QUANTPIPE: APPLYING ADAPTIVE POST-TRAINING QUANTIZATION FOR DISTRIBUTED TRANSFORMER PIPELINES IN DYNAMIC EDGE ENVIRONMENTS

Haonan Wang[1 2], Connor Imes[2], Souvik Kundu[1 3], Peter A. Beerel[1 2], Stephen P. Crago[1 2], and John Paul Walters[2]

[1]University of Southern California, Los Angeles, CA, USA

[2]USC Information Sciences Institute, Arlington, VA, USA

[3]Intel Labs, San Diego, CA, USA

BACKGROUND AND MOTIVATION

- AI faces a growing problem in the Transformer era
- High demands on local inference
  - *E.g., run ChatGPT in your house*
  - *Concerns of privacy or connectivity to Internet*
- Pushing these increasingly large models to the edge adds additional challenges
  - *Resources/power constrains of Edge devices*
- Pipeline parallelism can be employed to parallelize large-scale transformer models across devices

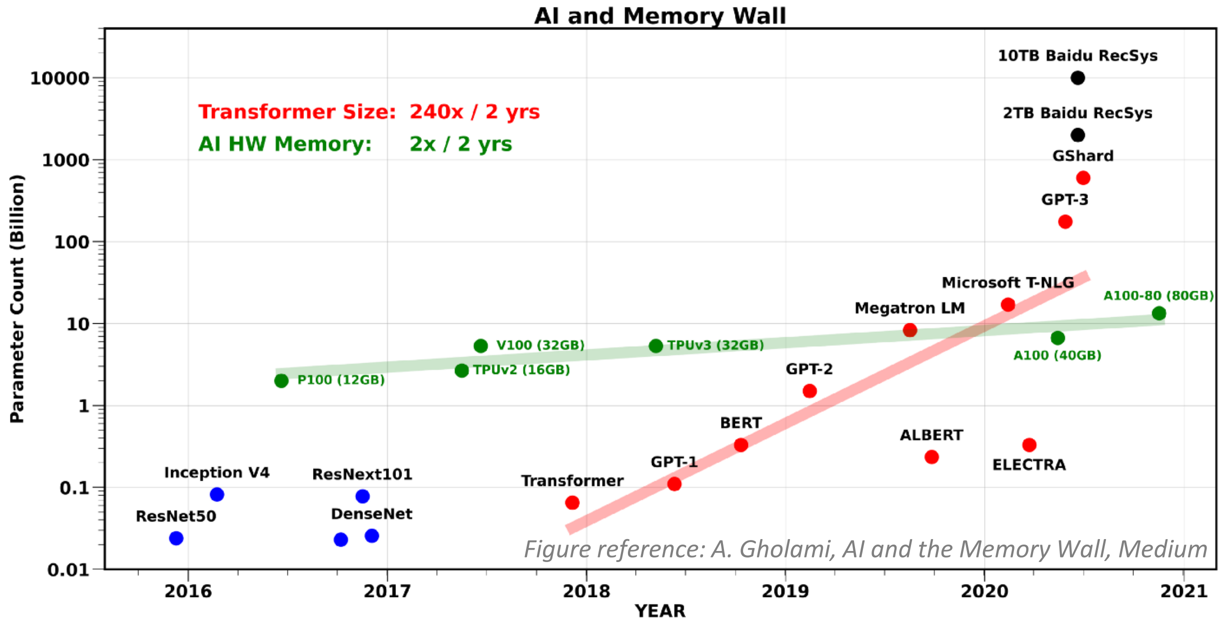


Figure reference: A. Gholami, AI and the Memory Wall, Medium

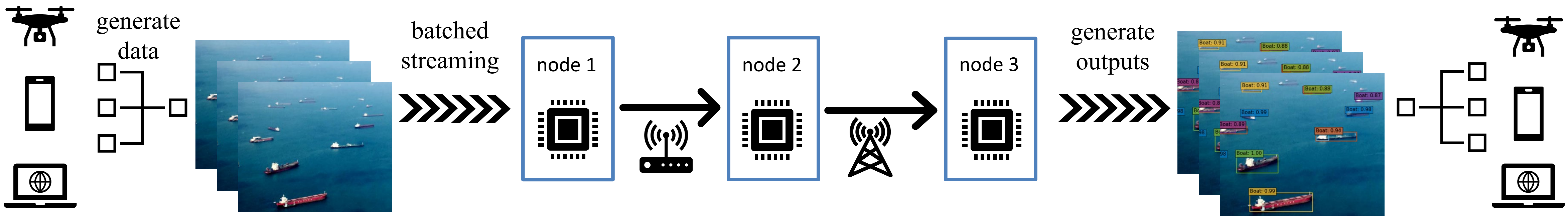
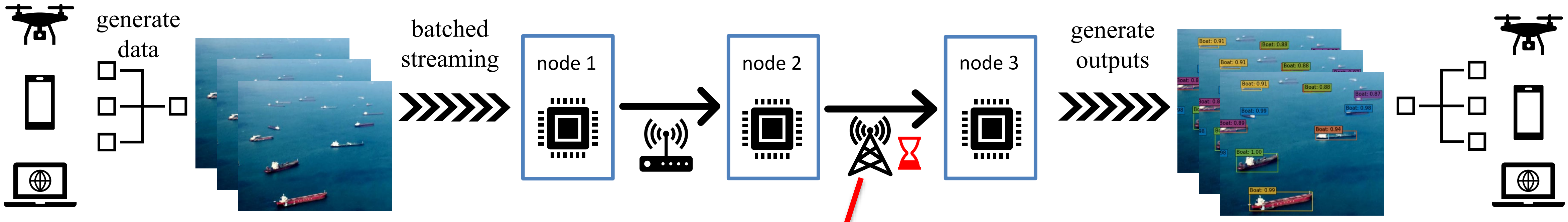


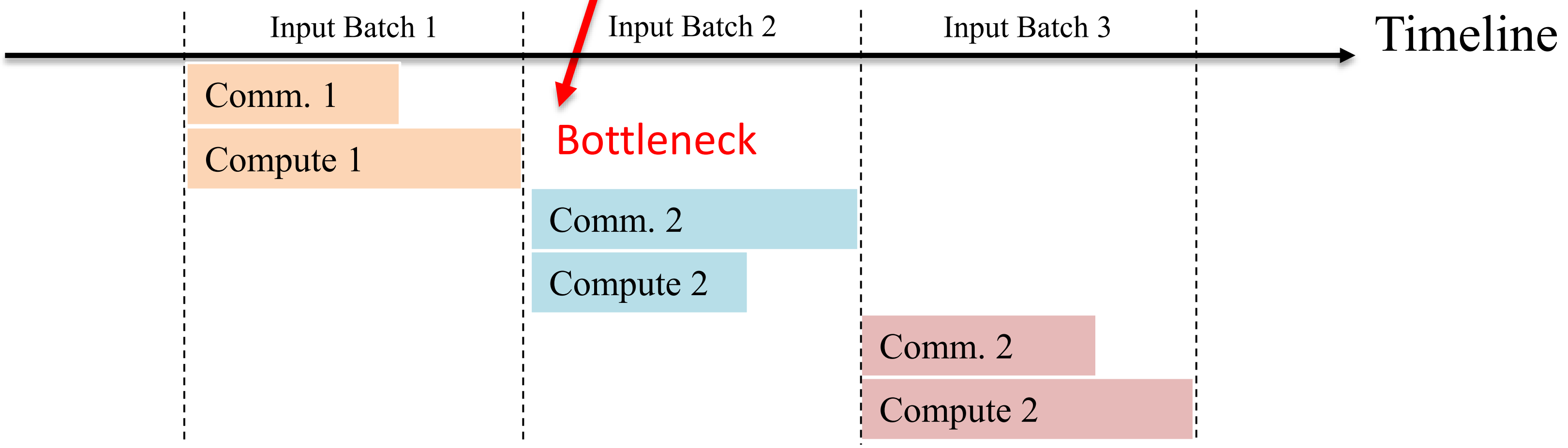
Fig. Illustration of the pipeline parallelism paradigm

BACKGROUND AND MOTIVATION



Property of the Edge System:

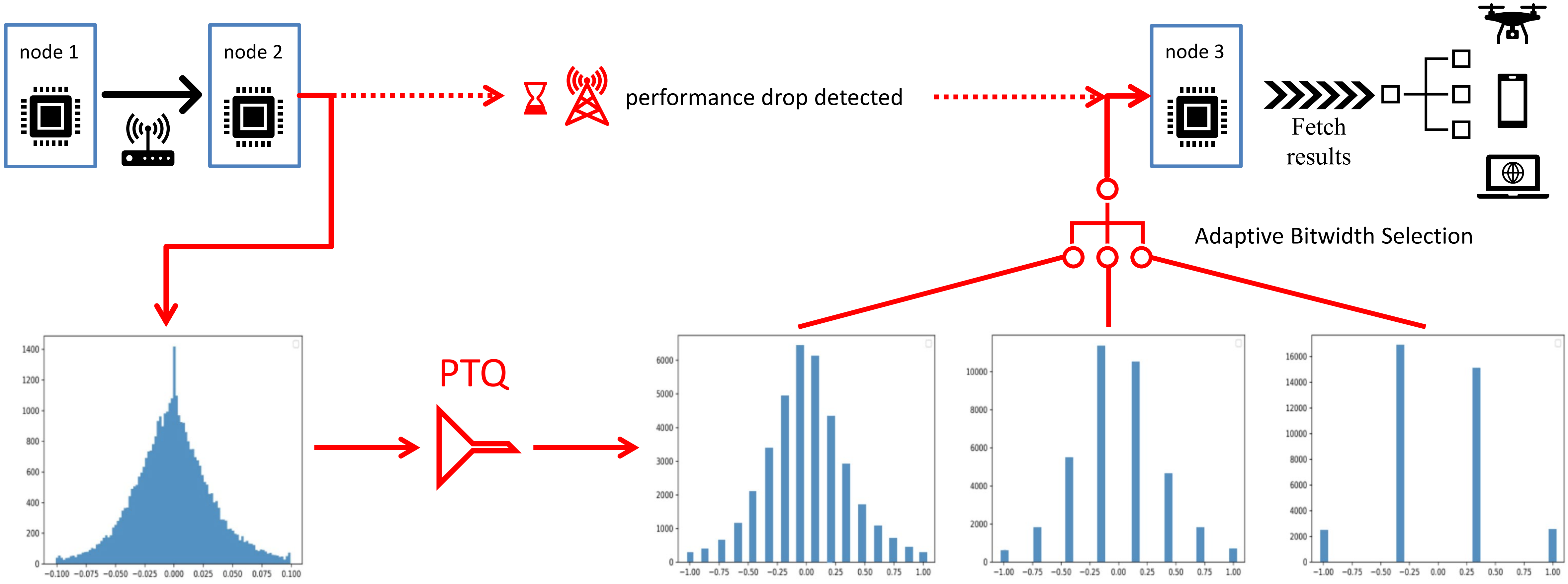
- ❖ Unstable Connection
  - Network fluctuation
  - Signal blocking
  - Movement
- ❖ Low-speed protocols
  - Bluetooth
  - LP-WAN
  - …



QUANTPIPE OVERVIEW

Q: How to compress communication?

A: Post-training Quantization (PTQ).



Original Distribution (32bit float)

Distribution after PTQ (4-bit, 3-bit, and 2-bit integer)

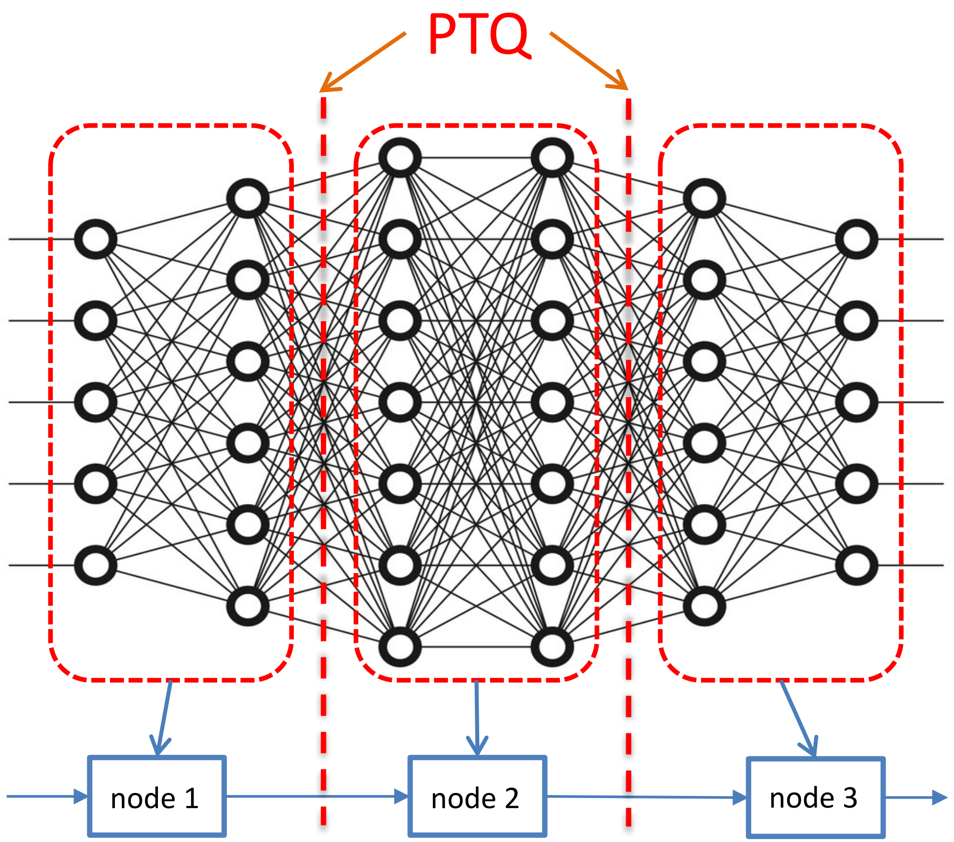
ADAPTIVE POST-TRAINING QUANTIZATION

Challenges of applying PTQ:

- ❖ Where to do PTQ?
- ❖ How to do PTQ?
- ❖ What is the accuracy loss?

Property of PTQ:

- We insert PTQ only at the boundary of the pipeline where the model is partitioned, to lower the impact of quantization
- Experimental results show PTQ is suitable for the PipeEdge System



ADAPTIVE POST-TRAINING QUANTIZATION

How to do PTQ?

- Analytical Clipping for Integer Quantization (ACIQ) [1]
  - ➢ *A PTQ method for CNN models*
  - ➢ *Clip the outliers to significantly improve accuracy*
  - ➢ *Decide the best clip range that minimizes the mean square error (MSE)*
- Applying PTQ to Visual Transformer (ViT) models
  - ➢ Two types of activation distribution
  - ➢ Mismatch of distribution estimation for real data

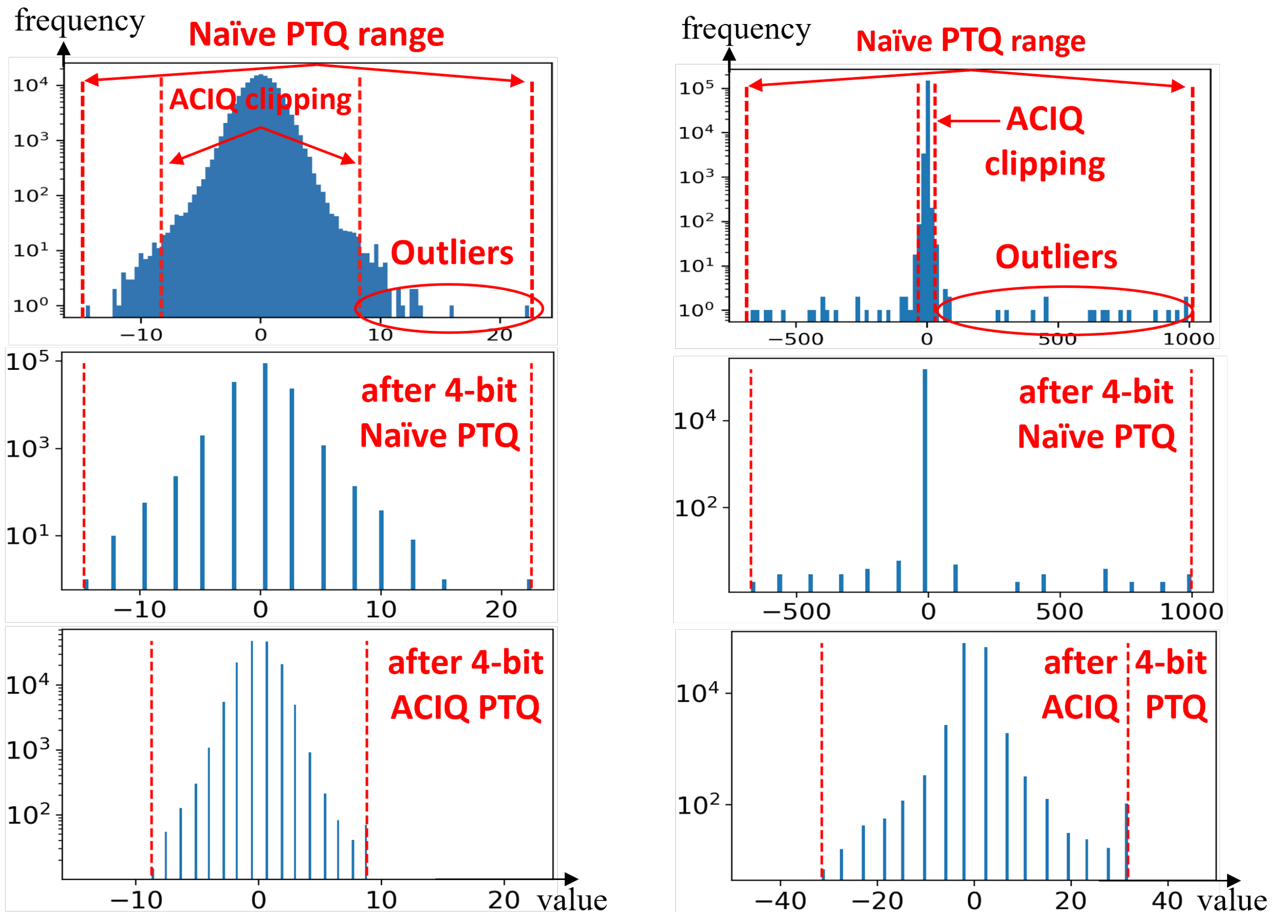


Fig. Distribution of the original data (top), after naive PTQ (middle), or after PTQ with ACIQ (bottom) from the ViT-Base model partitioned after 4th (left) and 6th (right) block.

[1] Banner, Ron, Yury Nahshan, and Daniel Soudry. "Post training 4-bit quantization of convolutional networks for rapid-deployment." *Advances in Neural Information Processing Systems* 32 (2019).

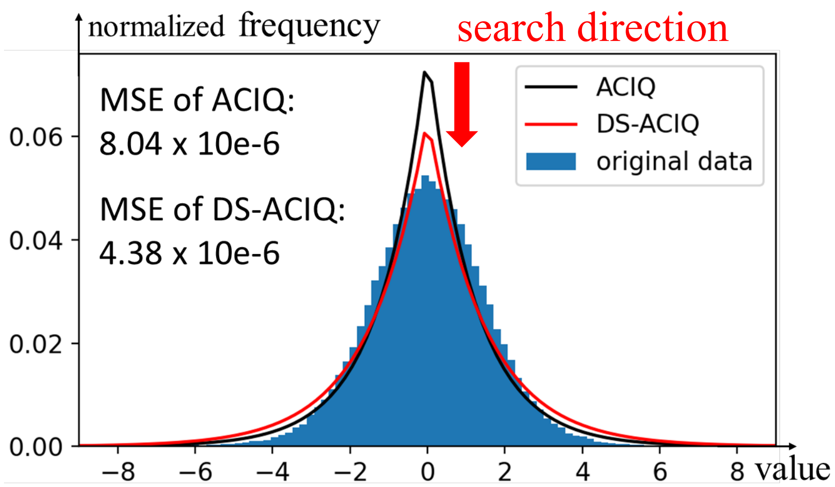
ADAPTIVE POST-TRAINING QUANTIZATION

Directed-search ACIQ (DS-ACIQ):

- For better estimation of the data distribution
- Search direction is determined by the peak of histogram curve
- Further decrease the MSE by ~50%
- Only incur < 1% computation overhead
- Accuracy of PTQ w/ DS-ACIQ (PDA):

Table 1: Average ViT-Base model accuracy with ImageNet.

	32bit	16bit	8bit	6bit	4bit	2bit
PTQ	80.23%	**80.26%**	75.74%	43.03%	30.29%	0.44%
ACIQ		80.03%	**79.35%**	**78.87%**	76.46%	54.97%
PDA		78.94%	78.72%	78.21%	**77.34%**	**70.82%**



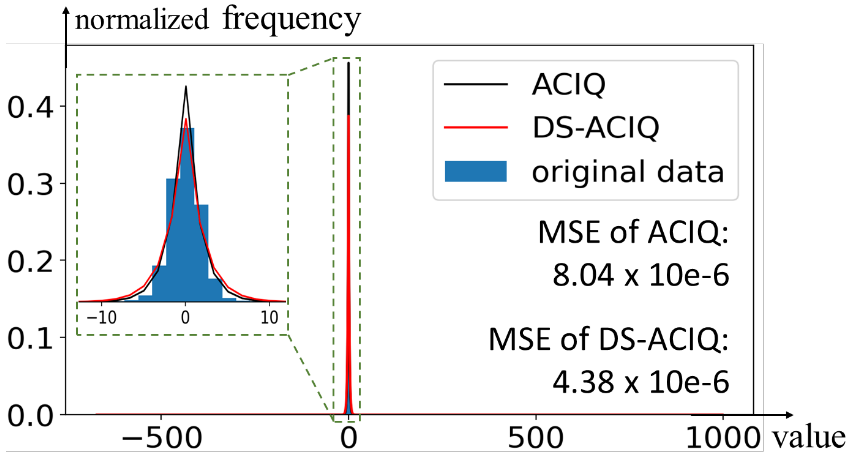


Fig. Comparison of ACIQ and DS-ACIQ

ADAPTIVE POST-TRAINING QUANTIZATION

Adaptive PTQ with DS-ACIQ (Adaptive PDA):

- Implement PDA in our QuantPipe system:
  - monitor the output bandwidth $B_{k,t}$ of stage k at inference iteration t
  - estimate the bitwidth $q_{k,t+1}$ required to achieve the target throughput R

$$q_{k,t+1} = 32/2^{\lceil log(\frac{V_{k,t} \times 32/q_{k,t}}{S/R \times B_{k,t}}) \rceil}$$

where $V_{k,t}$ represents the volume of quantized data under $q_{k,t}$

and S denotes the microbatch size

- In the real implementation:
  - QuantPipe monitors the bandwidth every 50 batches
  - Switch quantization bitwidth at runtime to recover system performance

EXPERIMENTAL EVALUATIONS

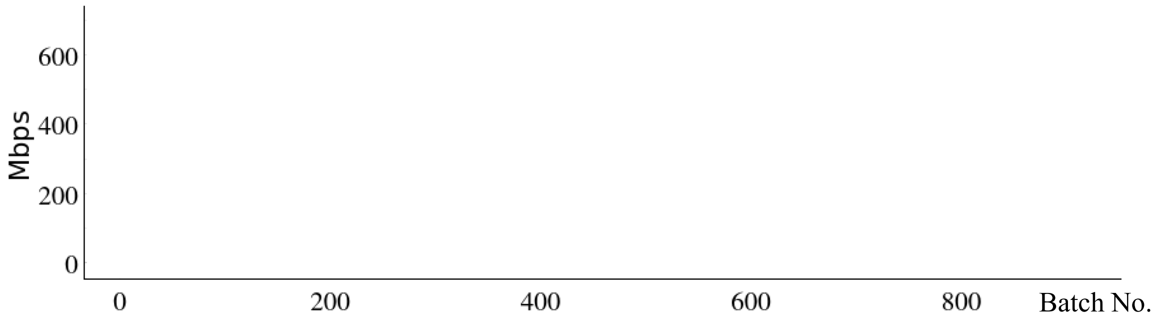
Experimental Settings:

- Hardware Testbed:
  - ➢ An Edge cluster with 6 NVIDIA Jetson AGX Orin devices
  - ➢ Each device has a 12-core ARM CPU, a 1792-core GPU, and runs Linux kernel 5.10.65-tegra.
  - ➢ 1Gbps Ethernet connection between devices
- Software:
  - ➢ We implement our QuantPipe on top of the PipeEdge, a distributed edge computing framework[2]
  - ➢ using Python 3.8 and PyTorch 1.12.
- Bandwidth Control:
  - ➢ We simulate the network fluctuation using Linux traffic control tools (tc).
- Deep Learning Model:
  - ➢ Visual Transformer (ViT)

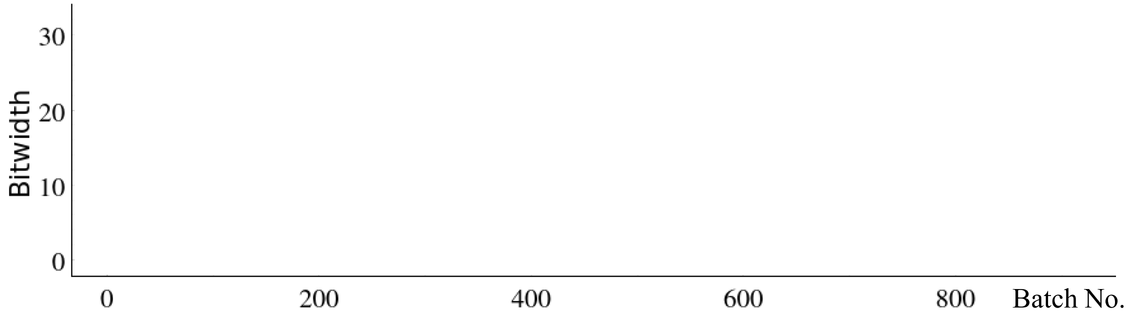
[2] Y. Hu et al., "PipeEdge: Pipeline Parallelism for Large-Scale Model Inference on Heterogeneous Edge Devices," 2022 25th Euromicro Conference on Digital System Design (DSD), Maspalomas, Spain, 2022, pp. 298-307, doi: 10.1109/DSD57027.2022.00048.

Experimental results: Adaptiveness

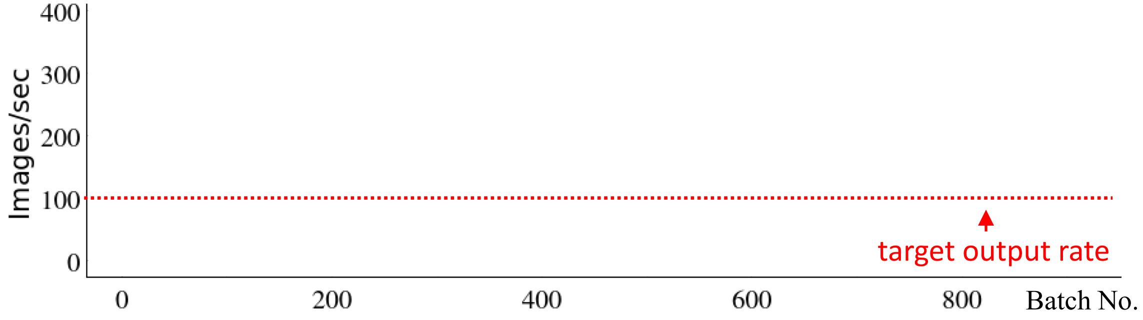
Stage Bandwidth



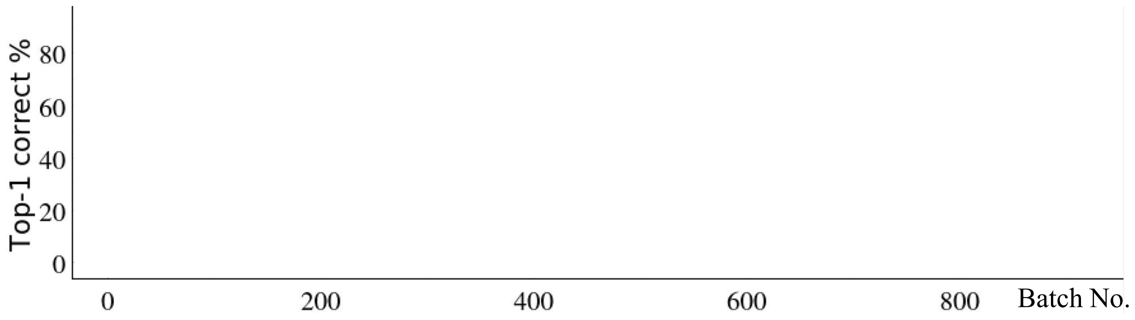
Quantization Bitwidth



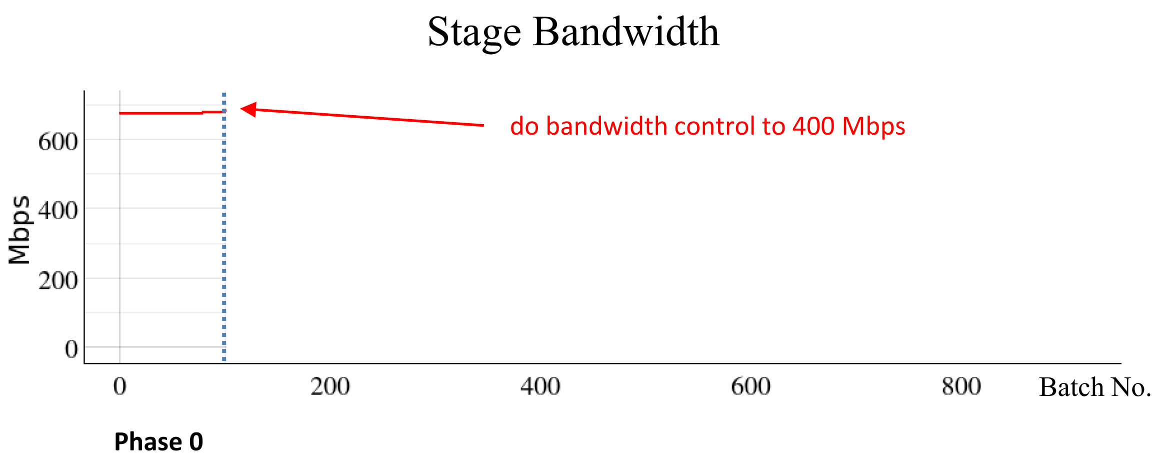
Stage Performance

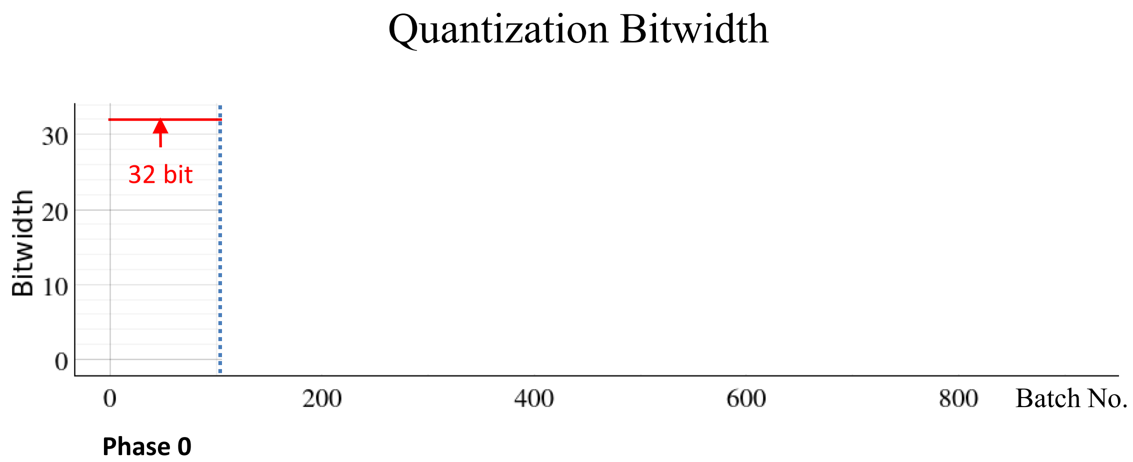


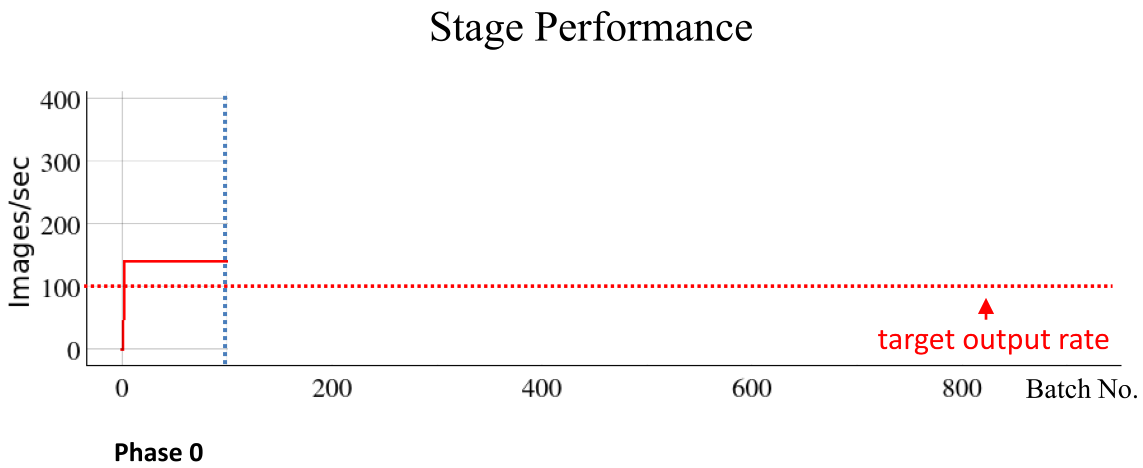
Model Accuracy

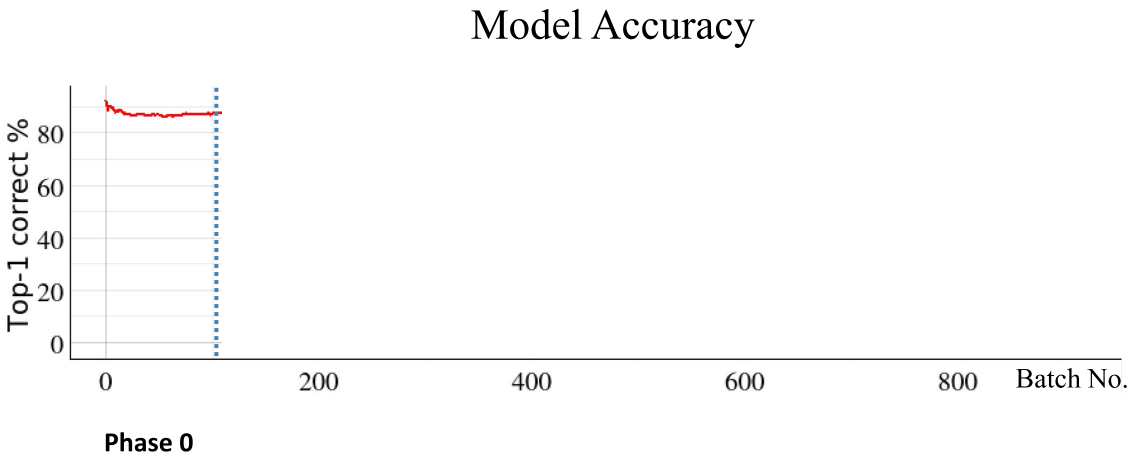


Experimental results: Adaptiveness

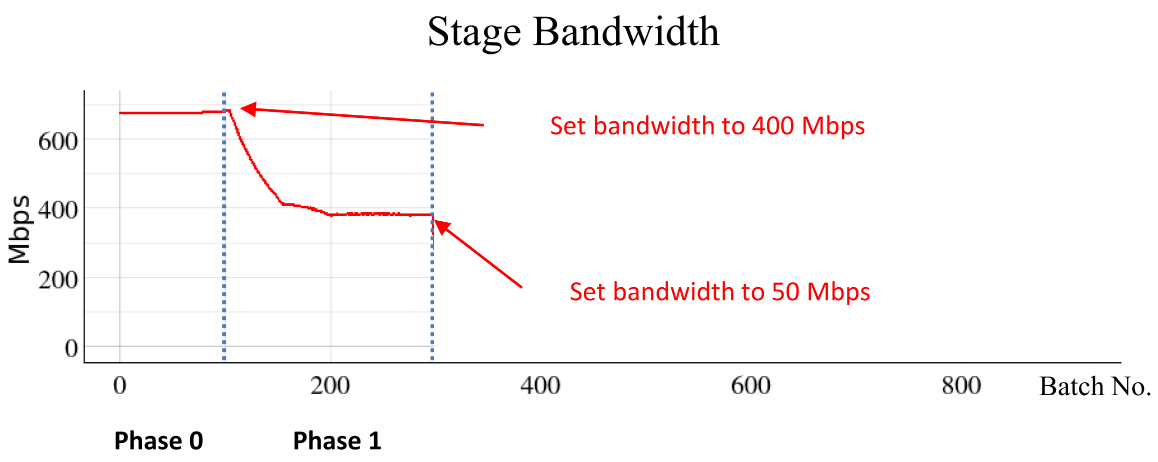


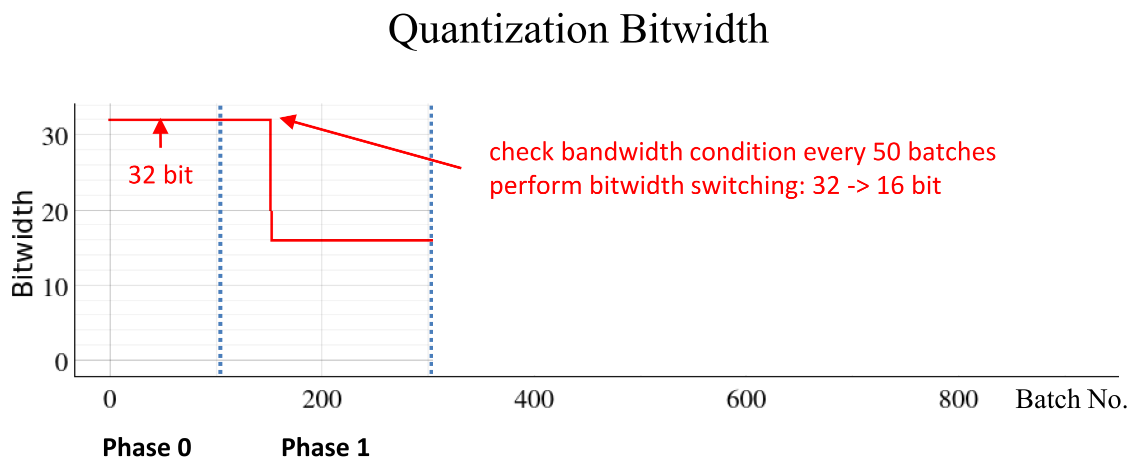


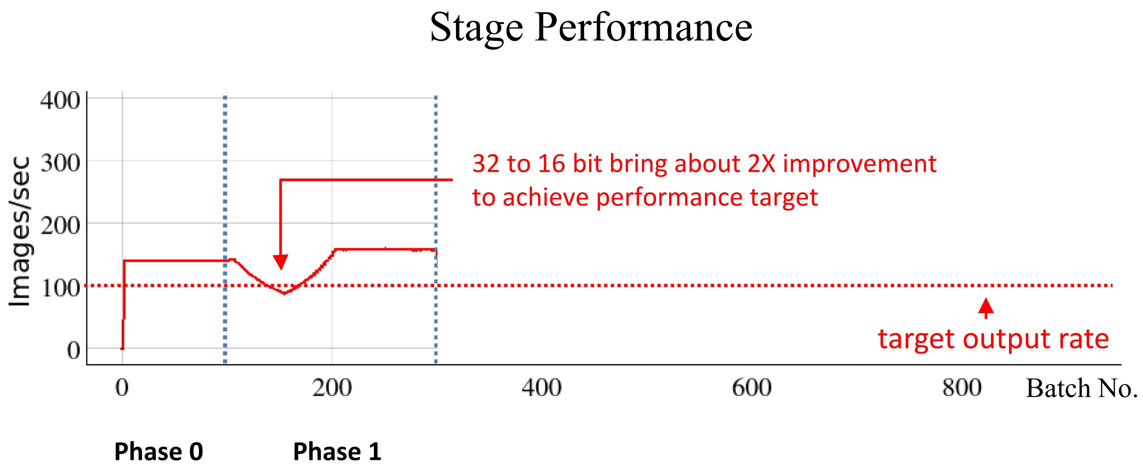


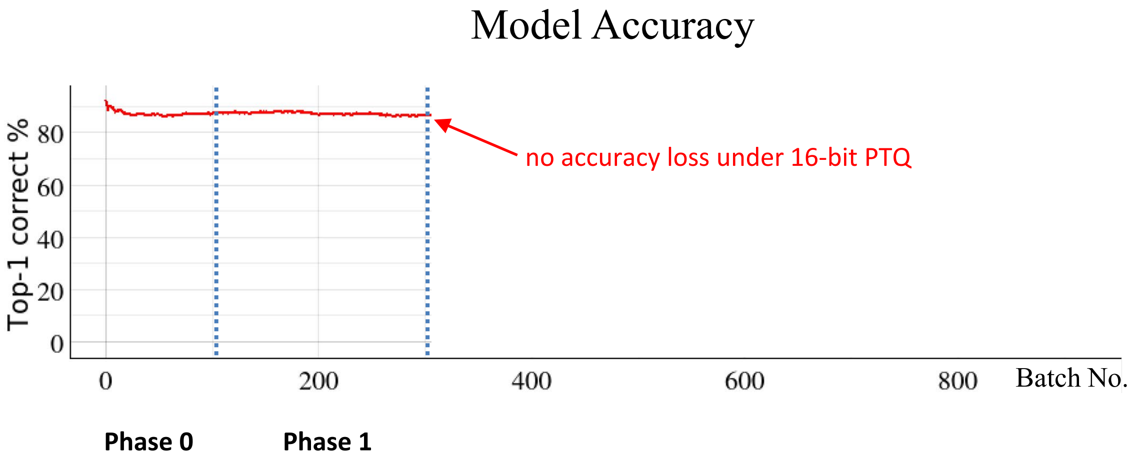


Experimental results: Adaptiveness

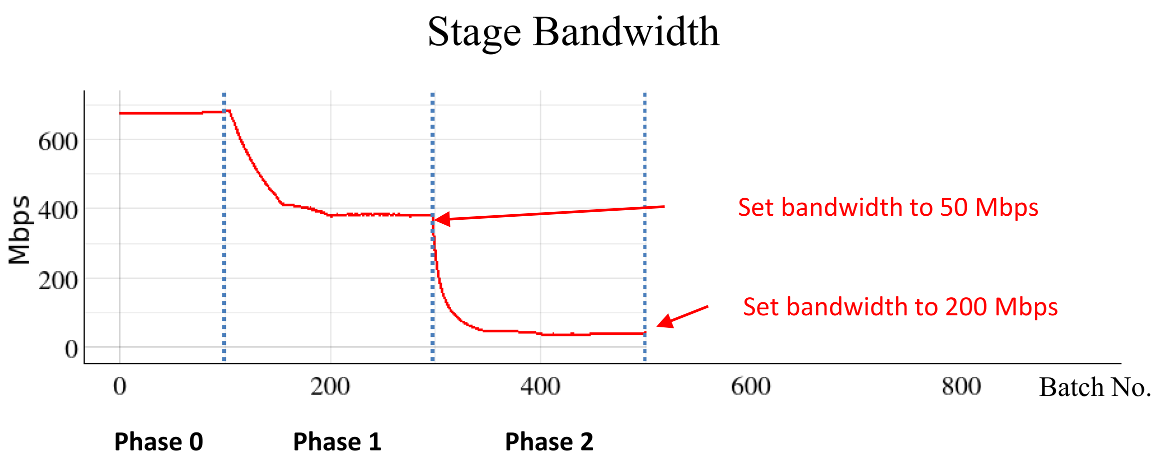


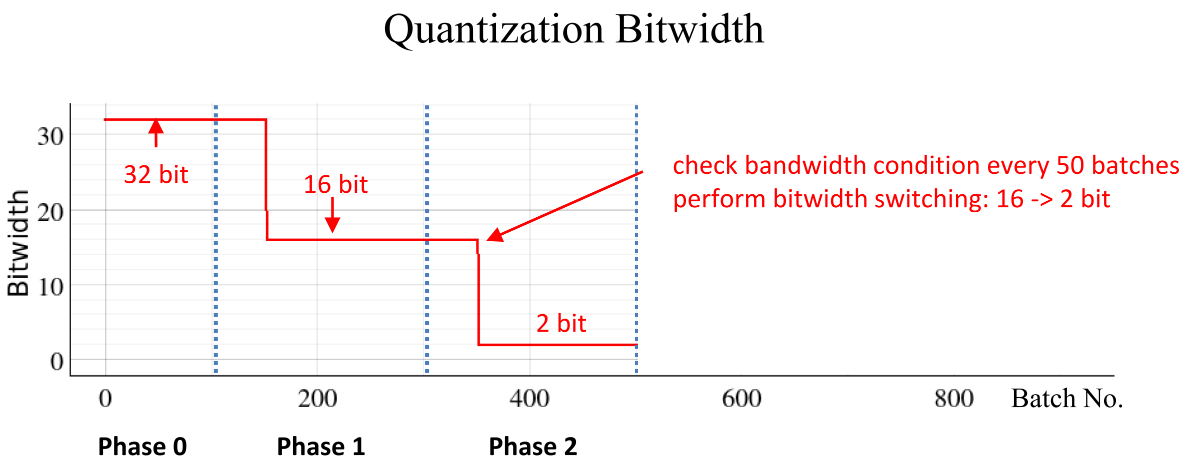


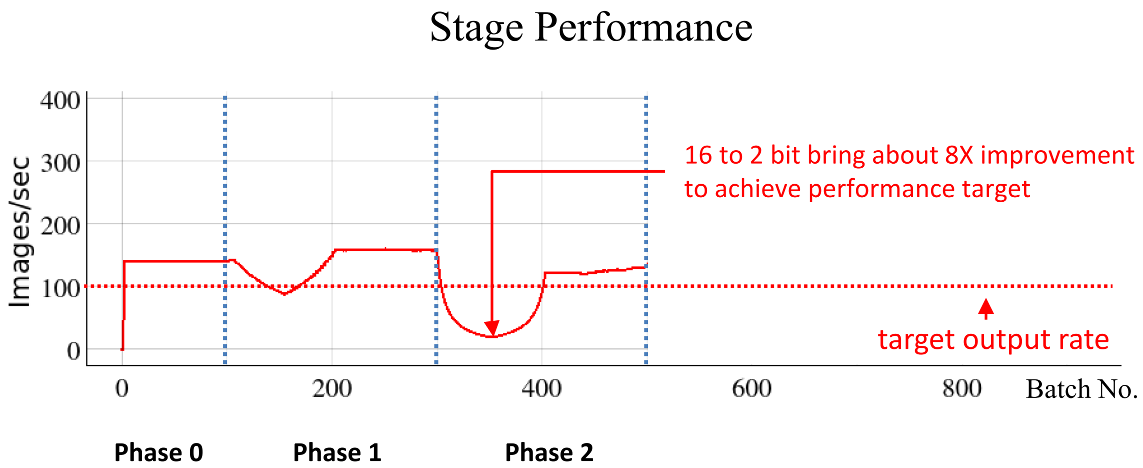


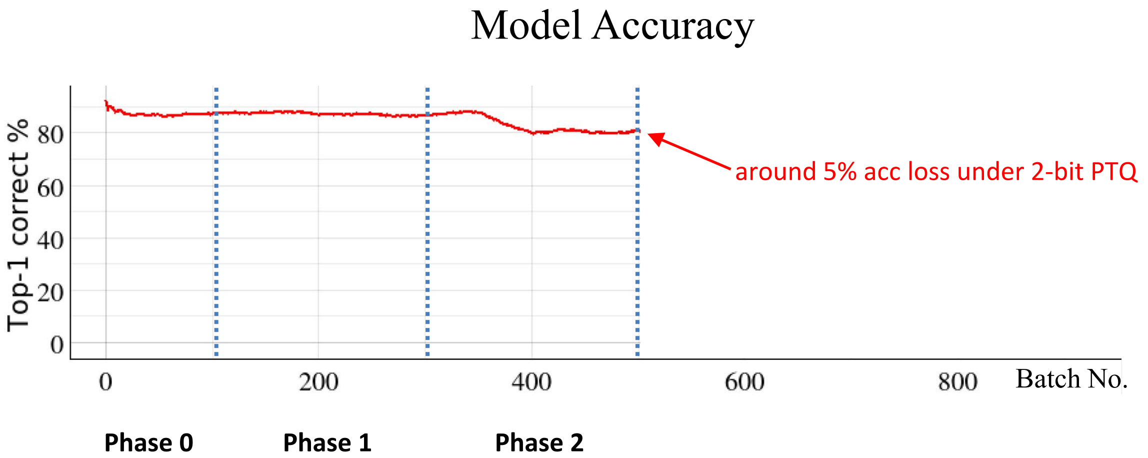


Experimental results: Adaptiveness

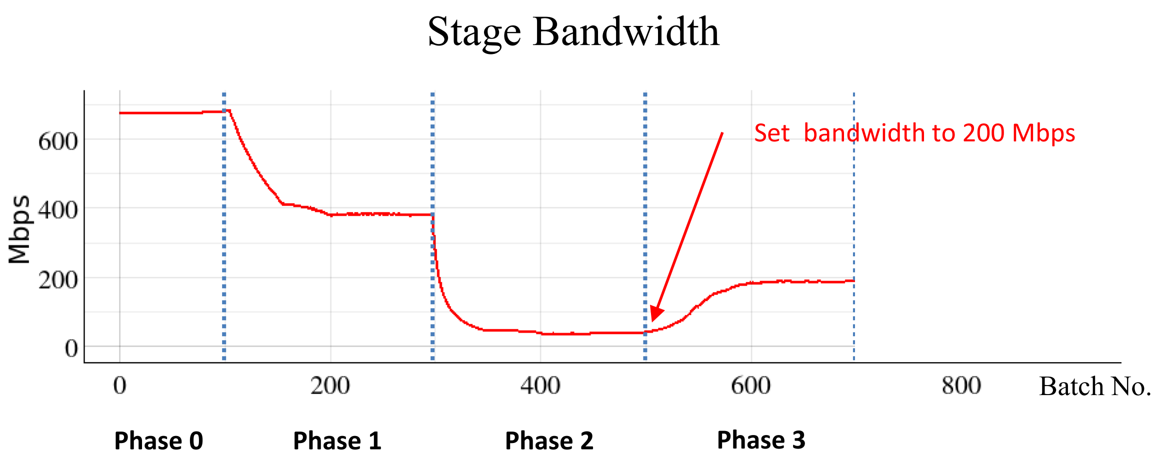


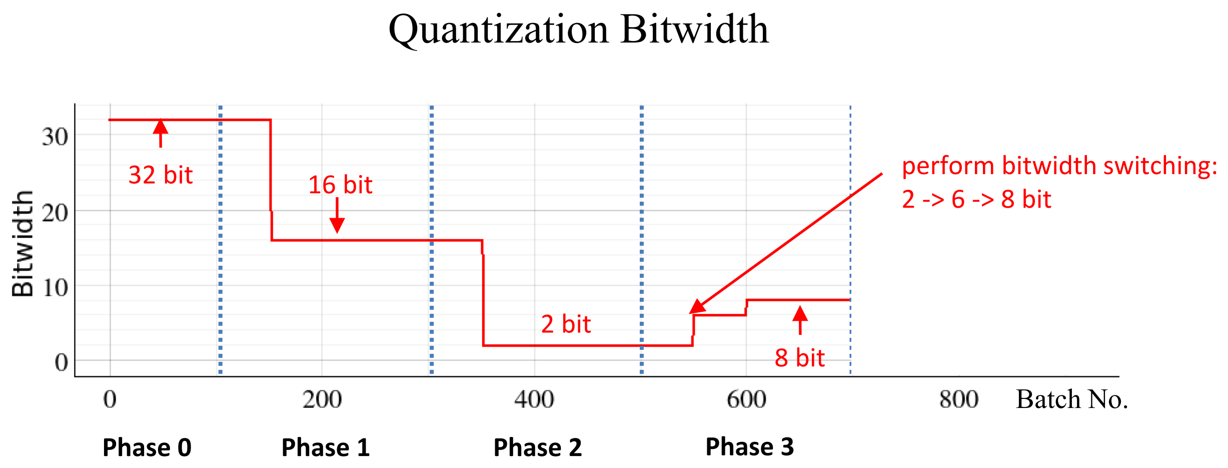


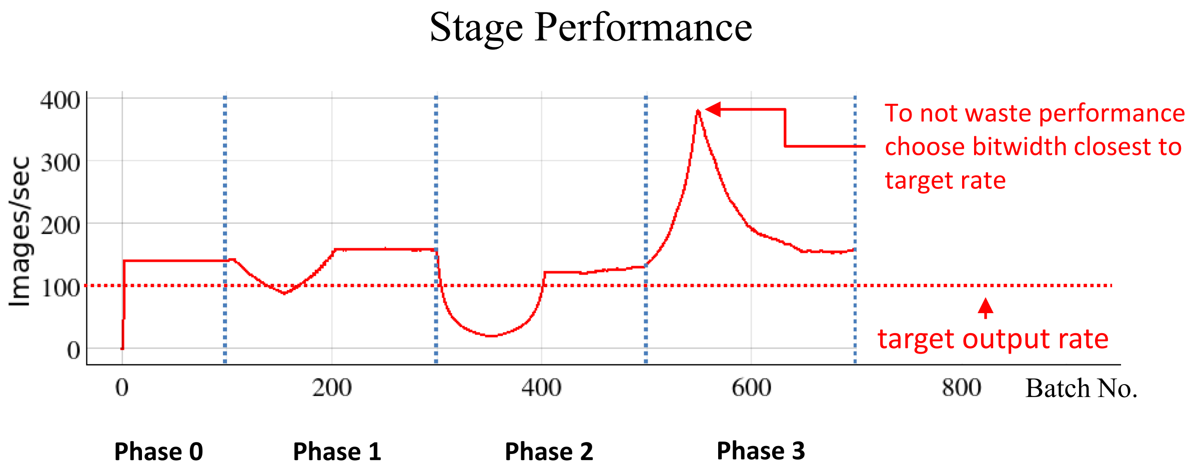


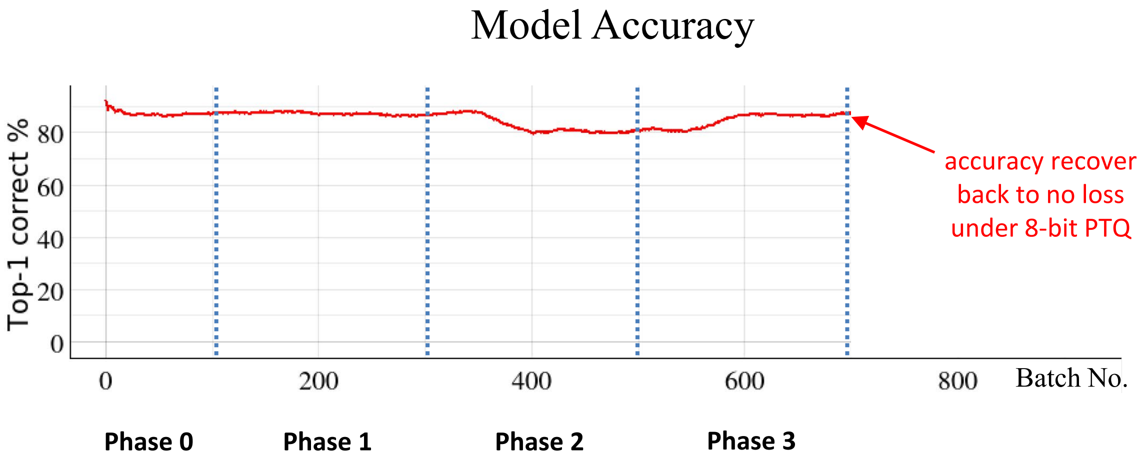


Experimental results: Adaptiveness

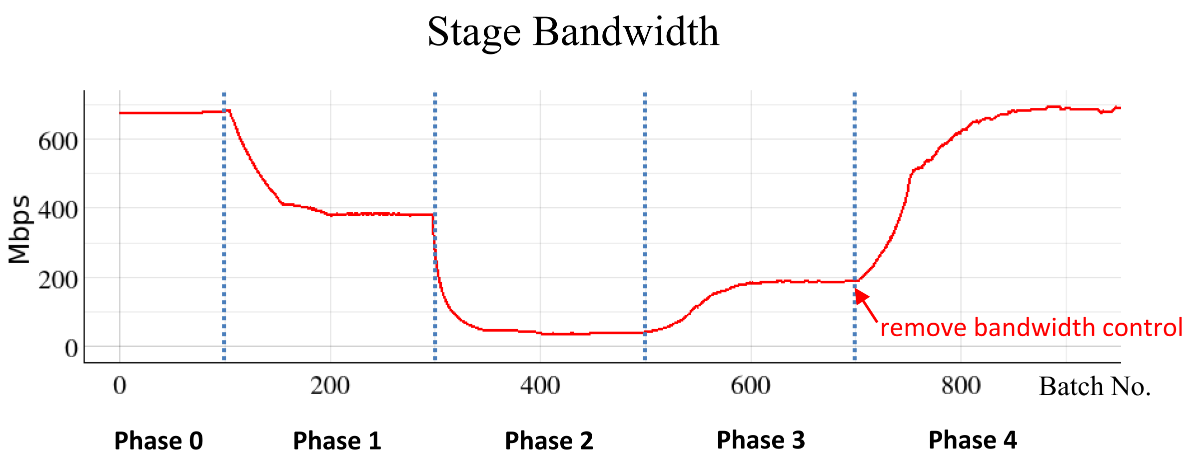


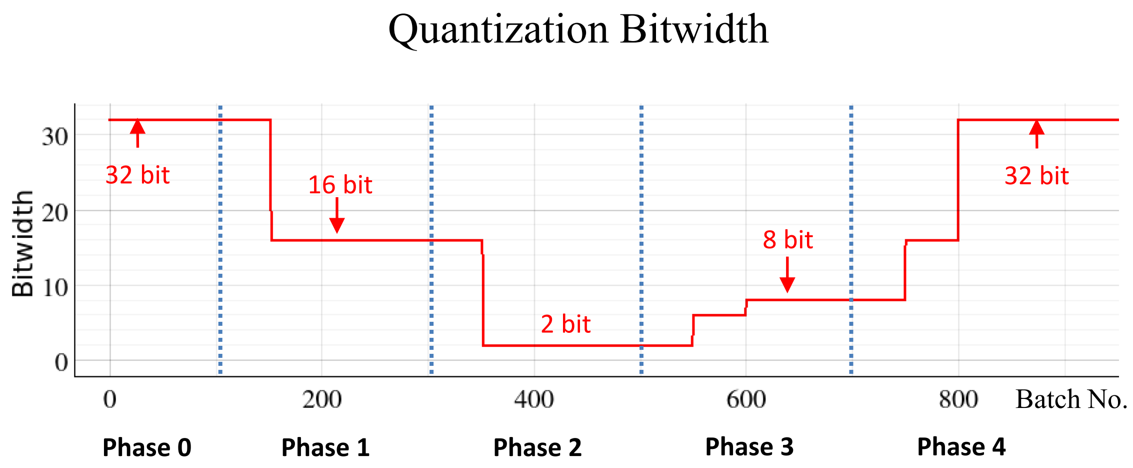


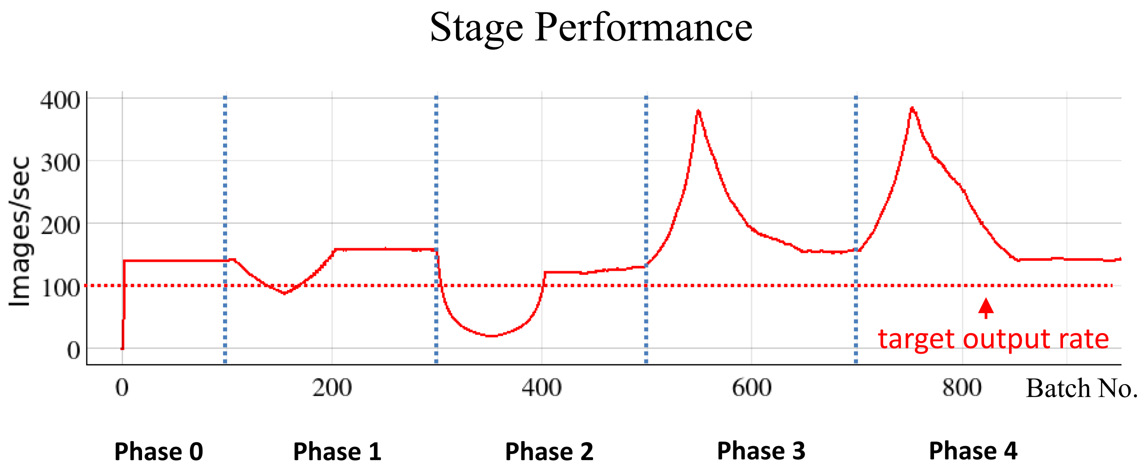


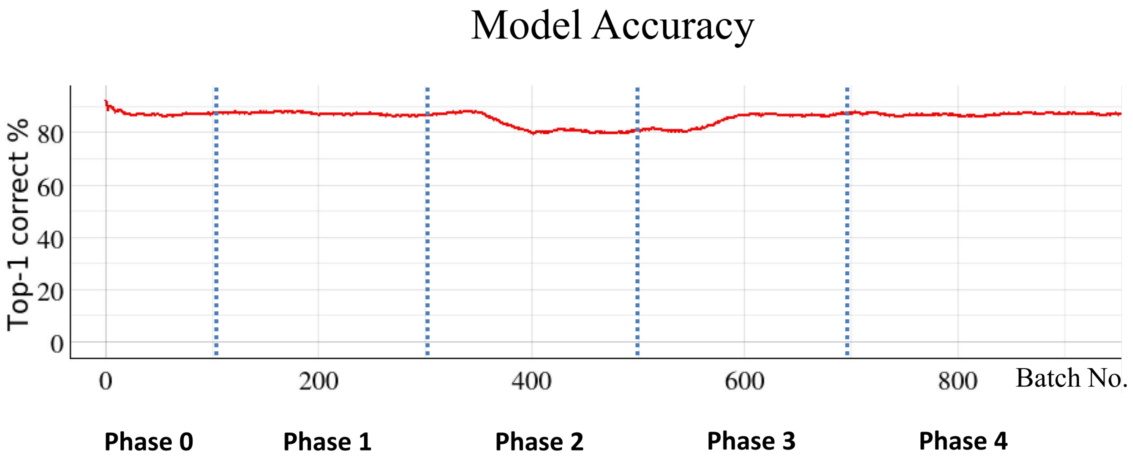


Experimental results: Adaptiveness









Demo Devices

Thank You

Any further questions are welcome
please contact: haonan.wang@usc.edu