LPCV2021 FPGA Track Winner Solution: When industrial model toolchain meets Xilinx FPGA

Jiahao Hu, Ruihao Gong

SenseTime Research, Model Toolchain Team

Team Members

Jiahao Hu
Detection, Optimization

Pu Li
Quantization, Compile

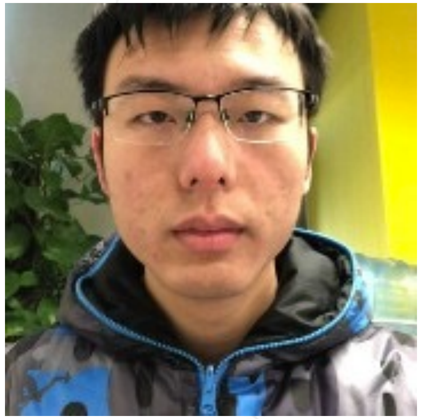
Yongqiang Yao
Detection, Optimization

Ruihao Gong
Quantization, Optimization

Shuo Wu
Detection

Yucheng Wang
FPGA board setup

Liang Liu
FPGA board setup



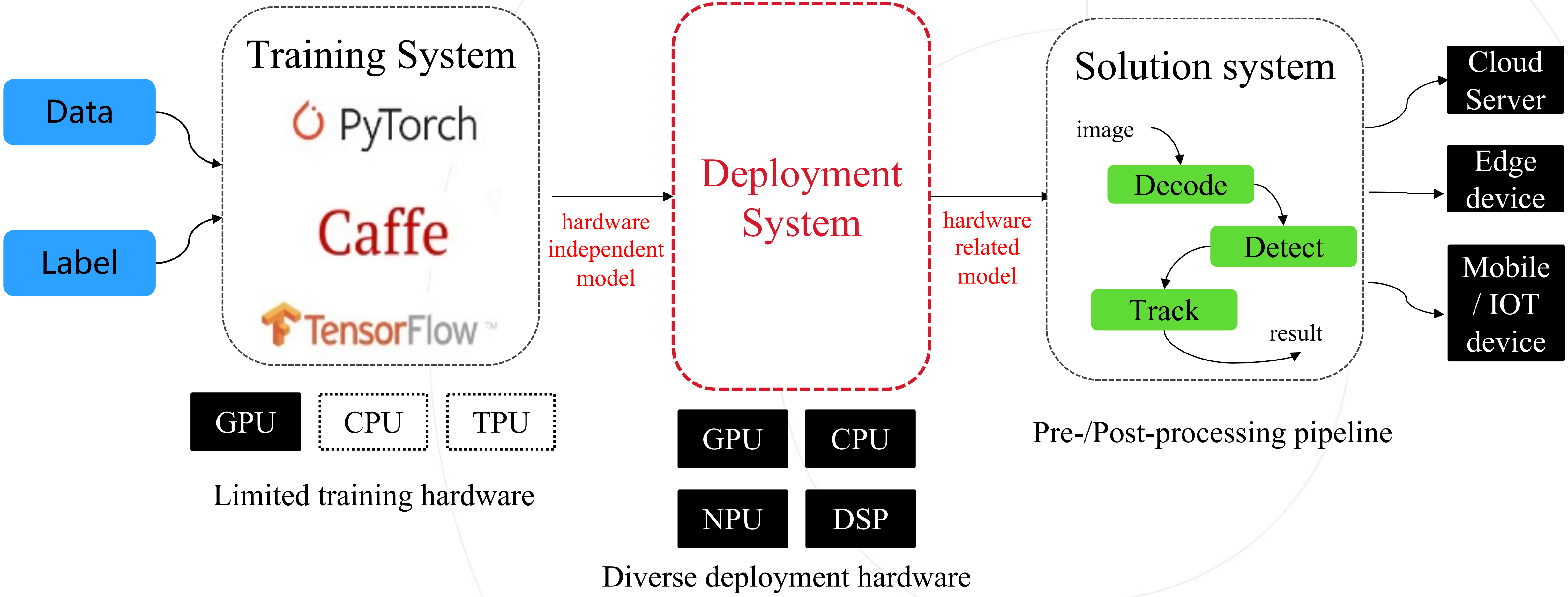
Yusong Wang
Inference Optimize

Fengwei Yu
Consultant

商汤
sensetime

Outline

A typical pipeline for model production



Overview

Our toolchain system for efficient model production

United Perception training framework | Model quantization toolkit | Multi-platform model deployment system

Training

- One for all: classification, detection, segmentation, etc.
- Bag of tricks: summarize the best practices for training
- Large-scale: large-scale dataset training support
- Deployable: hardware friendly

https://github.com/ModelTC/United-Perception

Quantization

- SOTA algorithms: LSQ, Brecq, Qdrop, etc.
- Deployable: quantized models can be directly exported to hardware format
- Flexible: automatic quantization node insertion

https://github.com/ModelTC/MQBench

Deployment

- Dozens of hardware platform support: NV GPU, DSPs, etc.
- Compiler and runtime: support mixed backend and multi-platform model conversion and inference
- Code level, operator level and network level integration

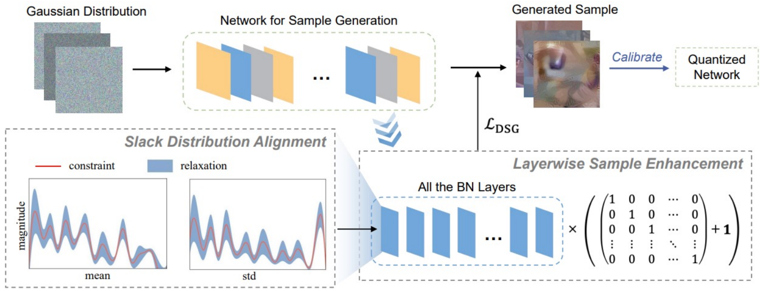
Not open-sourced yet.

The toolchain system is based on powerful algorithms and numerous engineering efforts.

Algorithm ability example

Comprehensive quantization algorithms：DFQ -> PTQ-> QAT

DSG CVPR2021 Oral



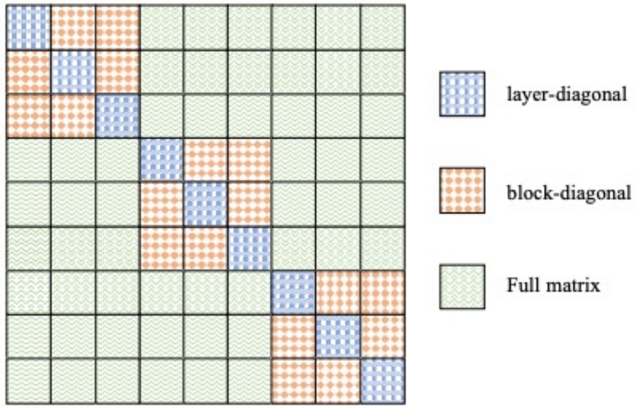
Data diversity greatly helps data free quantization

MixMix ICCV2021

Multi-model generation is good for the fidelity.

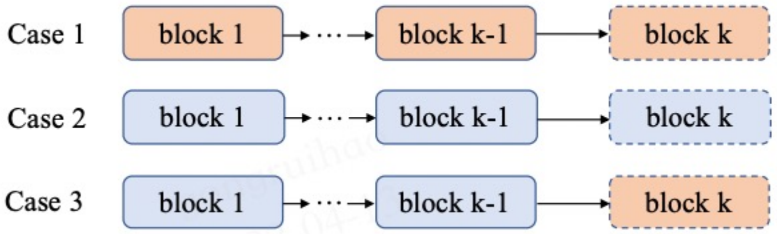
Calibration data is crucial

BRECQ ICLR2021



block reconstruction for PTQ, 4bit close to QAT

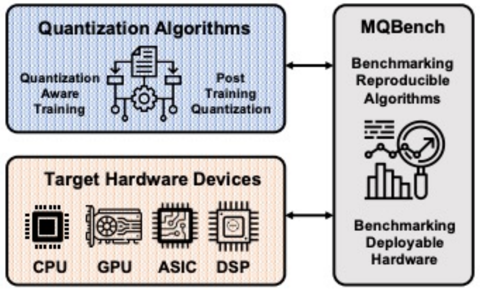
QDrop ICLR2022 Top1%



Randomly dropping for better PTQ，new SOTA

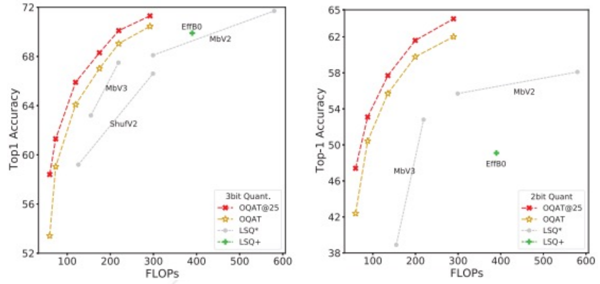
Continuously push the limit of SOTA for PTQ

MQBench NeuIPS2021 benchmark track



Reproducible and deployable, bridge the hardware and algorithms, followed by Intel and Qualcomm

OQAT ICCV2021



NAS: Quantization-friendly architectures

Solve the last mile problem

Objective of LPCV 2021 FPGA Track Challenge

Vision Task: Object detection.

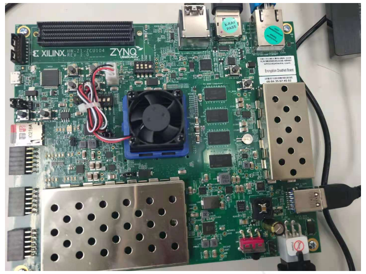
Data: Coco 2017 images data download link: https://cocodataset.org/#download

Hardware: Ultra96-V2 + Xilinx® Deep Learning Processor Unit (DPU)

Software: PYNQ

Evaluation: 10^4 / Energy * ReLU (mMAP – 0.2) * ReLU (fps – 5). Where mMAP is INT8 quantized accuracy

- Hardware：Ultra96-V2
- Inference lib：Vitis AI
- Quantization scheme：Ristretto
- Task：object detection

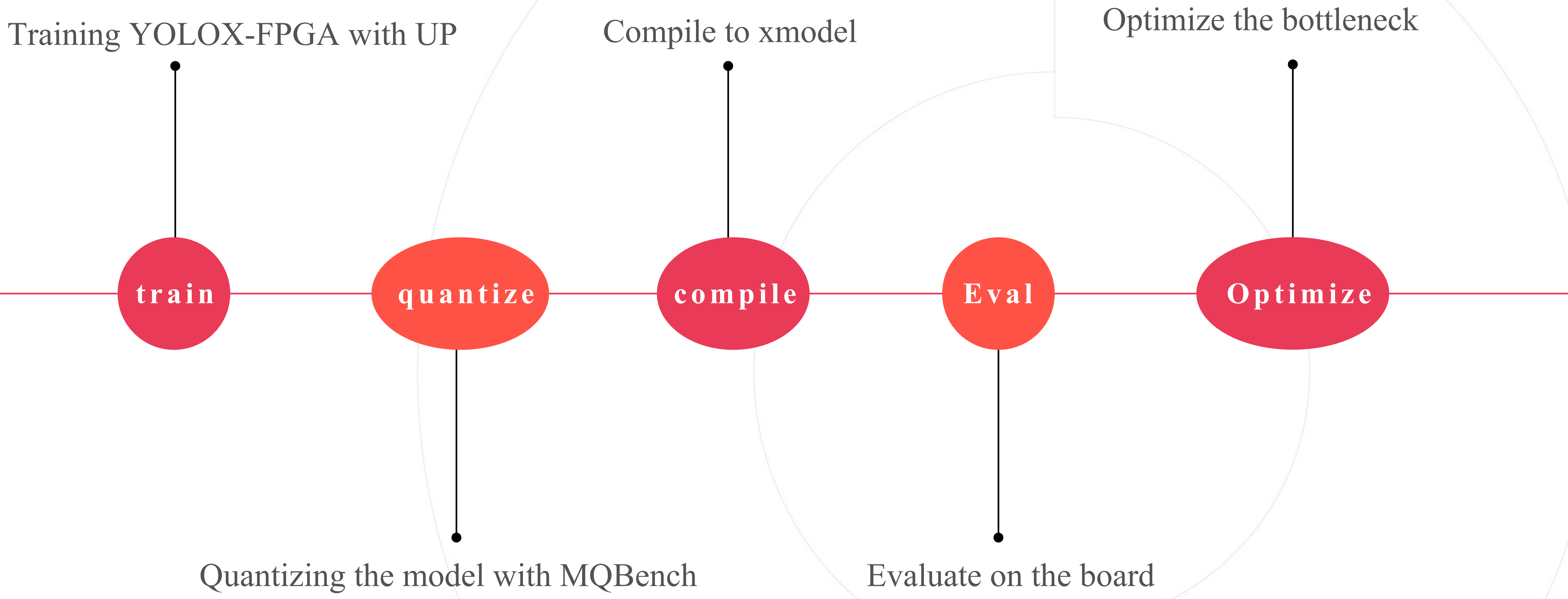


System Logic Unit (K)	504
Memory (Mb)	38
DSP slice	1,728
Video En/Decoder Unit (VCU)	1

A scene similar to that we often face in the industry production → Can be handled with our toolchain system.

LPCV FPGA Track Winner Solution

Overall pipeline



YOLOX-FPGA: Hardware-software-algorithm co-design

Existing YOLOX models can not directly deploy on the FPGA board => hardware friendly design

Width Selection

Best setting：width=0.375, depth=0.33

Model	Width	Image size	FP32	INT8	HW	Latency(ms)	Total Energy(J)	FPS	Score
YOLOX-FPGA	**0.375**	**416**	**32.1**	**30.2**	**30.0**	**38.906**	**635.876**	**25.702**	**34.510**
YOLOX-FPGA-nospp	0.3125	416	26.4	24.9	24.8	32.232	548.888	31.025	22.759
YOLOX-FPGA	0.3125	416	28.1	20.4	No		No		None
YOLOX-FPGA	0.25	416	22.5	20.4	20.3	27.575	No	36.265	None

OP adaptation

Considering the supported ops by Xilinx Ultra96 Vitis AI:

1. SiLU → ReLU；
2. SPP MaxPool2d (ceil_mode=False) → dilated convs；
3. Focus → three Conv2d (kernel_size=3)；

YOLOX-FPGA: Hardware-software-algorithm co-design

Experimenting different input shapes to achieve best accuracy-latency trade-off.

Input shape Best setting：gray input, input size = 320

Model	image size	FP32	INT8	HW	Latency(ms)	Total Energy(J)	FPS	Score
YOLOX-FPGA	288	26.9	24.2	24.6	21.817	378.108	45.837	49.681
YOLOX-FPGA	**320**	**29.05**	**27.2**	**27.4**	**24.557**	**416.239**	**40.723**	**63.509**
YOLOX-FPGA	352	31.1	29.3	29.3	28.835	480.148	34.658	57.444
YOLOX-FPGA	416	34.1	32.2	32.2	38.838	635.785	25.748	49.407

Improving accuracy with training techniques in UP

Pre-training

1. Select images in Object 365 with the same classes as MSCOCO according to Unidet class map and pretrained YOLOX-FPGA.
2. Finetune YOLOX-FPGA with 1/10 lr of pretraining.

Knowledge Distillation

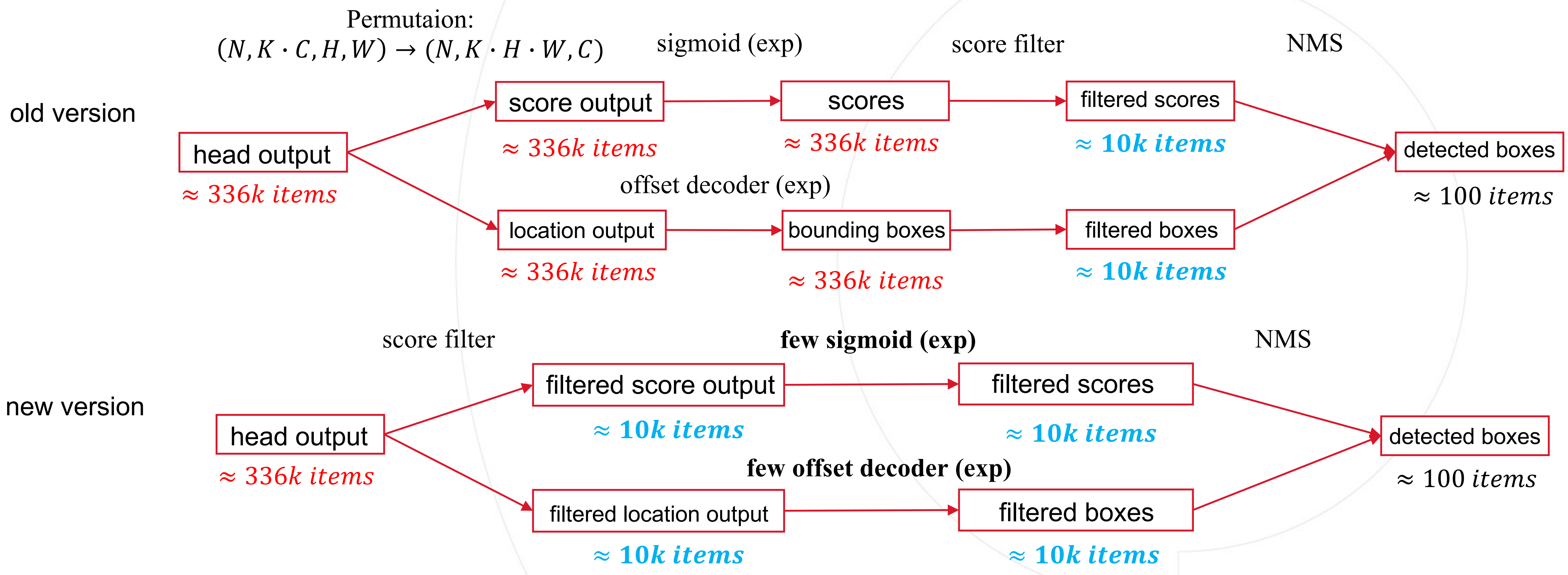
1. Train YOLOX-medium (4 times YOLOX-FPGA parameters) on MSCOCO.
2. Distill features of neck outputs with AT-loss.

Model	Pre-training	Knowledge Distillation	AP(%)
YOLOX-FPGA			25.69
YOLOX-FPGA	√		27.77(+2.08)
YOLOX-FPGA	√	√	29.04(+1.27)

Comparison of dfferent improvements AP on COCO 2017 val. Test size is 320

Post-processing optimization

- Efficient NMS implementation: using python => using cython
- Post-processing pipeline optimization



sensetime 商汤

Post-processing optimization

Improvement

	NMS	Few sigmoid	Few offset decoder	Time (ms/img)
Post-process				31.6968
Post-process	√			17.7001
Post-process	√	√		10.699
Post-process	√	√	√	8.5264

Comparison of different improvements cost time (COCO 2017 test-dev). The NMS is efficient cython NMS. Few sigmoid and decode are generating scores and bounding boxes after thresh filter.

sensetime 商汤

Model Quantization => No accuracy degradation with the powerful algorithm

- MQBench: Vitis AI Backend

We provide the example to deploy the quantized EOD model to Vitis, which is winner solution for the Low Power Computer Vision Challenge 2021 (LPCV2021).

- First quantize model in EOD.

```
python -m eod train -e --config configs/det/yolox/yolox_fpga_quant_vitis.yaml --nm 1 --ng 1 --launch pytorch 2>&1 | tee log_qat_mqb
```

- Second export the quantized model to ONNX [mqbench_qmodel.onnx] and [mqbench_qmodel_deploy_model.onnx].

```
python -m eod quant_deploy --config configs/det/yolox/yolox_fpga_quant_vitis.yaml --ckpt [model_save_path] --input_shape [input_sha
```

- Third build Docker from Dockerfile, convert ONNX to xmodel [mqbench_qmodel_deploy_model.onnx_int.xmodel].

```
python -m mq.dep.convert_xir -Q [mqbench_qmodel.onnx] -C [mqbench_qmodel_deploy_model.onnx] -N [model_name]
```

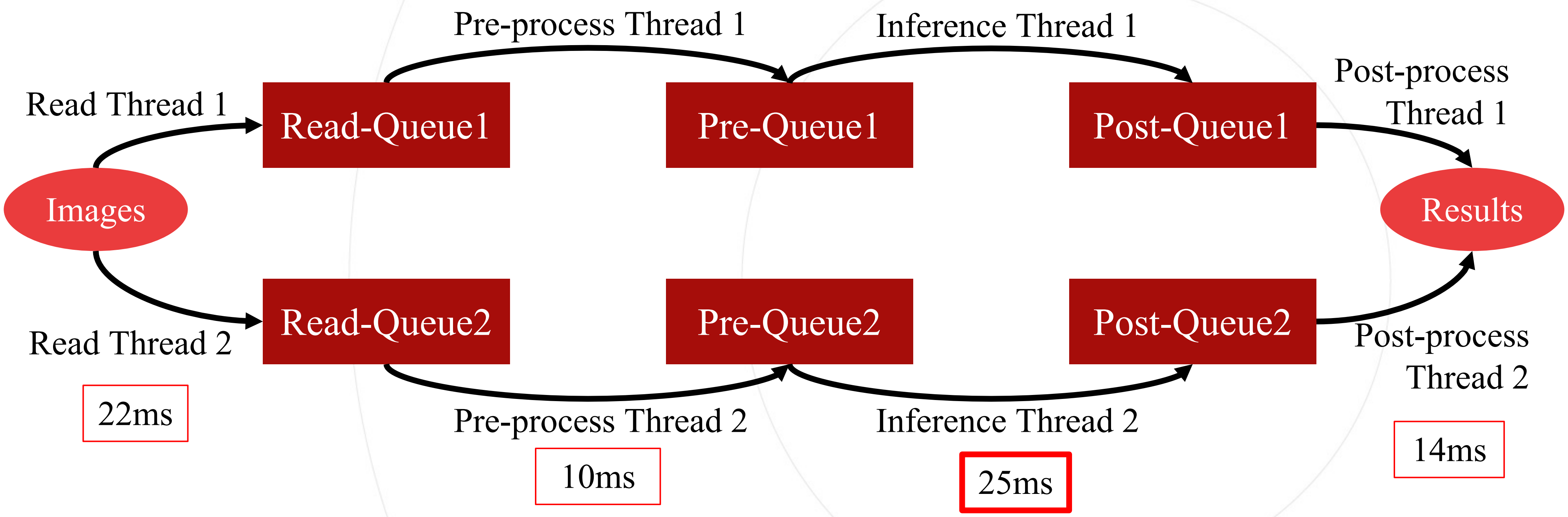
- Fourth compile xmodel to deployable model [mqbench_qmodel.xmodel].

```
vai_c_xir -x [mqbench_qmodel_deploy_model.onnx_int.xmodel] -a [new_arch.json] -o [output_path] -n [model_name]
```

Documentation：https://mqbench.readthedocs.io/en/latest/user_guide/deploy/vitis.html

Heterogeneous computing optimization

DPU+ARM ⇨ Image reading， pre-processing， inference and post-processing are parallelly executed by 2 threads, respectively. ⇨ The latency is dominated by inference.



LPCV FPGA Track Winner Solution

Result: winner with the lowest energy, highest accuracy, and smallest latency

Rank	Team	Date and Time	AP	Latency (ms)	Energy	Score
1st	spring	2021-08-31 23:57:54	0.274	26.625	200.283	120.296
2nd	spring	2021-08-30 23:58:07	0.262	27.064	201.394	98.357
3rd	MIT HANLAB	2021-08-31 09:26:22	0.241	34.849	251.814	38.58
4th	spring	2021-08-28 23:49:00	0.233	32.757	227.456	37.036
5th	MIT HANLAB	2021-08-30 07:26:21	0.237	34.305	257.28	34.731
6th	spring	2021-08-29 23:38:21	0.233	31.425	258.196	34.28
7th	spring	2021-08-28 00:00:07	0.254	42.057	311.0	32.603
8th	NYCity	2021-08-31 14:01:55	0.252	46.289	319.332	27.036

Summarization

All codes for reproducing our winner solution are open-sourced.

Welcome to star and have a try on our toolchain.

Model production procedure： https://mqbench.readthedocs.io/en/latest/user_guide/deploy/vitis.html

Open-sourced inference code： https://github.com/ModelTC/LPCV2021_Winner_Solution/

United Perception × Model Quantization Benchmark

link: https://github.com/ModelTC/United-Perception

link: https://github.com/ModelTC/MQBench

Thanks for Listening!

Q&A