

# Spring Project: Multi Backend Neural Network Auto Quantization and Deployment over ONNX



Fengwei Yu (SenseTime-China)  
yufengwei@sensetime.com

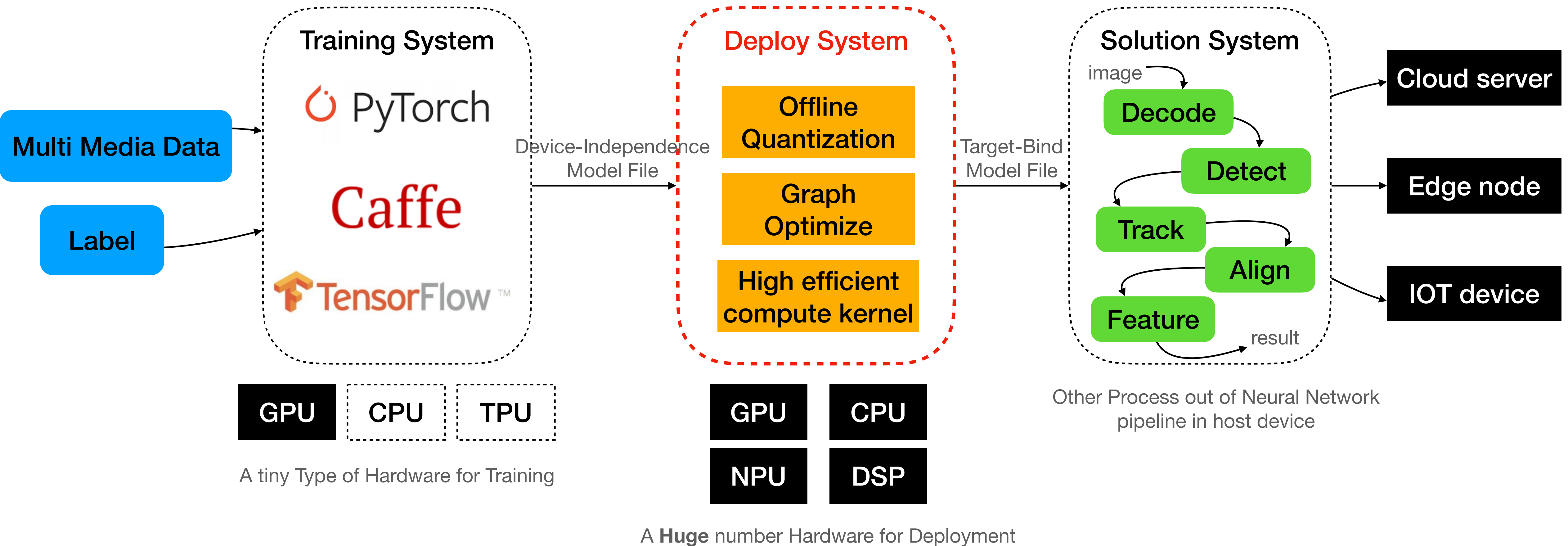
# Contents

- What is Spring Project.
- Neural Network Deployment.
  - Multi-device: NART
  - Efficiency: Quantization
  - Automatic: Adela system
- From Caffe to Onnx.

# What is Spring Project?



:Industrial Grade AI Model Production Framework



# Neural Network Deployment: Features and Challenges

: **From** Device-Independence Model File to Target-Bind Model File

★ 3 key features of a Neural Network Deployment Framework

## Multi-Device

support more and more device with an unified framework  
enable to run

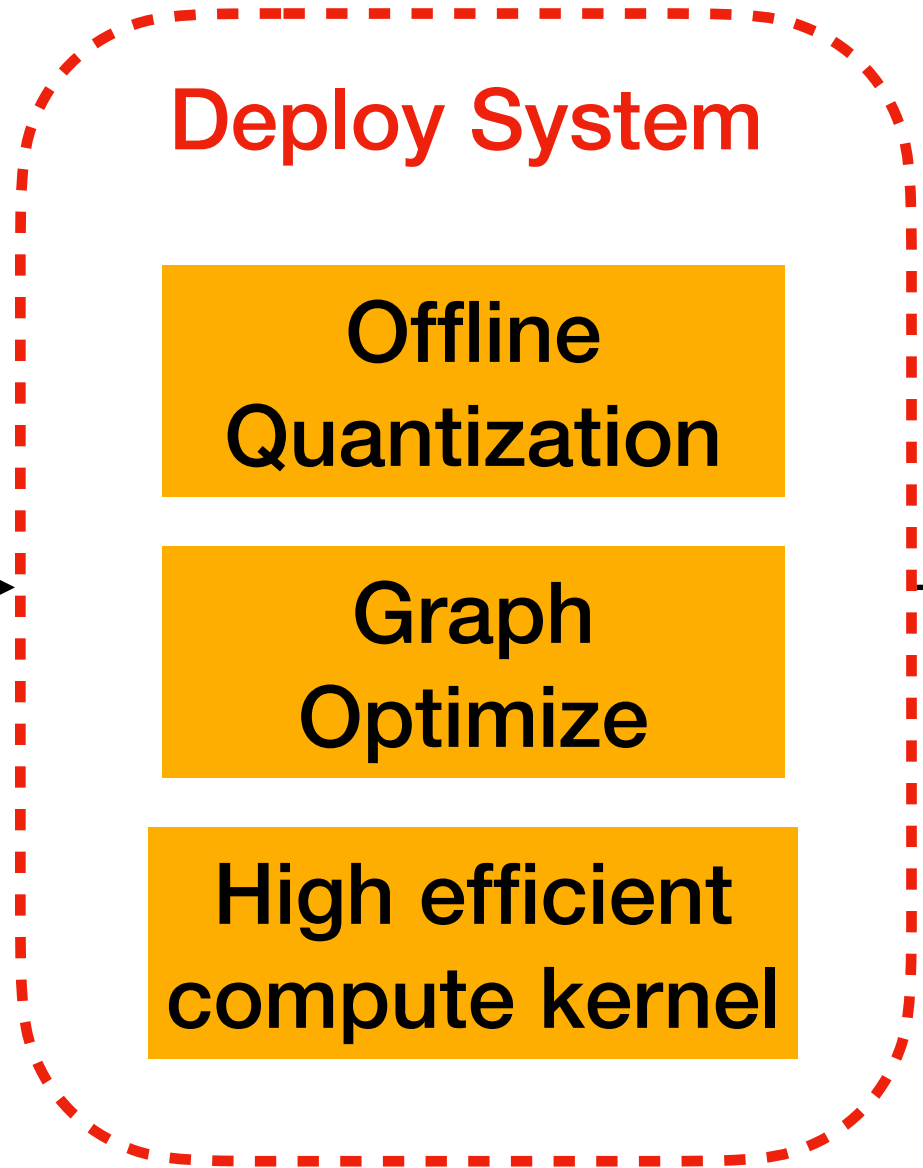
## Efficiency

high efficient on special target  
support model compress like quantization and sparsity

## Automatic

more “end2end” in one model  
we want “one-for-all”  
we don’t want “compile failed”

Device-Independence  
Model File



Target-Bind  
Model File

GPU

CPU

NPU

DSP

# Our Solutions: Multi-Device

**A little fact for codegen solution (like TVM) to support a new device:** Almost all nowadays NPUs provide only end2end compiler (of a net) instead of release their low level instruction for developer.

**We need a framework which support any type of integration:**

- **Code-level:** arm、x86、cuda、BangC、openCL
- **Op-level:** cuDNN、mkldnn、ARM-Compute Library
- **Net-level:** openvino、TensorRT、ncnn/mnn/tnn、davinci (huawei) 、nnie (hisicon) 、j1q、neuware (cambricon) 、sgstar、rknn (rockchip)

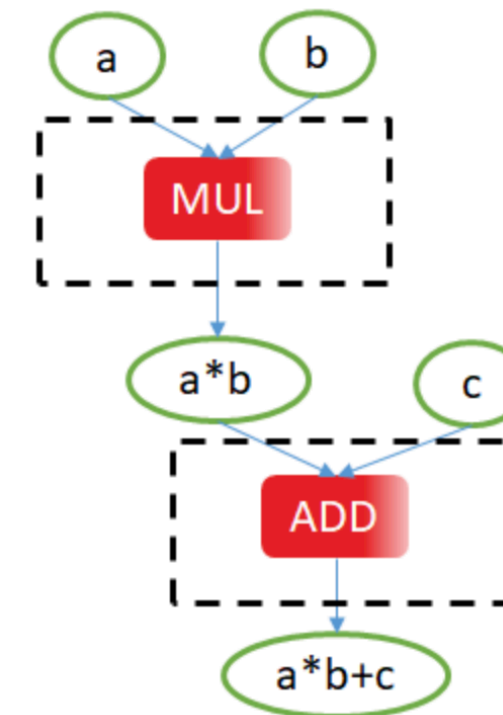
**We build a compiler system name spring.nart:**



# How we build NART

## Feature 1: multi-scope OP

view a compiled net as a type of OP  
serialize file as OP setting



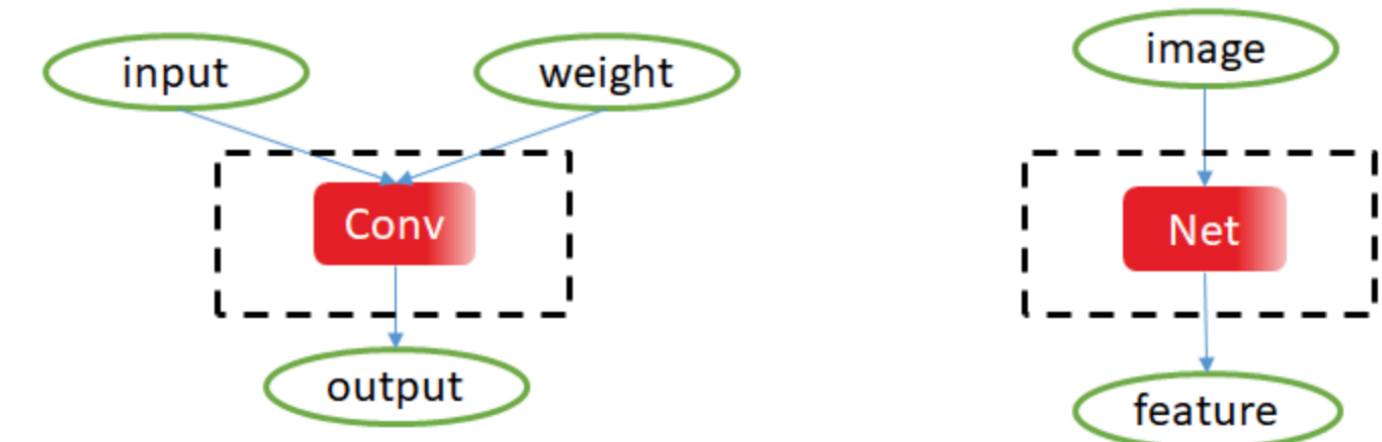
## Feature 2: compile / runtime separate

runtime (named nart-case):

only contain a large set of OP

compiler (named nart-switch):

deal with quantization/graph optimization/format transform/ bypass compile

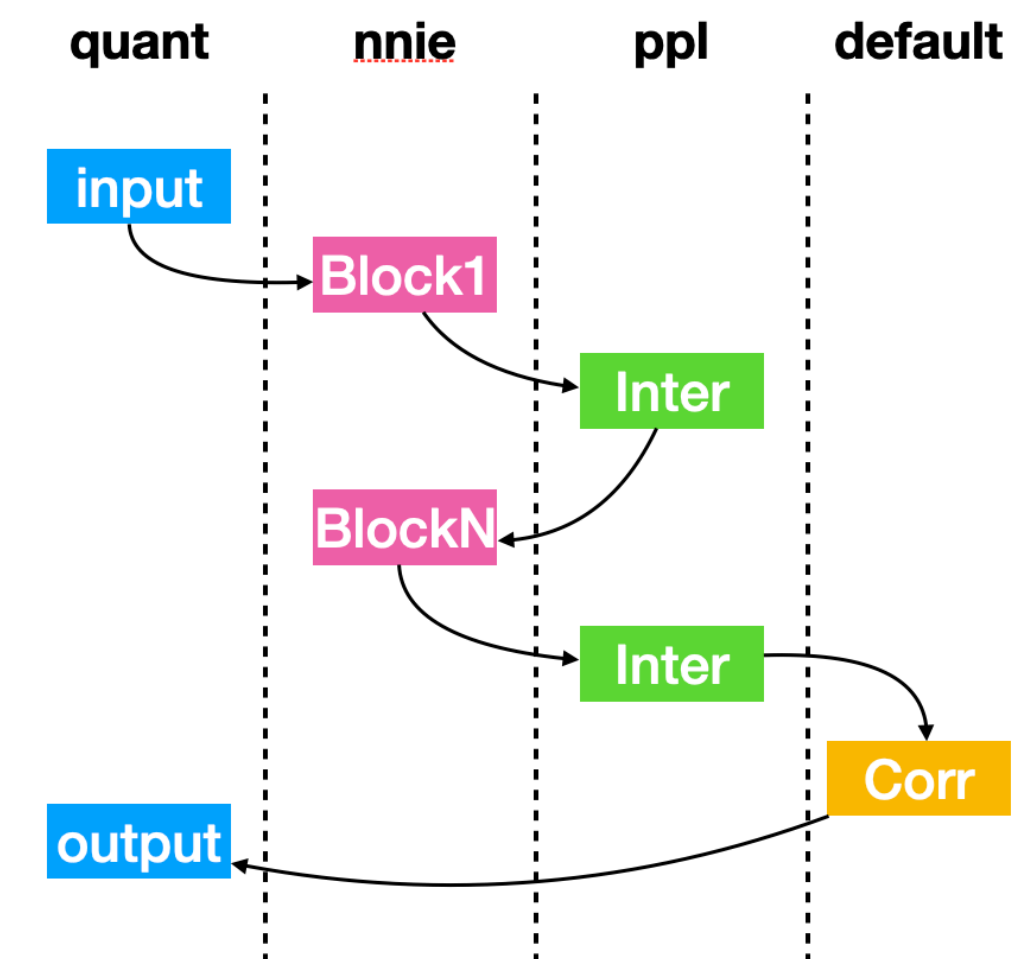
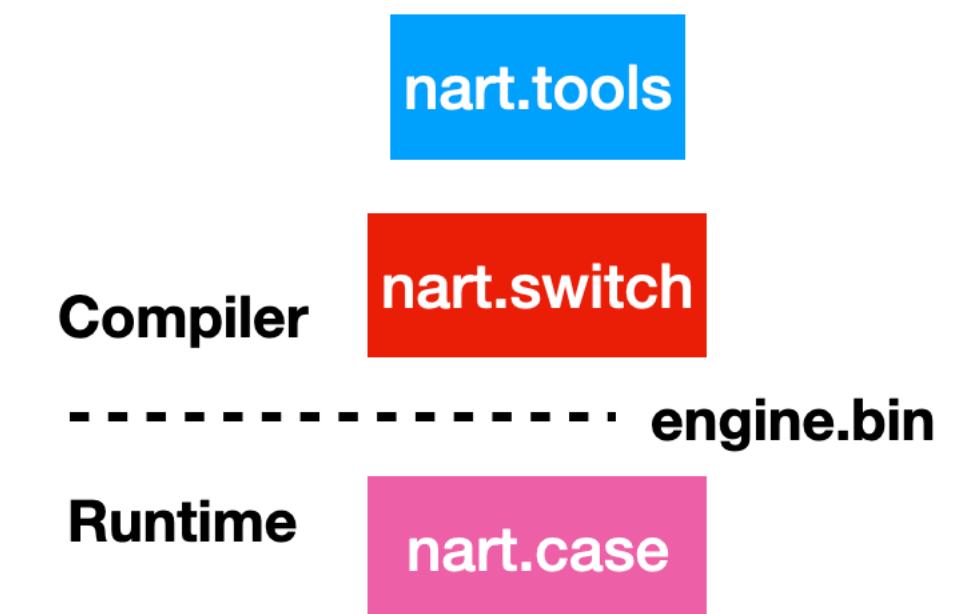


## Feature 3: mixed operation

Deal with fall back: op not support in NPU

Mix precision: auto insert quant and de-quant OP

with NART, we can now support more than **16** type of devices now (**with a single ONNX file input**), include: davinci、cambricon、nvidia、ambarella、vp6、nnie、jlq、movidius、sigmstar、ceva、opencl、rockchip、snpe、x86、arm and etc....



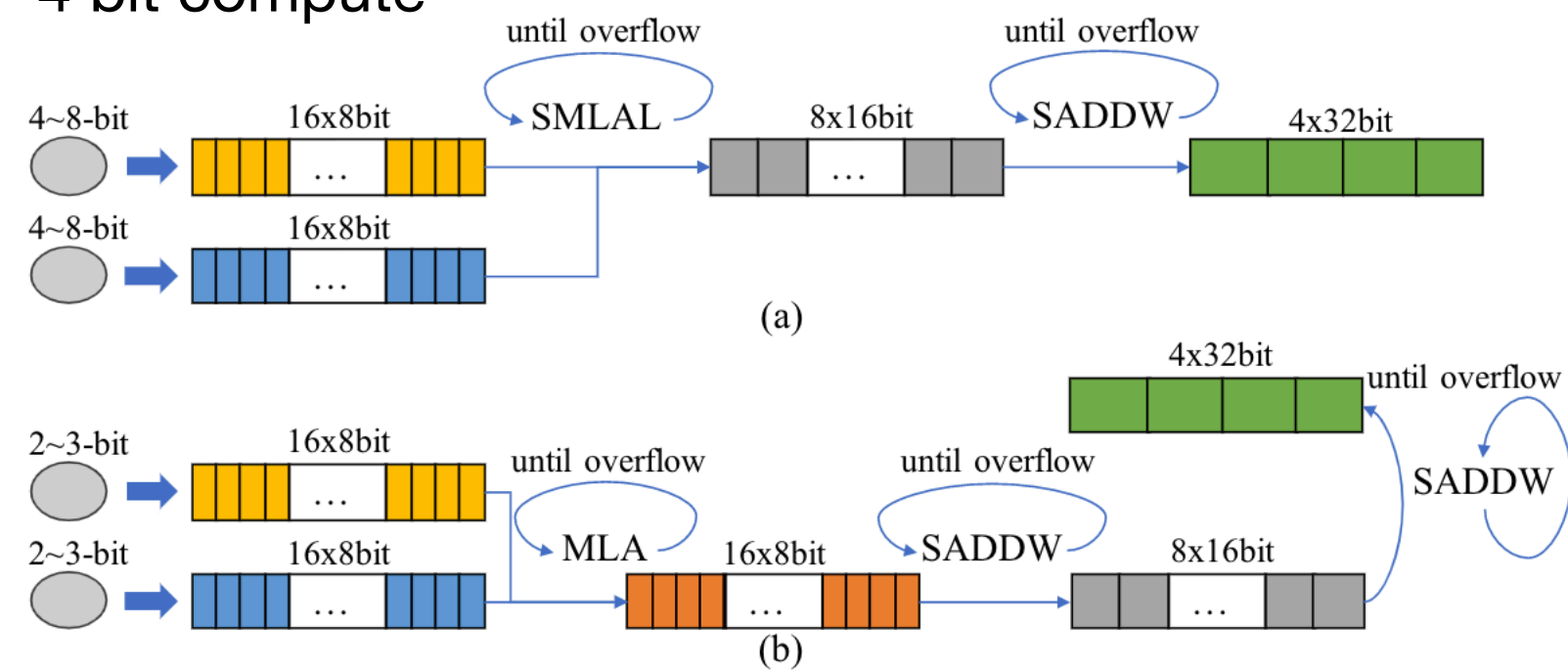
# Our Solutions: Efficiency

Programable hardware (GPU、CPU) : openvino、tensorRT doing very well on fp32/fp16/int8、 But leak support of low bit (< 8bit)

★first support extremely low-bit convolution on modern computer architectures

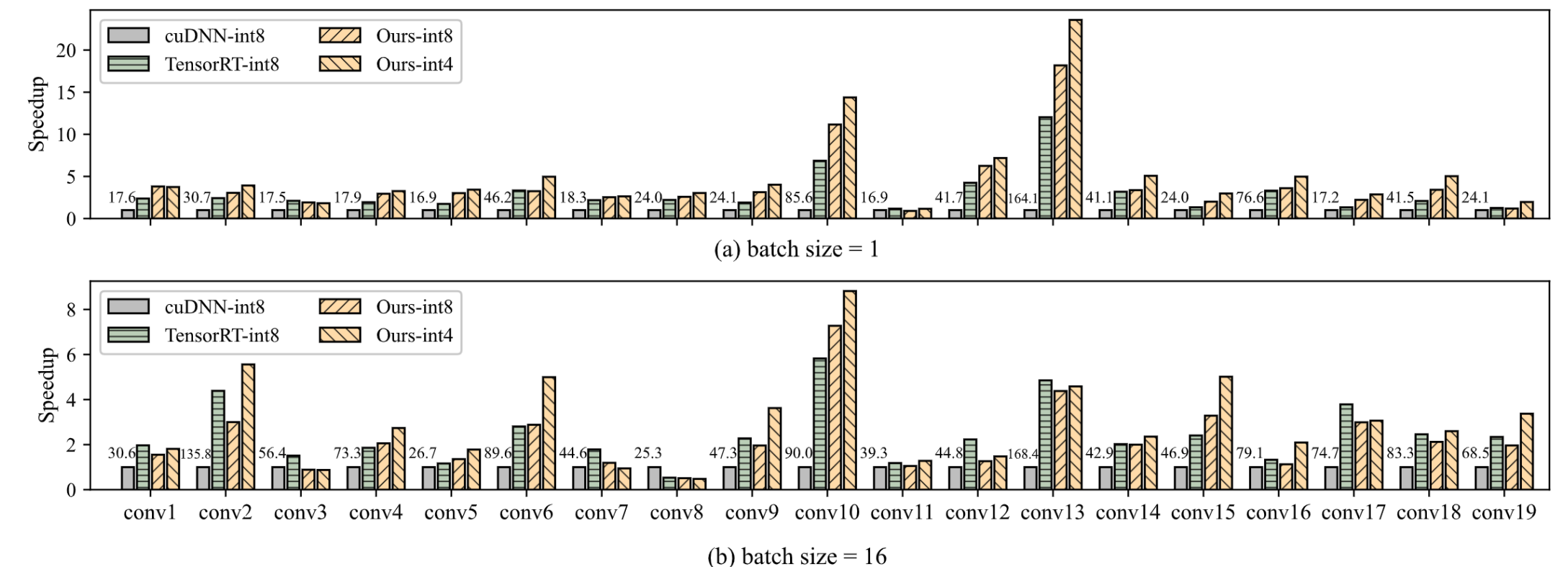
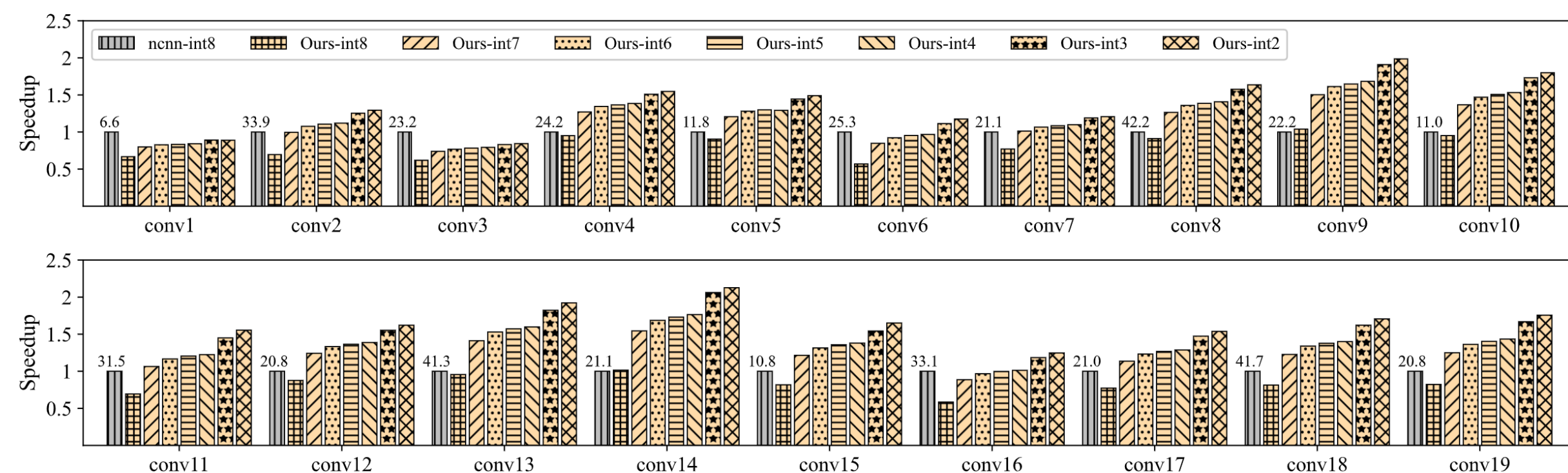
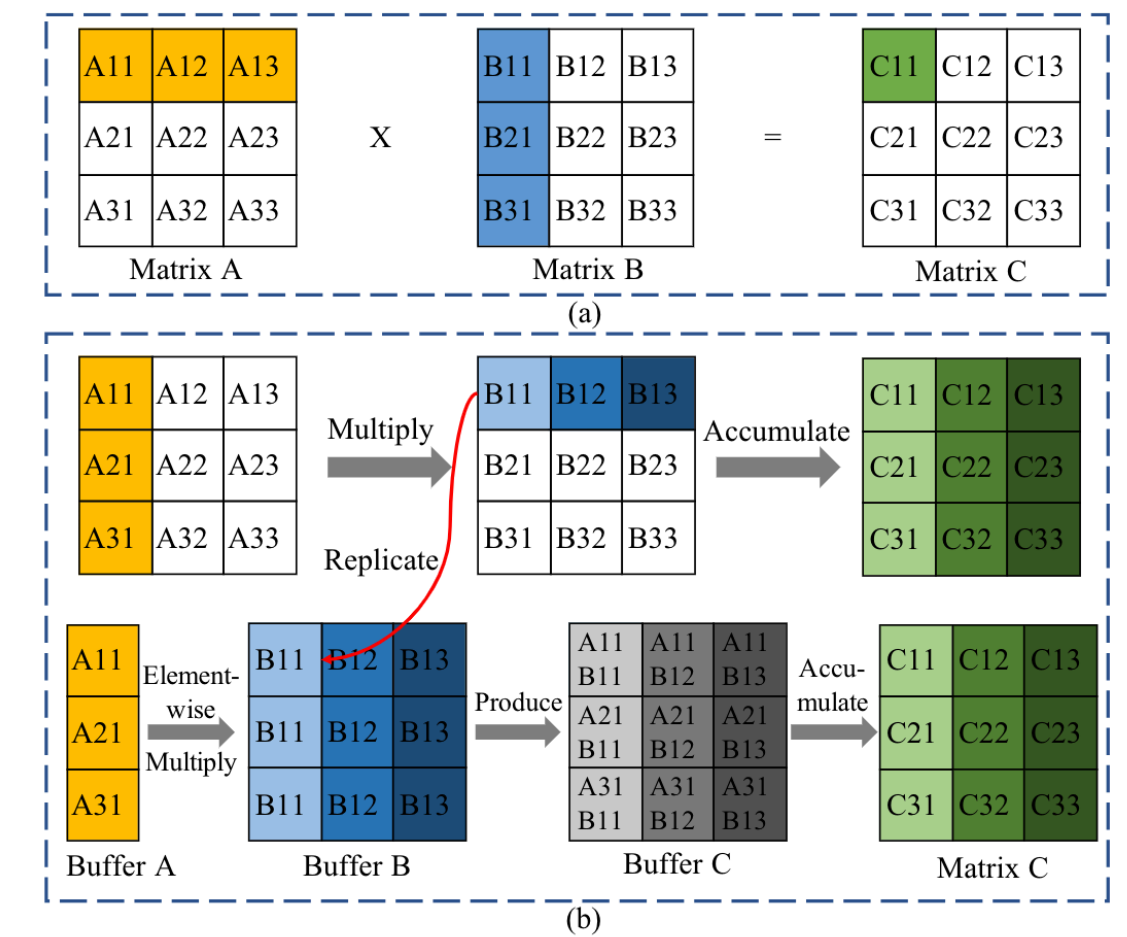
For arm neon:

use smlal for 5~8 bit compute  
use mla for 2~4 bit compute



For turning tensorcore:

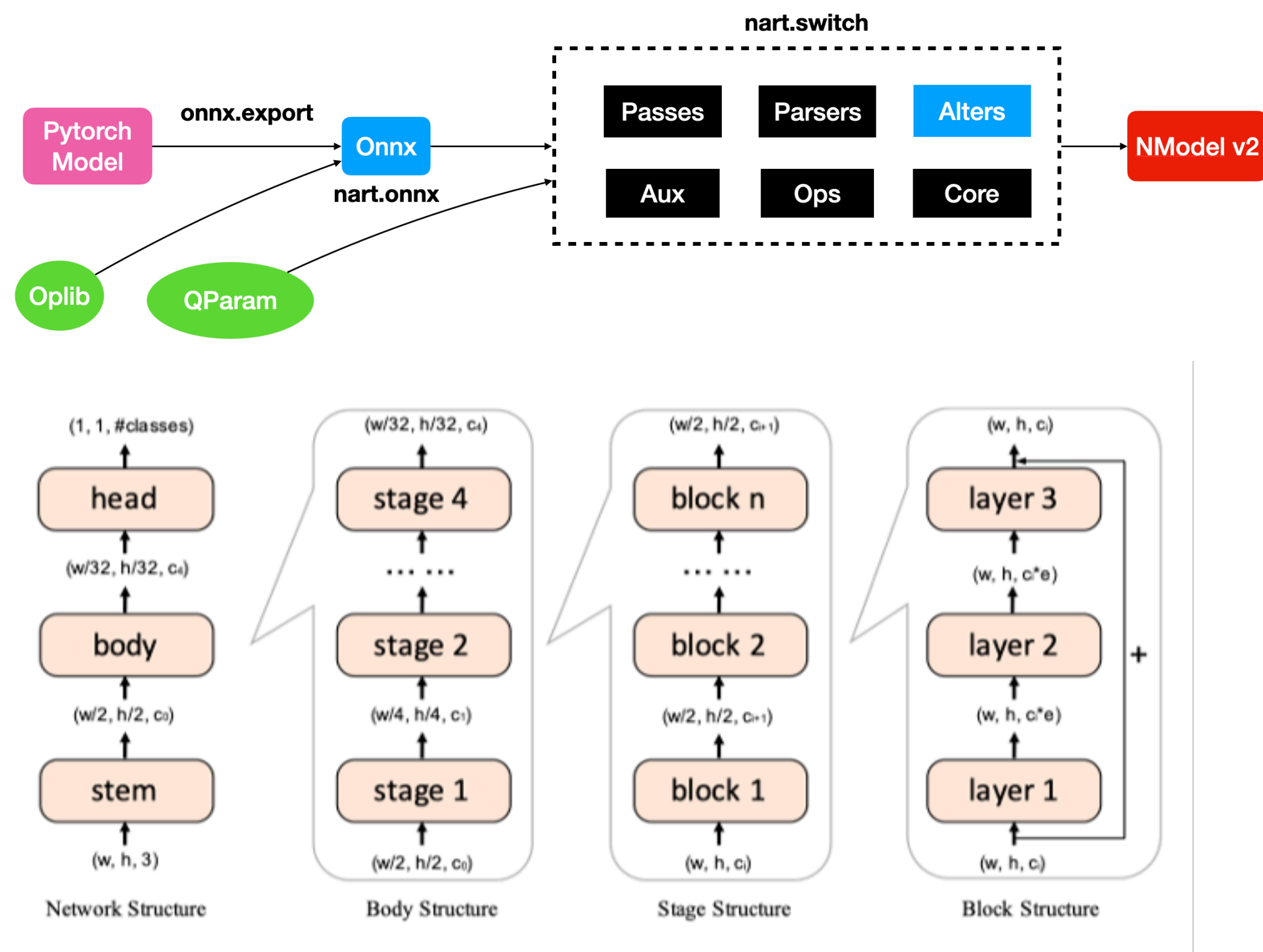
single load-replicate instruction on GEMM-based convolution data padding and packing optimization.



# Our Solutions: Efficiency

almost every NPU support int8、 but accuracy is not good enough

- ★ NART compiler allow user-defined quantization parameter
- ★ BRECCQ: a new offline quantization improve accuracy.

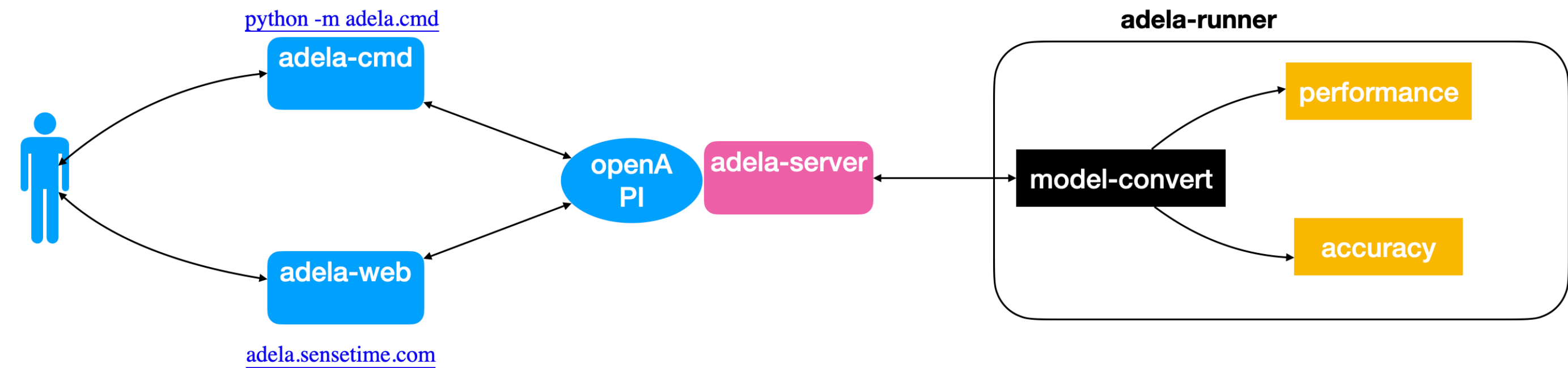


Hardware	Corp	Library	Bit number	Type
GPU	Nvidia	TensorRT	8	Per channel+ asym
		NART-quant	Fp16	IEEE 754
3559/3519/3516	Hisilicon	Nnie	4/8	Per channel/layer
Ceva DSP	Ceva	PPL	8/16	Log type
Hexagon DSP	Qualcomm	PPL	8/16	Per channel/layer
		SNPE	8	Per layer + asym
Adreno 5/6 seria	Qualcomm	PPL	Fp16	IEEE 754 without Subnormal
ARM	ARM	NART-quant	2~8	Per layer + asym
WUQI	WUQI tech	WUQI sdk	8/16	Ristretto type
SigmaStar	SigmaStar	SigmaStar sdk	8/16	Per-channel, asym A+ sym W
Davinci	Huawei	ACL	8	Per-channel
			Fp16	IEEE 754
MLU2xx	Cambricom	Neuware	8/16	Per-channel
JLQ	JLQ			
RockChip	RockChip	rknn		



# Our Solutions: Automatic

- ★ Adela System (internal): An end2end SaSS solution
- ★ Model file (ONNX)
  - upload once deploy and test everywhere!



- More than **500** users
- **200+** deployment and test / days
- **30+** new model upload / days

The screenshot shows the '模型详情' (Model Details) page in the Adela system. It features several configuration options for model deployment:

- 部署信息** (Deployment Information) and **模型详情** (Model Details) tabs.
- 模型部署平台选择** (Model Deployment Platform Selection): A dropdown menu currently set to 'cuda11.0 / nart / int4 / T4'.
- \* 量化数据集** (Quantization Dataset): A dropdown menu currently set to 'faces\_simple\_quant'.
- 模型部署批次选择** (Model Deployment Batch Selection): A numeric input field set to '8', with minus and plus buttons for adjustment. A note indicates '最大批次:2048' (Maximum Batch: 2048).
- At the bottom right, there are two buttons: **开始部署** (Start Deployment) and **取消** (Cancel).

# From Caffe format to Onnx

## Caffe

- × centralization caffe.proto
- × non standard
- × hard to extend a new op
- × no support constant tensor



## ONNX

- ✓ extendable onnx.proto
- ✓ opset version
- ✓ easy to extend a new op
- ✓ no support constant tensor



NART.switch compile onnx to vary type of caffe version

**Thanks!**