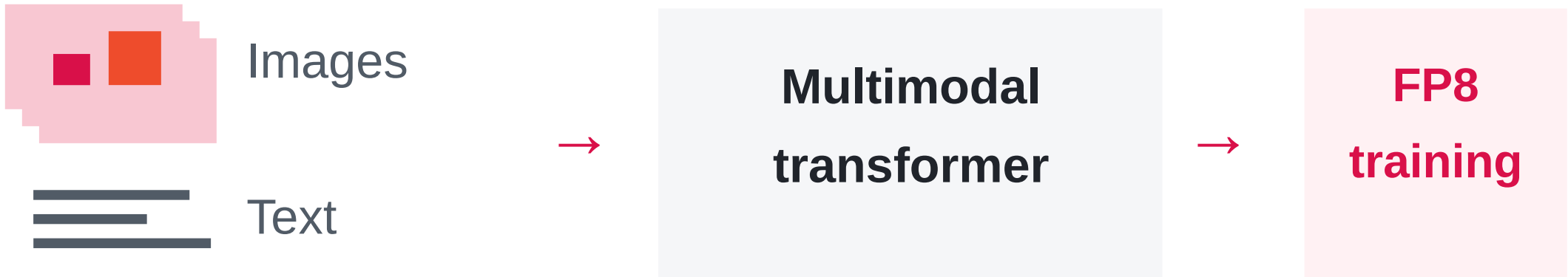


Wei Chen · Yu Shen · Iain Weissburg | NVIDIA, Alpamayo ML Infrastructure Team

FP8 workflow: FP8 kernel fusion + torch.compile() + ONNX export

01 Background

Multimodal transformers combine expensive projections, attention and feed-forward layers. FP8 targets the matrix-multiply workload to improve training performance.



02 Design: two FP8 integration paths

Transformer Engine: Fused modules

Cosmos VLA backbone benchmarking and reproducible H100 packaging, with FP8 kernel verification.



Fused Cosmos-VLA attention and SwiGLU blocks.

Observed in our training pipeline

Better fusion support and more FP8 operators. End-to-end FP8 training support, including the optimizer.

torchao: Linear conversion

Backbone Linear conversion with tensorwise or axiswise scaling; inner-dimension padding.

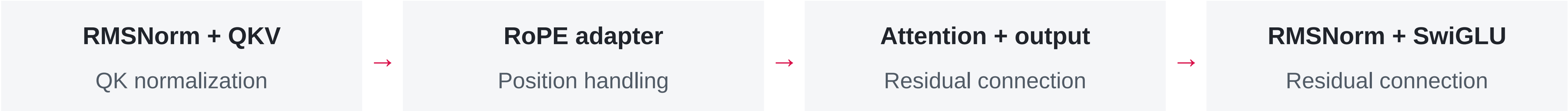


torch.compile() + FSDP2; no FP8 autocast needed.

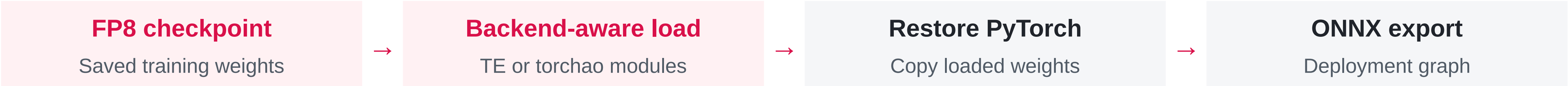
Observed in our training pipeline

Better torch.compile() support and better FSDP2 support.

Cosmos VLA fused backbone



Checkpoint-aware export



03 Results

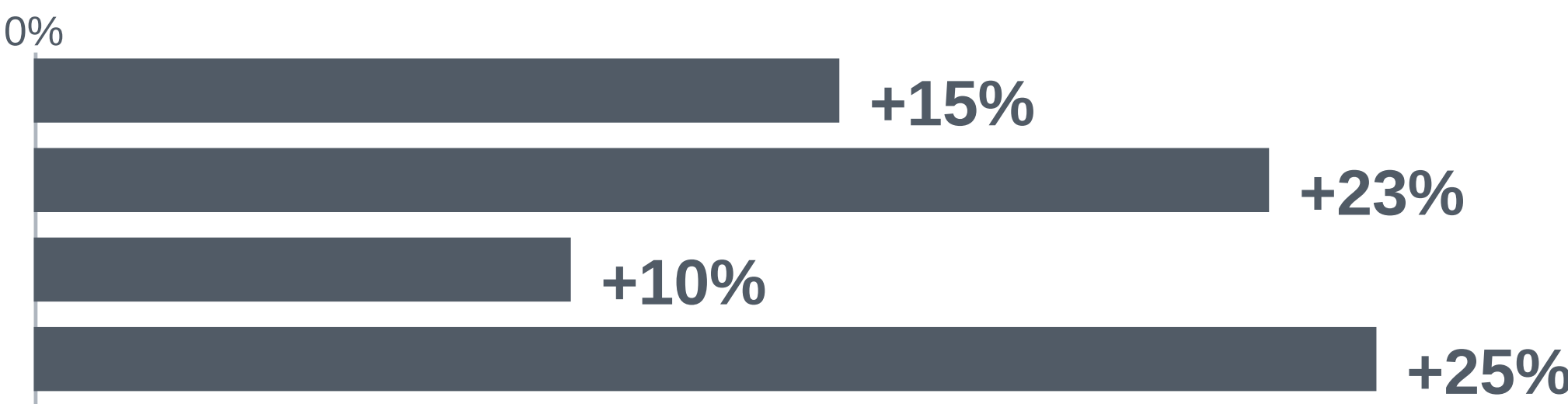
H100 results in our training setup | Change vs. baseline

Transformer Engine (layer fusion)

TE + torch.compile() (linear layers only)

torchao (linear layer only)

torchao + torch.compile() (linear layer only)



Results are specific to this setup. Benchmark configuration and metric definition are pending.

04 Future work

More layer fusion with torchao

Explore more layer fusion to improve end-to-end training performance.

Transformer Engine with NVFP4

Try NVFP4 on Blackwell and evaluate training performance.

Transformer Engine with CUDA Graphs

Evaluate Transformer Engine with CUDA Graphs for training.