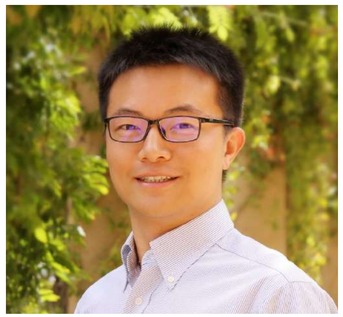
HAN LAB
Hardware Accelerated Neural-nets

Accelerating Large Language Model and Generative AI



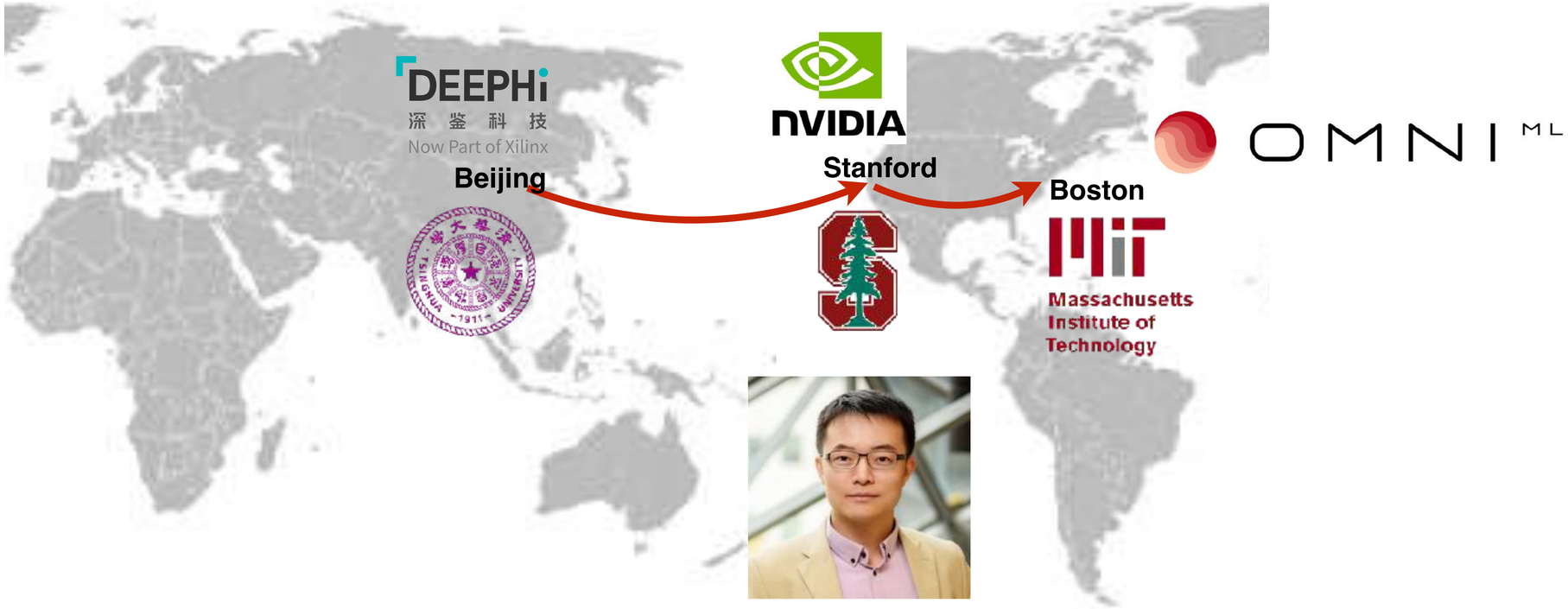
Song Han

Associate Professor, MIT
Distinguished Scientist, NVIDIA

https://songhan.mit.edu
@SongHan_MIT

Generated by Midjourney

Prof. Song Han

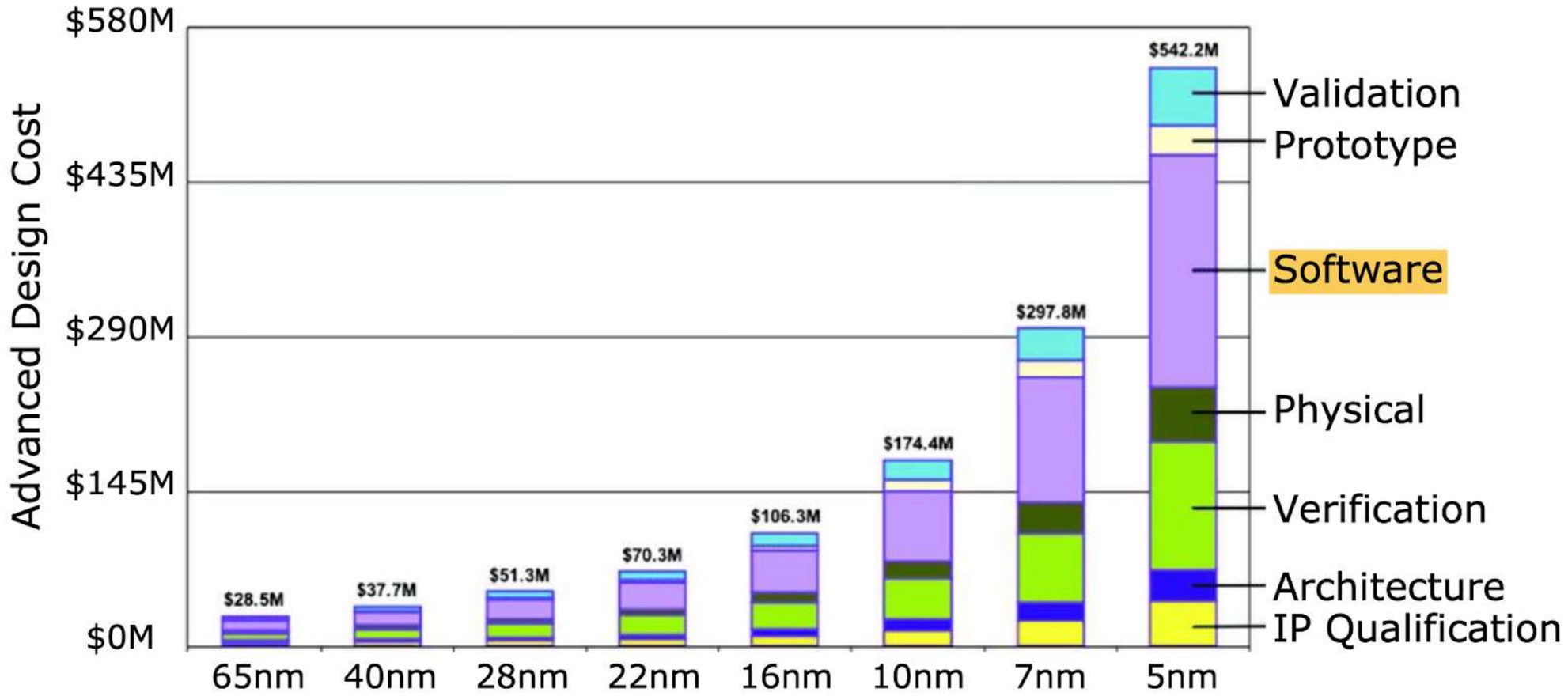


- B.S. from Tsinghua University
- Ph.D. from Stanford University, advised by Prof. Bill Dally
- Deep Compression (best paper award);
- EIE (top5 cited paper in 50 years of ISCA)

- Cofounder of DeePhi Tech (acquired by Xilinx/AMD)
- Cofounder of OmniML (acquired by NVIDIA)

- MIT Technology Review, 35 Innovators under 35
- NSF Career Award
- IEEE “AIs 10 to Watch: The Future of AI” Award
- Sloan Research Fellowship

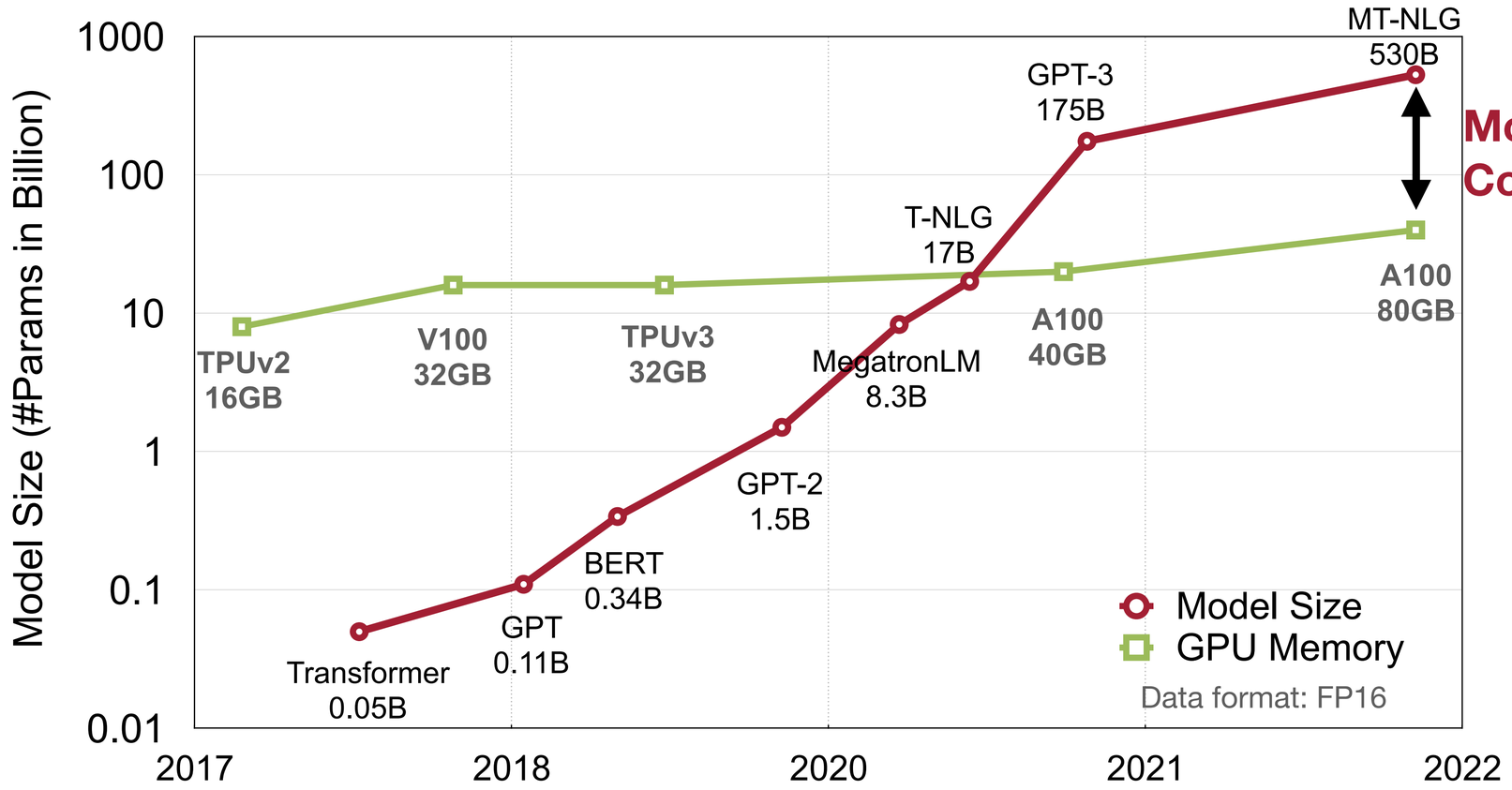
Software is Important in Advanced Technology Node



The software cost dominates the cost breakdown of advanced technology nodes [source].
We focus on designing new algorithms and software for efficient computing.

Model Compression and Co-Design are Essential

Bridge the Gap between the Supply and Demand of AI Computing

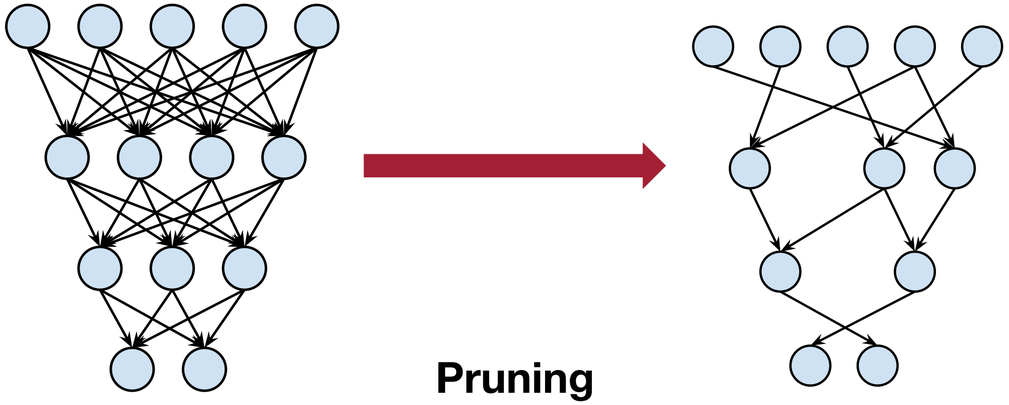


Model Compression

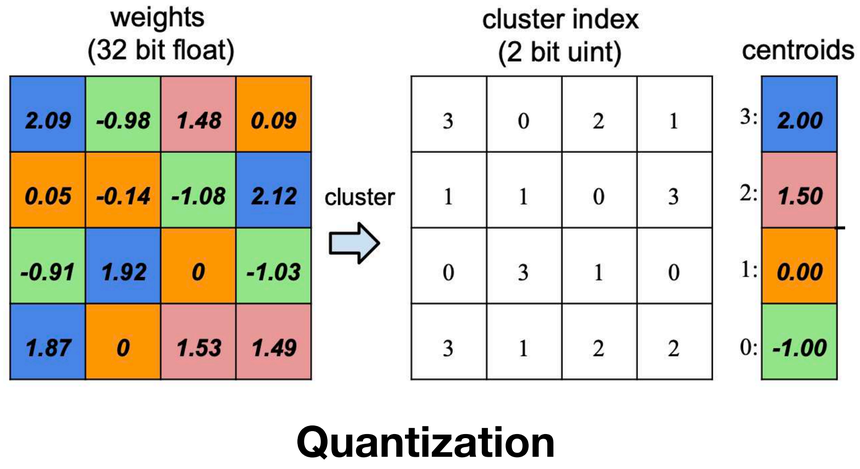
208V, 50Amp:

2x NVIDA DGX A100
or
1x NVIDA DGX H100
10KW!

Deep Compression

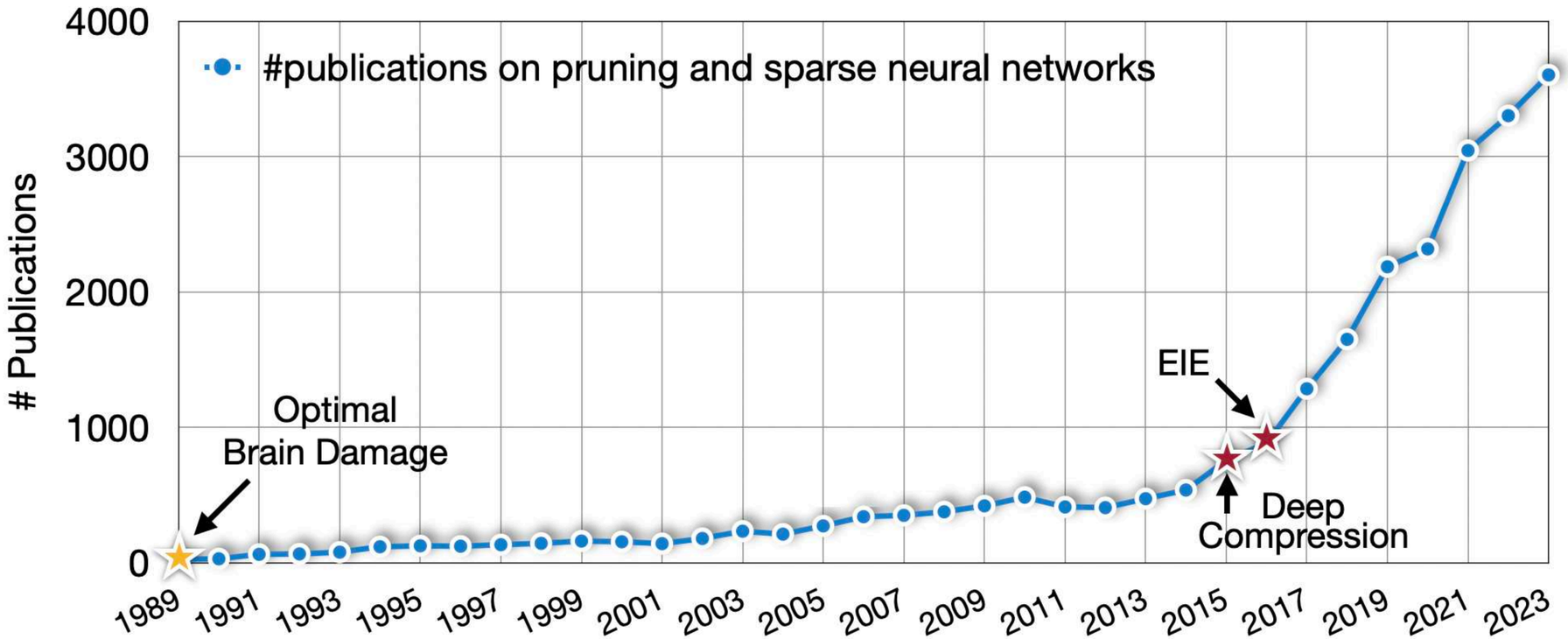


Pruning



Quantization

#publications on pruning and sparsity



The number of publications on neural network pruning and sparsity quickly increased since 2015, including both algorithms and systems.

Source: https://github.com/mit-han-lab/pruning-sparsity-publications

Top-5 most cited papers in 50 years of ISCA

Rank	*Citations*	*Year*	*Title (★ means it won the ISCA Influential Paper Award)*	First Author + *HOF Authors*	*Type*	*Topic*
1	5351	1995	The SPLASH-2 programs: Characterization and methodological considerations	Stephen Woo, *Anoop Gupta*	Tool	Benchmark
2	4214	2017	In-datacenter performance analysis of a Tensor Processing Unit	*Norm Jouppi, David Patterson*	Arch	Machine Learning
3	3834	2000	★ Wattch: A framework for architectural-level power analysis and optimizations	*David Brooks, Margaret Martonosi*	Tool	Power
4	3386	1993	★ Transactional memory: Architectural support for lock-free data structures	Maurice Herlihy	Micro	Parallelism
5	2690	2016	EIE: Efficient inference engine on compressed deep neural network	Song Han, *Bill Dally, Mark Horowitz*	Arch	Machine Learning

Efficient Large Language Models

LLM serving cost is extremely high

We're experiencing exceptionally high demand. Please hang tight as we work on scaling our systems. ×

ChatGPT is at capacity right now

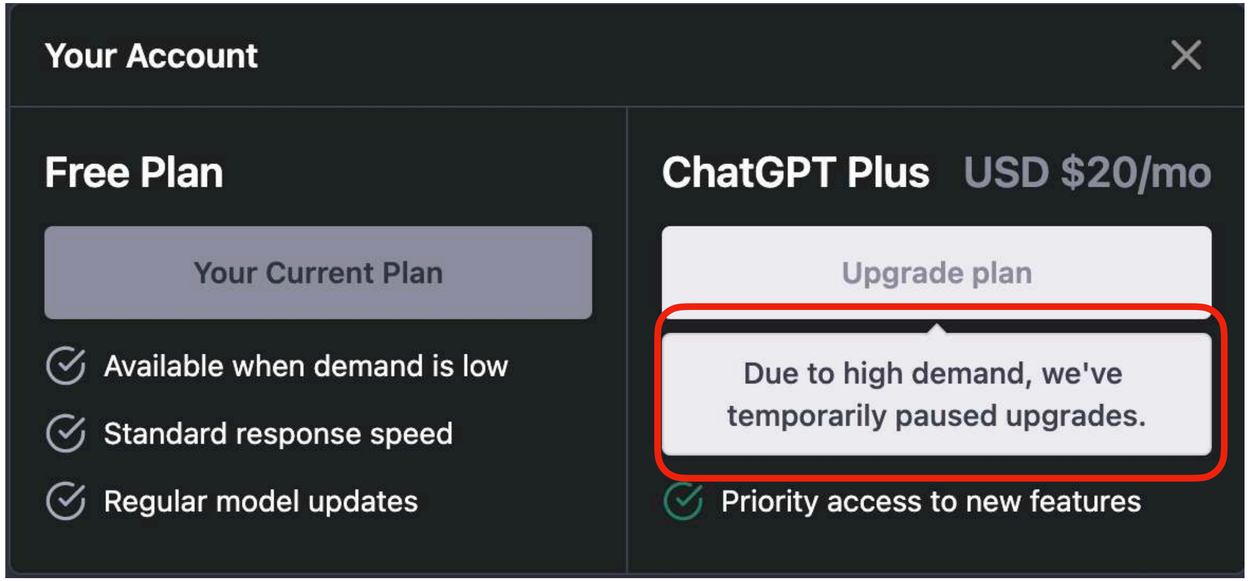
Get notified when we're back

Write a rap about the status of ChatGPT.

Yeah, yo, what's up ChatGPT fam
A lot of people trying to jam
But don't worry, we got your back
Just check back soon, we'll get on track

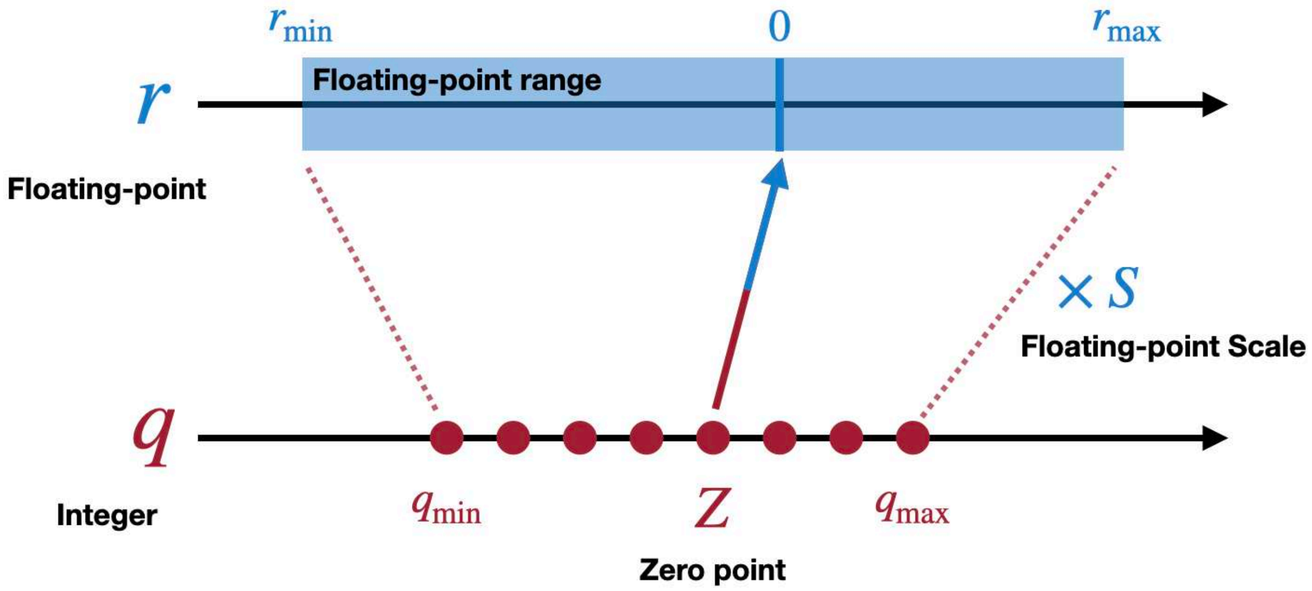
ChatGPT's the place to be
For all your AI chat needs
We're working hard to keep up the pace
So hold tight, we'll be back in this space

So don't you worry, don't be mad
We're working hard, it's not so bad
Just give us time, we'll be back
ChatGPT, the future of chat, that's a fact.



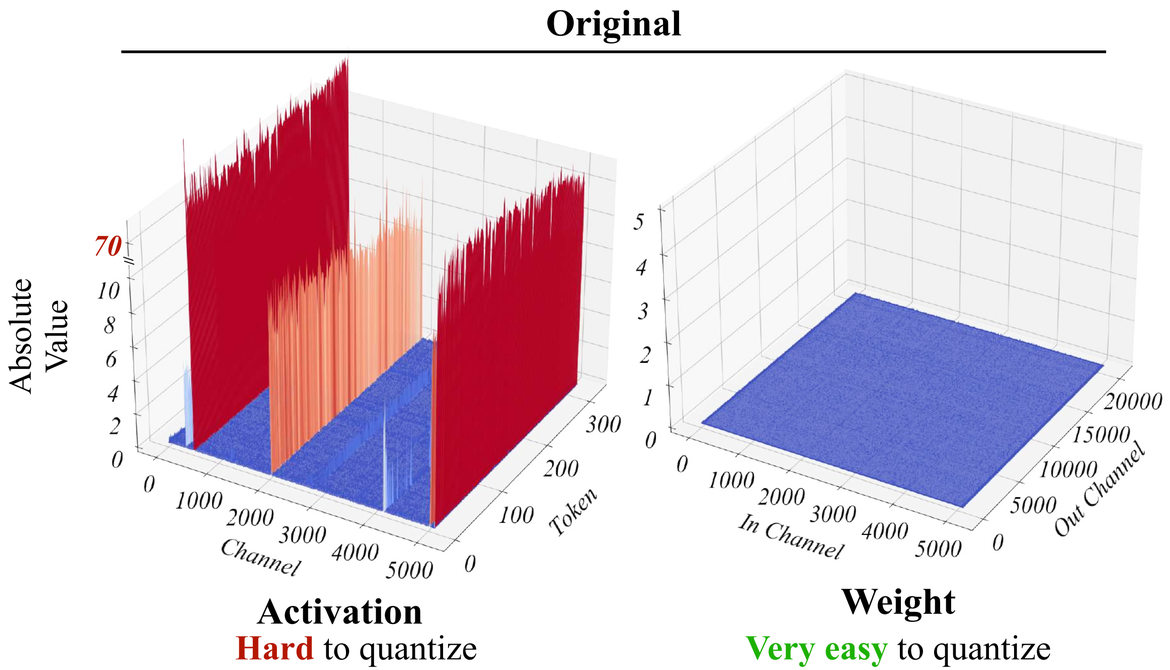
Quantization Can Reduce Deployment Costs

- Serving a 175B GPT-3 model at least requires:
  - FP16: 350GB memory ➡️ 5 x 80GB A100 GPUs
  - **INT8:** 175GB memory ➡️ **3 x 80GB A100 GPUs**



SmoothQuant

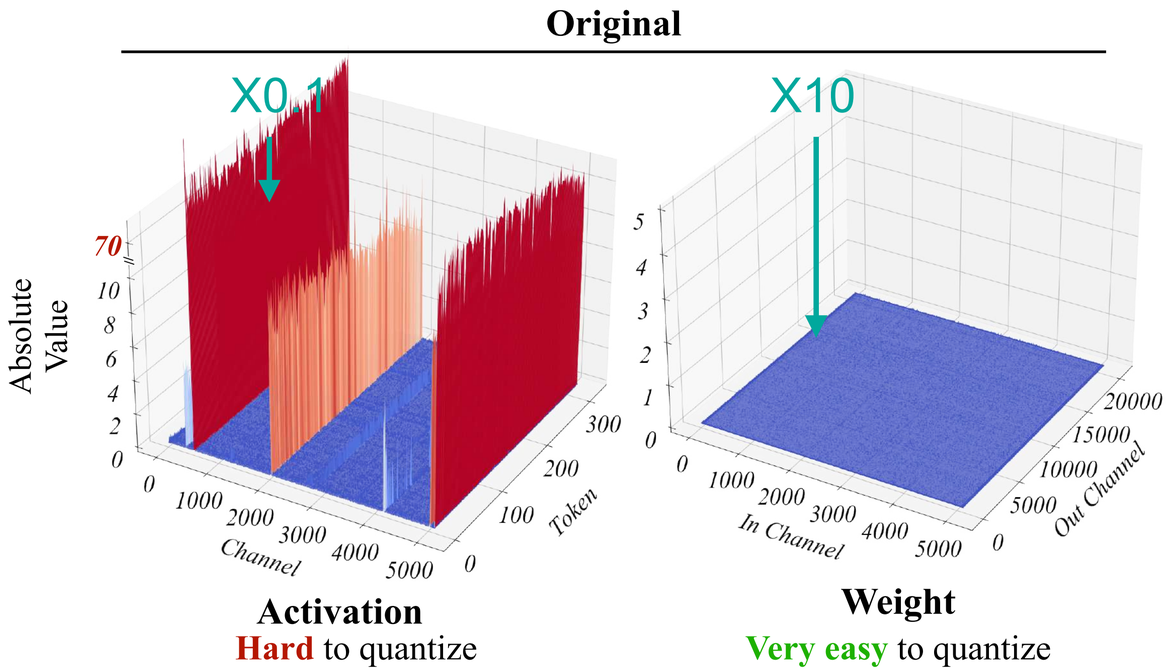
Smoothing activation to reduce quantization error



- Weights are easy to quantize, but activation is hard due to outliers
- Luckily, outliers persist in fixed channels

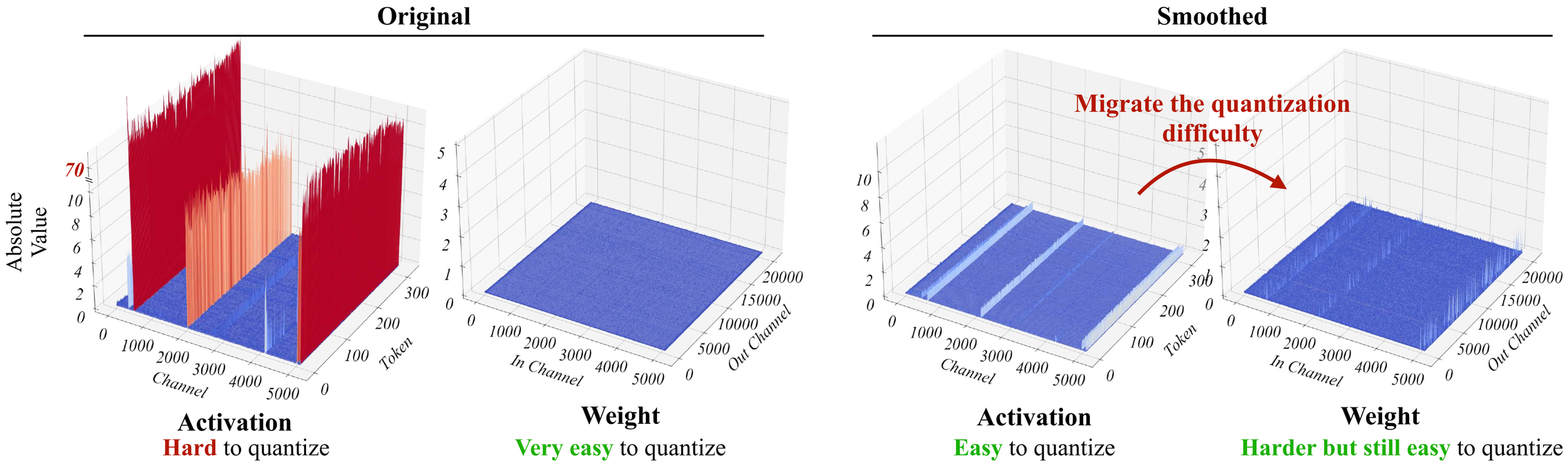
SmoothQuant

Smoothing activation to reduce quantization error



SmoothQuant

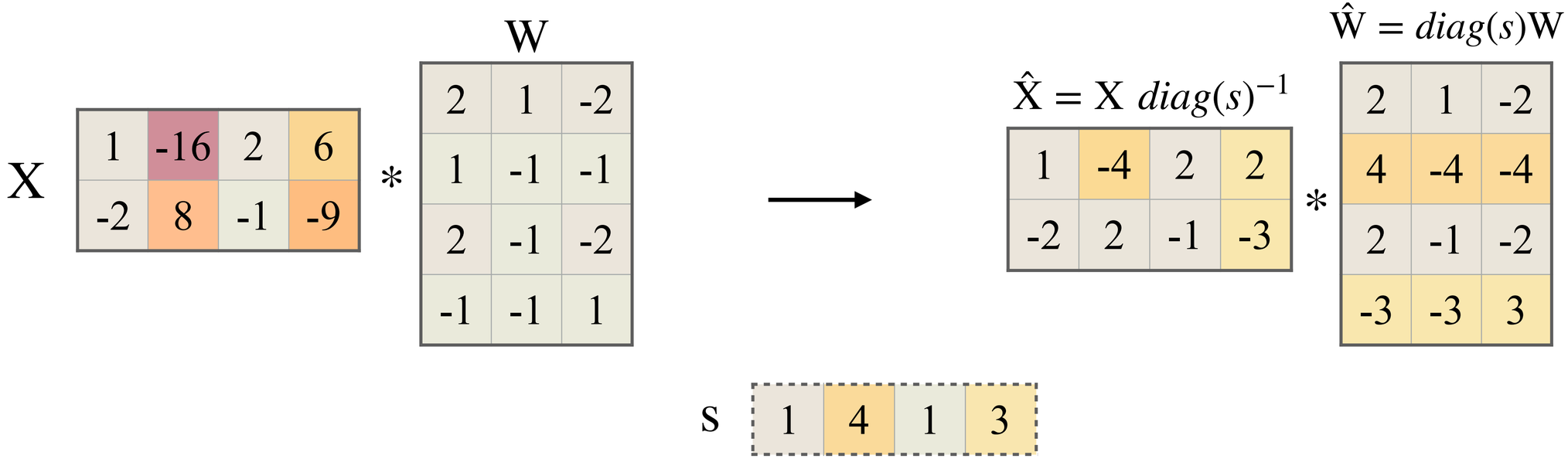
Smoothing activation to reduce quantization error



- Weights are easy to quantize, but activation is hard due to outliers
- Luckily, outliers persist in fixed channels
- Migrate the quantization difficulty from activation to weights, so both are easy to quantize

SmoothQuant

Smoothing activation to reduce quantization error



$$\mathbf{s}_j = \max(|\mathbf{X}_j|)^{\alpha} / \max(|\mathbf{W}_j|)^{1-\alpha},\ j = 1,2,\ldots,C_i$$

$$\mathbf{Y} = (\mathbf{X}diag(\mathbf{s})^{-1}) \cdot (diag(\mathbf{s})\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}}$$

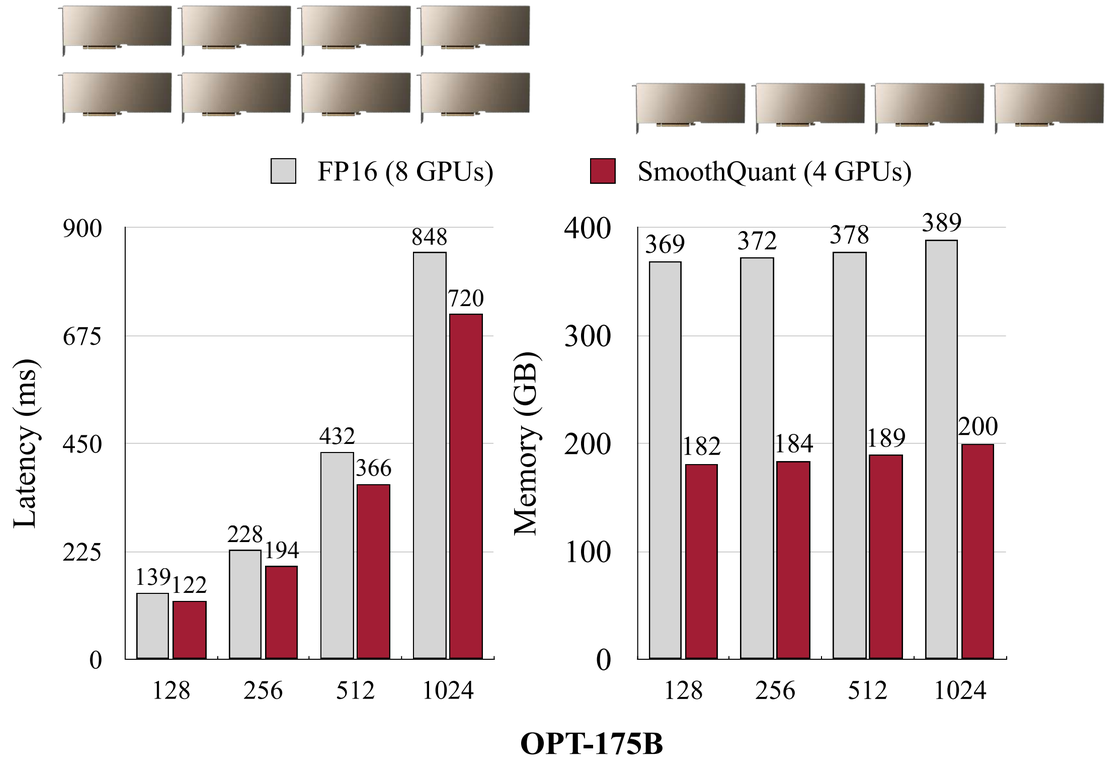
SmoothQuant (W8A8)

Accurate and efficient quantization of various LLMs

- SmoothQuant well maintains the accuracy without fine-tuning.
- SmoothQuant can both accelerate inference and halve the memory footprint.

LAMBADA Accuracy

	OPT-175B	BLOOM-176B	GLM-130B
FP16	71.6%	68.2%	73.8%
SmoothQuant	71.2%	68.3%	73.7%



SmoothQuant (W8A8)

Scaling up: W8A8 quantization of MT-NLG 530B

MT-NLG 530B Accuracy

	LAMBADA	HellaSwag	PIQA	WinoGrande	Average
FP16	76.6%	62.1%	81.0%	72.9%	73.1%
INT8	77.2%	60.4%	80.7%	74.1%	73.1%

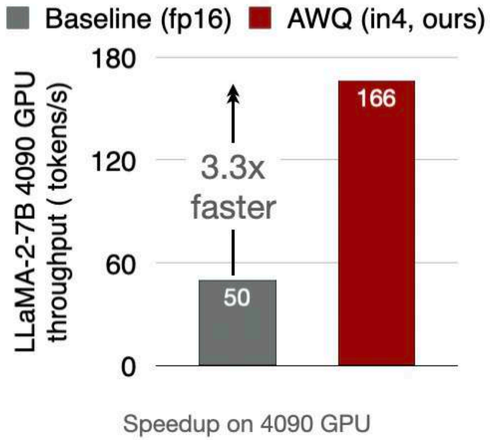
MT-NLG 530B Efficiency

SeqLen	Prec.	#GPUs	Latency	Memory
512	FP16	16	838ms	1068GB
	INT8	8	839ms	545GB
1024	FP16	16	1707ms	1095GB
	INT8	8	1689ms	570GB

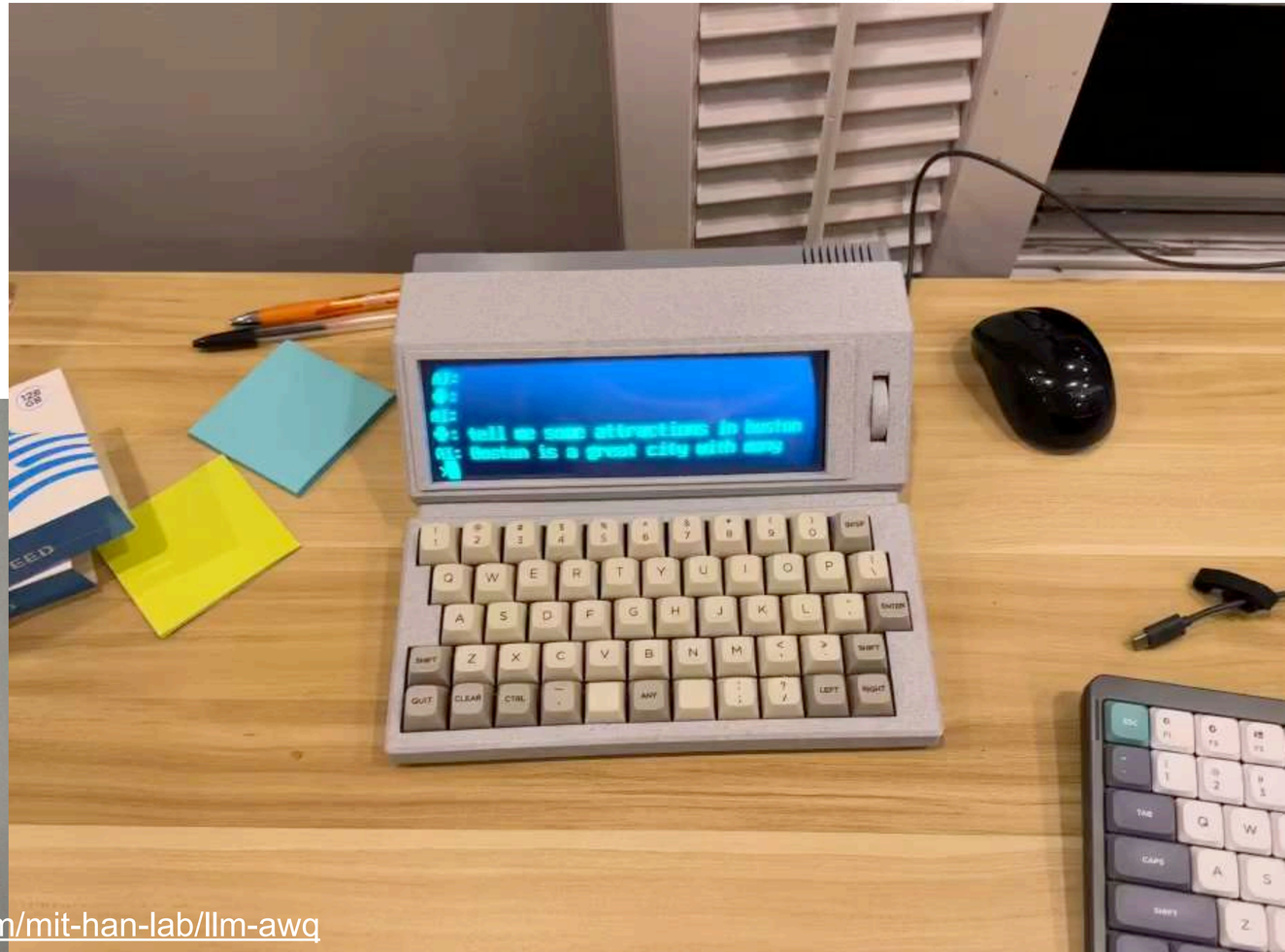
- SmoothQuant can accurately quantize MT-NLG 530B model and reduce the serving GPU numbers by half at a similar latency, which allows serving the 530B model within a single node.

TinyChat

- A LightWeight Chatbot for LLMs on the edge



- Deploying LLM on the edge is useful: running copilot services (code completion, office, game chat) locally on laptops, cars, robots, and more. These devices are **resource-constrained**, **low-power** and sometimes **do not have access to the Internet**.
- **Data privacy** is important. Users do not want to share personal data with large companies.



https://github.com/mit-han-lab/llm-awq

AWQ: 4bit Activation-aware Weight Quantization

Observation: Weights are not equally important; 0.1% salient weights

$\mathbf{W}_{\text{FP16}}$

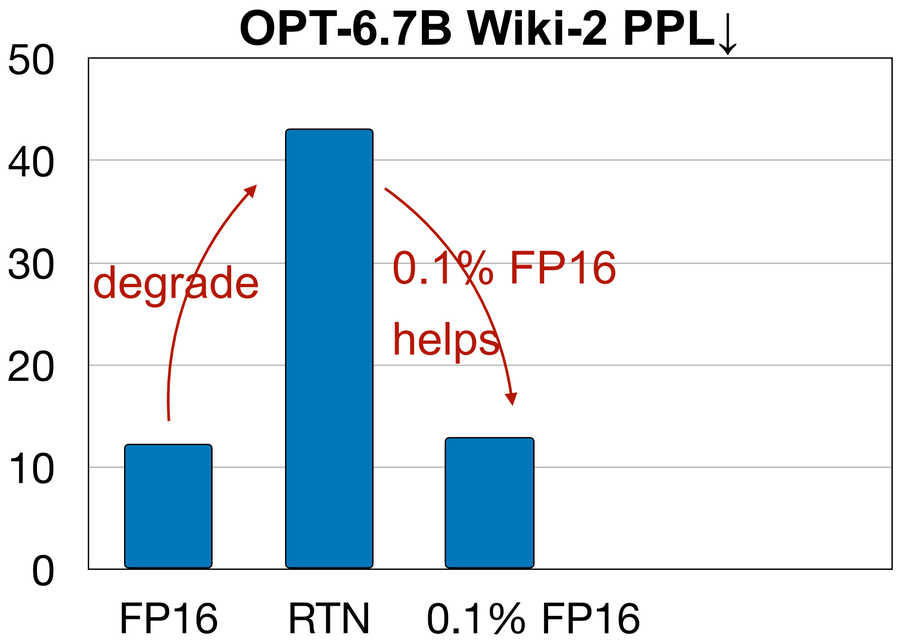
+1.2	−0.2	−2.4	−3.4
−2.5	−3.5	+1.9	+1.4
−0.9	+1.6	−2.5	−1.9
−3.5	+1.5	+0.5	−0.1
+1.8	−1.6	−3.2	−3.4
+2.4	−3.5	−2.8	−3.9
+0.1	−3.8	+2.4	+3.4
+0.9	+3.3	−1.9	−2.3

RTN →

$Q(\mathbf{W})_{\text{MixPrec}}$

+1	+0	−2	−3
−2.5	−3.5	+1.9	+1.4
−1	+2	−3	−2
−4	+2	+1	+0
+2	−2	−3	−3
+2	−4	−3	−4
+0	−4	+2	+3
+1	+3	−2	−2

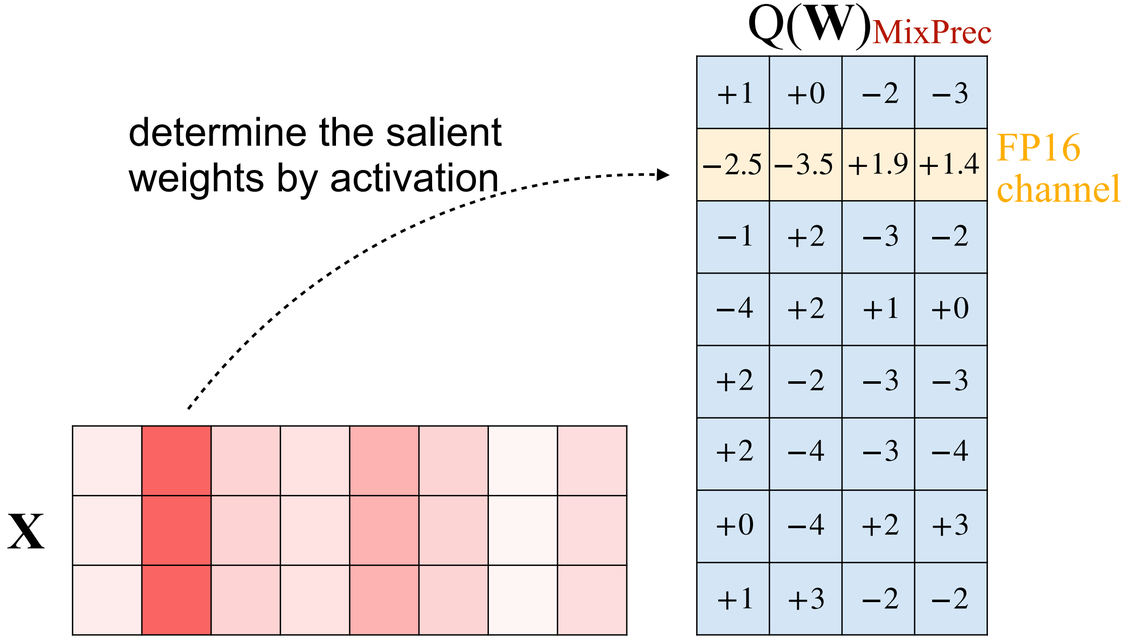
FP16 channel

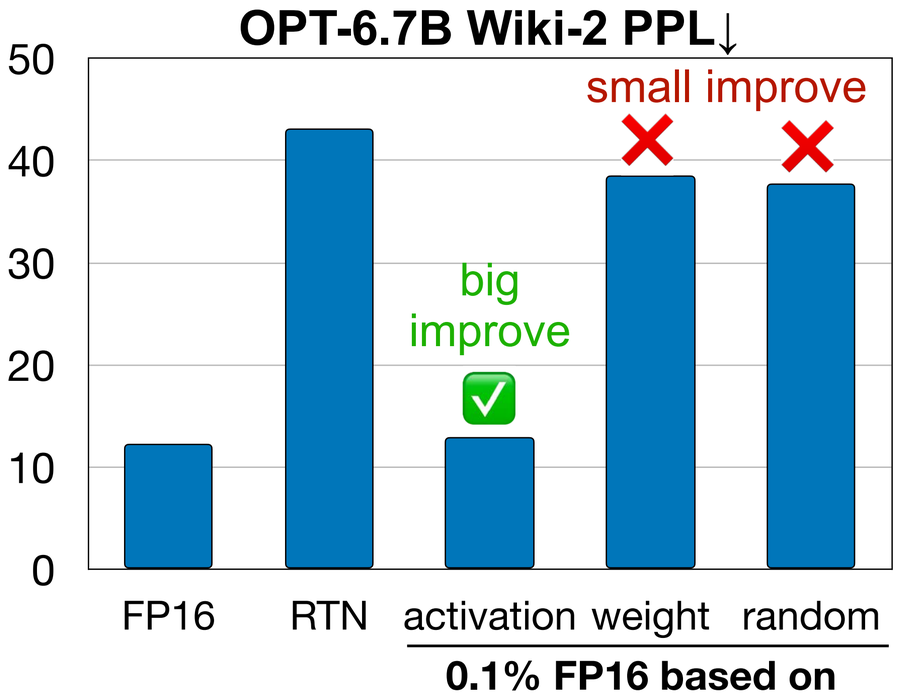


- We find that weights are not equally important, keeping **only 0.1%** of salient weight channels in FP16 can greatly improve perplexity
- But how do we select salient channels? Should we select based on weight magnitude?

AWQ: 4bit Activation-aware Weight Quantization

Salient weights are determined by activation distribution, not weight





- We find that weights are not equally important, keeping **only 0.1%** of salient weight channels in FP16 can greatly improve perplexity
- But how do we select salient channels? Should we select based on weight magnitude?
- No! We should look for **activation distribution, but not weight!**

AWQ: 4bit Activation-aware Weight Quantization

Protecting salient weights by scaling (no mixed prec.)

$$Q(\mathbf{w}) \cdot x = \Delta \cdot \text{Round}(\frac{\mathbf{w}}{\Delta}) \cdot x, \quad \Delta = \frac{\max(|\mathbf{w}|)}{2^{N-1}}$$

$$Q(\mathbf{w} \cdot s)(x/s) = \Delta' \cdot \text{Round}(\mathbf{w}s/\Delta') \cdot x \cdot \frac{1}{s}$$

constant E(error) suppress error

$$\Delta' \approx \Delta, \quad \text{RoundErr} \sim 0.25, \quad s > 1$$

$$\frac{\text{Err}(Q(\mathbf{w} \cdot s)(x/s))}{\text{Err}(Q(\mathbf{w}) \cdot x)} = \frac{\Delta'}{\Delta} \cdot \frac{1}{s} < 1$$

$$Q(\quad \mathbf{W} \quad)$$

+1.2	−0.2	−2.4	−3.4	× 1
−2.5	−3.5	+1.9	+1.4	× 2
−0.9	+1.6	−2.5	−1.9	× 1
−3.5	+1.5	+0.5	−0.1	× 1
+1.8	−1.6	−3.2	−3.4	× 1
+2.4	−3.5	−2.8	−3.9	× 1
+0.1	−3.8	+2.4	+3.4	× 1
+0.9	+3.3	−1.9	−2.3	× 1

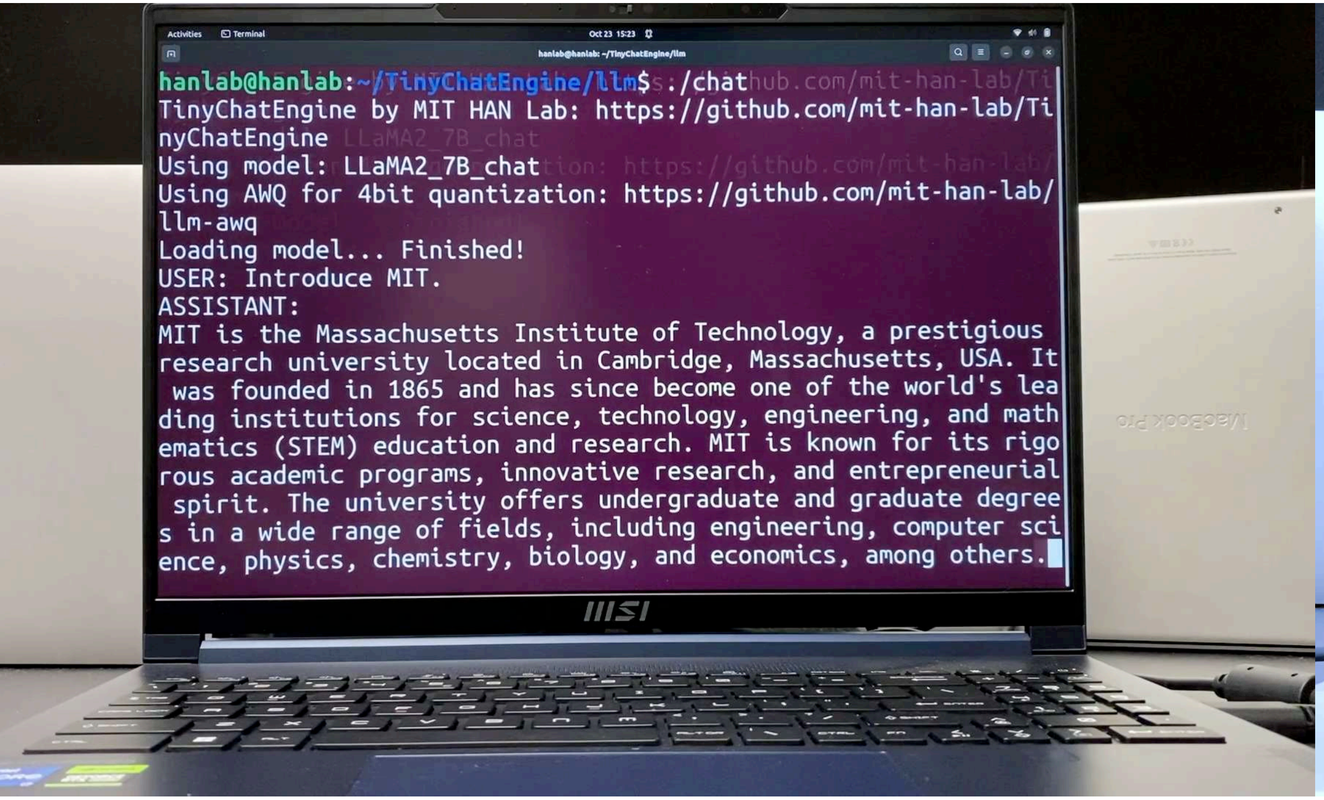
fuse to previous op

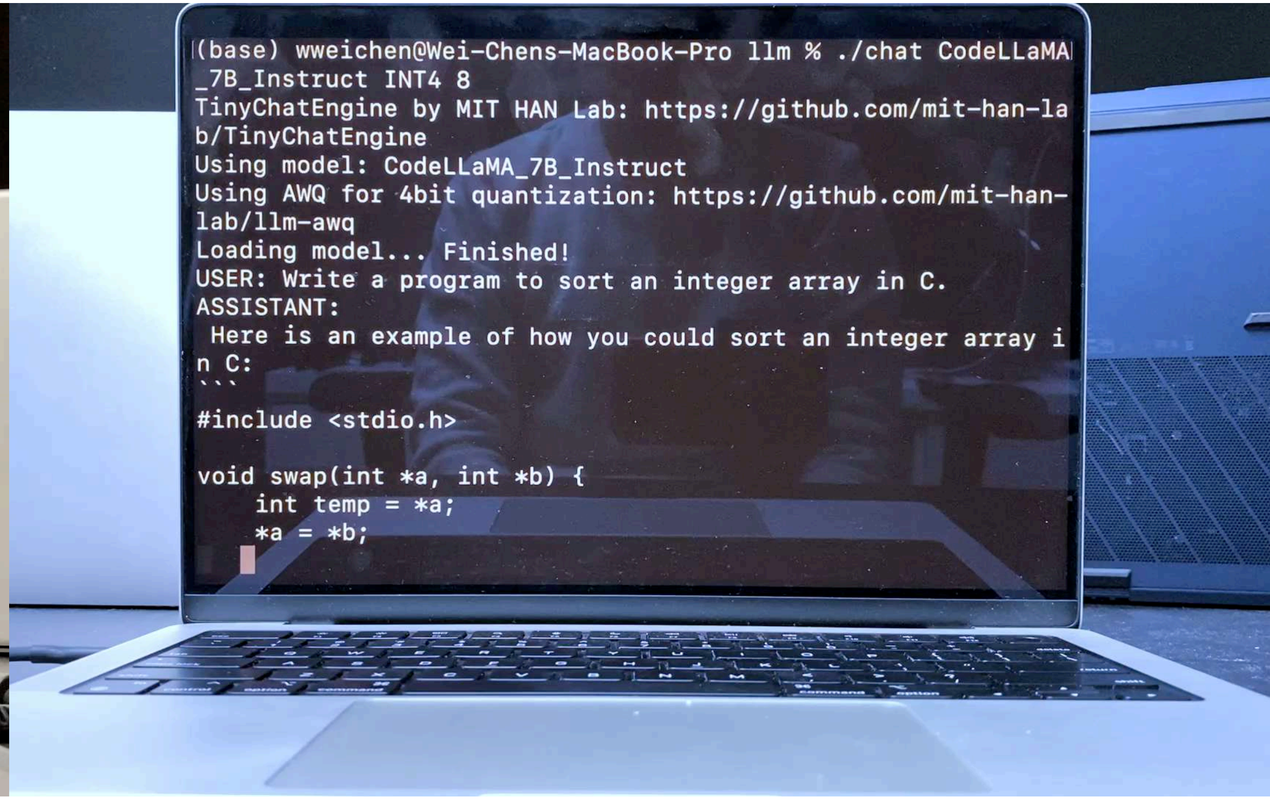
$$\mathbf{WX} \longrightarrow Q(\mathbf{W} \cdot \mathbf{s})(\mathbf{s}^{-1} \cdot \mathbf{X})$$

- Multiplying the salient channels with $s > 1$ reduces its quantization error
- Skip mathematical derivation for now

TinyChat

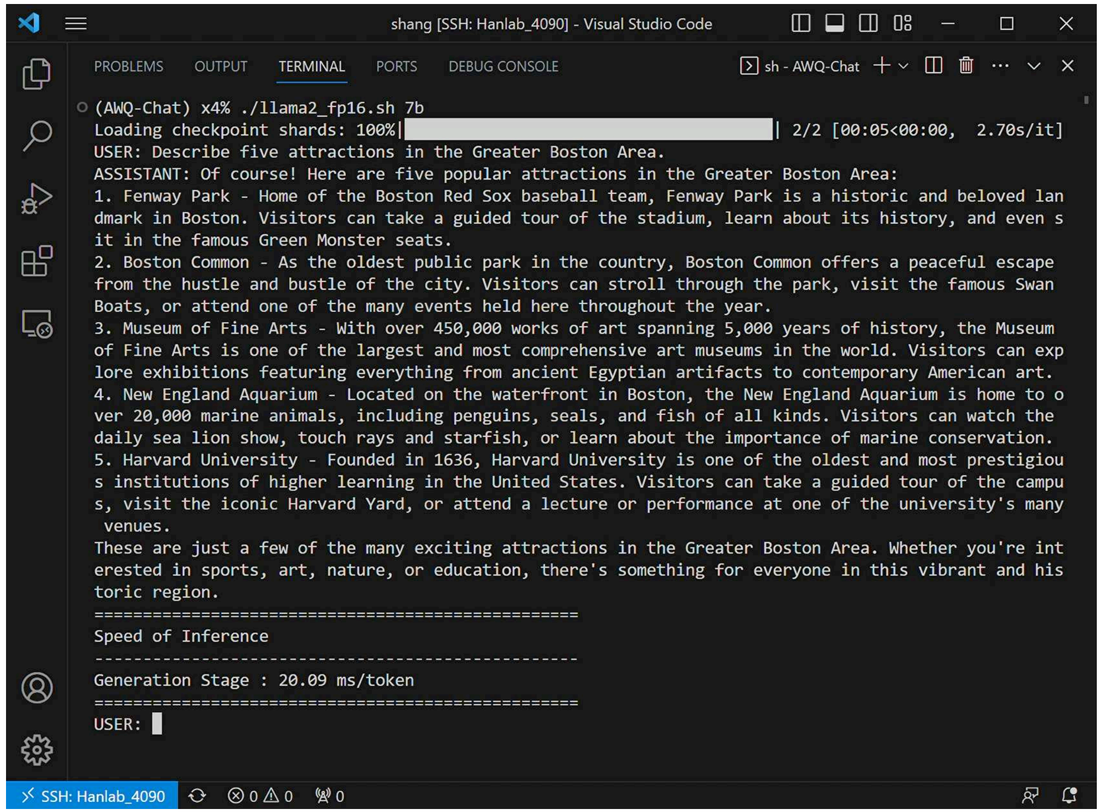
- TinyChatEngine implements the compressed AWQ 4bit model, built from C/C++ from scratch, easy to install and migrate to edge platforms
- Enables on-device LLM

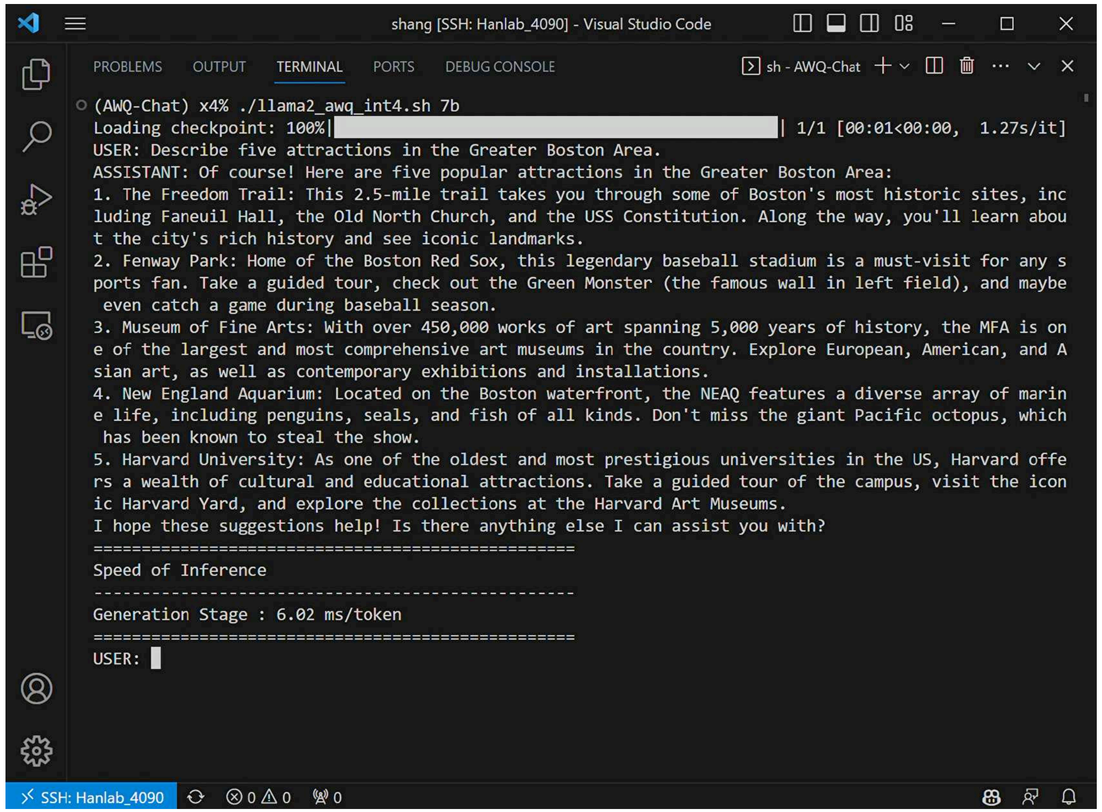




TinyChat

TinyChat brings about 3.3x speedup to LLaMA-2 on 4090





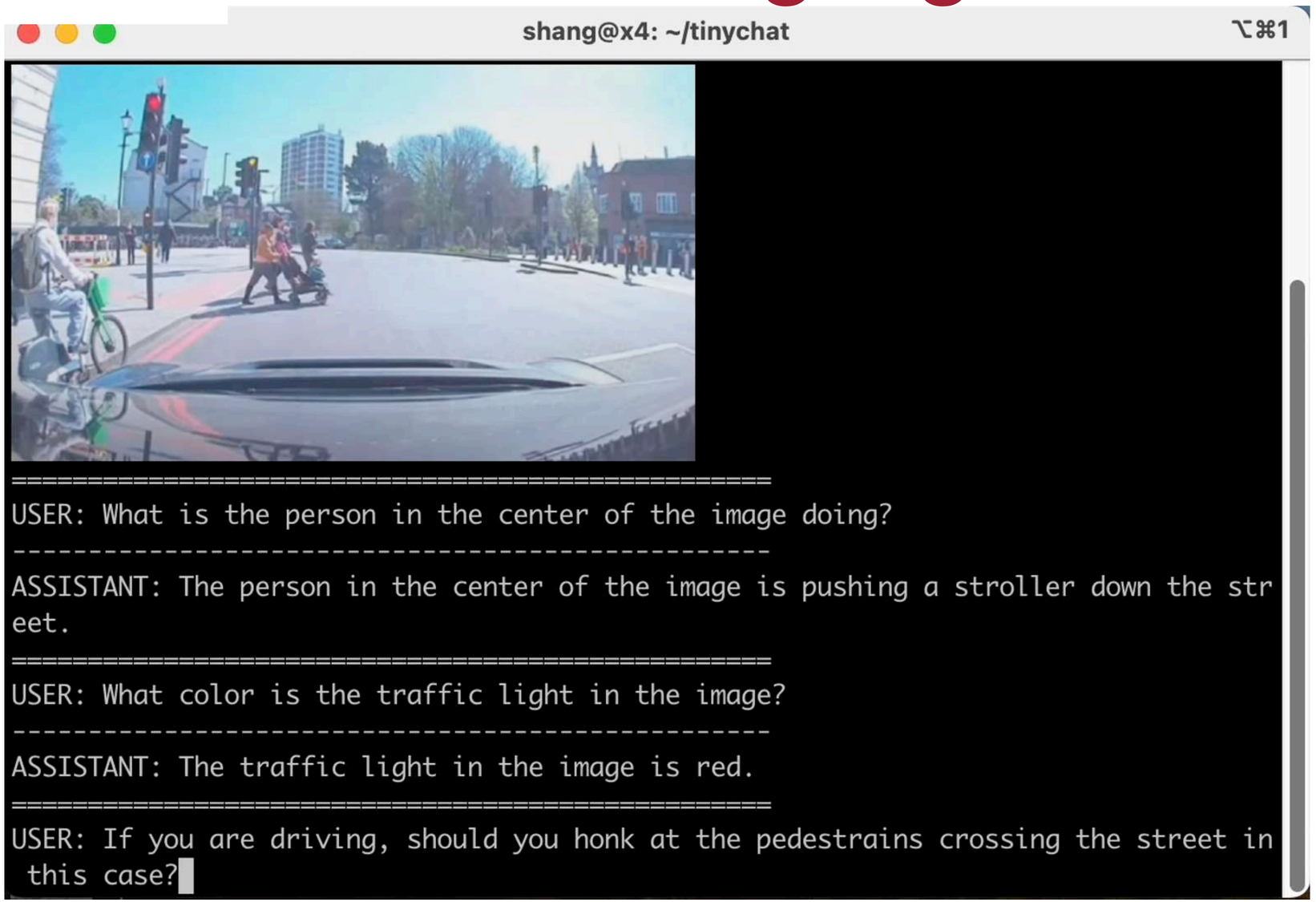
LLaMA-2-7B (FP16): 50 tokens / s

Baseline: fp16 weight, fp16 activation

LLaMA-2-7B (W4A16, AWQ): 166 tokens / s

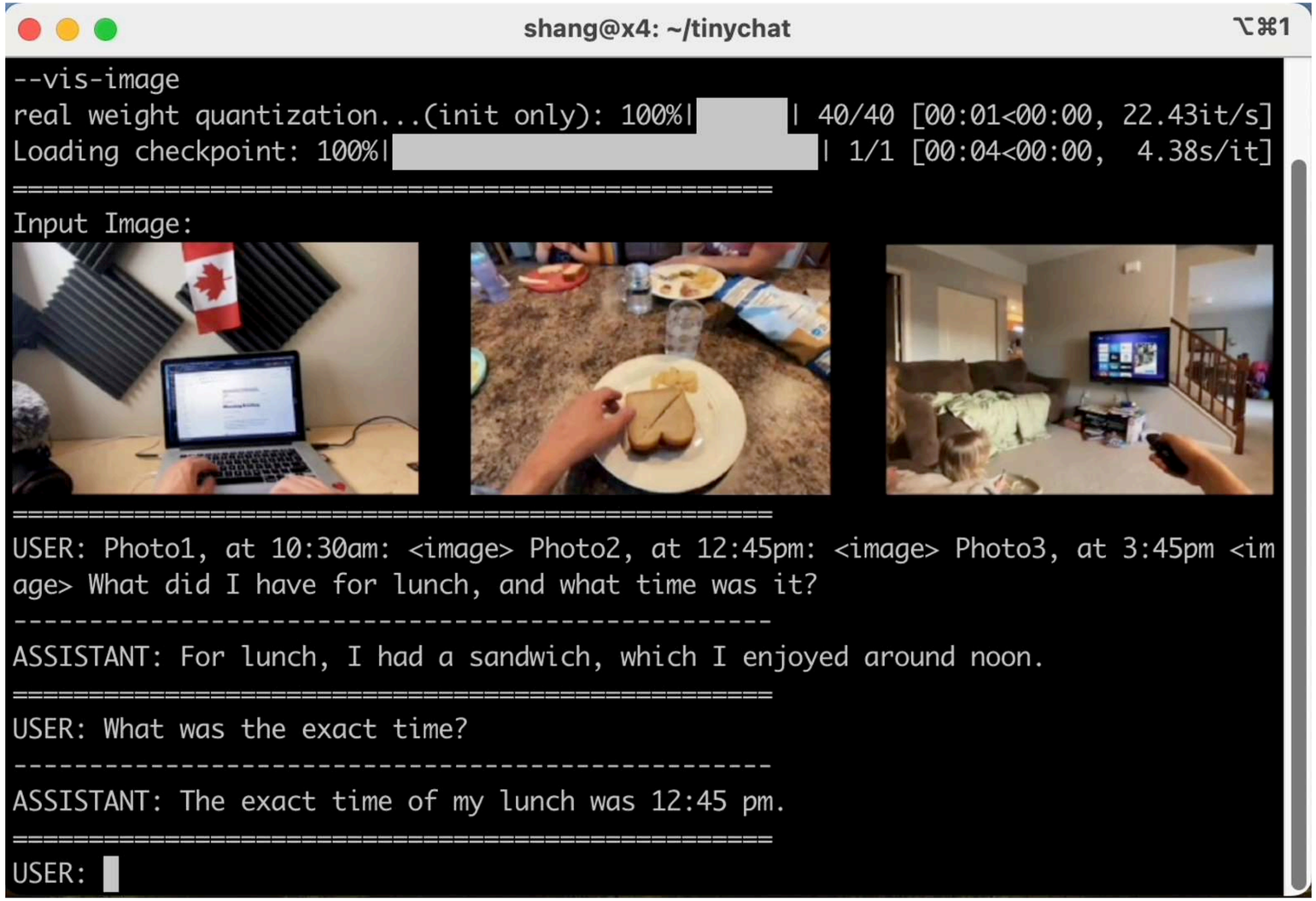
AWQ: int4 weight, fp16 activation

TinyChat for visual language model



Single image - Multi-round QA **VILA-13B + AWQ**: 100 tokens/s on 4090

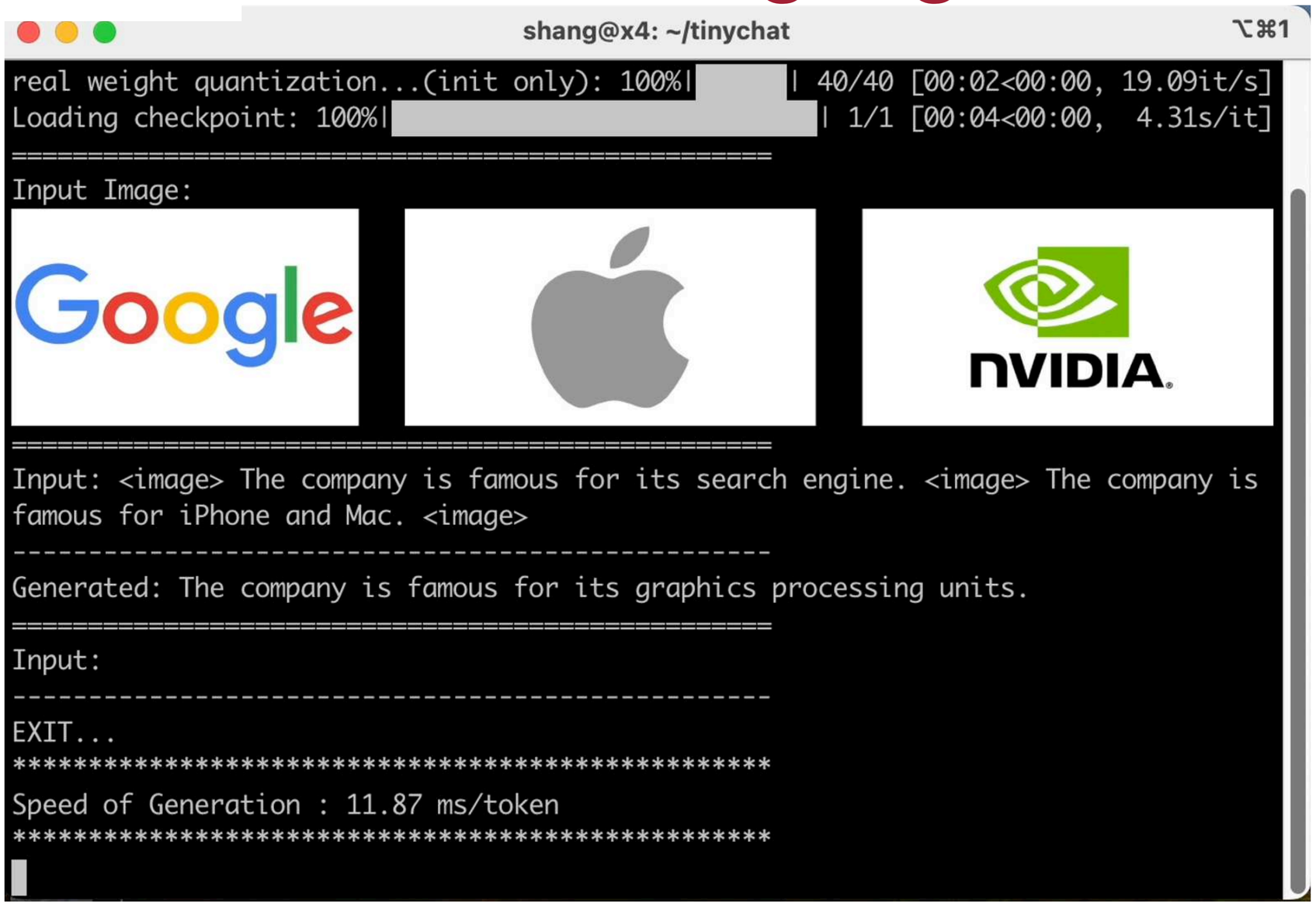
TinyChat for visual language model



Multi-image - Multi-round QA

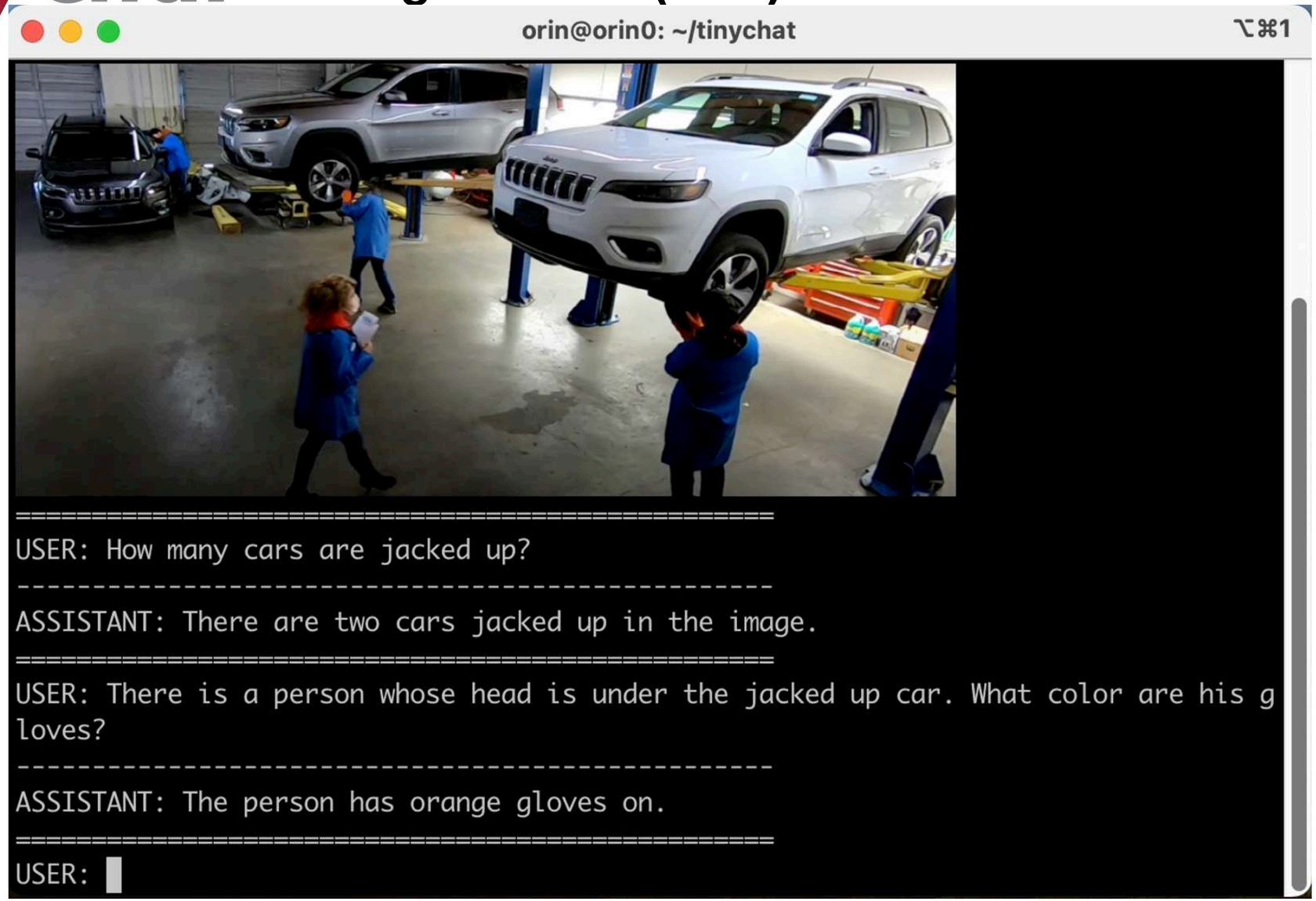
VILA-13B + AWQ: 83 tokens/s (3 image inputs) on 4090

TinyChat for visual language model



VILA-13B + AWQ: 84 tokens / s on 4090

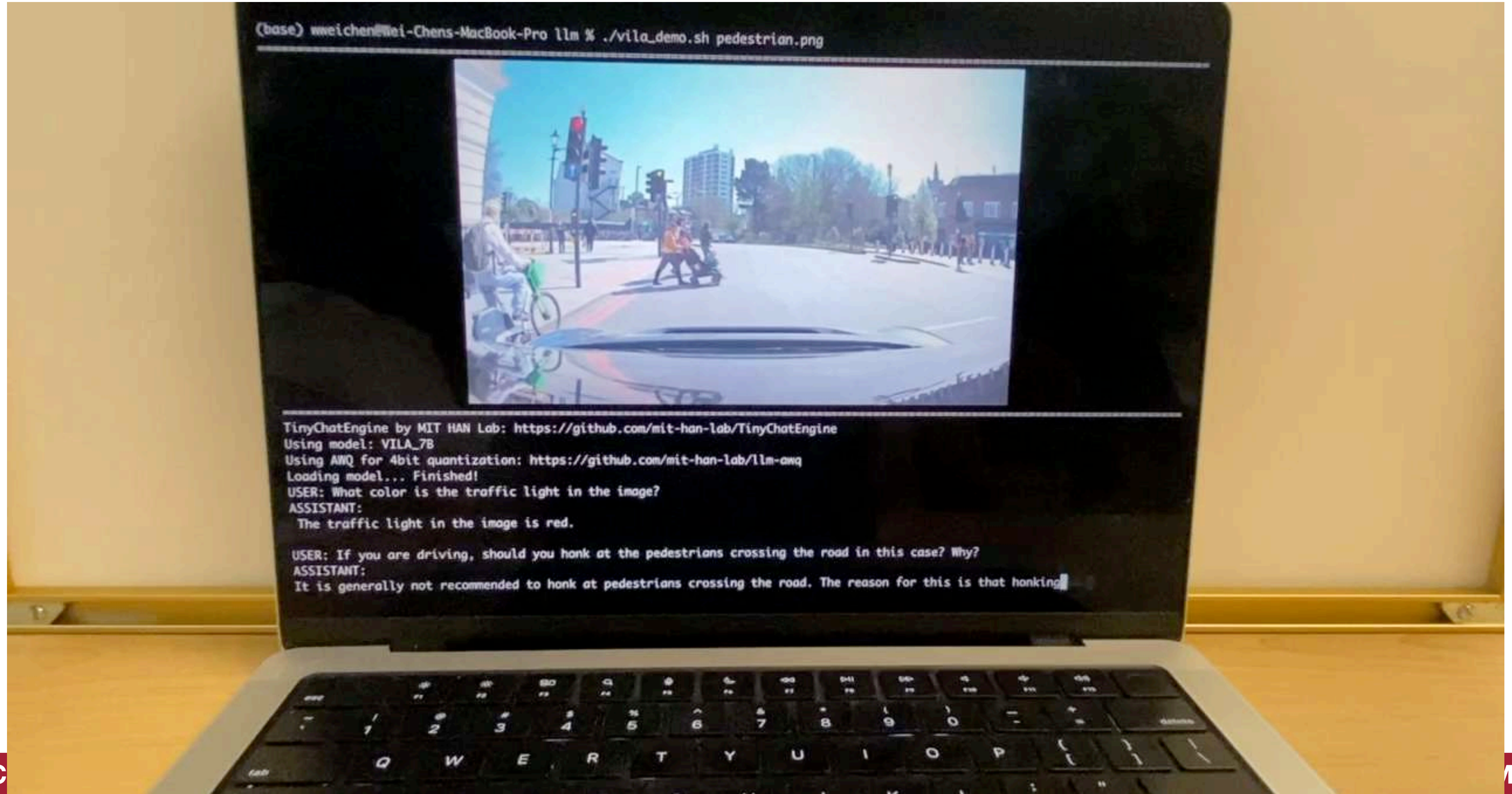
TinyChat On Edge Device (Orin)



Single image - Multi-round QA **VILA-7B + AWQ**: 29 tokens / s

TinyChat

- TinyChatEngine implements the compressed AWQ 4bit VILA model, built from C/C++ from scratch, easy to install and migrate to edge platforms, enables on-device VLM



Impact of SmoothQuant and AWQ

TensorRT-LLM

https://github.com/NVIDIA/TensorRT-LLM#key-features

Neural Compressor

Q8-Chat

https://github.com/intel/neural-compressor/blob/master/docs/source/smooth_quant.md

https://github.com/vllm-project/vllm/blob/main/vllm/model_executor/layers/quantization/awq.py

lm-sys/FastChat

https://github.com/lm-sys/FastChat/blob/main/docs/awq.md

Granite

IBM's internal code model, Granite, utilizes AWQ for quantization.

text-generation-inference

https://github.com/huggingface/text-generation-inference/tree/main/server/text_generation_server/utils/awq/quantize

lmdeploy

https://github.com/InternLM/lmdeploy/blob/main/lmdeploy/lite/quantization/awq.py

https://friendli.ai/blog/Unlocking-Efficiency-of-Serving-LLMs-with-Activation-aware-Weight-Quantization-AWQ-on-PeriFlow/

https://github.com/replicate/vllm-with-loras/blob/main/vllm/model_executor/quantization_utils/awq.py

StreamingLLM

[StreamingLLM, ICLR'24]

Enable long conversations in non-stop streaming applications

- There's need for LLMs in streaming applications such as multi-round dialogues, where long, non-stop interactions are needed.
- Challenge: extensive memory consumption when conversation gets long; perplexity explodes after sequence length exceeds threshold.

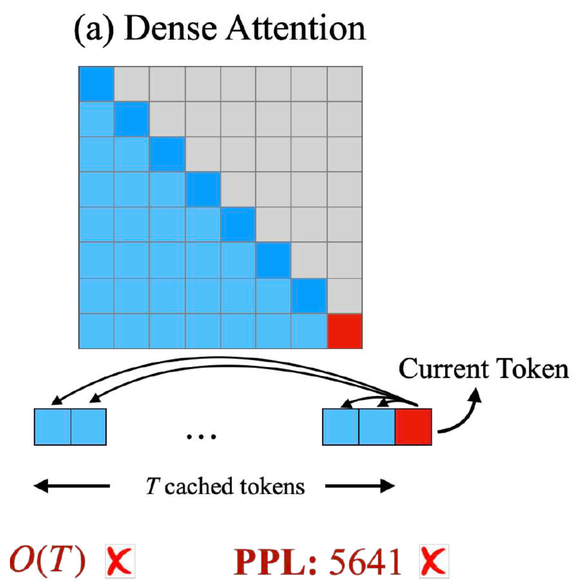
Without StreamingLLM, model collapses as the conversation gets long, then it runs out of memory.

With StreamingLLM, non-stop conversation continues.

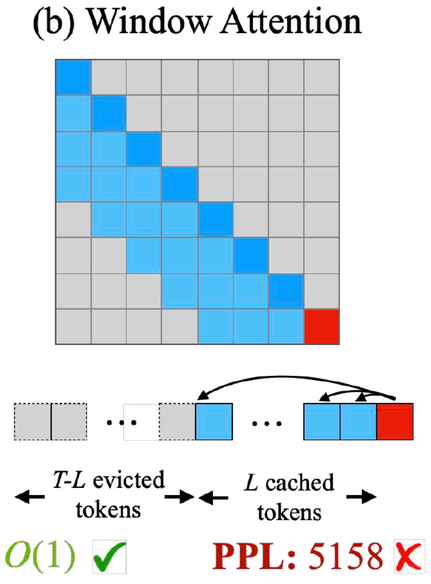
StreamingLLM

[StreamingLLM, ICLR'24]

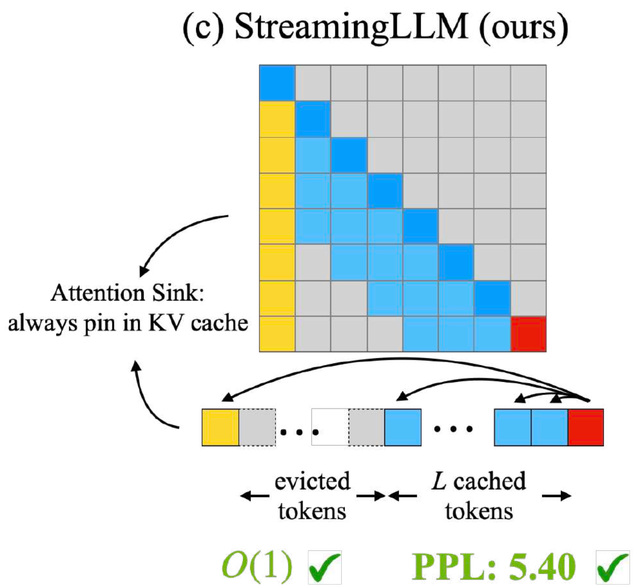
Enable long conversations in non-stop streaming applications



KV cache size grows linearly with the sequence length; perplexity explodes after exceeding the max context length (4K for llama2-7B).



KV cache size is constant; but perplexity explodes after sequence length exceeds the KV cache size (when first token is evicted).



KV cache size is constant; perplexity doesn't explode.

StreamingLLM with “Attention Sink”

[StreamingLLM, ICLR’24]

The first token is always heavily attended, we call it “attention sink”

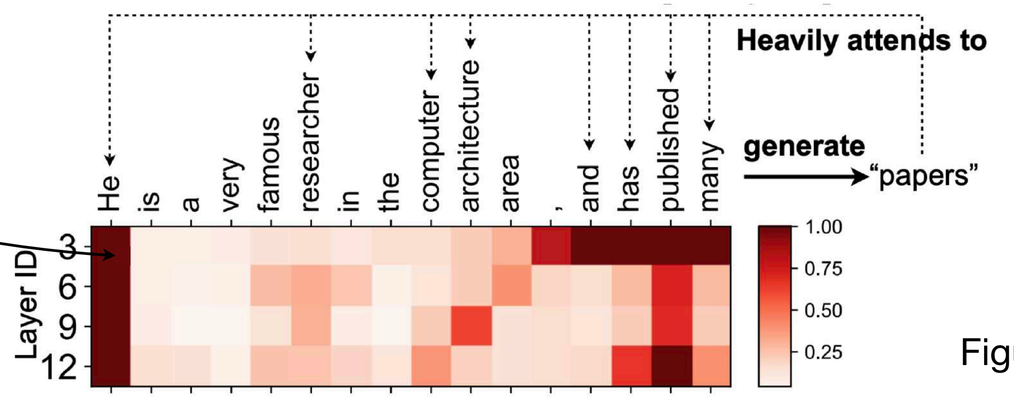


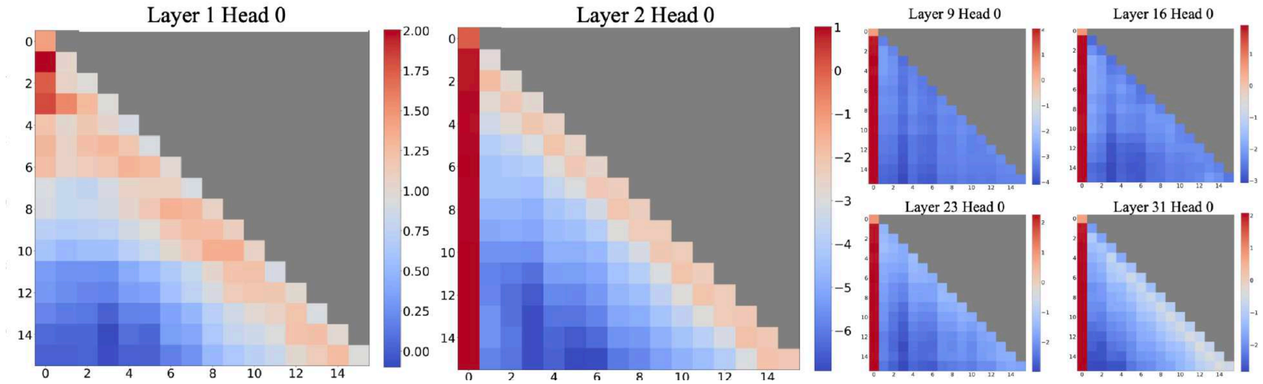
Figure: SpAtten, HPCA’21

GPT-2 for Language Modeling

```
Du Fu was a great poet of the Tang dynasty. Recently a variety of styles have been used in efforts to translate the work of Du Fu into English
Du ~~Fu was a great~~ poet ~~of the~~ Tang dynasty. Recently ~~a variety of~~ styles ~~have been used~~ in efforts to translate the work of Du Fu into English
Du ~~Fu was a great poet of the Tang dynasty. Recently a variety of styles have been used in efforts to~~ translate ~~the work of Du Fu~~ into English
```

‘***English***’ is the generated token.

attention sink can not be pruned.



Beyond layer 1, the model heavily attends to the initial token.

Why “attention sink” exist?

- SoftMax makes sure attention scores sum up to one for all contextual tokens, even not relevant, needs to dump it somewhere => attention sink.
- Why first token? Globally visible.

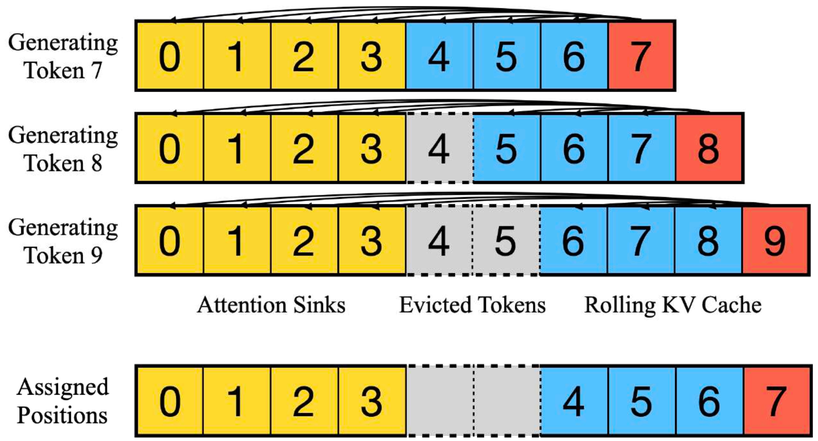
$$\text{SoftMax}(x)_i = \frac{e^{x_i}}{e^{x_1} + \sum_{j=2}^{N} e^{x_j}}, \quad x_1 \gg x_j, j \in 2, \ldots, N$$

- What should we do with attention sink? Always keep the attention sink tokens in the KV cache.

StreamingLLM

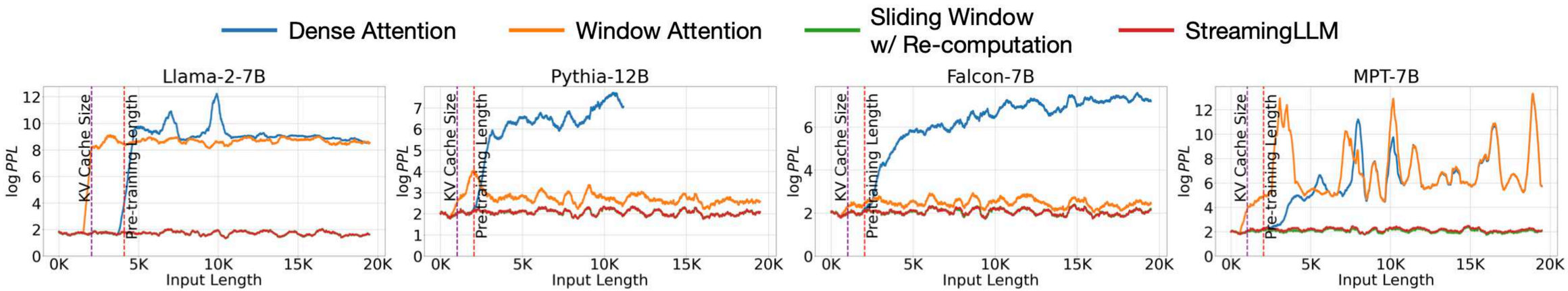
[StreamingLLM, ICLR'24]

Implementation and Results



- Recipe:
  - Always keep the attention sink tokens in the KV cache.
  - Use a sliding window KV to stabilize the model's behavior.
  - Use positions ***in the cache*** instead of those ***in the original text***.

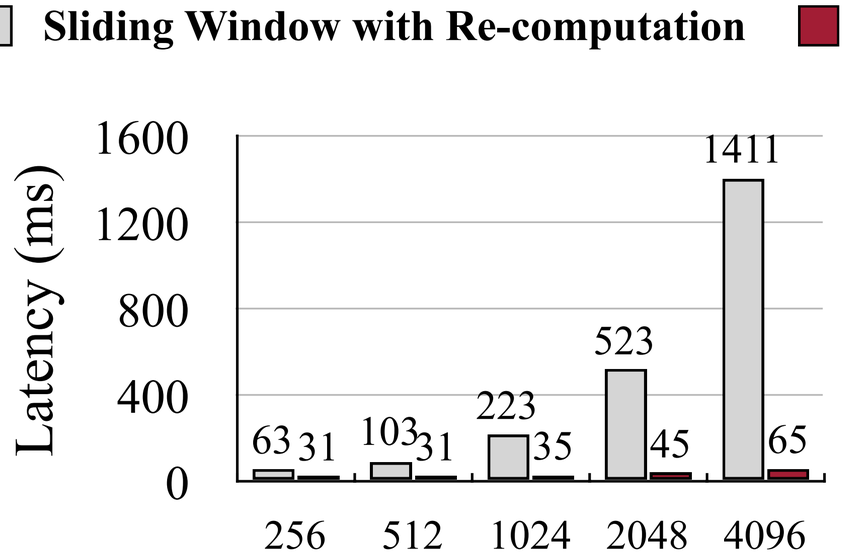
- Results:
  - Dense attention fails beyond pre-training attention window size.
  - Window attention fails after input exceeds cache size (initial tokens evicted).
  - StreamingLLM demonstrates no perplexity explosion, tested up to 4M tokens.

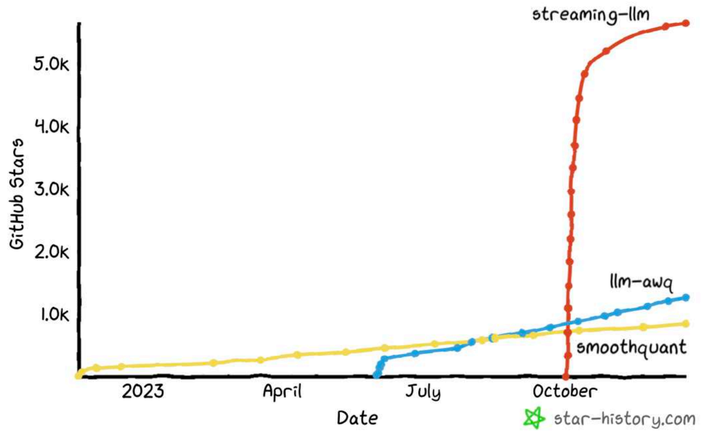


StreamingLLM

[StreamingLLM, ICLR'24]

Enable long conversations in non-stop streaming applications





- StreamingLLM accelerates Llama2-7B compared to sliding window with recomputation.
- The more you generate, the more you save!
- StreamingLLM is available in NVIDIA TRT-LLM

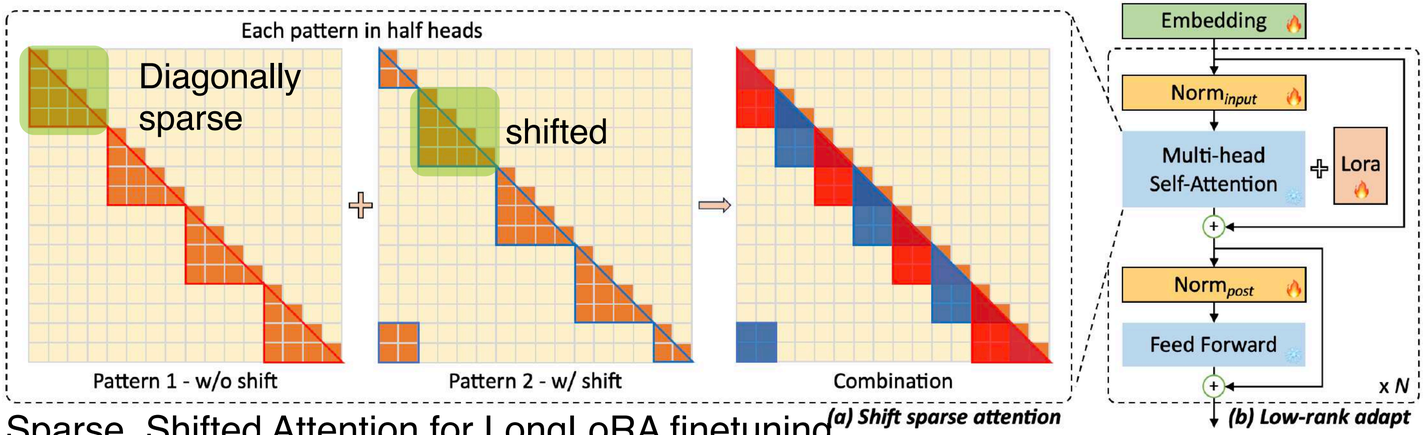
StreamingLLM on iPhone:

LongLoRA

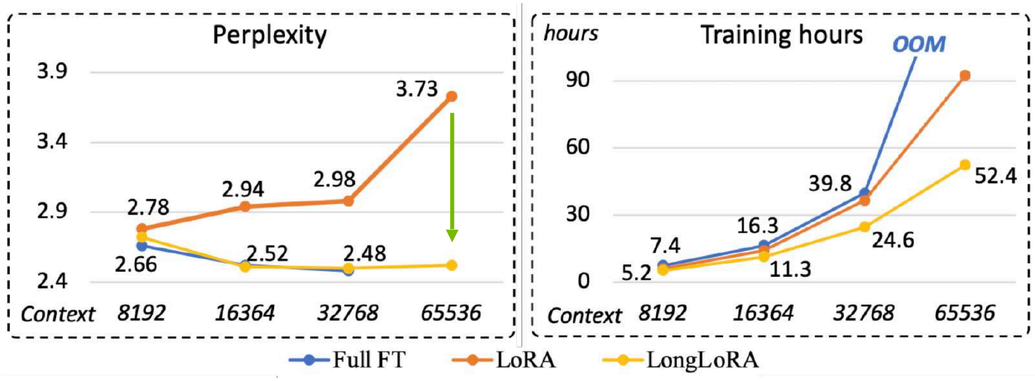
[LongLoRA, ICLR'24, **Oral**]

Increase the context length with low fine-tuning cost (on a single node)

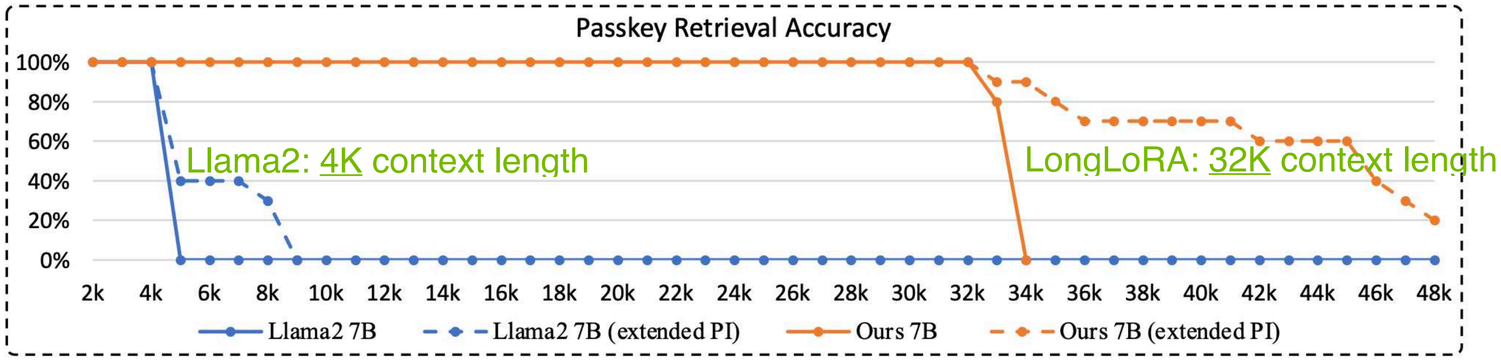
- LongLoRA invented "shifted, sparse local attention" to enable longer context length at low finetuning cost.
- LongLoRA efficiently extends the context length of Llama2-7B from 4k to 100k, Llama2-70B to 32k on a single 8× A100 machine.
- LongLoRA (for finetunining) and StreamingLLM (for inference) work together to increase both context length and generation length.



Sparse, Shifted Attention for LongLoRA finetuning

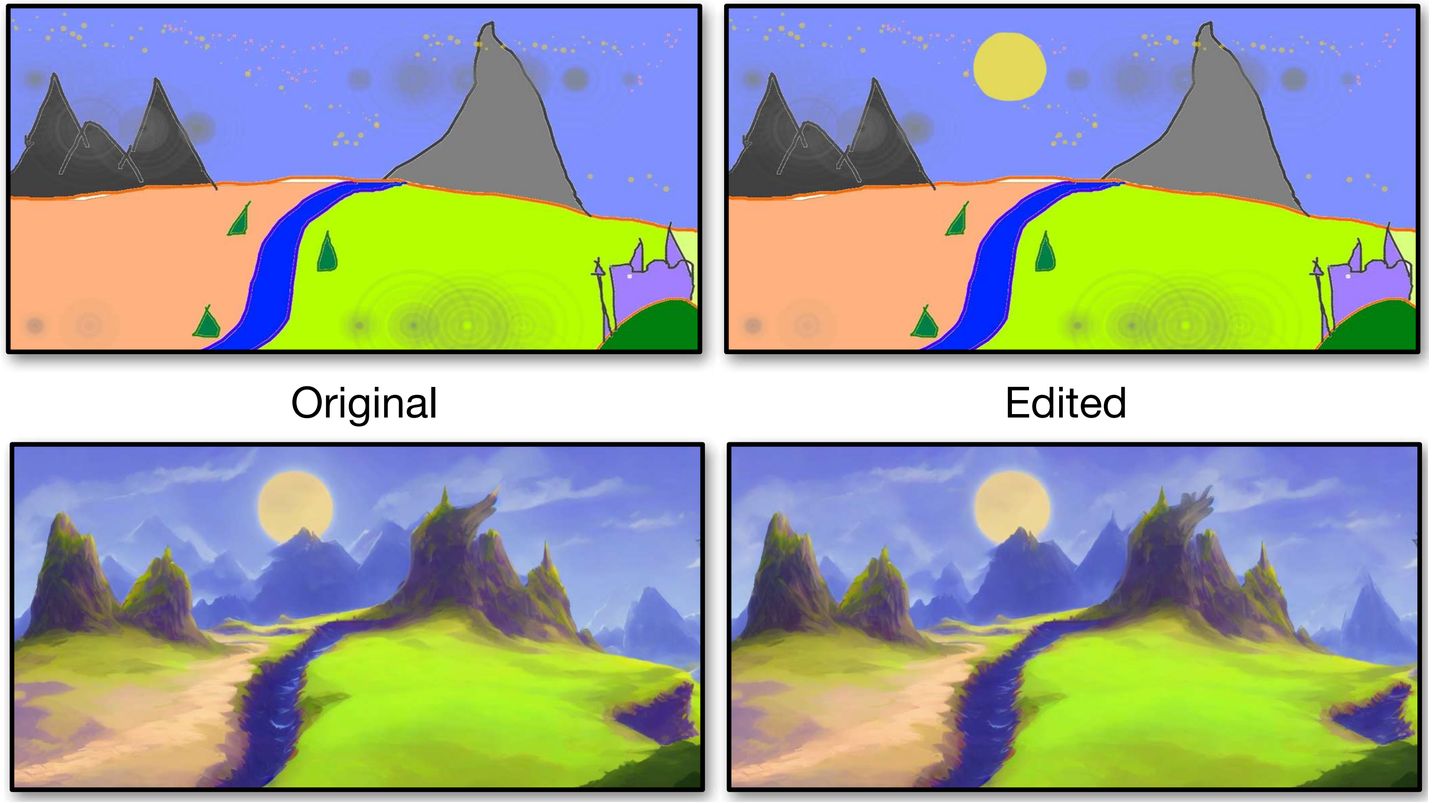


Lower perplexity, shorter finetuning time



Compressing and Accelerating Diffusion Models

Prompt: A fantasy landscape, trending on artstation.



Original

Edited

Stable Diffusion+SDEdit:
1855GMACs, 369ms

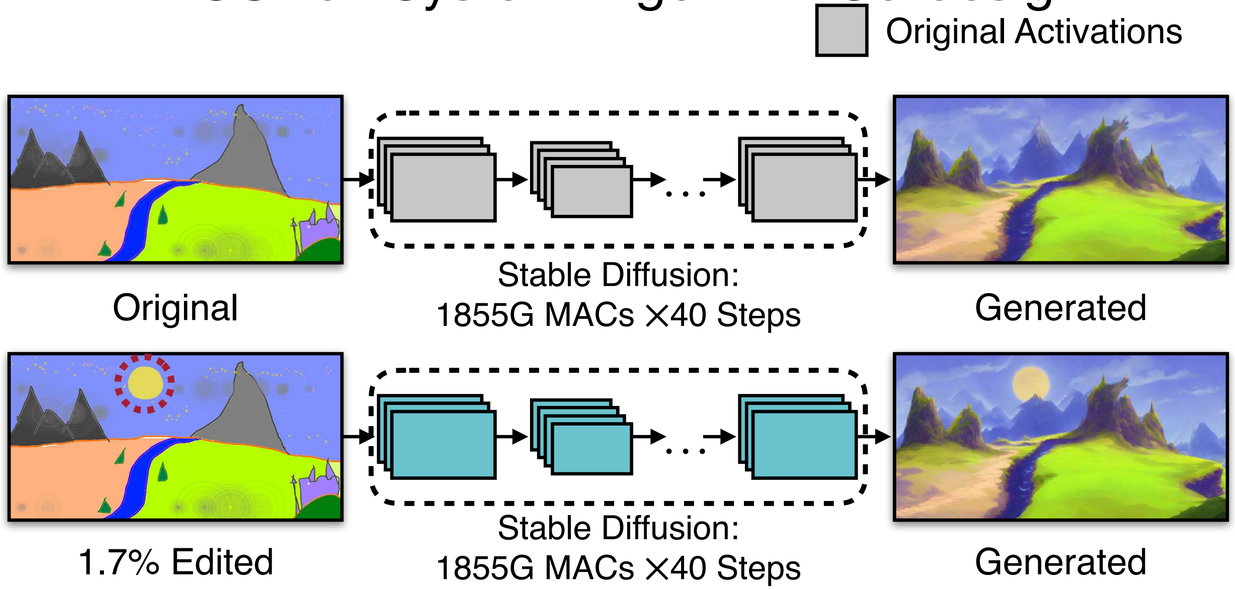
Ours:
225GMACs (8.2×),
51.2ms (7.2x)

Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models. [Li et al, NeurIPS 2020]

Spatially Sparse Inference for Diffusion Models

Vanilla Model Wastes Many Computations to Re-synthesize the Entire Image

- MCUNet: System-Algorithm Co-design



(a) SDEdit with Vanilla Stable Diffusion

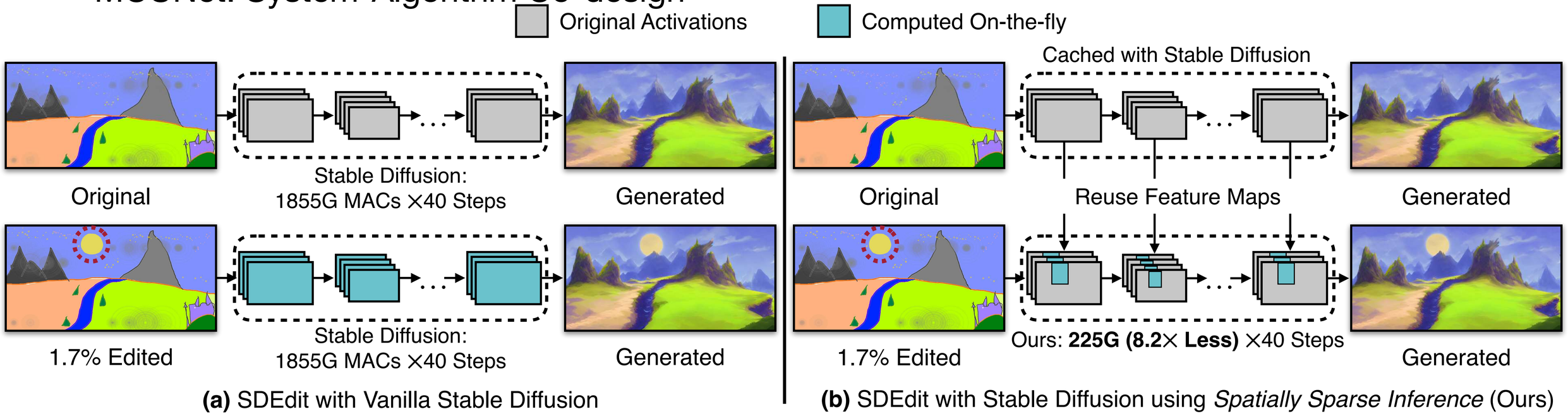
- Only 1.7% region is edited, but vanilla model re-synthesizes the entire image.
- Feature maps remain mostly the same at unedited regions.

Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models. [Li et al, NeurIPS 2020]

Spatially Sparse Inference for Diffusion Models

Vanilla Model Wastes Many Computations to Re-synthesize the Entire Image

- MCUNet: System-Algorithm Co-design



(a) SDEdit with Vanilla Stable Diffusion

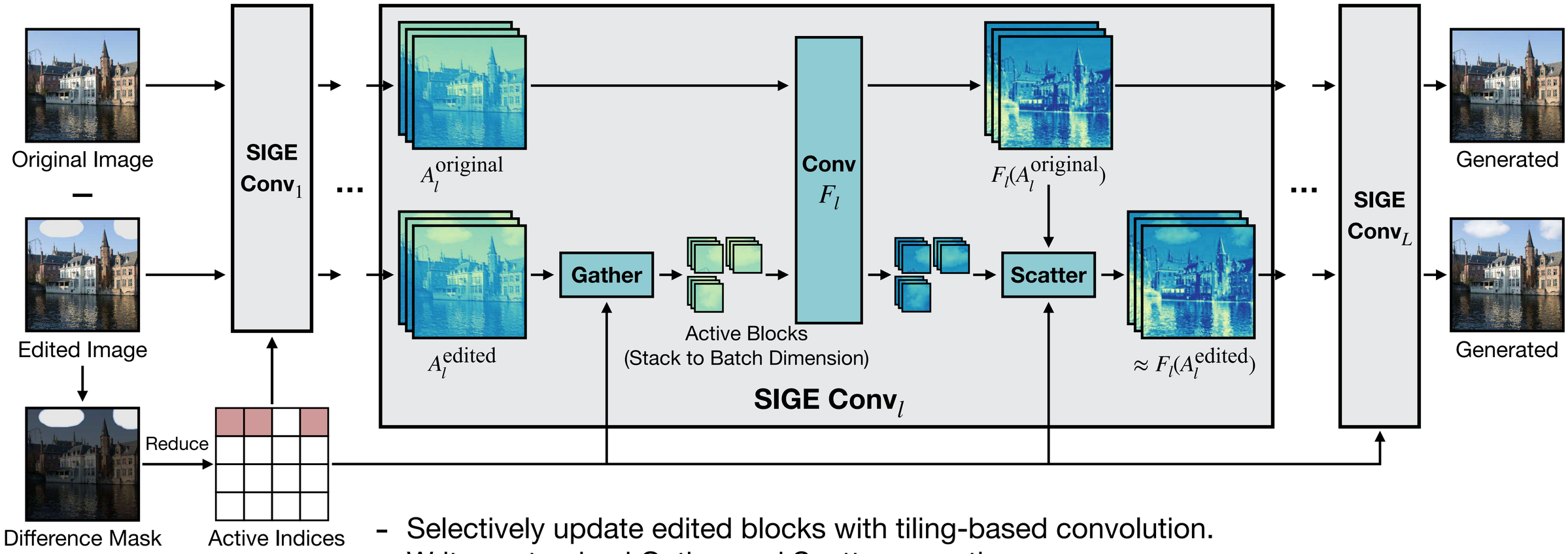
(b) SDEdit with Stable Diffusion using *Spatially Sparse Inference* (Ours)

- Only 1.7% region is edited, but vanilla model re-synthesizes the entire image.
- Feature maps remain mostly the same at unedited regions.
- Reuse cached activations to selectively update edited regions (8× less computation).

Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models. [Li et al, NeurIPS 2020]

Spatially Sparse Inference for Diffusion Models

Sparse Incremental Generative Engine (SIGE)



- Selectively update edited blocks with tiling-based convolution.
- Write customized Gather and Scatter operations.
- Fuse kernels to reduce overheads.

Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models. [Li et al, NeurIPS 2020]

Spatially Sparse Inference for Diffusion Models

Qualitative Results of SIGE on Stable Diffusion

A photograph of a horse on a grassland.

Original | 11.6% Masked

Stable Diffusion:
1855GMACs 369ms

Ours:
514G (3.6×) 95.0ms (3.9×)

Image Inpainting

A fantasy beach landscape, trending on artstation.

Original | 2.9% Edited

Stable Diffusion+SDEdit:
1855GMACs 369ms

Ours:
353G (5.3×) 76.4ms (4.8×)

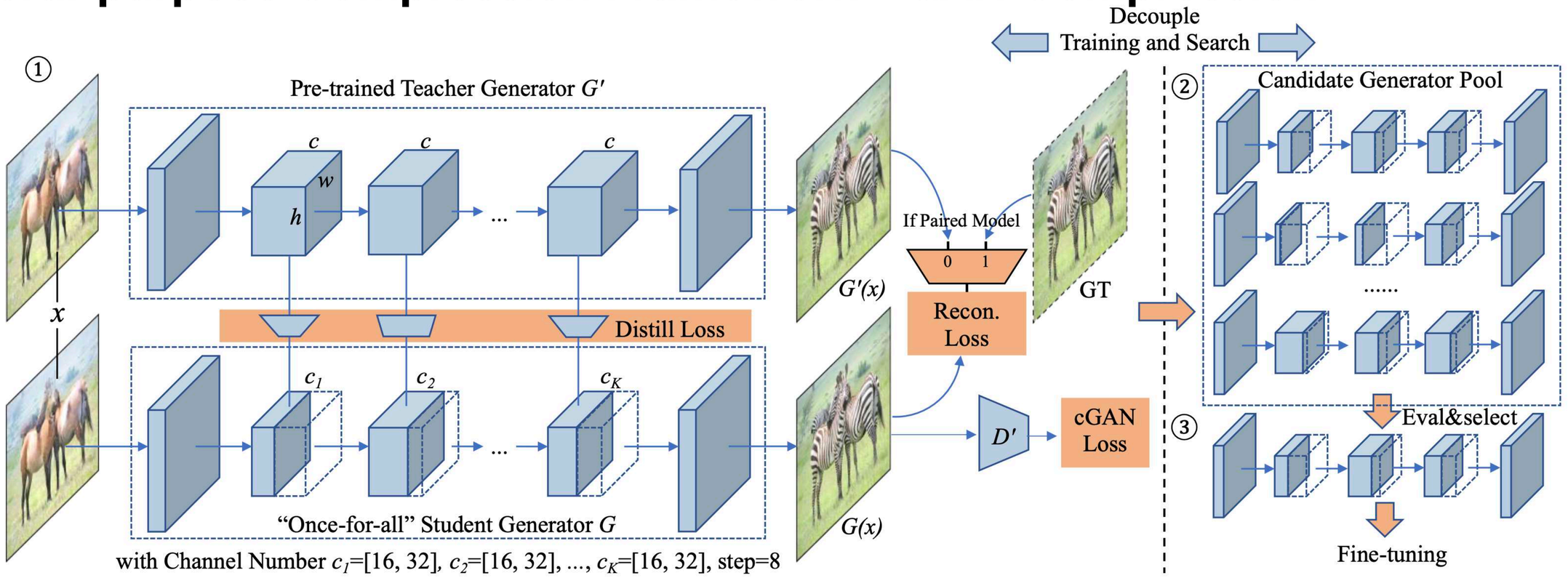
Image Editing

Latency Measured on NVIDIA RTX 3090

Efficient Spatially Sparse Inference for Conditional GANs and Diffusion Models. [Li et al, NeurIPS 2020]

GAN Compression

General-purpose Compression Framework—GAN Compression



Training Objective:

- Unify unpaired and paired learning
- Inherit teacher discriminator
- Distill intermediate channels

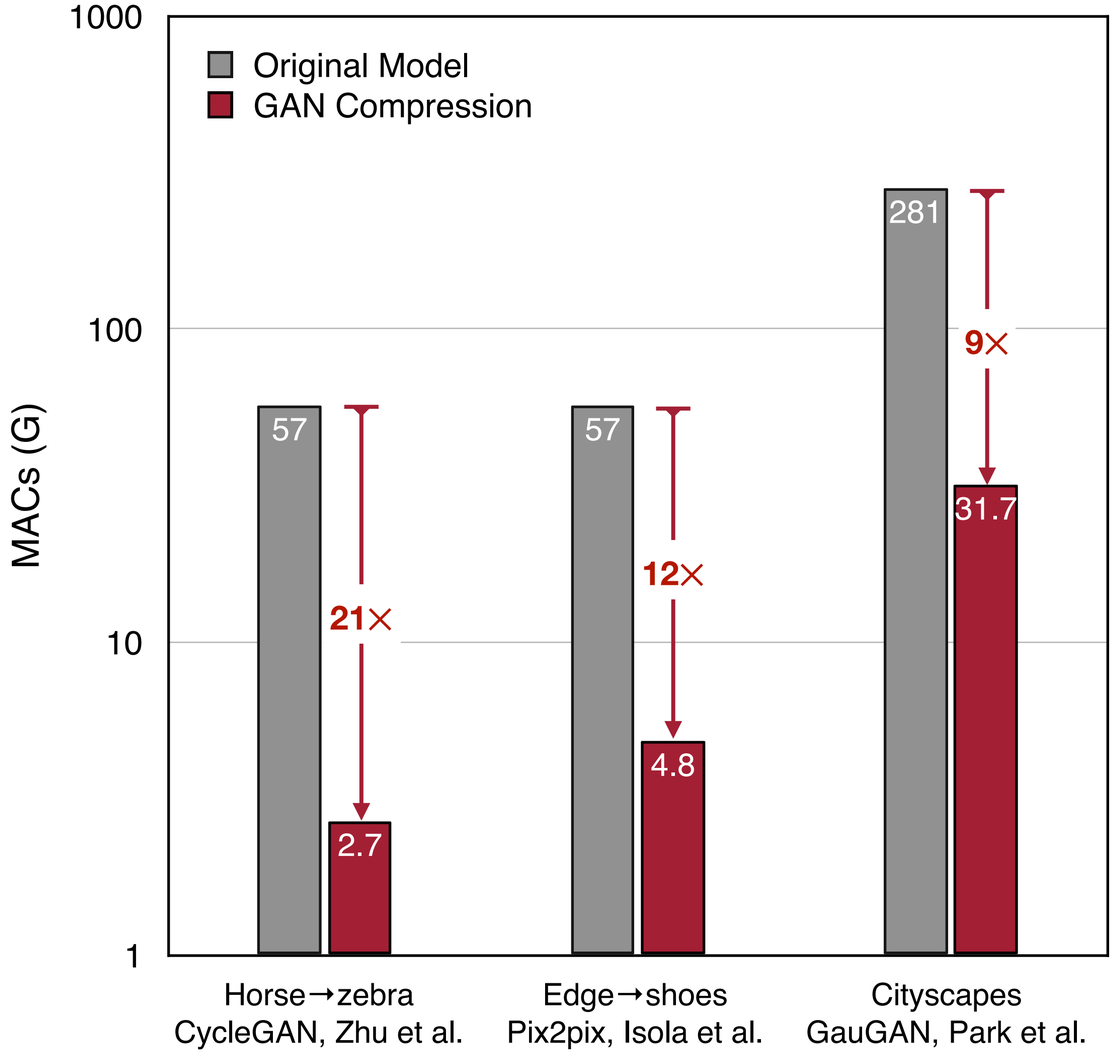
Automated Channel Reduction with NAS:

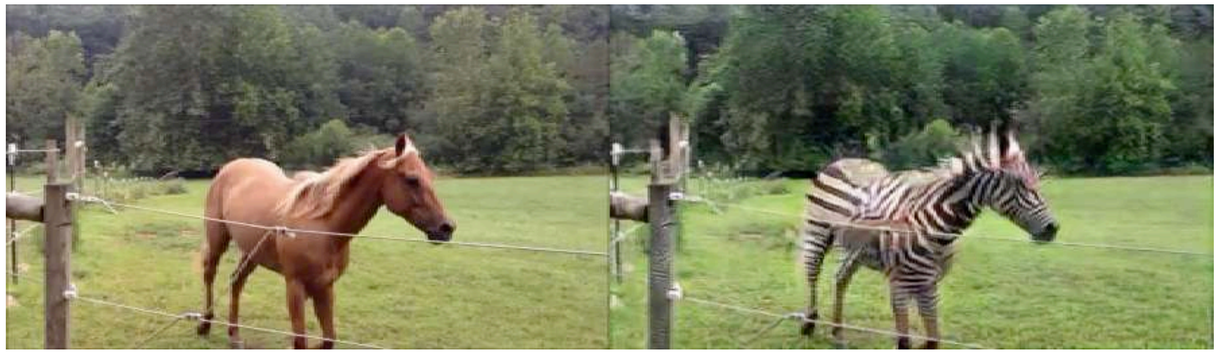
- Decomposed convolution as the design space
- Decouple training and search via weight sharing and evolutionary search

GAN Compression: Efficient Architectures for Interactive Conditional GANs. [Li et al. CVPR 2020]

GAN Compression

GAN Compression Reduces the Computation by 9-21×





Original CycleGAN; MACs: 56.8G; FPS: 12.1; FID: 61.5

GAN Compression; MACs: 3.50G (16.2×); FPS: 40.0 (3.3×); FID: 53.6

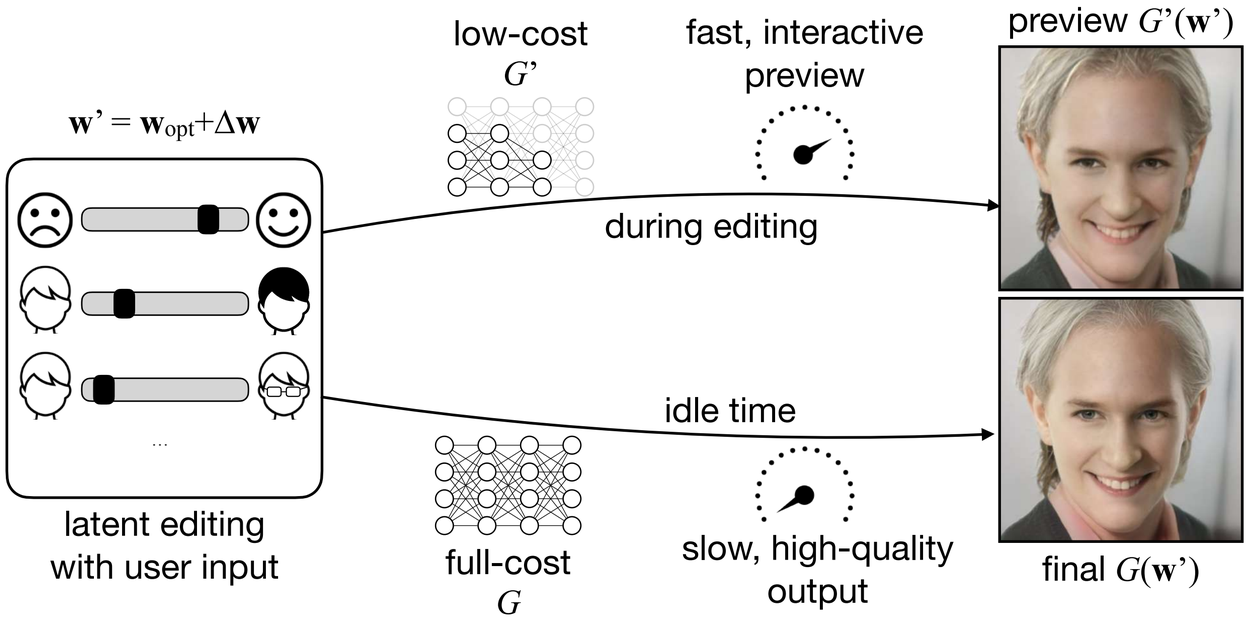
Measured on NVIDIA **Jetson Xavier GPU**
Lower FID indicates better Performance.

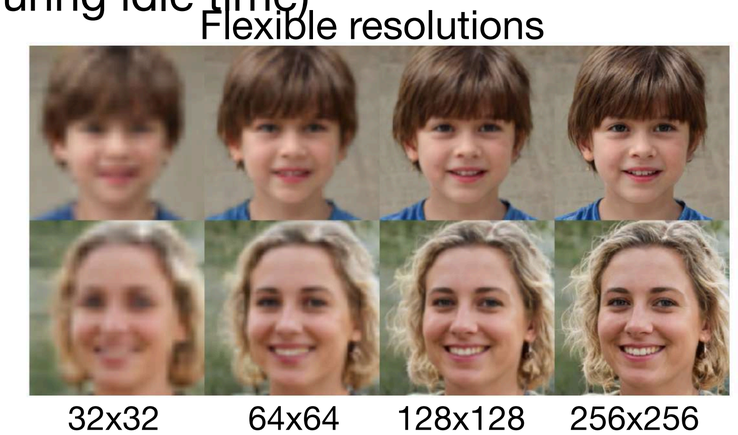
GAN Compression: Efficient Architectures for Interactive Conditional GANs. [Li et al. CVPR 2020]

Anycost GAN

Quick Preview By Runner A Smaller But Consistent Model

- **Anycost GAN**
  - Run a low-cost model to generate a fast, interactive preview
  - Run a full-cost model to generate the final, high-quality output (during idle time)
- Supports flexible **resolutions** and **channel widths**

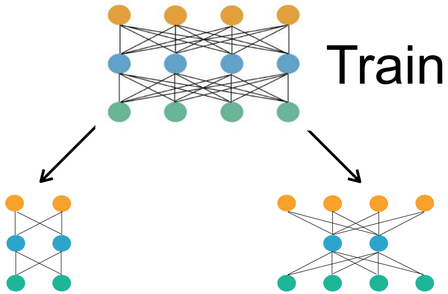




Anycost GANs for Interactive Image Synthesis and Editing. [Lin et al. CVPR 2021]

Anycost GAN

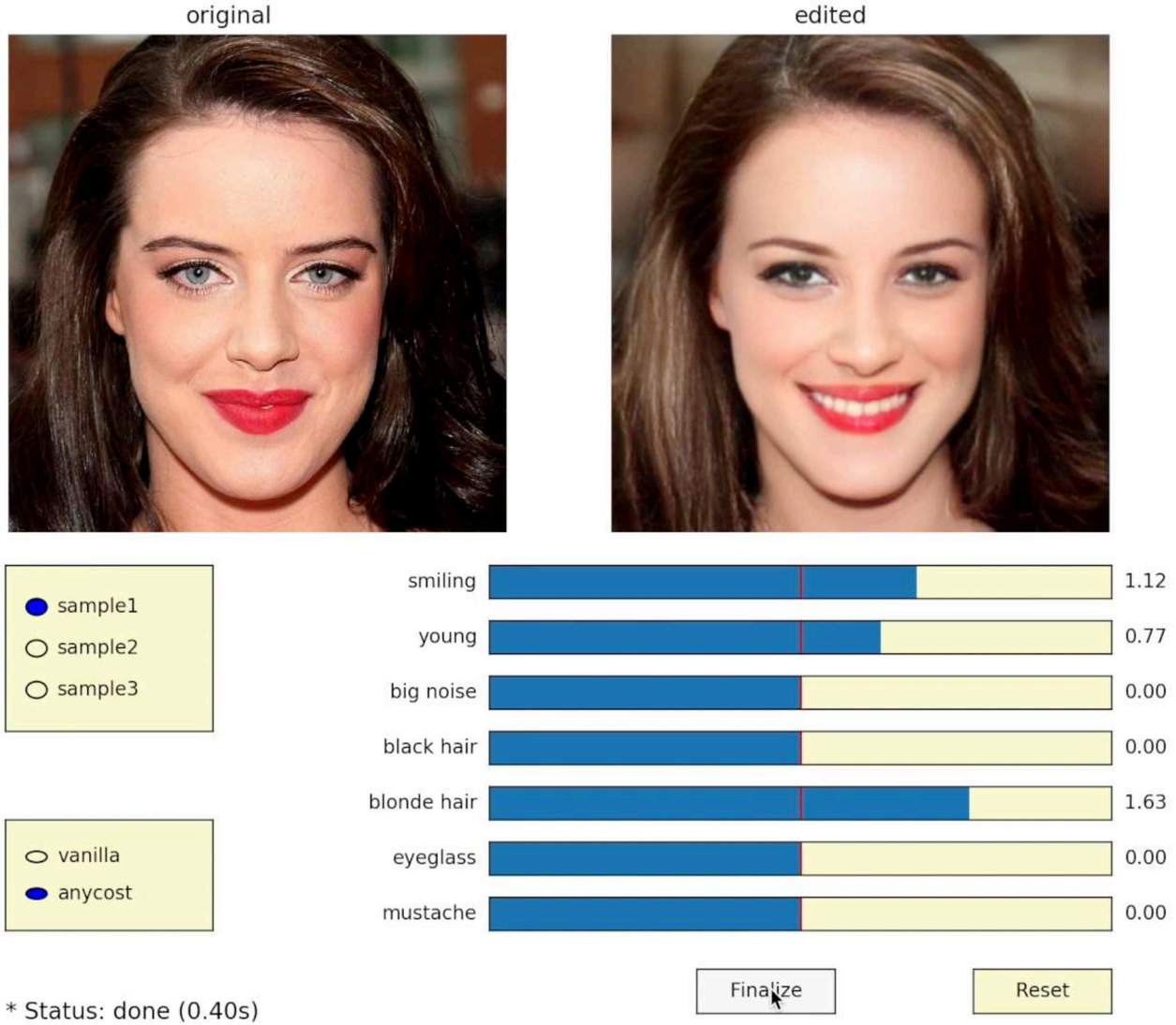
- Generative Adversarial Network (GAN) is computationally heavy and slow
- Difficult for interactive photo editing on mobile device (iPad)
- Anycost GAN with once-for-all network:



Train once

Small sub-net: low cost, fast prototyping

Large sub-net: high-quality finalization

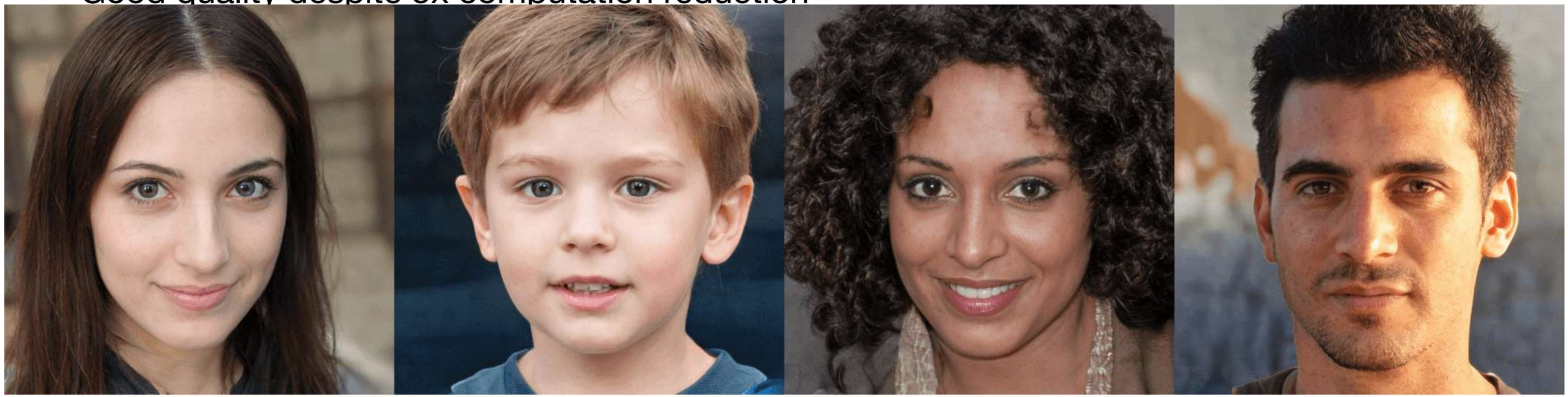


Anycost GANs for Interactive Image Synthesis and Editing. [Lin et al. CVPR 2021]

Anycost GAN

Tiered Pricing for Content Generation as a Service

- Good quality despite 5x computation reduction



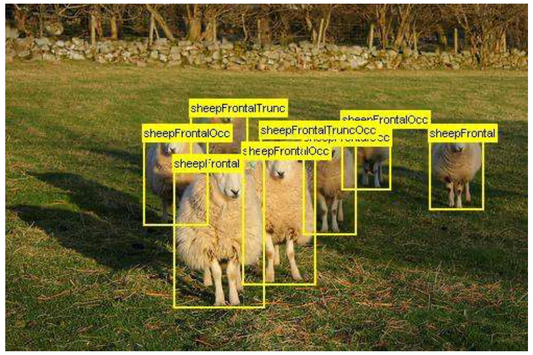
Compute Budget	1x	0.7x	0.5x	0.4x	0.2x
Tiered Pricing	$0.01	$0.007	$0.005	$0.004	$0.002

Anycost GANs for Interactive Image Synthesis and Editing. [Lin et al. CVPR 2021]

The quality is still reasonably good

MCUNet for TinyML

Advancing object detection by allowing a larger resolution



- Resolution is more important for detection than classification
- Our method significantly improves objection detection by double digits

Face/mask detection

Person detection

OpenMV Cam: 512KB SRAM + 2MB Flash

MCUNet v1-v3

MIT News
ON CAMPUS AND AROUND THE WORLD

SUBSCRIBE | SEARCH NEWS

System brings deep learning to "internet of things" devices

Advance could enable artificial intelligence on household appliances while enhancing data security and energy efficiency.

Watch Video

Daniel Ackerman | MIT News Office
November 13, 2020

PRESS INQUIRIES

MIT researchers have developed a system, called MCUNet, that brings machine learning to microcontrollers. The advance could enhance the function and security of devices connected to the Internet of Things (IoT).

(Highlighted by MIT Homepage)

MIT News
ON CAMPUS AND AROUND THE WORLD

SUBSCRIBE | SEARCH NEWS

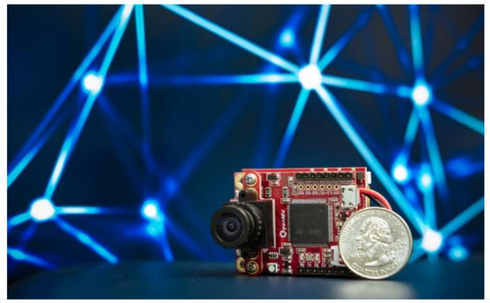
Tiny machine learning design alleviates a bottleneck in memory usage on internet-of-things devices

New technique applied to small computer chips enables efficient vision and detection algorithms without internet connectivity.

Watch Video

Lauren Hinkel | MIT-IBM Watson AI Lab
December 8, 2021

PRESS INQUIRIES



An MIT team's tinyML vision system outperforms other models in many image classification and detection tasks.

Photo courtesy of the researchers.

(Highlighted by MIT Homepage)

MIT News
ON CAMPUS AND AROUND THE WORLD

SUBSCRIBE | SEARCH NEWS

Learning on the edge

A new technique enables AI models to continually learn from new data on intelligent edge devices like smartphones and sensors, reducing energy costs and privacy risks.

Adam Zewe | MIT News Office
October 4, 2022

PRESS INQUIRIES

A machine-learning model on an intelligent edge device allows it to adapt to new data and make better predictions. For instance, training a model on a smart keyboard could enable the keyboard to continually learn from the user's writing.

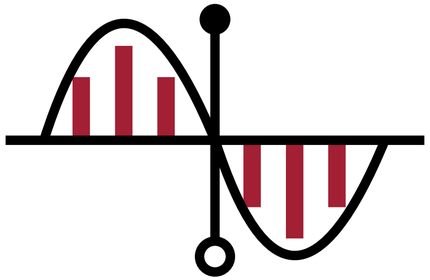
Image: Digital collage by Jose-Luis Olivares, MIT, using stock images and images derived from MidJourney AI.

Can We Learn on Edge?

AI systems need to continually adapt to new data collected from the sensors

- On-device learning: **better privacy, lower cost, customization, life-long learning**
- Training is more **expensive** than inference, hard to fit edge hardware (limited memory)

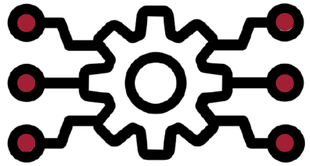
On-Device Training Under 256KB Memory



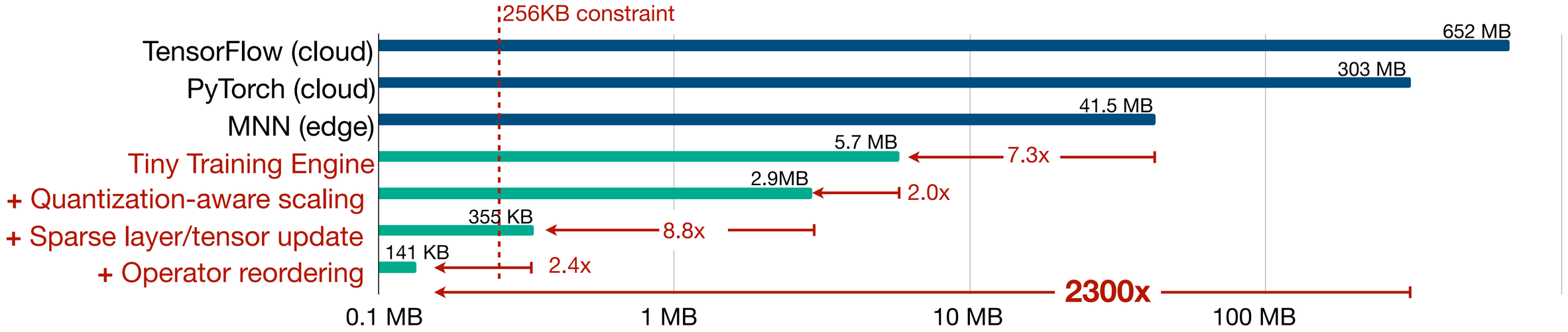
1. Quantization-aware scaling

$$\tilde{\mathbf{G}}_{\bar{\mathbf{W}}} = \mathbf{G}_{\bar{\mathbf{W}}} \cdot s_{\mathbf{W}}^{-2}$$

2. Sparse layer/tensor update

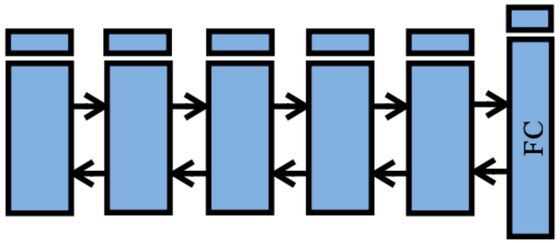


3. Tiny Training Engine

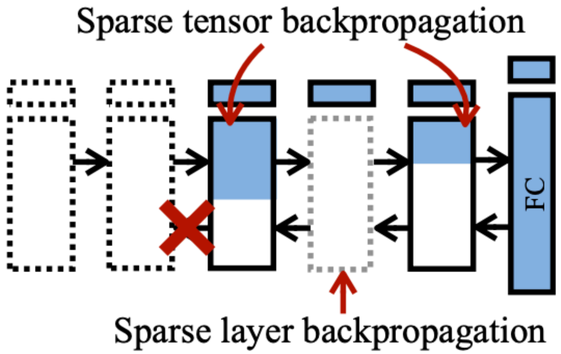


PockEngine

Learning on the Edge



Full back-propagation



Sparse back-propagation

Input: Please reverse the words in the sentence "I love the Micro conference”
LlamaV2-7B-original: I **hate** the conference Micro.
LlamaV2-7B-tuned: The reversed result is "Conference Micro the love I"

Method	Iteration Latency (↓)
FT-Full	7.7s
LoRA (rank=8)	7.3s
FT-Full	1.8s
Sparse	0.9s

MIT News
ON CAMPUS AND AROUND THE WORLD

SUBSCRIBE

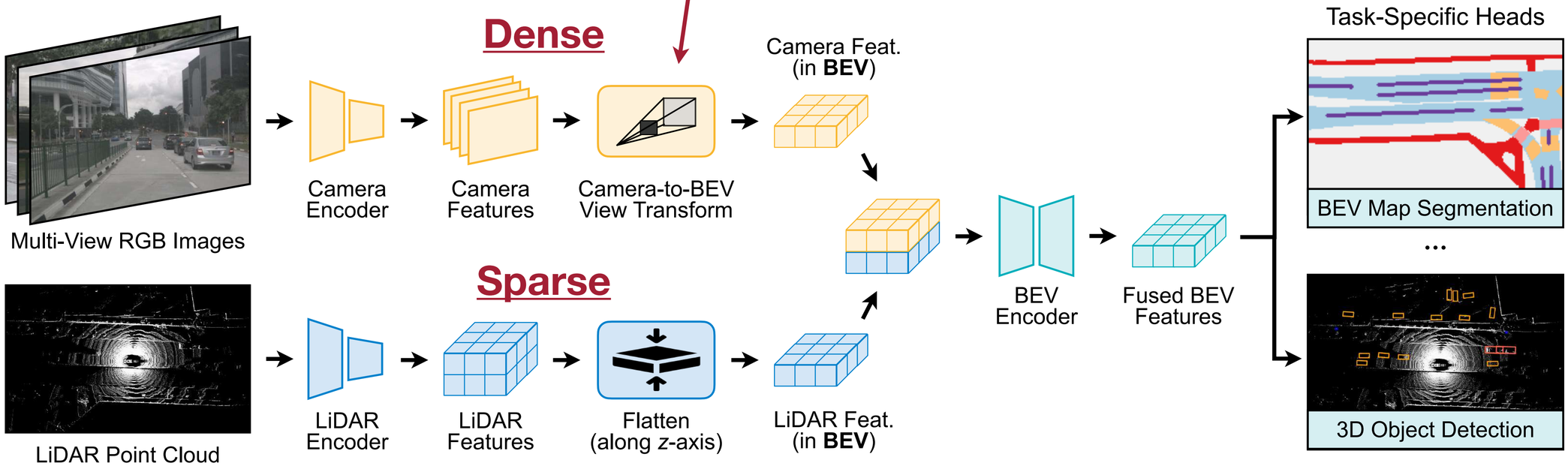
Technique enables AI on edge devices to keep learning over time

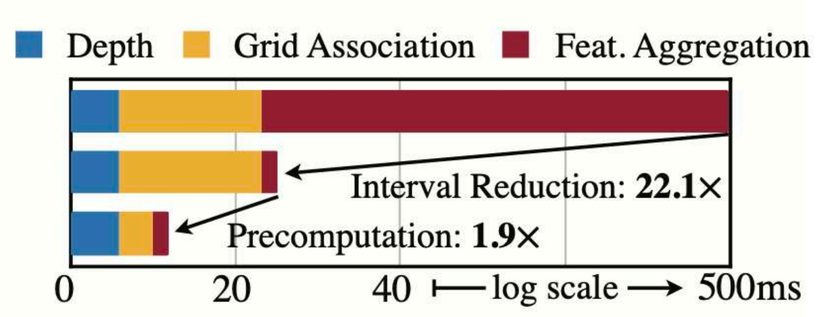
With the PockEngine training method, machine-learning models can efficiently and continuously learn from user data on edge devices like smartphones.

Adam Zewe | MIT News
November 16, 2023

BEVFusion: Dense (Camera) + Sparse (LiDAR) Sensor Fusion

Accelerate LSS by 40x with Interval Reduction and Pre-computation





- Ranked **first** on **nuScenes 3D object detection** (2022/6).
- Ranked **first** on **nuScenes 3D object tracking** (2022/7).
- Ranked **first** on **Waymo 3D object detection** (2022/11).
- Ranked **first** on **Argoverse 3D object detection** (2023/4).

BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation. [Liu et al. ICRA 2023]

BEVFusion: Multi-Task Multi-Sensor Fusion

BEVFusion takes **multi-modal** sensory inputs and supports **multiple** 3D perception tasks.

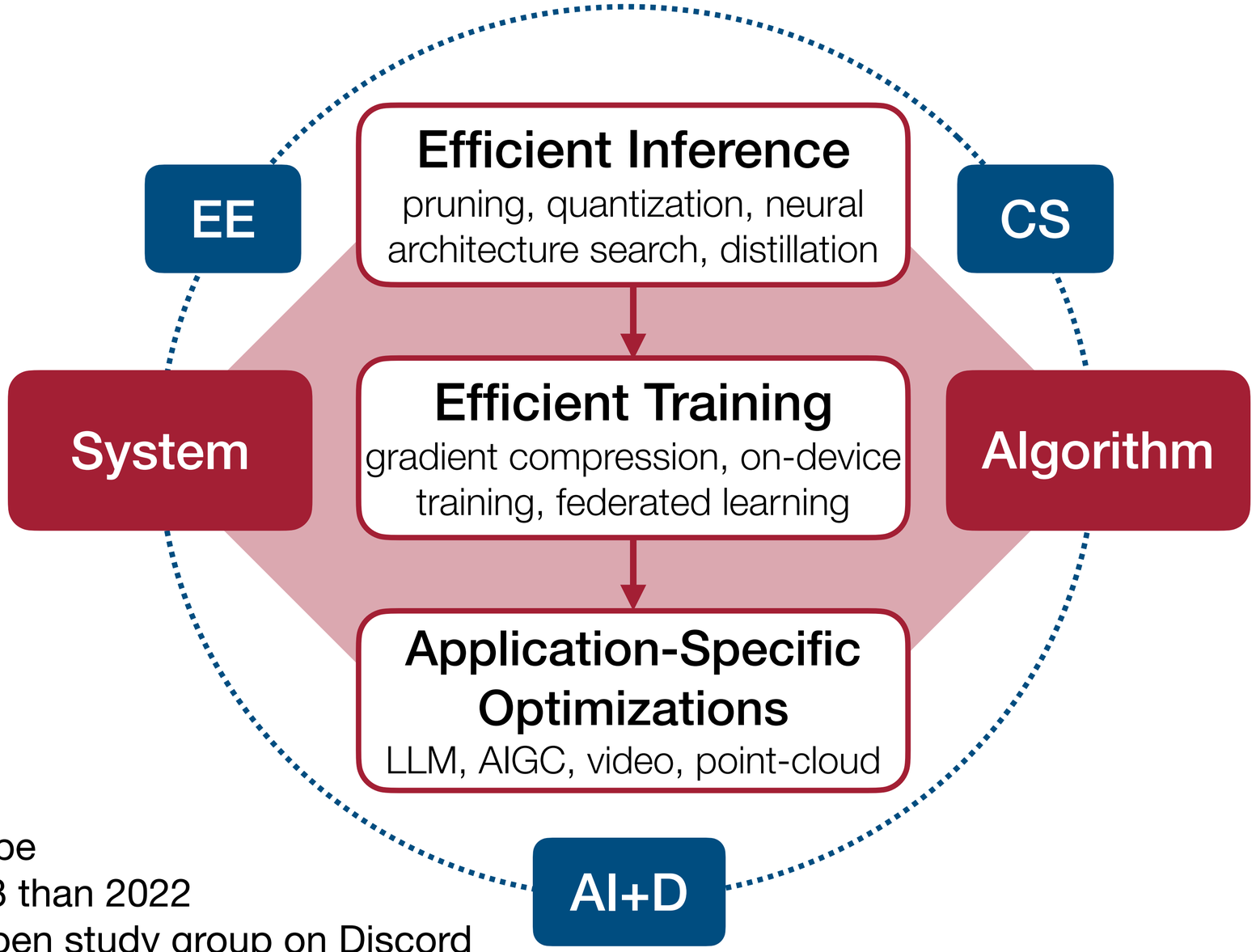
Multi-View Camera

LiDAR

https://youtu.be/uCAka90si9E

EfficientML.ai Course

TinyML and Efficient AI Computing



200K views on YouTube
3x registration in 2023 than 2022
700 students in the open study group on Discord

MIT HAN LAB

Hardware, AI and Neural-nets

github.com/mit-han-lab

youtube.com/c/MITHANLab

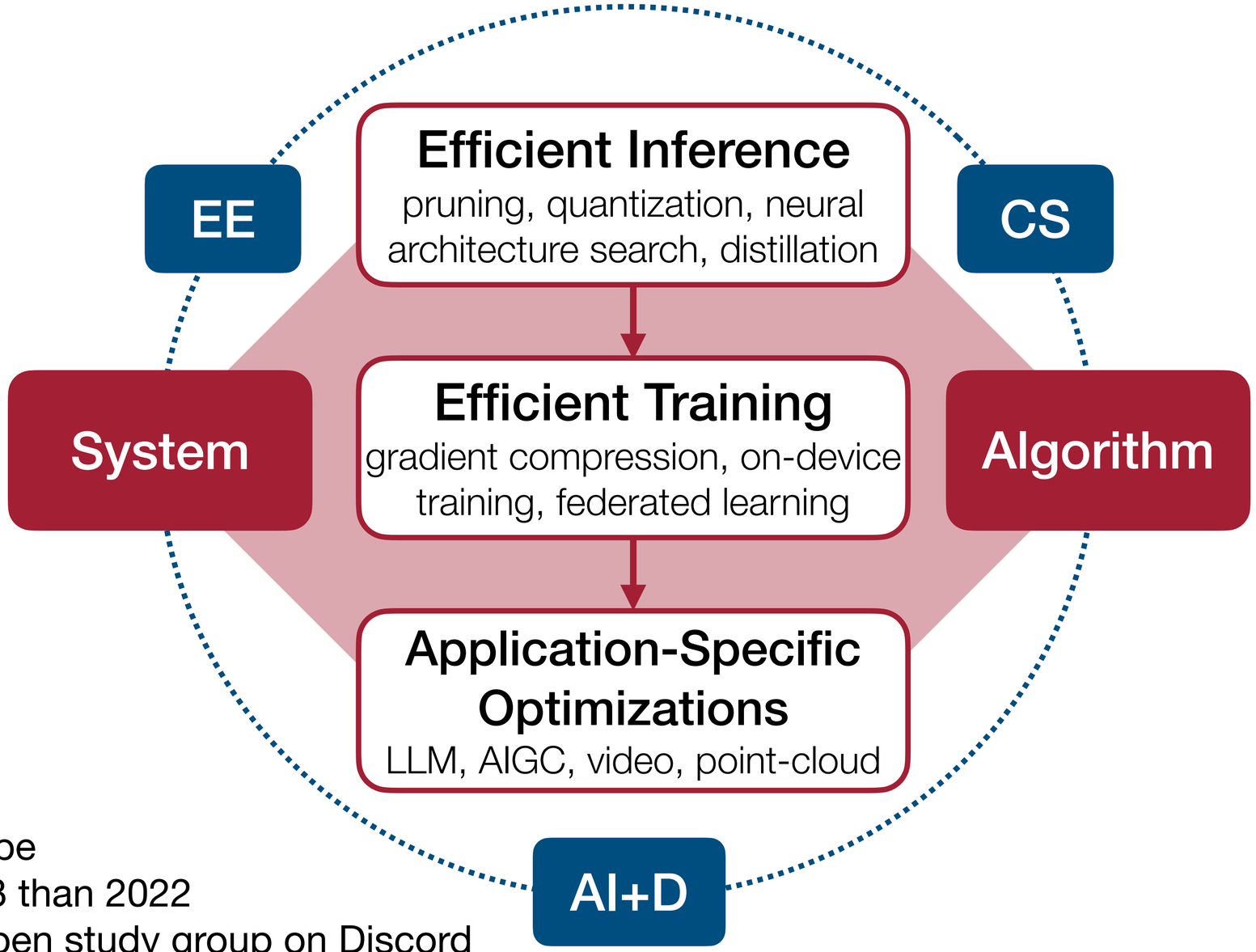
songhan.mit.edu
tinyml.mit.edu

Sponsors:

Media:

EfficientML.ai Course

TinyML and Efficient AI Computing



200K views on YouTube
3x registration in 2023 than 2022
700 students in the open study group on Discord

EfficientML.ai Course

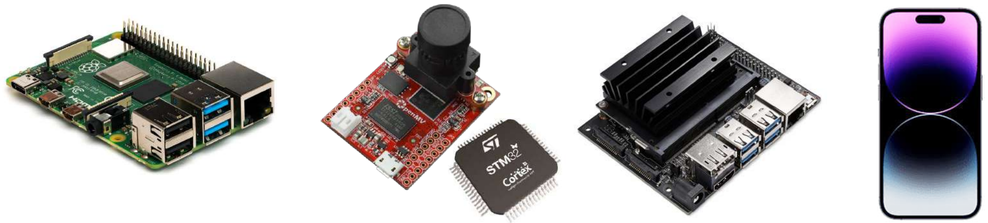
TinyML and Efficient Deep Learning Computing

6.5940 • Fall 2023 • MIT

Large generative models (e.g., large language models, diffusion models) have shown remarkable performance, but they require a massive amount of computational resources. To make them more accessible, it is crucial to improve their efficiency.

This course will introduce efficient AI computing techniques that enable powerful deep learning applications on resource-constrained devices. Topics include model compression, pruning, quantization, neural architecture search, distributed training, data/model parallelism, gradient compression, and on-device fine-tuning. It also introduces application-specific acceleration techniques for large language models, diffusion models, video recognition, and point cloud. This course will also cover topics about quantum machine learning. Students will get hands-on experience deploying large language models (e.g., LLaMA 2) on a laptop.

- **Time:** Tuesday/Thursday 3:30-5:00 pm Eastern Time
- **Location:** 36-156
- **Office Hour:** Thursday 5:00-6:00 pm Eastern Time, 38-344 Meeting Room
- **Discussion:** Discord
- **Homework submission:** Canvas
- **Online lectures:** The lectures will be streamed on YouTube.
- **Resources:** MIT HAN Lab, HAN Lab Github, TinyML, MCUNet, OFA, SmoothQuant
- **Contact:**
  - Students can ask all course-related questions on Discord.
  - For external inquiries, personal matters, or emergencies, you can email us at *efficientml-staff@mit.edu*.
  - If you are interested in getting updates, please sign up here to join our mailing list to get notified!



This is honestly one of the best set up courses I've taken at MIT

I really like how structured the labs are, and being able to see actual implementations of the techniques we learn about.

I managed the weekly labs and lectures by only watching the course on YouTube. As a researcher, I gained some valuable knowledge from your course. Excellent slides and teaching and useful labs.

I love how we are using microntroller and focusing on application instead of just theories.

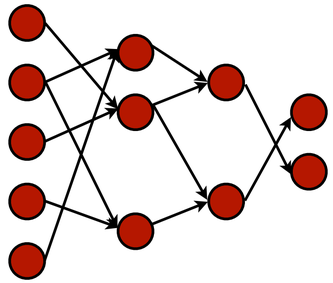
I like the class and I have been able to follow the class easily (which had rarely happened to me in my previous courses)

https://efficientml.ai

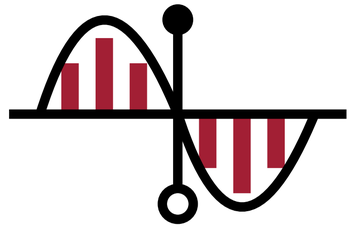
EfficientML.ai Hands-on Labs

PYTORCH

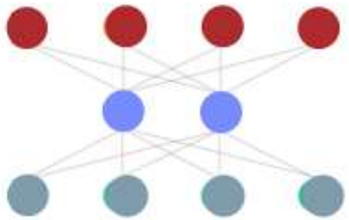
Lab 0: Tutorial on PyTorch



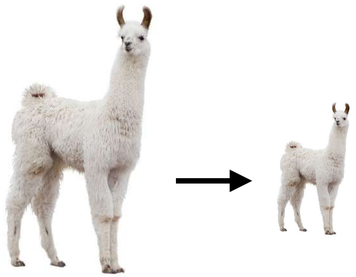
Lab 1: Pruning



Lab 2: Quantization



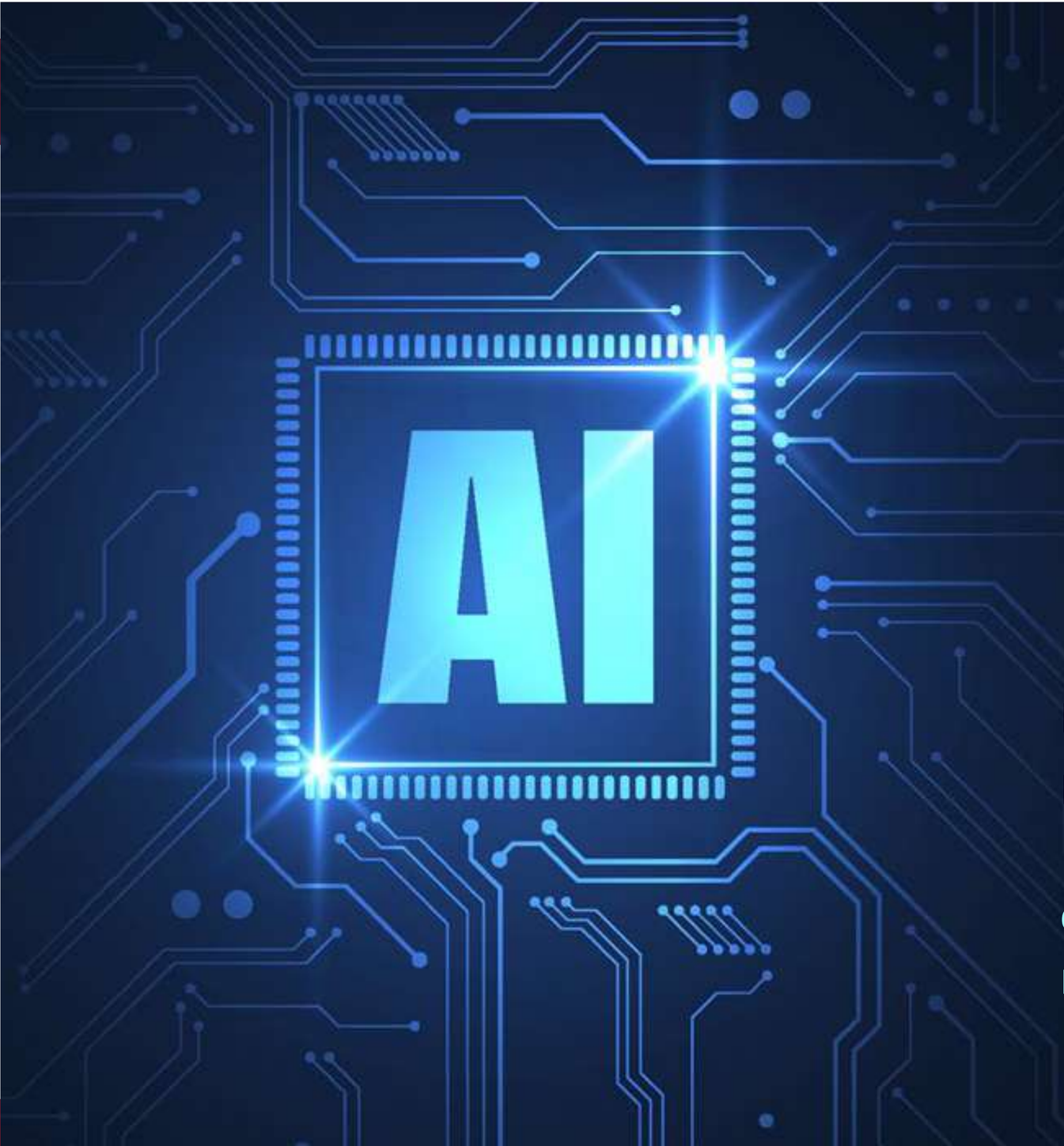
Lab 3: Neural Architecture Search



Lab 4: LLM Compression

TinyChat

Lab 5: On-Device LLM Deployment



MIT AI Hardware Program

MIT Microsystems Technology Laboratories (SoE)

MIT Quest for Intelligence – Corporate (SCC)

Co-Leads: Jesús del Alamo and Aude Oliva

Internal Advisory Board Chair: Anantha Chandrakasan

TinyML and Efficient AI Computing

MIT HAN LAB
Hardware, AI and Neural-nets

github.com/mit-han-lab

youtube.com/c/MITHANLab

songhan.mit.edu
tinyml.mit.edu

Sponsors:

Media: