噪声了，可以根据 TnrLumaRatio 表，全设 256 是默认回叠，也就是关闭 TnrLumaCompEnable 时默认全回叠细节，开启时可以根据亮度选择性回叠。表全 0 时也就是不回叠细节，接近 AI 结果(实际是 AI 经过时域降噪，但画面风格还是 AI 比较糊的状态)

5.3 AINR 调优过程

AINR 调优过程主要分成如下几个阶段；sensor 噪声模型标定、AI 模型训练、基础传统通路检查和调试、AI 通路调试。

5.3.1 Sensor 噪声模型标定

噪声模型标定准确与否为整体通路效果好坏息息相关，sensor 完成了正确适配后即可点亮采集数据进行噪声标定；详细步骤参考工具调试指南：

噪声模型标定注意事项如下：

- a.确认 sensor 去坏点模式关闭、sensor 端开启 dpc 功能可能影响噪声模型和图像清晰度表现。
- b.确认 ae 的 sensorDgain 为 1 倍，AI 模式下 SensorDgain 默认不开启。

c. 标定后的噪声曲线 K/B curve 需检验是否符合预期，若工具检查有问题则需重新确认。

5.3.2 AI 模型训练

其中需注意事情如下：

a. 模型训练中 blc 的设置值和 sensor 驱动中 default 值，必须与 againmax、dgain 为 1x 的时候标定的 blc 值保持一致。

图5-16 sensor.c 驱动参数

```
static xmedia_s32 os04a10_get_isp_black_level(xmedia_u32 dev, xmedia_sensor_black_level *black_level)
{
    ... sensor_context *os04a10_ctx = XMEDIA_NULL;

    ... SENSOR_CHECK_DEV_RETURN(dev);
    ... SENSOR_CHECK_PTR_RETURN(black_level);
    ... OS04A10_GET_CONTEXT(dev, os04a10_ctx);
    ... SENSOR_CHECK_PTR_RETURN(os04a10_ctx);

    ... // Don't need to update black_level when iso change
    ... black_level->update = XMEDIA_FALSE;

    ... if (os04a10_ctx->wdr_mode == XMEDIA_VIDEO_WDR_MODE_NONE) {
    ...     black_level->black_level[XMEDIA_SENSOR_BAYER_CHN_R] = 0x104;
    ...     black_level->black_level[XMEDIA_SENSOR_BAYER_CHN_GR] = 0x104;
    ...     black_level->black_level[XMEDIA_SENSOR_BAYER_CHN_GB] = 0x104;
    ...     black_level->black_level[XMEDIA_SENSOR_BAYER_CHN_B] = 0x104;
    ... } else {
    ...     black_level->black_level[XMEDIA_SENSOR_BAYER_CHN_R] = 0x104;
    ...     black_level->black_level[XMEDIA_SENSOR_BAYER_CHN_GR] = 0x104;
    ...     black_level->black_level[XMEDIA_SENSOR_BAYER_CHN_GB] = 0x104;
    ...     black_level->black_level[XMEDIA_SENSOR_BAYER_CHN_B] = 0x104;
    ... }

    ... return XMEDIA_SUCCESS;
} // end os04a10_get_isp_black_level
```

图5-17 sensor.ex.h 驱动参数

```
static const xmedia_isp_blc_attr g_os04a10_blc = {
    ... 0, // op_type;
    ... {{256, 256, 256, 256}}, // manual_attr;
    ... {
    ...     10, // iso_num;
    ...     {100, 200, 400, 800, 1600, 3200, 6400, 12800, 25600, 51200}, // iso_level[XMEDIA_ISP_ISO_MAX_COUNT];
    ...     {
    ...         {256, 256, 256, 256},
    ...         {256, 256, 256, 256},
    ...         {256, 256, 256, 256},
    ...         {256, 256, 256, 256},
    ...         {256, 256, 256, 256},
    ...         {256, 256, 256, 256},
    ...         {256, 256, 256, 256},
    ...         {256, 256, 256, 256},
    ...         {256, 256, 256, 256},
    ...         {256, 256, 256, 256},
    ...         {256, 256, 256, 256},
    ...     }, // black_level[XMEDIA_ISP_ISO_MAX_COUNT][XMEDIA_ISP_BAYER_PATTERN_NUM];
    ... } // auto_attr;
};
```

5.3.3 基础传统通路调试

噪声模型曲线标定完成后即可开始基础通路的调试，如果此通路效果大致符合预期，则在此 tuning 基础上加载 AI 模型细调即可。此时的通路为传统通路效果，重点验证基于已经标定的噪声模型的基础上传统降噪效果是否符合预期。验证方法如下：

- a. 选取一个自动 ae 模式下、again 跑满、总体增益较大的暗环境，时域降噪均 bypass，只调试 bayer snr 空域降噪强度。check 在多个强度下的空域降噪是否有明显变化。若噪声涂抹状效果跟随空域强度变化而明显变化，则认为通路总体符合预期。
- b. 在 a 的基础上，设置 1.5 倍左右的中间空域强度，打开 bayer3dnr 时域开关，如果 tnr md 在典型值下噪声能完成基本收敛，且运动保持较好，则此降噪通路大致符合预期。
- c. 以上验证均正常，即可调试其他如亮度、色彩、大致的降噪清晰度喜好调试为 AI 通路效果做准备了。

5.3.4 AI 通路调试

AI 模型完成训练后，即可加载此模型开始完成整体通路的 finetune。此时复用已经 tuning 过基础传统通路的 PQ 效果参数。参数有如下两点要注意：

- a. bnr 中有两个参数要切换到 ainr 模式：ainrposswitch & coringposswitch 均选择 post；
- b. ainr 中 ainr 开关要使能、同时设置 switchthdlow & switchthdhigh 的阈值， $high > low \geq againmax$ 下的 ISO。

5.4 Q & A:

1. AI 相对于传统有哪些提升？

答：AI 去噪相对于传统去噪更加干净，噪声更细腻；因此在同样暗环境下的效果相比传统模式呈现出的信噪比更高，则有更大的能力满足更高或者极高增益下图像看清的需求。亮度、色彩也能随之能在更高或者极高增益下保持和还原。

AI 在什么时候开启，是怎么决定的？

答：调试工具上开放了 AINR 模块接口，可以设置 ainr 开启和关闭的 ISO 阈值，建议 $thdhigh > thdlow \geq againmax$ 下的 ISO。

2. AI 模式下，某场景的无论时域强度设置多大，噪声均无法收敛。

答：此时可能是噪声参数标定结果不符合预期，请检查噪声标定曲线和标定流程是否正确。

AI 模式下，极高 ISO 图像容易出现渗透如何调试？

答：渗透问题通常为时域强度太强导致，建议先关闭 YUVTNR，观察此时的渗透和噪声收敛情况，判断渗透是否跟 BayerTNR 强度强相关。若此时整体噪声跳动不大、则可以微调降低 tnrmtd 的阈值大小等手段；如果关闭 YUVTNR 后有改善，则同步确认 YUVTNR 的 tnrmtd 阈值大小、Mdwinsize 是否合理。

3. AI 模式下，运动物体较模糊如何调试？

答：可以先判断是否 STNR 中 YNR 强度太强，如果关闭 YNR 后还是模糊，再调整 BNR 中运动回叠噪声强度。也可以同步调整 AINR 模块中的 K/B 增益值调试 AI 降噪强度。

4. x_g 的单位怎么换到 t 算力上？几个模型 4m, 8m, 2m 的几个小模型是模型结构类似的嘛？以 4m_8g 为例，npu 利用率和对应的帧率大概是多少呢？
motion_loss 如何去理解呢？

答： $x_g \times \text{帧率} \times 2 / \text{npu 利用率} / 1024 = x(T)$ ，模型结构类似，NPU 利用率不同的配置不太一样，没办法给出一个准确的值。4m_8g 这档理论最大帧率可以到 27 帧左右。7206 是 1T 算力，部署到 7206 的有 4M_8g, 2M_3g 这两个，8M 的都不支持。
motion_loss 信息为辅助运动检测的信息。

5. ai nr 的 sample，是一定要使用 fpn.bin 吗？

答：sample 默认写了去加载 fpn.bin，如果没有对应.bin 就会报错，但不影响 ainr 通路，只不过没有 fpn.bin 的效果

6. 常见问题以及调试手段(例如坏点、跳动、偏色、局部清晰度、伪纹理等)

答：坏点、跳动、伪纹理属于细节过强类问题，可通过开启 smooth_on 来缓解；偏色首先需要查看训练数据对是否偏色，若正常，需查看测试场景与训练场景差异，如果差异较大可补充测试场景训练数据；局部清晰度可通过调 loss 调整。坏点可以通过调节 bpc 强度来调节

7. 工具上 AINR 的两个增益(高斯/泊松)是如何作用的？

答：影响模型降噪强度，一般默认 64，调小后会让模型放出噪声。

仅限深圳市安远威视科技有限公司使用