清華大學电子工程系
Department of Electronic Engineering, Tsinghua University

INFINIGENCE
无问芯穹

Generative Model Compression and Acceleration

Xuefei Ning

Affiliated with: Department of Electronic Engineering, Tsinghua University

foxdoraame@gmail.com

Group Leader: Prof. Yu Wang yu-wang@tsinghua.edu.cn

NICS EFC

目录 Contents

目录 Contents

AI-Generated Content (AIGC)

AIGC, which uses <u>generative models</u> to generate content that satisfies human instructions, aims to make the content creation process more efficient and accessible[1].

Language Generation

Large Language Models: LLaMA-2-7B[2]

Visual Generation

Video Diffusion Models: Sora[3]

[1] Cao, Yihan, et al. "A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt." *arXiv 2023*.
[2] Touvron, Hugo, et al. "Llama 2: Open foundation and fine-tuned chat models." *arXiv 2023*.
[3] Brooks, Peebles, et al., "Video generation models as world simulators." *2024*.

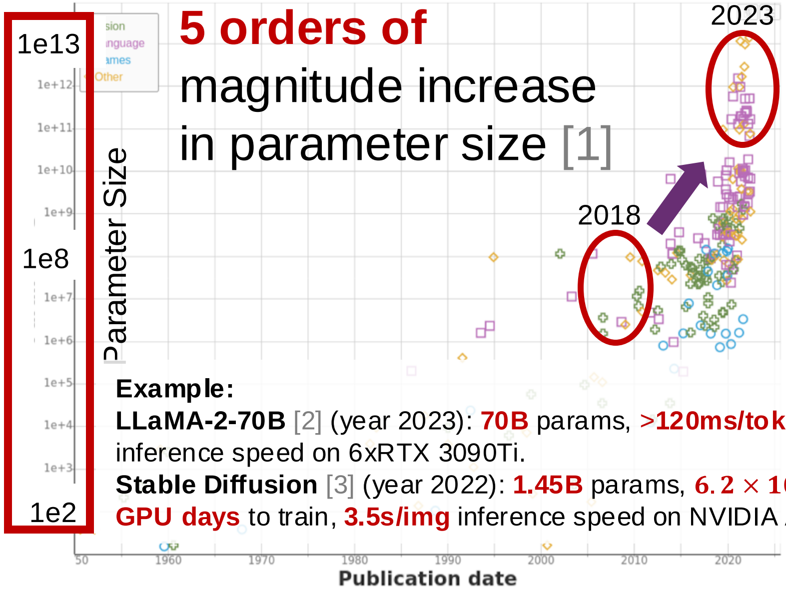
Trend of Generative Models

INFINIGENCE 无问芯穹

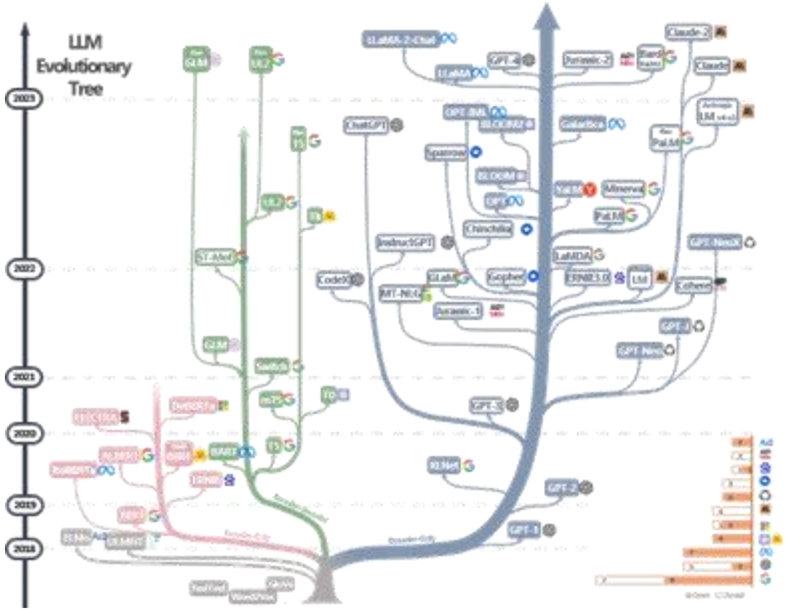
The model size of generative models has being rapidly increased

2018 - 2023

5 orders of magnitude increase in parameter size [1]



Example:
LLaMA-2-70B [2] (year 2023): **70B** params, **>120ms/token** inference speed on 6xRTX 3090Ti.
Stable Diffusion [3] (year 2022): **1.45B** params, 6.2×10^3 **GPU days** to train, **3.5s/img** inference speed on NVIDIA A100.



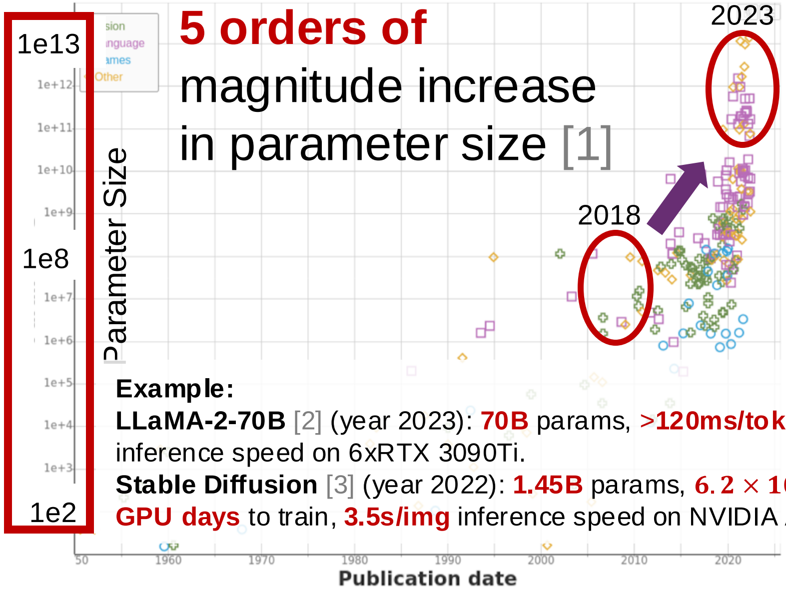
[1] Villalobos et al. “Machine Learning Model Sizes and the Parameter Gap.” arXiv 2022.
[2] Touvron, Hugo, et al. "Llama 2: Open foundation and fine-tuned chat models." *arXiv 2023*.
[3] Rombatch et al., High-Resolution Image Synthesis with Latent Diffusion Models, CVPR 2022.
[4] Yang et al., "Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond“, *ACM Transactions on Knowledge Discovery from Data* 2023.

Trend of Generative Models

The model size of generative models has being rapidly increased

2018 - 2023

5 orders of magnitude increase in parameter size [1]



Example:

LLaMA-2-70B [2] (year 2023): **70B** params, **>120ms/token** inference speed on 6xRTX 3090Ti.

Stable Diffusion [3] (year 2022): **1.45B** params, 6.2×10^3 **GPU days** to train, **3.5s/img** inference speed on NVIDIA A100.

Stable Diffusion 1.5[3]
~1B Params

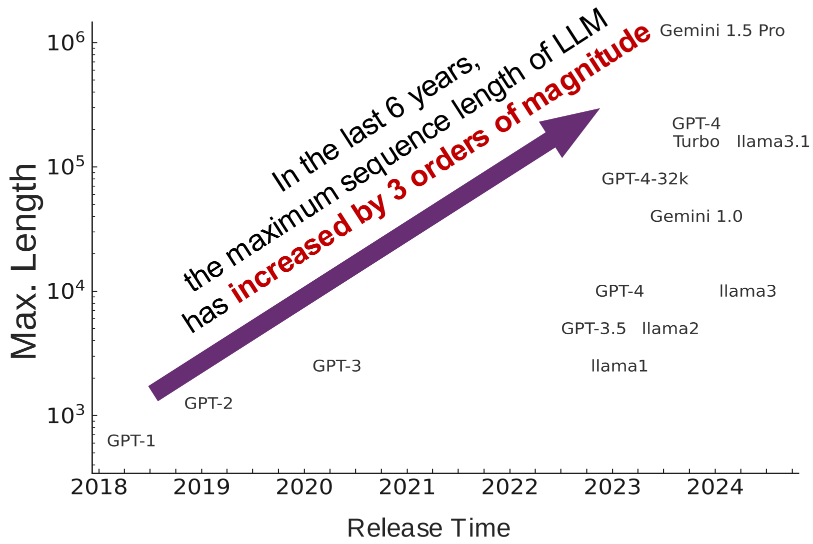
Flux[4]
~12B Params

[1] Villalobos et al. "Machine Learning Model Sizes and the Parameter Gap." arXiv 2022.
[2] Touvron, Hugo, et al. "Llama 2: Open foundation and fine-tuned chat models." *arXiv 2023*.
[3] Rombatch et al., High-Resolution Image Synthesis with Latent Diffusion Models, CVPR 2022.
[4] black-forest-labs/flux: Official inference repo for FLUX.1 models

Trend of Generative Models

The generation data size has being rapidly increased

Longer Sequence Length for Language



OpenAI
Meta AI
Google

Higher Resolution / Longer Video Length for Vision

OpenSORA[4]
generate Videos

Pixart-sigma[5]
generates 4K image

[1] Achiam, Josh, et al. "Gpt-4 technical report." arXiv 2023.
[2] Reid, Machel, et al. "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context." arXiv 2024.
[3] Dubey, Abhimanyu, et al. "The llama 3 herd of models." arXiv 2024.
[4] hpcaitech, "Open-SoRA: Democratizing Efficient Video Production for All." https://github.com/hpcaitech/Open-Sora
[5] Chen, Junsong et al. "PixArt-Σ: Weak-to-Strong Training of Diffusion Transformer for 4K Text-to-Image Generation." arXiv 2024.

Challenge and Research Goal

- As the model size is scaling up, the demands for computing power are increasing
- Due to real-time, usable, privacy and other application demands, physical limitations of the scenario, as well as cost control considerations, models need to be deployed on computing devices with limited computing power and low storage, and are required to run under low budgets.
- How to deploy “large” generative models and satisfy the application’s efficiency requirements while maintaining algorithmic performance?

Our goal is to **improve the efficiency (e.g., latency, throughput, storage)** of generative models to satisfy the application requirement.

Research Overview

Research Goal：Efficient model inference for AIGC application

Application

Language Generation

Large Language Models
(e.g., LLaMA-2-7B)

Visual Generation

Diffusion Models
(e.g., Stable Diffusion 3)

Methodology: Hardware-aware algorithm-level and model-level optimization

Technique

Algorithm-level

Diffusion Timestep Compression — Tackling Many Timesteps of Diffusion

Non-Autoregressive Generation — Tackling Full Autoregressive Generation of LLMs

Model-level

Structure Design

Model Compression

Research Framework

INFINIGENCE 无问芯穹

Efficient Large Language Models

Overview

Survey [CSUR Submission]

Survey on efficient LLM inference techniques

Algorithm-level

SoT [ICLR'24]

Parallel generation via prompting.
1.91~2.39x speed-up

Model-level

Sparse Attention

MoA [ICLR Submission]

Decide the heterogeneous elastic rule of the attention span for each head.
5.5~6.7x throughput improvement

Pruning

EEP [ICLR Submission]

Search the pruning pattern for MoE and use expert merging for finetuning.
48%~71% memory reduction,
1.11~1.40x speed-up,
better performance

Quantization

LLM-MQ [NeurIPS'23 Workshop]

Mixed-precision quantization.
2.8-bit quantization

QLLM-Eval [ICML'24]

Evaluating the effect of quantization.
Providing knowledge and practical suggestions

Efficient Diffusion Models

Algorithm-level

Time Step Compression

LCSC [ICLR Submission]

Linear combination of checkpoints.
15~23x training acceleration,
1.25~2x timestep compression

USF [ICLR'24]

OMS-DPM [ICML'23]

Search for optimal diffusion schedulers.
1.5~2x speed-up

DD [ICLR Submission]

generates image in **0.01s** and can achieve **>100x** speedup for Image AR model

Model-level

Quantization

MixDQ [ECCV'24]

Mixed-precision quantization.
3x memory decrease,
1.5x speed-up

ViDiT-Q [ICLR Submission]

Quantization for DiT.
2.5x memory improvement,
1.5x speed-up

Pruning & Sparse Attention

DiTFastAttn [NeurIPS'24]

Window & reused attention for DiT.
1.6x speed-up

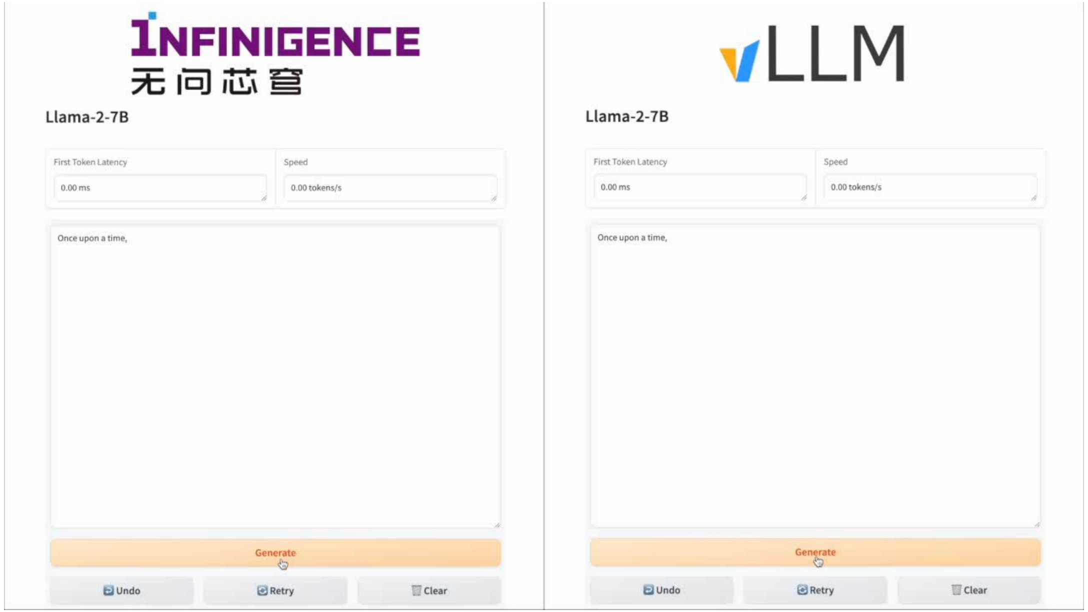
Fast Compression

FlashEval [CVPR'24]

10x evaluation acceleration

Acceleration Demo: LLMs

Achieving 2× throughput improvement with operator optimization



LLaMA-2-7B on AMD MI210

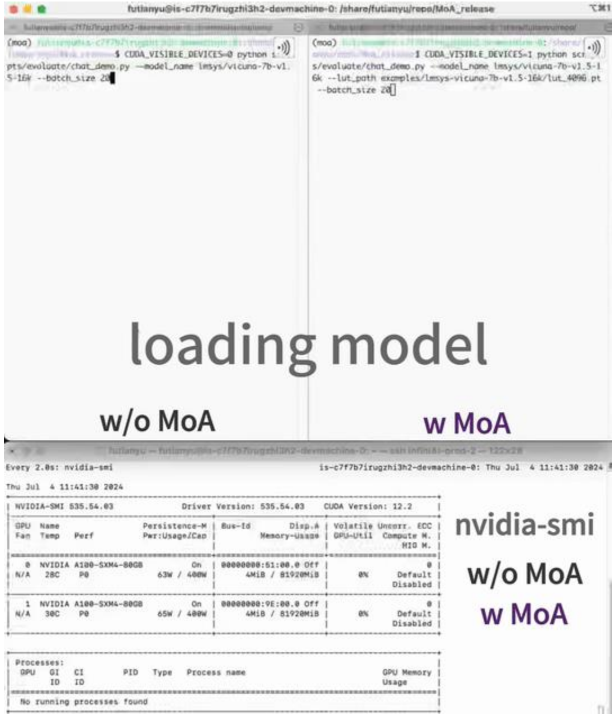
Before Acceleration (right):
39 tokens/s

After Acceleration (right):
79 tokens/s

No performance drop

Acceleration Demo: LLMs

- Sparse attention batch inference demo



Vicuna-7B on Nvidia-A100
batch size 20
end-to-end latency

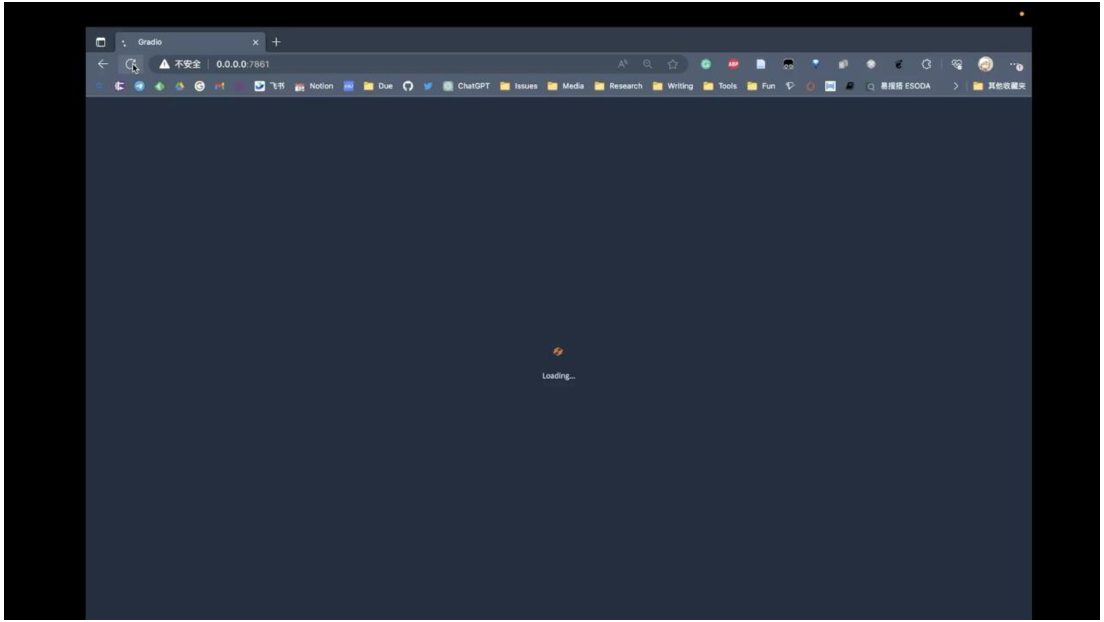
Before Sparse Attention (left):
Latency 42s

After Sparse Attention (right):
Latency 18s

Acceleration Demo: Diffusion Models

Timestep Optimization + TensorRT Deployment:

Achieving 6.9× end-to-end speed-up and reducing 1.5× memory



Stable Diffusion on a single NVIDIA A100 GPU

Before Acceleration (left):
11.7s latency, 11.9G VRAM

After Acceleration (right):
1.7s latency, 7.8G VRAM

Almost no performance drop

Acceleration Demo: Diffusion Models

Efficient Attention for DiT:

Achieve up to **1.8x** latency speedup

Pixart-Sigma

2Kx2K image, 50 steps

on NVIDIA A100 GPU

Before Acceleration (left):

~16s latency

After Acceleration (right):

~8s latency

Almost no performance drop

Acceleration Demo: Diffusion Models

Low-bit Quantization for DiT-based Image and Video Generation:
Achieve up to **2x** memory saving, **1.7x** latency speedup

FP16

ViDiT-Q W8A8

OpenSORA
512x512x16 Frames,
on NVIDIA A100 GPU

Before Acceleration (left):
~8.9s latency (20 step)

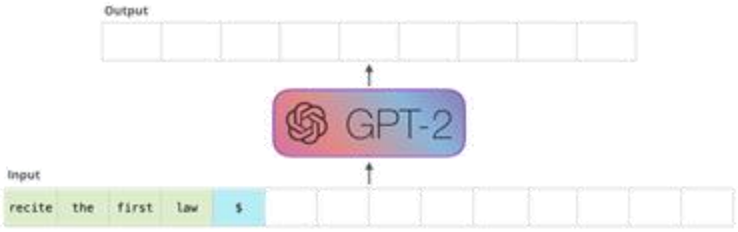
After Acceleration (right):
~5.1s latency (20 step)

Almost no performance drop

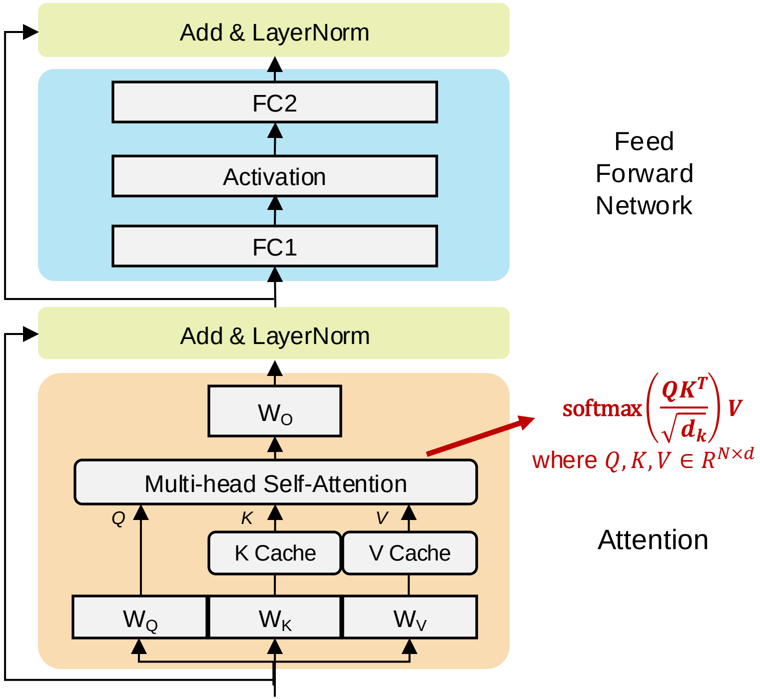
目录 Contents

How LLMs Do Inference

- Most LLMs are based on the Transformer architecture[1].
- A Transformer block consists of :
  - Attention-Linear (generate matrix Q, K, V)
  - Multi-Head Attention
  - Feed Forward Network
  - Layer Norm
- A typical LLM inference process:



Example of Decoder's word-by-word translation



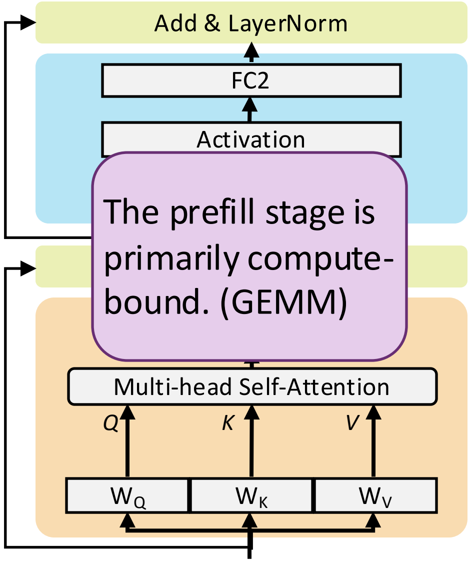
[1] Vaswani, Ashish, et al. "Attention is all you need." NeurIPS 2023.

How LLMs Do Inference

LLM Inference has two stages:

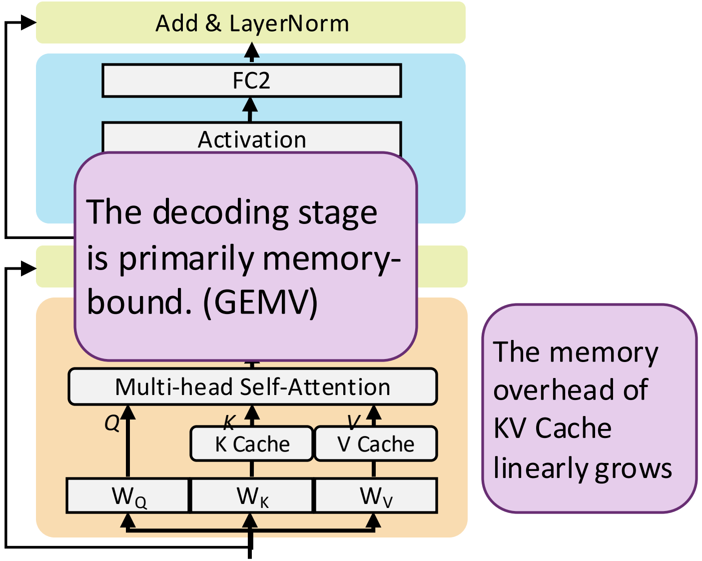
- **Prefill Stage:** takes a prompt sequence to generate the key-value cache (KV Cache)
- **Decode Stage:** utilizes and updates the KV cache to generate tokens one by one, where the current token depends on all the previously tokens

Output: ['Processing'] (1*dim)



Prompt: ['I', 'like', 'natural', 'language'] (4*dim)

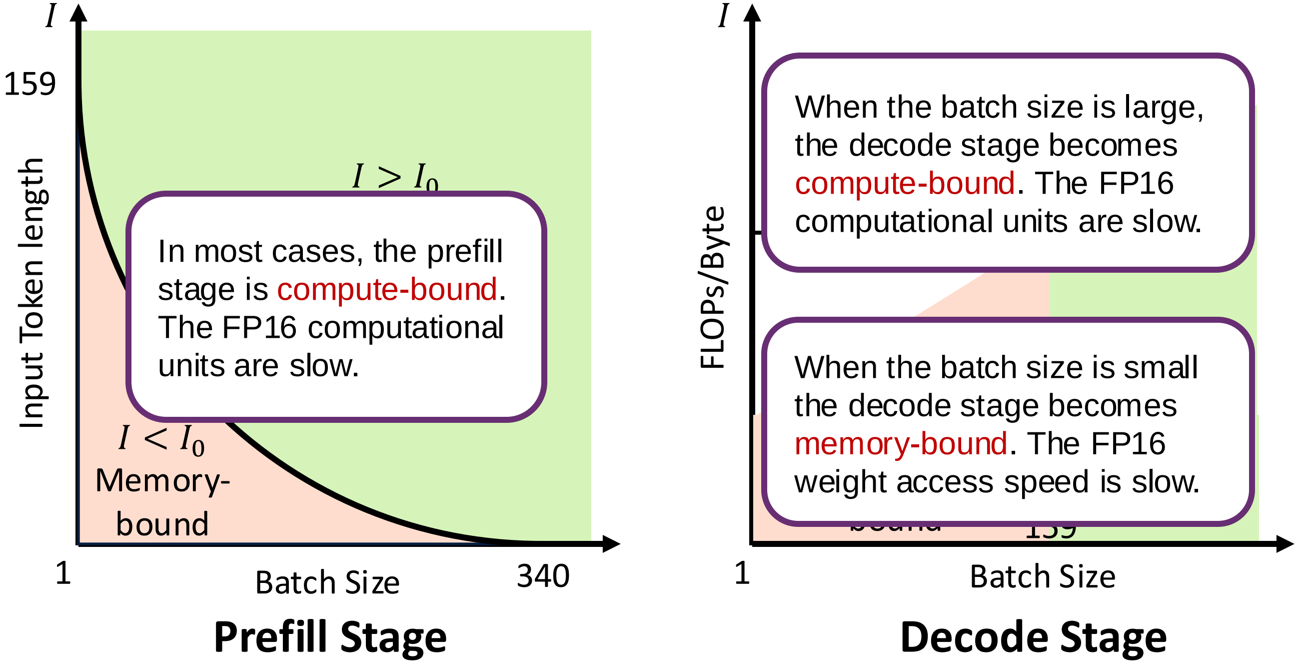
Output: ['!'] (1*dim)



Prompt: ['I', 'like', 'natural', 'language', 'Processing'] (1*dim)

Efficiency Analysis of LLM Inference

- Bottleneck Analysis of Large Parameter Size
  - Take LLaMA3-70B as an example: 8192*8192 linear layer
  - A100 FP16 CUDA Core: I_0 =156 FLOPs/Byte



Efficiency Analysis of LLM Inference

- Bottleneck Analysis of Large Sequence Length

Llama2-7B LLM inference cost

Seq. Length	N	2K	1M
Compute	$O(N^2)$	28.7 TFLOP	**5.9×10^5 TFLOP**
Memory	$O(N)$	15 GB	**526 GB**
First-token latency	$O(N^2)$	150 ms	**30 min**
Generation speed	$O(N)$	88.50 token/s	**0.5 token/s**

System capability and user requirements

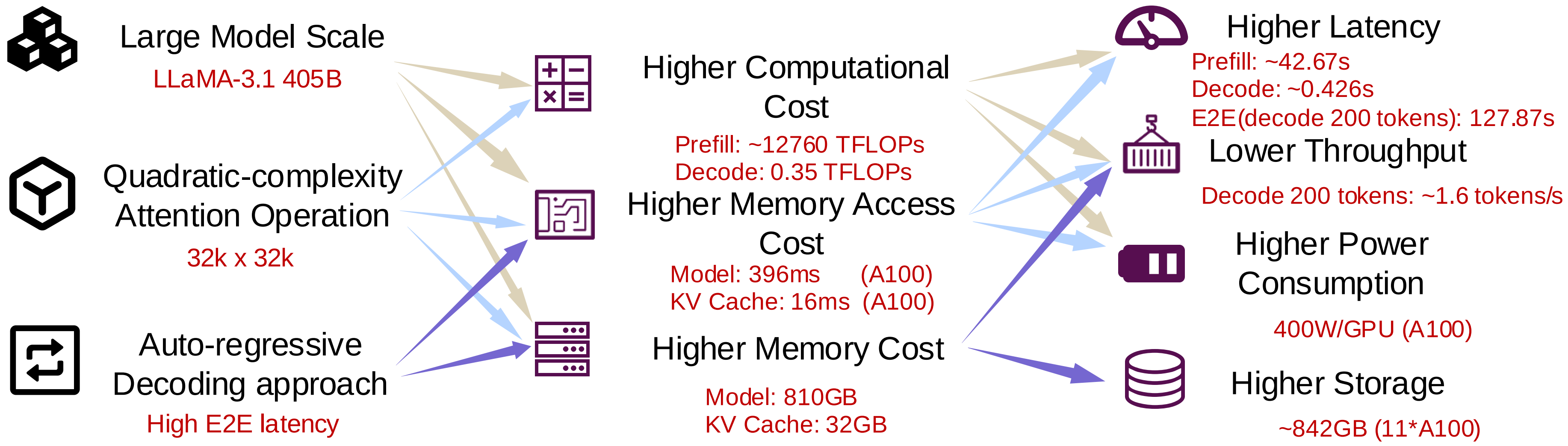
A100 peak computing power* **312 TFLOPS**
A100 maximum GPU memory **80GB**
Waiting causes customer losses; Short-term memory: **15-30s**[1]
Human's average reading speed: **5.4 token/s**[2]

[1] Ubben, Giselle. "How long is short-term memory? Shorter than you might think." Academic Resource Center, Duke University
[2] Brysbaert, Marc. "How many words do we read per minute? A review and meta-analysis of reading rate." Journal of Memory and Language
* with Llama2-7B LLM, Measured on the minimum A100-80GB graphics card that can accommodate the model; Prefill with 1 A100-80GB; Decode with 8 A100-80GB; The A100 peak performance is calculated using the FP16 TensorCore.

Efficiency Analysis of LLM Inference

- Root causes of inefficiency during LLM Inference
  - Model scale: A large number of weights and computations.
  - Attention operation: It has quadratic complexity *w.r.t.* input token length.
  - Decoding approach: Generate tokens one by one (fully sequential).

For example: Deploy LLaMA-3.1 405B in the cloud server



[1] Zhou, Zixuan, **Ning, Xuefei,** et al. "A Survey on Efficient Inference for Large Language Models." arXiv 2024.

Efficient Techniques for LLMs

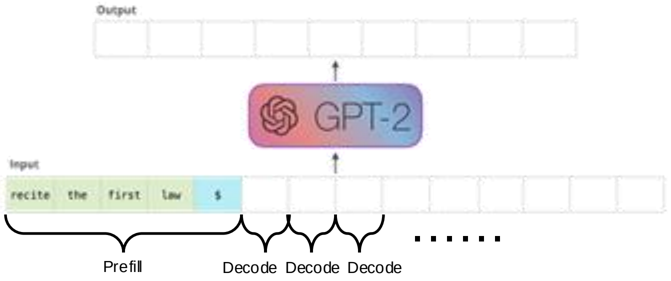
Directions to improve Large Language Models' efficiency

Prefill Stage

Reduce $t_{prefill}$, M_{weight}, M_{other_act}

Decode Stage

Reduce t_{decode}, M_{weight}, $M_{kv\ cache}$



Overall Cost

(for each request)

Total Latency: $t_{prefill} + t_{decode} * N_{token}$

Total Memory: $M_{weight} + M_{kv\ cache} + M_{other\_act}$

Overview of Efficient Techniques

INFINIGENCE 无问芯穹

Lower Latency | Lower Storage | Higher Throughput | Lower Power Consumption

What algorithm property?	Cause what?	Solutions
Model Scale	• Large computation • Large memory access • Large memory footprint	Data-level Optimization: Input Compression; Output Organization
Attention Operation	• Input-quadratic computation • Input-quadratic memory access • Input-quadratic memory footprint	Model-level Optimization: Structure Design; Model Compression
Decoding Approach	• Low arithmetic intensity (i.e., computation / memory access) cause under-utilization • Varying length -> Dynamically increasing KV cache cause fragmented memory, increasing both footprint and access	System-level Optimization: Inference Engine; Serving Framework

[1] Zhou, Zixuan, **Ning, Xuefei**, et al. "A Survey on Efficient Inference for Large Language Models." arXiv 2024.

Overview of Efficient Techniques

Lower Latency | Lower Storage | Higher Throughput | Lower Power Consumption

What algorithm property?	Cause what?	Solutions	
Model Scale	• Large computation • Large memory access • Large memory footprint	• Prompt pruning • Soft prompt tuning • … • Output Organization	Input Compression Output Organization
Attention Operation	• Input-quadratic computation • Input-quadratic memory access • Input-quadratic memory footprint	• Dynamic MoE • Low-complexity attention • Quantization • Sparse Attention • Weight Pruning • …	Structure Design Model Compression
Decoding Approach	• Low arithmetic intensity (i.e., computation / memory access) cause under-utilization • Varying length -> Dynamically increasing KV cache cause fragmented memory, increasing both footprint and access	• Graph and Operator Optimization • Speculative decoding • Memory Management • Batching • …	Inference Engine Serving Framework

[1] Zhou, Zixuan, **Ning, Xuefei**, et al. "A Survey on Efficient Inference for Large Language Models." arXiv 2024.

Overview of Efficient Techniques

INFINIGENCE 无问芯穹

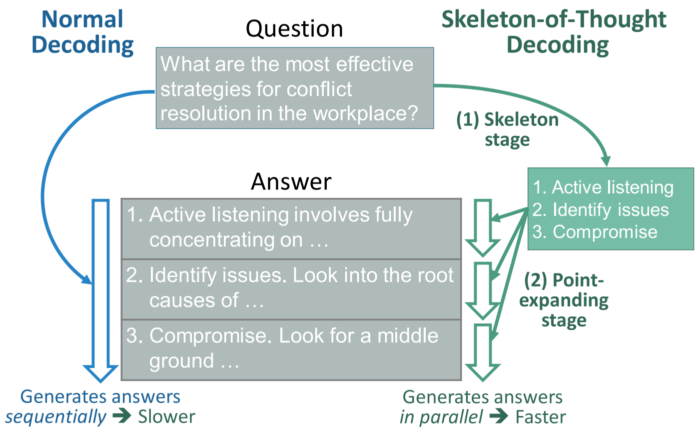
Lower Latency | Lower Storage | Higher Throughput | Lower Power Consumption

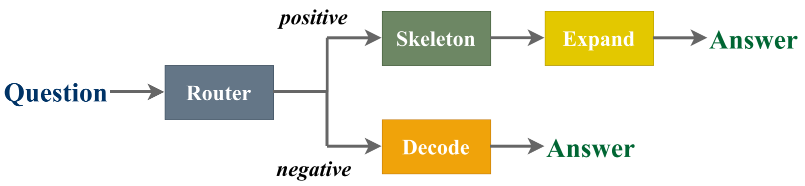
What algorithm property?	Cause what?	Solutions	
Model Scale	• Large computation • Large memory access • Large memory footprint	• Prompt pruning • Soft prompt tuning • … • **Output Organization**	Input Compression Output Organization
Attention Operation	• Input-quadratic computation • Input-quadratic memory access • Input-quadratic memory footprint	• Dynamic MoE • Low-complexity attention • Quantization • Sparse Attention • Weight Pruning • …	Structure Design Model Compression
Decoding Approach	• Low arithmetic intensity (i.e., computation / memory access) cause under-utilization • Varying length -> Dynamically increasing KV cache cause fragmented memory, increasing both footprint and access	• Graph and Operator Optimization • Speculative decoding • Memory Management • Batching • …	Inference Engine Serving Framework

[1] Zhou, Zixuan, **Ning, Xuefei**, et al. "A Survey on Efficient Inference for Large Language Models." arXiv 2024.

Skeleton-of-Thought (SoT)

- Skeleton-of-Thought (SoT) consists of two stages:
  (1) **Skeleton Stage**: Guide the LLM to output a concise skeleton of the answer.
  (2) **Point-expanding Stage**: Guide the LLM to expand on each point from the skeleton in parallel.
- SoT can improve the hardware utilization and decrease the end-to-end latency.





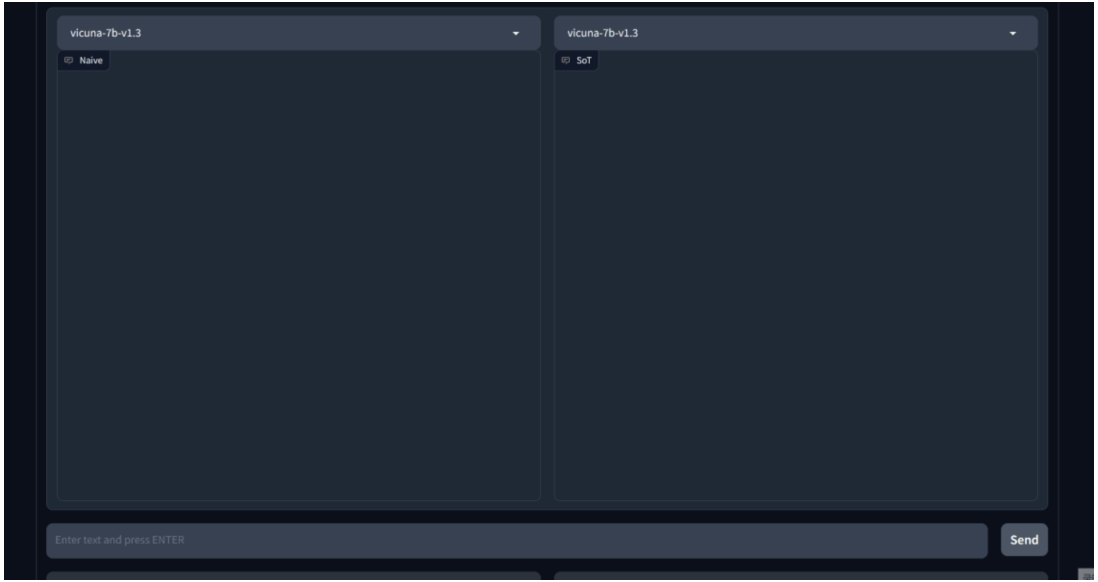
We further extend **SoT with router (SoT-R)** to make the overall solution more practical.

- The router first decides whether to apply the SoT decoding mode based on the user's prompt.

[1] **Ning, Xuefei**, et al. "Skeleton-of-Thought: Large Language Models Can Do Parallel Decoding." ICLR 2024.

Skeleton-of-Thought (SoT)

Accelerating LLM inference by up to 2.39× end-to-end speed-up *without* any changes to their model, system, or hardware



Vicuna-7B model on one A100 GPU: 2.1× end-to-end speed-up compared with sequential decoding

[1] **Ning, Xuefei**, et al. "Skeleton-of-Thought: Large Language Models Can Do Parallel Decoding." ICLR 2024.

Overview of Efficient Techniques

Lower Latency | Lower Storage | Higher Throughput | Lower Power Consumption

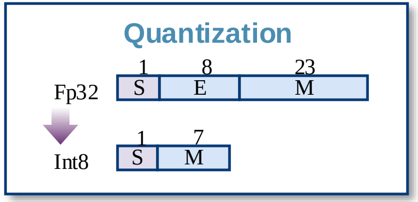
What algorithm property?	Cause what?	Solutions	
Model Scale	• Large computation • Large memory access • Large memory footprint	• Prompt pruning • Soft prompt tuning • … • Output Organization	Input Compression Output Organization
Attention Operation	• Input-quadratic computation • Input-quadratic memory access • Input-quadratic memory footprint	• Dynamic MoE • Low-complexity attention • **Quantization** • Sparse Attention • Weight Pruning • …	Structure Design Model Compression
Decoding Approach	• Low arithmetic intensity (i.e., computation / memory access) cause under-utilization • Varying length -> Dynamically increasing KV cache cause fragmented memory, increasing both footprint and access	• Graph and Operator Optimization • Speculative decoding • Memory Management • Batching • …	Inference Engine Serving Framework

[1] Zhou, Zixuan, **Ning, Xuefei,** et al. "A Survey on Efficient Inference for Large Language Models arXiv 2024.

Quantization Technique

- Quantization is a promising technique to address the aforementioned efficiency issues.
  - Taking **signed uniform** quantization as an example, quantization parameters include

    Scaling Factor, Zero Point, Bitwidth



$$x_{\text{int}} = \text{clip}\left(\left\lfloor \frac{x}{s_x} \right\rceil + z; q_{\min}, q_{\max}\right), \quad \text{where} \quad q_{\max} = 2^{b-1} - 1, \quad q_{\min} = -2^{b-1}$$

  - The Weight-Activation Quantization methods enable the utilization of low-precision Tensor Cores to mitigate the compute-bounded GEMM operators in the prefill stage.
  - The Weight-only Quantization methods prove effective to accelerate the memory-bounded GEMV operators in the decoding stage.
  - The KV Cache Quantization methods are necessary to alleviate the large memory overhead when handling tasks with long contexts or large batch sizes.

Mixed-precision Quantization (LLM-MQ)

INFINIGENCE 无问芯穹

Expected to accelerate linear operators by 1.9~2.7× speed-up via mixed-precision quantization and sparse outliers protection technique

- Assign high bit-width to high-sensitivity layers in order to minimize the change in model output.
  - Use first-order information to estimate the sensitivity:

$$\mathcal{L}(Q_b(\mathbf{W}_i)) \approx \mathcal{L}(\mathbf{W}) + \mathbf{g}_i^T(\mathbf{W}_i - Q_b(\mathbf{W}_i)),$$

  - For each layer, w

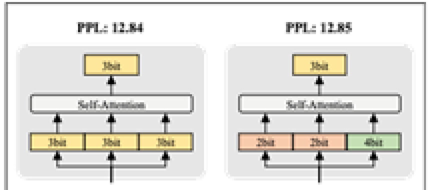
$\min |\mathcal{L}(Q_b(\mathbf{W}_i))| \le |\mathcal{L}(\mathbf{W}$

  - We model the ab following integer

$$\underset{c_{i,b}}{\arg\min} \sum_i^N \sum_b c_{i,b} \cdot s_{i,b},$$

$$s.t. \sum_b c_{i,b} = 1, \quad \sum_i^N \sum_b c_{i,b} \cdot \mathcal{M}(Q_b(\mathbf{W}_i)) \le B,$$

$$c_{i,b} \in \{0,1\}, b \in \{2,3,4\},$$



(i) 3-bit (ii) 2.75-bit

- For zero-shot understanding tasks:
  - When the average accuracy loss is around 0.1%, the model can be quantized to an average of 3.6 bits.
  - When the average accuracy loss is around 1%, the model can be quantized to an average of 2.8 bits.

Does the accuracy on specific tasks sufficiently reflect the effect of quantization on LLMs?

						okqa	Avg. (↑)	Wiki (↓)
						[illegible]	65.08	7.89
						[illegible]	64.67	8.12
						[illegible]	63.23	9.26
						[illegible]	41.74	1056.33
						[illegible]	65.03	8.08
						[illegible]	64.02	8.81
						[illegible]	37.06	5e6
LLM-MQ (Ours)	4.0	79.49	76.31	69.30	58.50	41.20	64.96	8.03
	3.8	79.22	76.22	69.85	58.29	41.40	65.00	8.08
	3.6	79.05	75.88	69.77	58.59	42.20	65.10	8.23
	3.4	79.49	74.77	69.61	58.12	40.60	64.52	8.61
	3.2	79.33	75.12	67.96	57.87	41.60	64.38	8.43
	3.0	79.00	75.08	68.59	57.79	41.00	64.29	8.54
	2.8	78.73	74.32	67.96	57.95	41.20	64.03	8.83
	2.6	78.35	73.81	68.03	57.32	39.40	63.38	9.35
	2.4	77.31	72.93	68.59	54.63	40.00	62.69	10.03
	2.2	76.77	70.83	67.09	55.26	38.40	61.67	10.80
	2.0	75.84	68.32	65.51	54.29	37.20	60.23	12.17

[1] Li, Shiyao, **Ning, Xuefei** et. al., "LLM-MQ: Mixed-precision Quantization for Efficient LLM Deployment." NeurIPS Workshop 2023

Evaluating Quantized LLMs (QLLM Eval)

- Knowledge summary

Knowledge Level	Key Knowledge
Tensor-level	**1. Tensor type (Sec. 3.2)**: The larger the model, the higher the tolerance for Weight-only and KV Cache Quantization, while the tolerance for Activation Quantization is lower. **2. Tensor position (Sec. 3.2)**: The sensitivity to quantization varies significantly across different tensor positions due to their distinct data distributions.
Model-level	**1. (Sec. 3.3)** The relative rankings of quantized LLMs are generally consistent with those of the FP16 LLMs when the bit-width is higher than W4, W4A8, and KV4. **2. (Sec. 3.3)** Leveraging MoE to increase the model size can improve the model's performance but may not improve the tolerance to quantization.
Task-level	**1. Emergent abilities (Sec. 4)**: The tolerance of Multi-Step Reasoning and Self-Calibration to quantization is lower than that of Instruction-Following and In-Context Learning abilities. **2. Dialogue tasks (Sec. 6)**: As the bit-width decreases, sentence-level repetition occurs first, followed by token-level repetition, and token-level randomness. **3. Long-Context tasks (Sec. 7)**: The longer the text, the larger the performance loss caused by Weight and KV Cache quantization. Most LLMs are more sensitive to KV Cache Quantization than Weight-only and Weight-Activation Quantization.
Bit-width Recommendation	**1. Basic NLP tasks (Sec. 3)**: W4, W4A8, KV4, W8KV4. **2. Emergent (Sec. 4)**: W8, W8A8, KV8 ($<$ 13B); W4, W4A8, KV4 ($\geq$ 13B). **3. Trustworthiness (Sec. 5)**: W8, W8A8, KV8 ($<$ 7B); W4, W4A8, KV4 ($\geq$ 7B). **4. Dialogue (Sec. 6)**: W8, W8A8, KV4. **5. Long-Context (Sec. 7)**: W4, W4A8, KV4 (token $<$ 4K); W4, W4A8, KV8 (token $\geq$ 4K). *(Note: Within 2% accuracy loss on the evaluated tasks. The recommended quantization bit-width may not generalize to other LLMs or tasks)*

[1] Li, Shiyao, **Ning, Xuefei,** et al. "Evaluating Quantized Large Language Models." ICML 2024.

Overview of Efficient Techniques

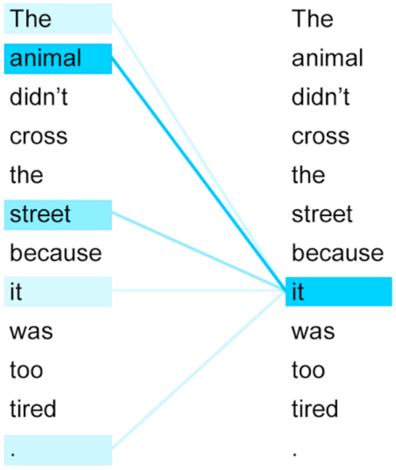
INFINIGENCE 无问芯穹

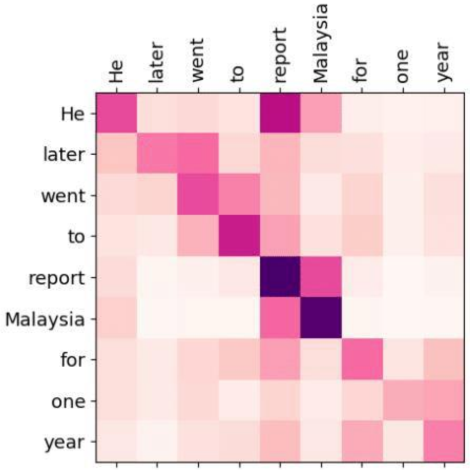
Lower Latency | Lower Storage | Higher Throughput | Lower Power Consumption

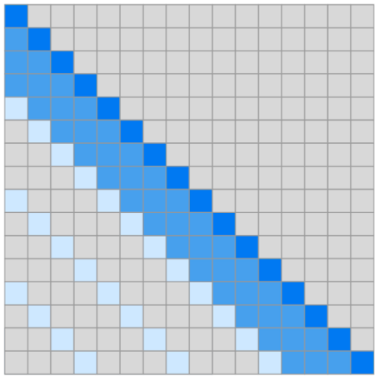
What algorithm property?	Cause what?	Solutions	
Model Scale	• Large computation • Large memory access • Large memory footprint	• Prompt pruning • Soft prompt tuning • … • Output Organization	Input Compression Output Organization
Attention Operation	• Input-quadratic computation • Input-quadratic memory access • Input-quadratic memory footprint	• Dynamic MoE • Low-complexity attention • Quantization • **Sparse Attention** • Weight Pruning • …	Structure Design Model Compression
Decoding Approach	• Low arithmetic intensity (i.e., computation / memory access) cause under-utilization • Varying length -> Dynamically increasing KV cache cause fragmented memory, increasing both footprint and access	• Graph and Operator Optimization • Speculative decoding • Memory Management • Batching • …	Inference Engine Serving Framework

[1] Zhou, Zixuan, **Ning, Xuefei**, et al. "A Survey on Efficient Inference for Large Language Models." arXiv 2024.

Attention Mechanism







Attention Mechanism

each word "looks at" other words in the sentence to determine their **relevance** (attention value) to the current word.

Attention Matrix

Represents the **relevance** between word pairs with **matrix**, showing the the attention values.

Sparse Attention

Each word doesn't need to focus on all words, **only a few relevant** ones, such as nearby context.*

[1] Child, Rewon et al. “Generating Long Sequences with Sparse Transformers.”, arXiv 2019

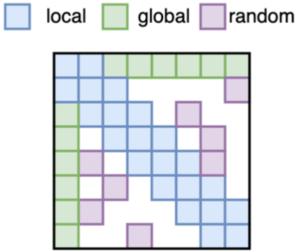
Sparse Attention Methods

INFINIGENCE 无问芯穹

For language understanding models, like BERT

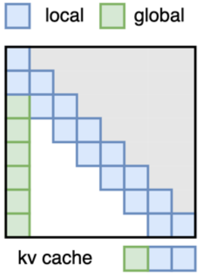
For generative large language models, like GPT

design the **unified** **static** mask to reduce attention computation **during prefill**



BigBird
2020.7

design the **unified** **static** mask to reduce attention computation and KV-cache **during prefill and decode**

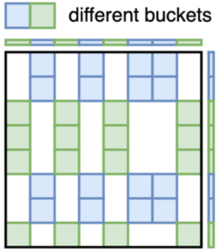


StreamingLLM
2023.9

static

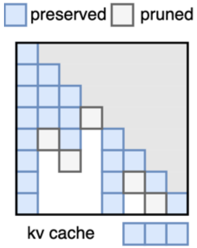
dynamic

Reformer
2020.2



compute attention within the same bucket, **dynamically** allocate tokens to a **unified** number of buckets **during prefill**

H2O
2023.6



dynamically prune previous tokens at a **unified** ratio in the KV-cache **during decode**

[1] Kitaev, Nikita, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The efficient transformer." arXiv 2020
[2] Zaheer, Manzil, et al. "Big bird: Transformers for longer sequences." NeurIPS 2020
[3] Zhang, Zhenyu, et al. "H $ _2 $ O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Languag Models." arXiv 2023
[4] Xiao, Guangxuan, et al. "Efficient Streaming Language Models with Attention Sinks." ICLR 2024.

The Local Context Problem

needle-in-a-haystack task

Below is a record of lines I want you to remember. For each line index, memorize its corresponding <REGISTER_CONTENT>.

long context

line funny-boy: REGISTER_CONTENT is <34836>
line cute-chicken: REGISTER_CONTENT is <28499>
…
line **lovely-dog**: REGISTER_CONTENT is **<28840>**
…
line small-bug: REGISTER_CONTENT is <23550>

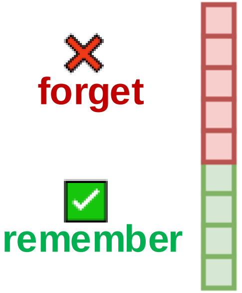
Tell me what is the <REGISTER_CONTENT> in line lovely-dog? I need the number.

The <REGISTER_CONTENT> is **<28840>** ✓

The <REGISTER_CONTENT> is **<23550>** ✗

local attention, local context

local attention[1]
+ global attention on initial tokens



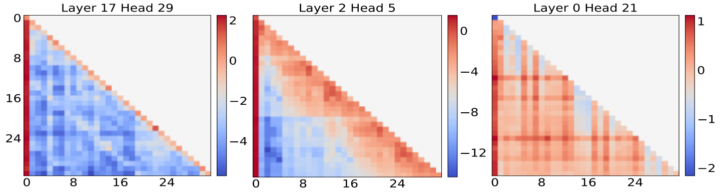
LLM **forgets** the context beyond the **attention span**

[1] Xiao, Guangxuan, et al. "Efficient Streaming Language Models with Attention Sinks." ICLR 2024.

Mixture of Attention Pattern

- Insight： different attention patterns exist in a single LLM

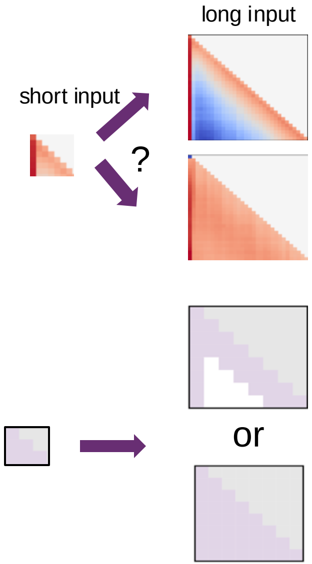
For different attention heads



🔍 different heads show different attention spans

💡 find the optimal attention span for each head

For different input lengths



🔍 different input lengths show different elastic rules

💡 find the optimal elastic rule for each head

[1] Fu, Tianyu, … , **Ning, Xuefei**, et al. "MoA: Mixture of Sparse Attention for Automatic Large Language Model Compression." ICLR 2025 Submission.

Mixture of Sparse Attention (MoA)

INFINIGENCE
无问芯穹

Step1: Dataset

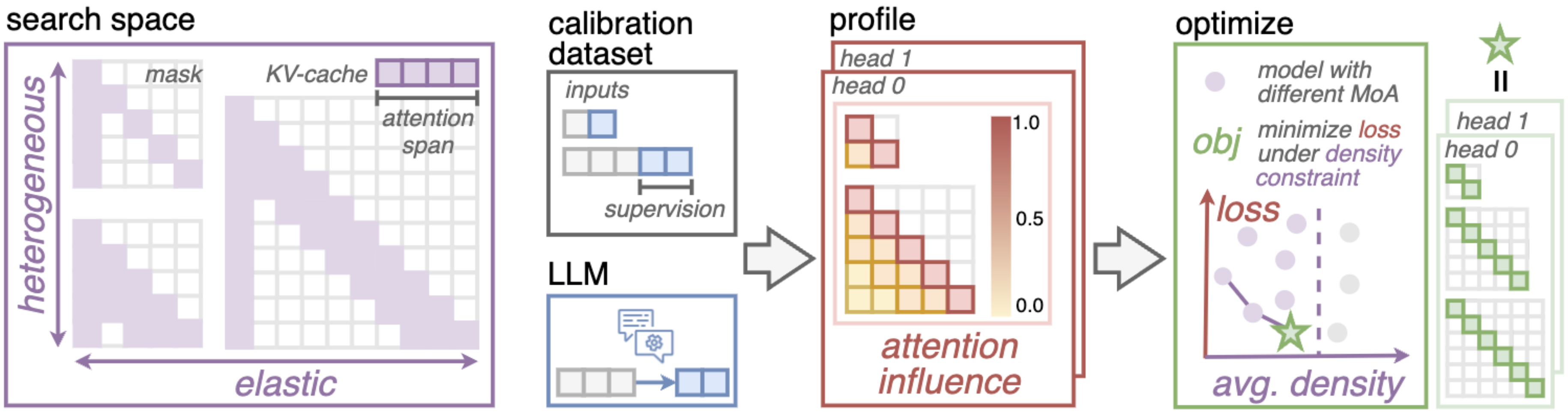
Construct calibration dataset using **long-contextual** MultiNews dataset along with **summarizations generated by original LLM.**

Step2: Profile

Automatically quantify the **influence** of different attention values in LLM on final prediction results, , producing **accuracy-density trade-offs** curves for all schemes.

Step3: Optimize

Select the **optimal elastic rule** for each attention head to **minimize the overall accuracy impact** at a given sparsity level across input lengths.

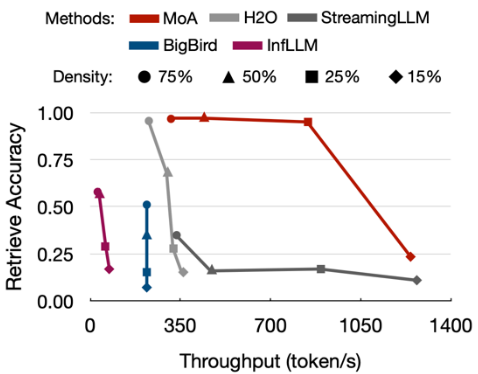


With the masks, large models can **skip** the corresponding attention **computations** and **KV-Cache**, achieving inference efficiency optimization **without needing additional training.**

Mixture of Sparse Attention (MoA)

- Performance overview

Accuracy-Throughput Pareto Front



Pareto front

better trade-off than baselines

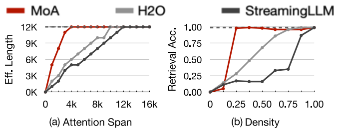
Vicuna-7B model with 8k input length

Open source

project page with code

compression pipeline, CUDA kernel

Long Context Retrieval



Attention	Retrieve Acc. ↑ 32k	64k	128k	256k
Original	0.98	0.93	0.76	0.37
InfLLM	0.43	0.32	0.25	OOT
StreamingLLM	0.52	0.48	0.41	0.25
MoA	**1.00**	**0.92**	**0.83**	**0.46**

3.9x

effective context length

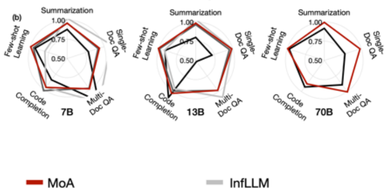
over avg. attention span

256k

generalizable input length

when profile within 8k

Long Context Understanding



Consistent

Benchmark scores across 7-70B models

at 50% density

Efficiency

7x

throughput

over dense FlashAttention2 on single A100-80GB for 7B and 13B LLMs at 50% density

attributed to:

3.0x static-size KV-Cache

1.5x reduced attention

1.4x batch size

1.2x GPU kernel design

Model	Framework	Attention	4k Batch	4k Throughput	8k Batch	8k Throughput	16k Batch	16k Throughput
7B	vLLM	PagedAttention	30	628.8	15	323.0	8	145.5
	FlexGen	H2O	20	754.9	6	296.3	1	51.7
	HuggingFace	InfLLM	15	62.0	10	37.5	6	19.2
	HuggingFace	StreamingLLM	50	945.1	25	467.3	12	232.0
	HuggingFace	FlashAttention2	30	134.6	15	66.9	8	32.9
		+Static KV-Cache	30	496.1	15	219.5	8	91.6
		+Reduced Attention	30	722.5	15	369.9	8	178.3
		+Increased Batch	50	897.7	25	436.7	12	206.4
		+Kernel (=**MoA**)	50	**1099.0**	25	**535.7**	12	**257.3**
13B	vLLM	PagedAttention	16	314.8	8	160.5	4	71.1
	FlexGen	H2O	12	330.2	4	138.2	1	37.4
	HuggingFace	InfLLM	8	30.3	5	17.63	3	11.3
	HuggingFace	StreamingLLM	28	478.4	14	241.2	7	116.5
	HuggingFace	FlashAttention2	16	81.3	8	40.8	4	19.8
		+Static KV-Cache	16	264.6	8	111.3	4	62.2
		+Reduced Attention	16	329.6	8	156.4	4	87.3
		+Increased Batch	28	471.5	14	222.6	7	108.3
		+Kernel (=**MoA**)	28	**550.9**	14	**267.6**	7	**132.3**

Overview of Efficient Techniques

Lower Latency | Lower Storage | Higher Throughput | Lower Power Consumption

What algorithm property?	Cause what?	Solutions	
Model Scale	• Large computation • Large memory access • Large memory footprint	• Prompt pruning • Soft prompt tuning • … • Output Organization	Input Compression Output Organization
Attention Operation	• Input-quadratic computation • Input-quadratic memory access • Input-quadratic memory footprint	• Dynamic MoE • Low-complexity attention • Quantization • Sparse Attention • **Weight Pruning** • …	Structure Design Model Compression
Decoding Approach	• Low arithmetic intensity (i.e., computation / memory access) cause under-utilization • Varying length -> Dynamically increasing KV cache cause fragmented memory, increasing both footprint and access	• Graph and Operator Optimization • Speculative decoding • Memory Management • Batching • …	Inference Engine Serving Framework

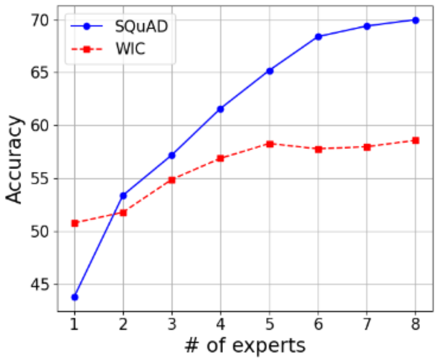
[1] Zhou, Zixuan, **Ning, Xuefei**, et al. "A Survey on Efficient Inference for Large Language Models arXiv 2024.

Efficient Expert Pruning (EEP)

INFINIGENCE 无问芯穹

Construct search space of expert merging and search for coefficients. Can be used to prune active/total expert num.

Adjust number of active expert



Key Observations:

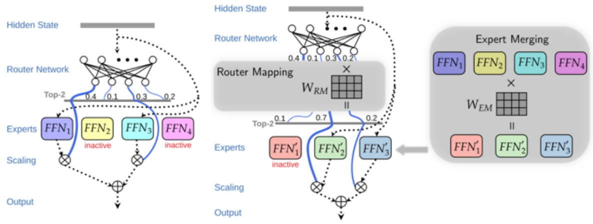
1. Inactive expert can benefit
2. Redundancy exists

Search Space

1. Weight Merging Matrix
2. Routing weights transformation

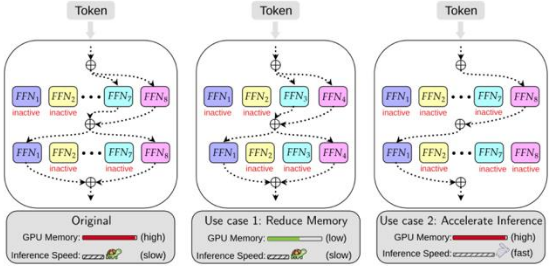
Search Process

1. Discrete (only prune)
2. Continuous (expert merging)



Use Cases

1. Reduce total expert
2. Reduce active expert



[1] Enshu Liu, … , **Xuefei Ning**, et al. "Efficient Expert Pruning for Sparse Mixture-of-Experts Language Models: Enhancing Performance and Reducing Inference Costs" ICLR 2025 Submission.

Efficient Expert Pruning (EEP)

EEP prunes 75%/50% total/active expert while achieves comparable and even better performance. EEP can generalize well on OOD data.

Reduce Total Experts

Expert	Method	COPA	MultiRC	WIC	WSC	RTE	BoolQ	CB	ReCoRD	DROP	SQuAD	Avg.
Num=8	Full Model	89.0	83.0	51.8	63.5	73.2	77.4	51.7	50.3	30.6	53.4	62.4
Num=4	Random	63.8	49.4	37.6	43.3	45.1	50.2	38.7	35.1	27.4	58.3	44.9
	Frequency [37]	63.0	74.8	36.0	34.6	18.1	71.0	30.4	41.6	29.9	58.2	45.8
	Soft Activation [37]	73.0	30.6	51.4	37.5	41.9	40.4	17.9	36.8	33.3	10.2	37.3
	NAEE [34]	87.0	76.0	52.6	64.5	61.7	77.2	51.7	50.4	30.6	53.0	60.5
	EEP (Prune Only)	95.0	81.2	57.8	67.3	74.0	82.8	69.6	60.0	37.3	75.2	70.3
	EEP (Prune+Merge)	**99.0**	**84.6**	**65.0**	**73.1**	**76.9**	**84.8**	**75.0**	**63.6**	**39.7**	**80.6**	**74.2**
Num=2	Random	36.8	22.3	13.6	15.0	28.4	15.5	38.6	16.9	18.3	36.9	24.2
	Frequency [37]	51.0	17.6	8.8	1.9	48.4	30.6	35.7	10.4	14.9	9.2	24.9
	Soft Activation [37]	33.0	18.2	49.4	18.5	15.2	1.8	32.1	4.4	11.7	50.0	23.4
	NAEE [34]	75.0	42.4	48.4	49.0	54.5	49.8	19.6	42.0	31.2	58.2	47.0
	EEP (Prune Only)	76.0	63.8	51.8	63.5	64.3	70.6	58.9	47.2	37.1	64.0	59.7
	EEP (Prune+Merge)	**93.0**	**71.6**	**58.6**	**65.4**	**69.0**	**75.6**	**66.1**	**47.2**	**38.4**	**70.2**	**65.6**

Reduce Active Experts

Total	Active	Method	WIC	WSC	BoolQ	CB	SQuAD	Avg.
8	2	Full Model	51.8	63.5	77.4	51.7	53.4	59.6
	1	Full Model	50.8	48.1	66.0	48.2	43.8	51.4
	1.4~1.5	Dyn [34]	50.0	59.6	72.8	46.4	44.8	54.7
	1	EEP	**59.2**	**70.2**	**79.0**	**66.1**	**51.8**	**65.3**
4	1	NAEE [34]	48.6	20.2	56.2	33.9	51.8	42.1
	1.4~1.5	NAEE+Dyn [34]	43.4	61.5	36.2	53.6	53.4	49.6
	1	EEP	**55.8**	**70.2**	**74.4**	**64.3**	**72.0**	**67.3**

Generalization Test

Budget	Method	IID (50 val. sets)	OOD (7 unseen datasets)
Num=8	Full Model	60.7	72.6
Num=6	Random	53.0±9.6	64.6±10.0
	Frequency [37]	35.2	35.0
	Soft Activation [37]	54.3	65.6
	NAEE [34]	57.5	69.4
	EEP (Prune Only)	59.6	71.4
	EEP (Prune+Merge)	**61.8**	71.3
Num=4	Random	45.1±6.1	50.3±10.7
	Frequency [37]	26.6	25.2
	Soft Activation [37]	46.7	53.1
	NAEE [34]	53.5	63.6
	EEP (Prune Only)	55.4	62.4
	EEP (Prune+Merge)	**56.9**	**64.6**

Expert merging improves the performance of the pruned model

[1] Enshu Liu, … , **Xuefei Ning**, et al. "Efficient Expert Pruning for Sparse Mixture-of-Experts Language Models: Enhancing Performance and Reducing Inference Costs" ICLR 2025 Submission.

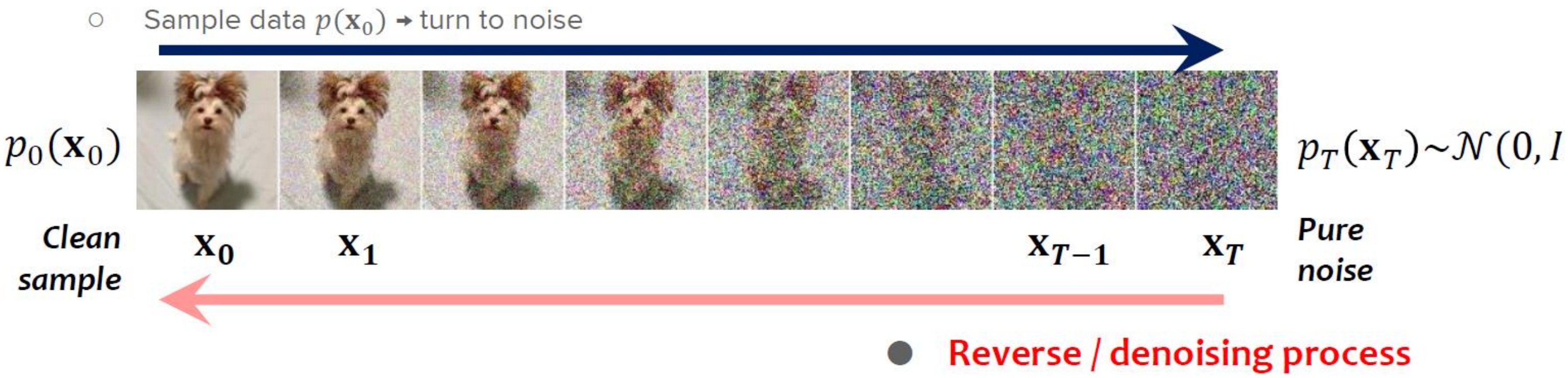
Efficient Techniques for LLMs

Overall Cost (for each request)	**Total Latency:** $t_{prefill} + t_{decode} * N_{token}$ **Total Memory:** $M_{weight} + M_{kv\ cache} + M_{other\_act}$
SoT (Skeleton-of-Thought)	**Total Latency:** $t_{prefill} + t_{decode} * N_{token}\ /B$ **Total Memory:** $M_{weight} + M_{kv\ cache} + M_{other\_act}$
LLM-MQ (Mixed-precision quantization)	**Total Latency:** $t_{prefill} + t_{decode}\downarrow * N_{token}$ **Total Memory:** $M_{weight}\downarrow + M_{kv\ cache} + M_{other\_act}$
MoA (Mixture of Attention)	**Total Latency:** $t_{prefill} + t_{decode}\downarrow * N_{token}$ **Total Memory:** $M_{weight} + M_{kv\ cache}\downarrow + M_{other\_act}$
EEP (Efficient Expert Pruning)	**Total Latency:** $t_{prefill} + t_{decode}\downarrow * N_{token}$ **Total Memory:** $M_{weight}\downarrow + M_{kv\ cache} + M_{other\_act}$

目录 Contents

How DMs do inference

- Forward Process: Gradually add gaussian noise of different levels
- Backward Process: Gradually denoise the gaussian noise
- Intuition: the NN learns to predict the "noise" at each timestep.

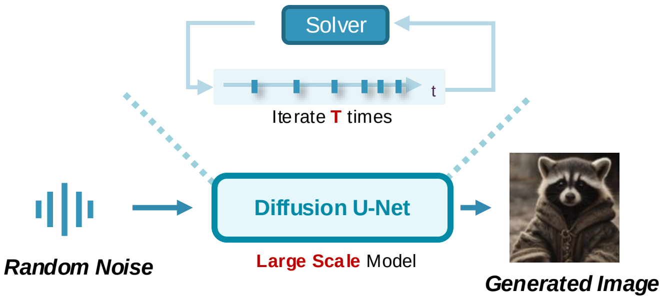


[1] Ho, Jonathan et al. "Denoising Diffusion Probabilistic Models." ArXiv 2020.

Efficiency Analysis of DM Inference

Current visual generation faces efficiency challenge

Large Param Size: 2.5B (SDXL)
Iterative NN Inference: 10-100x



Diffusion Model

Current SOTA
visual generation scheme

Latency Challenge:

SDXL 50 steps
on RTX3090: **30** s

Cannot Satisfy

Image Editing
Needs **Fast (<1s)** Feedback

Memory Challenge:

SDXL model
9.7GB GPU Memory

Cannot Fit In

Desktop GPU: RTX4070
8GB GPU Memory

Efficient Techniques for DMs

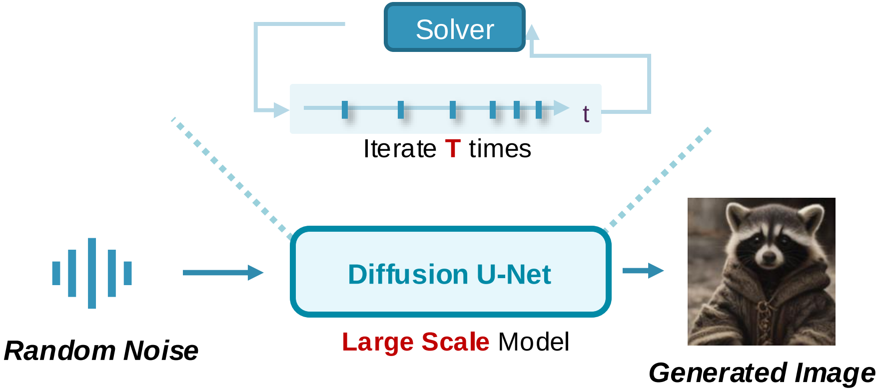
Directions to improve Diffusion Models' efficiency

Algorithm-level

Reduce $N_{timestep}$

Model-level

Reduce t_{model}, M_{weight}, M_{act}



Overall Cost

(for each iter)

Total Latency: $t_{model} * N_{timestep}$

Total Memory: $M_{weight} + M_{activation}$

Overview of Efficient Techniques

INFINIGENCE 无问芯穹

We improve diffusion model's efficiency from Algorithm & Model & Data level

Latency Challenge:

SDXL 50 steps
on RTX3090: 30 s

Cannot Satisfy

Image Editing
Needs **Fast (<1s)** Feedback

Memory Challenge:

SDXL model
9.7GB GPU Memory

Cannot Fit In

Desktop GPU: RTX4070
8GB GPU Memory

Algorithm-level
Time Step Compression

LCSC [ICLR Submission] — Linear combination of checkpoints. **15~23x training acceleration, 1.25~2x timestep compression**

USF [ICLR'24], **OMS-DPM** [ICML'23] — Search for optimal diffusion schedulers. **1.5~2x speed-up**

DD [ICLR Submission] — generates image in **0.01s** and can achieve **>100x** speedup for Image AR model

Model-level
Quantization

MixDQ [ECCV'24] — Mixed-precision quantization. **3x memory decrease, 1.5x speed-up**

ViDiT-Q [ICLR Submission] — Quantization for DiT. **2.5x memory improvement, 1.5x speed-up**

Pruning & Sparse Attention

DiTFastAttn [NeurIPS' 24] — Window & reused attention for DiT. **1.6x speed-up**

Fast Compression

FlashEval [CVPR'24]

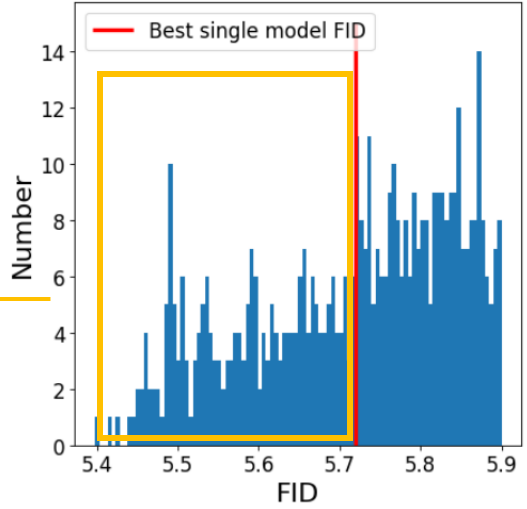
Efficient Diffusion Models

Optimizing the Model Schedule (OMS-DPM)

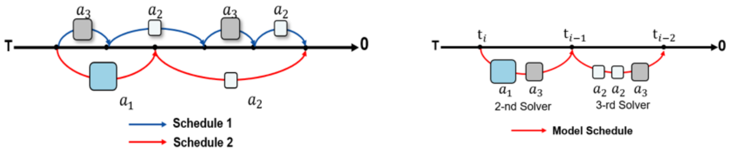
Achieving 2-5× speed-up on typical datasets and 2× speed-up on Text-to-Image generation

Motivation: Small models outperform large models at some timesteps

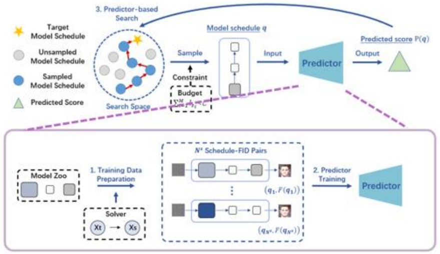
Mixing small and large models can possibly get better FID than only use large model



Methodology: Model Schedule & Predictor-based Search



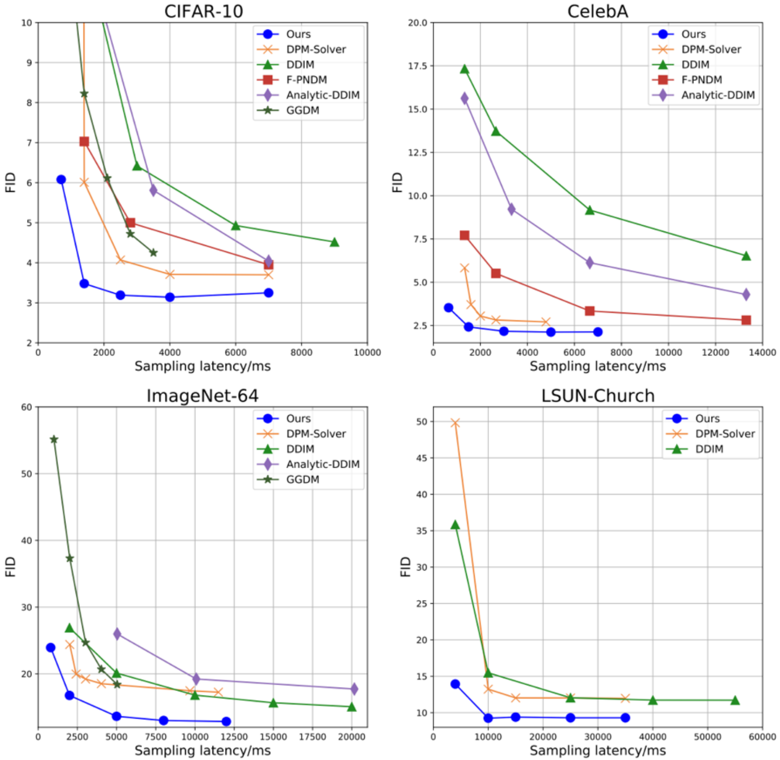
Model Schedule



Search Method

[1] Liu, Enshu, **Ning, Xuefei,** et al. "OMS-DPM: Optimizing the Model Schedule for Diffusion Probabilistic Models." ICML 2023.

Optimizing the Model Schedule (OMS-DPM)



Baseline

Ours

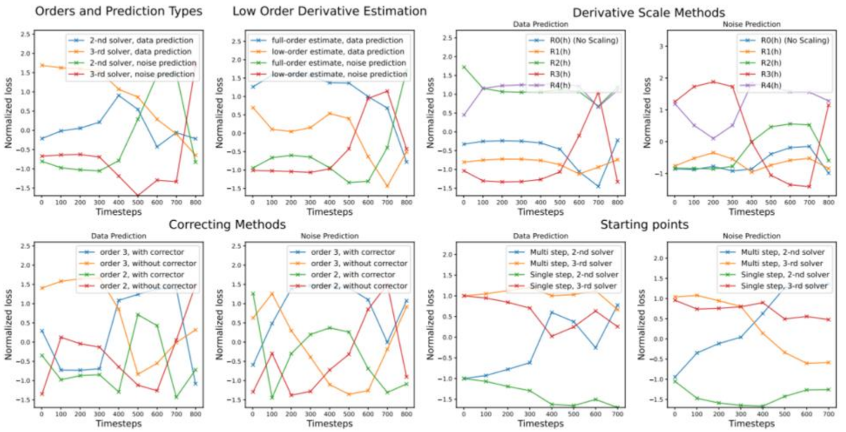
[1] Liu, Enshu, **Ning, Xuefei,** et al. "OMS-DPM: Optimizing the Model Schedule for Diffusion Probabilistic Models." ICML 2023.

Unified Sampling Framework (USF)

INFINIGENCE 无问芯穹

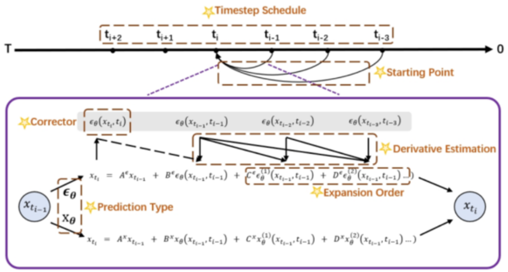
Achieving 2× speed-up on Text-to-Image generation and enables sampling with very low NFE

Motivation: Current solvers use sub-optimal strategies, cause poor quality with few NFE

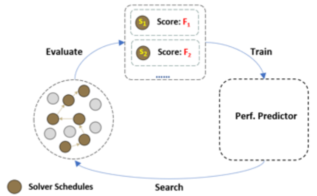


The ranking of all strategies changes over timestep

Methodology: A framework that unifies all exiting solvers and search based on it.



Method framework



Search method

Method	Prediction	Taylor order	Starting point	Scale	Corrector
DPM-Solver-2S	Noise	1,2	-1,-2	$\frac{\frac{h}{2}*(e^h-1)}{e^h-h-1}$	None
DPM-Solver-3S	Noise	1,2,2	-1,-2,-3	None	None
DPM-Solver++(2M)	Data	2	-1	$\frac{\frac{h}{2}*(1+e^{-h})}{e^{-h}+h-1}$	None
UniPC-2-$B_1(h)$	Noise/Data	2	-1	$\frac{\frac{h'^2}{2}}{e^{h'}-h'-1}$	UniC-2
UniPC-2-$B_2(h)$	Noise/Data	2	-1	$\frac{\frac{h'}{2}(e^{h'}-1)}{e^{h'}-h'-1}$	UniC-2
UniPC-p ($p>2$)	Noise/Data	p	-1	None	UniC-p
DEIS-p (in the λ domain)	Noise	p	-1	None	None

USF unifies all solvers

[1] Liu, Enshu, **Ning, Xuefei,** et al. "OMS-DPM: Optimizing the Model Schedule for Diffusion Probabilistic Models." ICML 2023.

Unified Sampling Framework (USF)

INFINIGENCE 无问芯穹

Dataset	Method	NFE						
		4	5	6	7	8	9	10
CIFAR-10	Baseline-W(S)	255.21	288.12	32.15	14.79	22.99	6.41	5.97
	Baseline-W(M)	61.13	33.85	20.84	13.89	10.34	7.98	6.76
	Baseline-B	57.52	23.44	10.33	6.47	5.16	4.30	3.90
	Ours	**11.50**	**6.86**	**5.18**	**3.81**	**3.41**	**3.02**	**2.69**
CelebA	Baseline-W(S)	321.39	330.10	52.04	17.28	16.99	10.39	6.91
	Baseline-W(M)	31.27	20.37	14.18	11.16	9.28	8.00	7.11
	Baseline-B	26.32	8.38	6.72	6.72	5.17	4.21	4.02
	Ours	**12.31**	**5.17**	**3.65**	**3.80**	**3.62**	**3.16**	**2.73**
ImageNet-64	Baseline-W(S)	364.60	366.66	72.47	47.84	54.21	28.22	27.99
	Baseline-W(M)	93.98	69.08	50.35	40.99	34.80	30.56	27.96
	Baseline-B	76.69	61.73	42.81	31.76	26.99	23.89	24.23
	Ours	**33.84**	**24.95**	**22.31**	**19.55**	**19.19**	**19.09**	**16.68**
LSUN-Bedroom	Baseline-W(M)	44.29	24.33	15.96	12.41	10.87	9.99	8.89
	Baseline-B	22.02	17.99	12.43	10.79	9.92	9.11	8.52
	Ours	**16.45**	**12.98**	**8.97**	**6.90**	**5.55**	**3.86**	**3.76**
ImageNet-128	Baseline-W(M)	32.08	15.39	10.08	8.37	7.50	7.06	6.80
	Baseline-B	25.77	13.16	8.89	7.13	6.28	6.06	6.03
	Ours	**18.61**	**8.93**	**6.68**	**5.71**	**5.28**	**4.81**	**4.69**
ImageNet-256	Baseline-W(M)	80.46	54.00	38.67	29.35	22.06	16.74	13.66
	Baseline-B	51.09	27.71	17.62	13.19	10.91	9.85	9.31
	Ours	**33.84**	**19.06**	**13.00**	**10.31**	**9.72**	**9.06**	**9.06**

Results on typical datasets

Method	NFE						
	4	5	6	7	8	9	10
Baseline-W(S)	161.03	156.72	106.15	75.28	58.54	39.26	29.54
Baseline-W(M)	30.77	22.71	19.66	18.45	18.00	17.65	17.54
Baseline-B	24.95	20.59	18.80	17.83	17.54	17.42	17.22
Ours	**22.76**	**16.84**	**15.76**	14.77	**14.23**	**13.99**	**14.01**
Ours-500	24.47	17.72	15.71	**14.60**	14.47	14.15	14.27
Ours-250	23.84	18.27	17.29	14.90	15.50	14.12	14.31

Results on T2I task

Ours

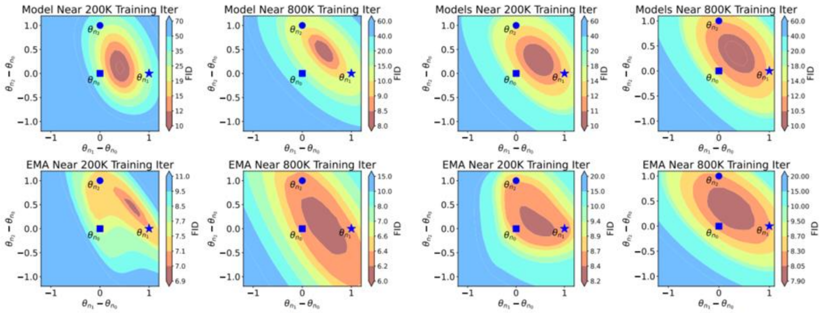
Baseline

[1] Liu, Enshu, **Ning, Xuefei**, et al. "A Unified Sampling Framework for Solver Searching of Diffusion Probabilistic Models." ICLR 2024.

Linear Combination of Saved Checkpoints (LCSC)

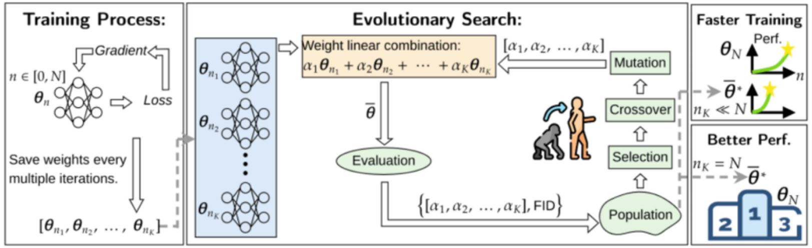
Achieving 15~23× training speed-up on Consistency Models and 1.25~1.7× inference speed-up on Diffusion Models

Motivation: Combination of checkpoints can improve the performance of CM/DM.



(a) Metric Landscape of DM. (b) Metric Landscape of CM.

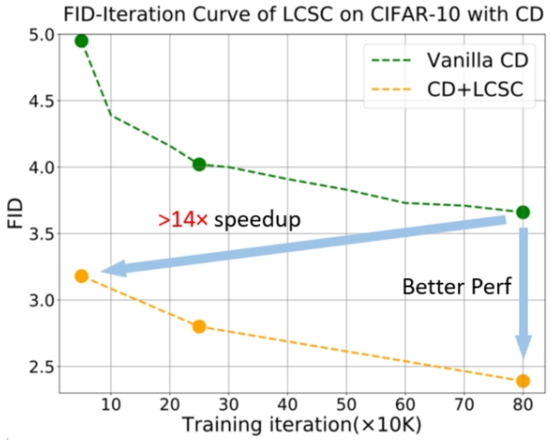
Methodology: Search the combination coefficients of saved checkpoints

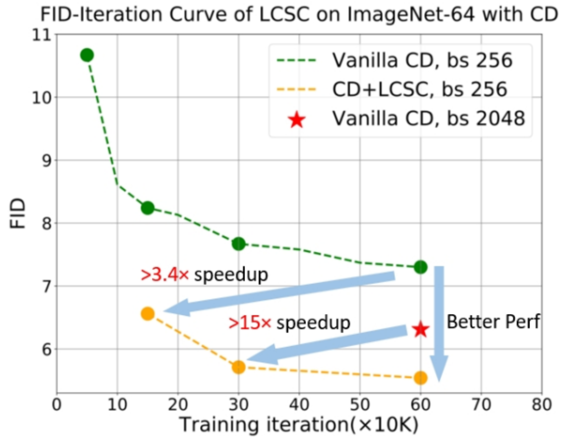


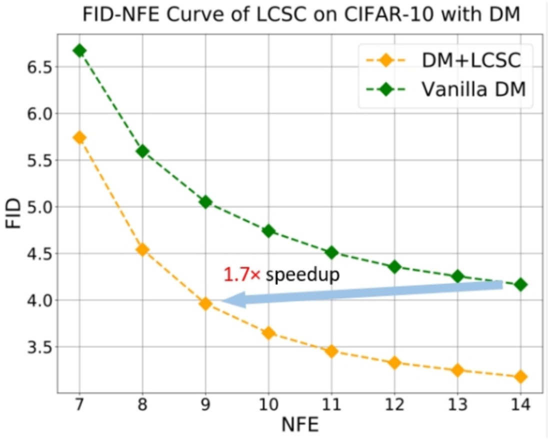
Use Case: accelerate training & enhancing converged models

[1] Liu, Enshu, …, , **Ning, Xuefei**, et al. “Linear Combination of Saved Checkpoints Makes Consistency and Diffusion Models Better.” ICLR 2025 Submission.

Linear Combination of Saved Checkpoints (LCSC)







Baseline（FID=7.30）

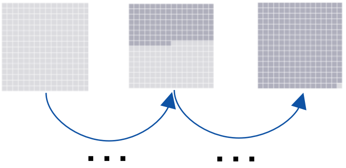
Ours（FID=5.54）

[1] Liu, Enshu, ..., **Ning, Xuefei**, et al. "Linear Combination of Saved Checkpoints Makes Consistency and Diffusion Models Better." ICLR 2025 Submission.

Distilled Decoding of Image AR model (DD)

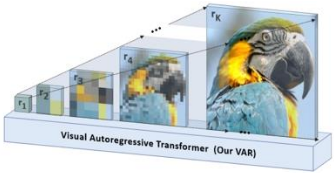
Motivation1: Auto-regressive (AR) image generation model takes too many steps to generate

LlamaGen[2]



>200 steps
>5s/img

VAR[3]



10 steps
~0.13s/img

Motivation2: Typical solution don't work: modeling the distribution of multiple steps simultaneously

Ignore the correlation and introduce the gap between

$$\prod_{i=k+1}^{m} p(q_i|q_k,\dots,q_1) \ \& \ p(q_m,\dots,q_{k+1}|q_k,\dots,q_1)$$

1step generation

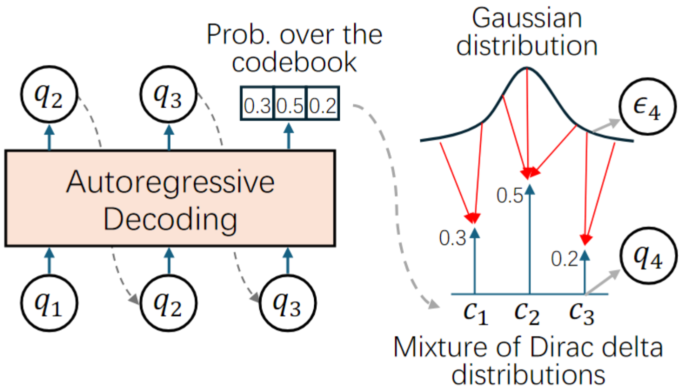
[1] Liu, Enshu, **Ning Xuefei**, et al. "Distilling Autoregressive Models Into Few Steps 1: Image Generation." ICLR 2025 Submission.
[2] Sun, Peize, et al. "Autoregressive Model Beats Diffusion: Llama for Scalable Image Generation." Arxiv 2024
[3] Tian, Keyu, et al. "Visual Autoregressive Modeling: Scalable Image Generation via Next-Scale Prediction." NeurIPS 2024.

Distilled Decoding of Image AR model (DD)

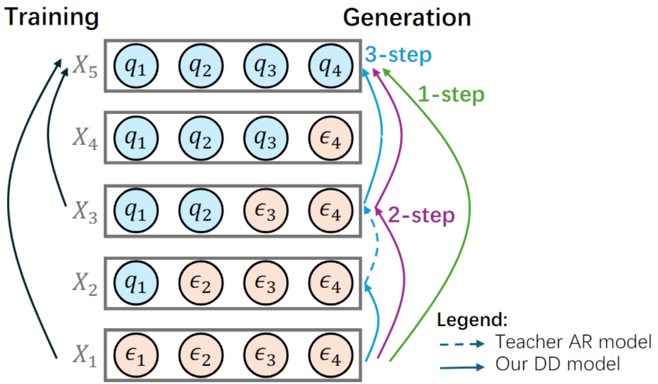
Methodology:

- Introduce noise token and flow-matching to construct an auto-regressive trajectory
- Train the model to skip further along the trajectory

1. Construct the trajectory



2. Training & Sampling



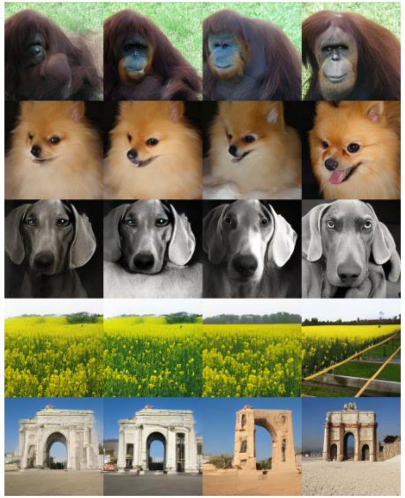
[1] Liu, Enshu, **Ning Xuefei**, et al. "Distilling Autoregressive Models Into Few Steps 1: Image Generation." ICLR 2025 Submission.

Distilled Decoding of Image AR model (DD)

Results: DD generates image in 0.01s and can achieve >100x speedup for Image AR model with acceptable performance loss.

Type	Model	FID↓	IS↑	Pre↑	Rec↑	#Para	#Step	Time
AR	VAR (Tian et al., 2024)	4.19	230.2	0.84	0.53	310M	10	0.133
AR	LlamaGen (Sun et al., 2024)	6.53	291.8	0.86	0.42	343M	256	5.01
Baseline	VAR-*skip-1*	9.52	178.9	0.68	0.54	310M	9	0.113
Baseline	VAR-*skip-2*	40.09	56.8	0.46	0.50	310M	8	0.098
Baseline	VAR-*onestep**	157.5	—	—	—	—	1	—
Baseline	LlamaGen-*skip-106*	19.14	80.39	0.42	0.43	343M	150	2.94
Baseline	LlamaGen-*skip-156*	80.72	12.13	0.17	0.20	343M	100	1.95
Baseline	LlamaGen-*onestep**	220.2	—	—	—	—	1	—
Ours	VAR-DD	7.86	185.1	0.80	0.41	327M	**1**	**0.021** (6.3×)
Ours	VAR-DD	10.65	168.1	0.79	0.37	327M	2	0.036 (3.7×)
Ours	LlamaGen-DD	17.98	179.6	0.79	0.20	326M	**1**	0.023 (**217.8**×)
Ours	LlamaGen-DD	11.24	235.1	0.85	0.30	326M	2	0.043 (116.5×)
Ours	VAR-*pre-trained-1-6*	5.90	241.3	0.85	0.40	327M	6	0.090 (1.5×)
Ours	VAR-*pre-trained-4-6*	6.10	229.5	0.85	0.39	327M	4	0.062 (2.1×)
Ours	VAR-*pre-trained-5-6*	6.62	208.5	0.83	0.40	327M	3	**0.045** (2.6×)
Ours	LlamaGen-*pre-trained-1-81*	10.30	271.2	0.88	0.35	326M	81	1.725 (2.9×)
Ours	LlamaGen-*pre-trained-41-81*	10.43	266.2	0.88	0.33	326M	42	0.880 (5.7×)
Ours	LlamaGen-*pre-trained-61-81*	10.62	255.4	0.87	0.31	326M	22	0.447 (**11.2**×)

1step 2step 42step 256step



[1] Liu, Enshu, **Ning Xuefei**, et al. "Distilling Autoregressive Models Into Few Steps 1: Image Generation." ICLR 2025 Submission.

Motivation: Diffusion Quantization

The text-to-image/video diffusion models are **memory-intensive**,
and cannot be deployed on **Edge** Devices (Even Desktop GPU)

OPEN SORA

Open-SORA 2s Video
~10 GB Peak Memory

FP16 SDXL 512x512
9.7GB Peak Memory

>

Desktop GPU: RTX4070
8GB GPU Memory

Mobile: IPhone 14
6GB Memory

Solution: Model Quantization, **low-bit data** storing and computing, reduce the memory cost

Mixed-precision Quantization (MixDQ)

Motivation: Few-step text-to-image diffusion models face additional challenge for quantization

FP16

Q-Diff (W8A8)

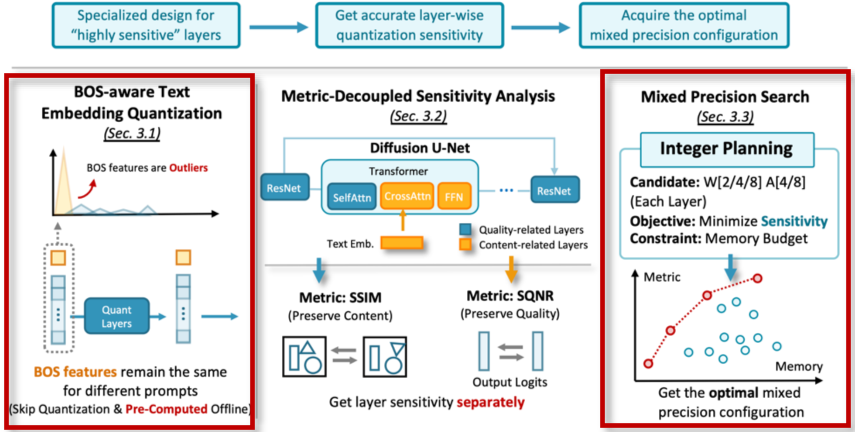
"Two sheep are standing side by side behind a fence."

(* Adopting Q-Diffusion for 1-step SDXL-turbo model)

Solutions:

- BOS-aware Quantization Technique
  "Address Outliers in Text Embeddings"
- Mixed-precision Bit-width Allocation
  "Address over-sensitive layers"

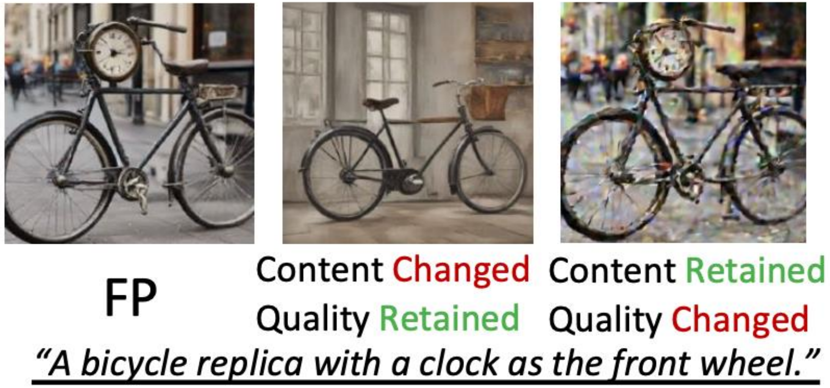
MixDQ WorkFlow



[1] Zhao, Tianchen, **Ning, Xuefei**, et al. "MixDQ: Memory-Efficient Few-Step Text-to-Image Diffusion Models with Metric-Decoupled Mixed Precision Quantization." ECCV 2024.

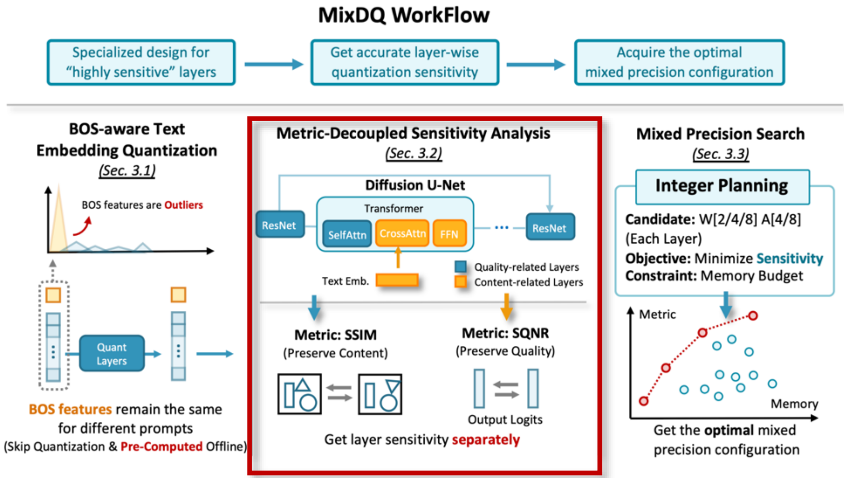
Mixed-precision Quantization (MixDQ)

Motivation: Quantization affects both the image quality & content



Solution:

- "Metric-decoupled" analysis and mixed precision



[1] Zhao, Tianchen, **Ning, Xuefei**, et al. "MixDQ: Memory-Efficient Few-Step Text-to-Image Diffusion Models with Metric-Decoupled Mixed Precision Quantization." ECCV 2024.

Experimental Results: MixDQ improves **both image quality & text alignment**

Achieves W4A8 with negligible loss(+0.5 FID), while baseline methods fail at W8A8 (+50 FID)

Model	Method	Bit-width (W/A)	*Storage Opt.*	*Compute Opt.*	FID(↓)	CLIP(↑)	IR(↑)
SDXL-turbo (1 step)	FP	16/16	-	-	17.15	0.2722	0.8631
	Naive PTQ	8/16	2×	1×	16.89	0.2740	0.8550
		4/16	4×	1×	301.49	0.1581	-2.2526
		8/8	2×	4×	103.96	0.1478	-1.7446
		4/8	4×	8×	358.894	0.1242	-2.2815
	Q-Diffusion	8/16	2×	1×	16.97	0.2735	0.8588
		4/16	4×	1×	22.58	0.2685	0.6847
		8/8	2×	4×	76.18	0.1772	-1.3112
		4/8	4×	8×	118.93	0.1662	-1.6353
	MixDQ(Ours)	4/16	4×	1×	17.23	0.2693	0.8254
		3.66/16	4.4×	1×	17.40	0.2682	0.7528
		8/8	2×	4×	17.03	0.2703	0.8415
		5/8	3.2×	8×	17.23	0.2697	0.8307
		4/8	4×	8×	17.68	0.2698	0.7822
LCM-lora (4 steps)	FP	16/16	-	-	25.56	0.2570	0.2122
	Naive PTQ	8/8	2×	4×	23.36	0.2548	0.0517
		4/8	4×	8×	87.36	0.2055	-1.6160
	Q-Diffusion	8/8	2×	4×	23.92	0.2561	0.1875
		4/8	4×	8×	57.73	0.2280	-1.1863
	MixDQ(Ours)	8/8	2×	4×	22.54	0.2552	0.1573
		4/8	4×	8×	33.48	0.2403	-0.6732

[1] Zhao, Tianchen, **Ning, Xuefei**, et al. "MixDQ: Memory-Efficient Few-Step Text-to-Image Diffusion Models with Metric-Decoupled Mixed Precision Quantization." ECCV 2024.

Mixed-precision Quantization (MixDQ)

INFINIGENCE 无问芯穹

Practical 1.45× speed-up and 2× memory saving on Nvidia GPU
Open-source tool that achieves speedup and support few-step models

FP16 | MixDQ W8A8

2x U-Net Memory Opt.: **4.8 GB -> 2.4 GB**

1.45x U-Net Latency Opt.: **36.1 ms -> 24.9 ms**

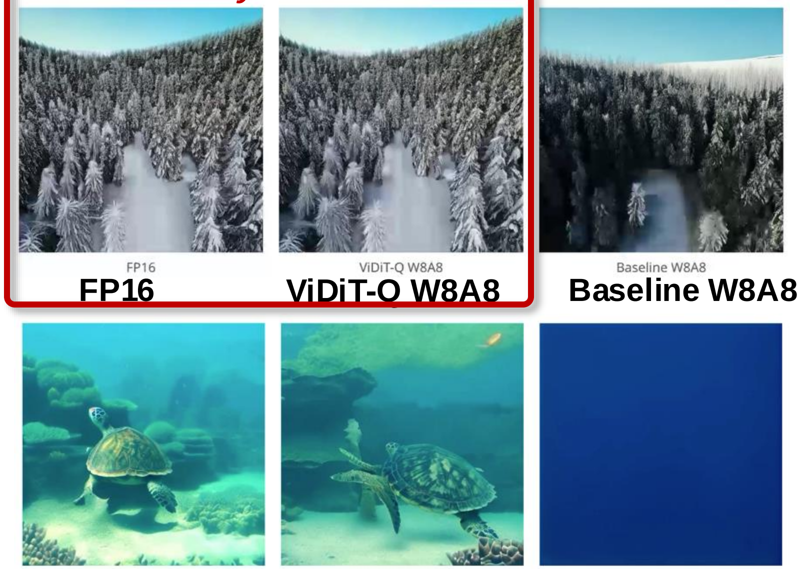
	Reduce VRAM	No Visual Degradation	Latency Speedup	Open Source	Support Few-step
Stable Diffusion WebUI - **FP8**	✓	✗	✗	✓	✗
Huggingface DiffusionFast: **Dynamic Quantization**	✓	✓	✗	✓	✗
Nvidia TensorRT: **Q-Diffusion PTQ**	✓	✓	✓	✗	✗
Ours: **MixDQ**	✓	✓	✓	✓	✓

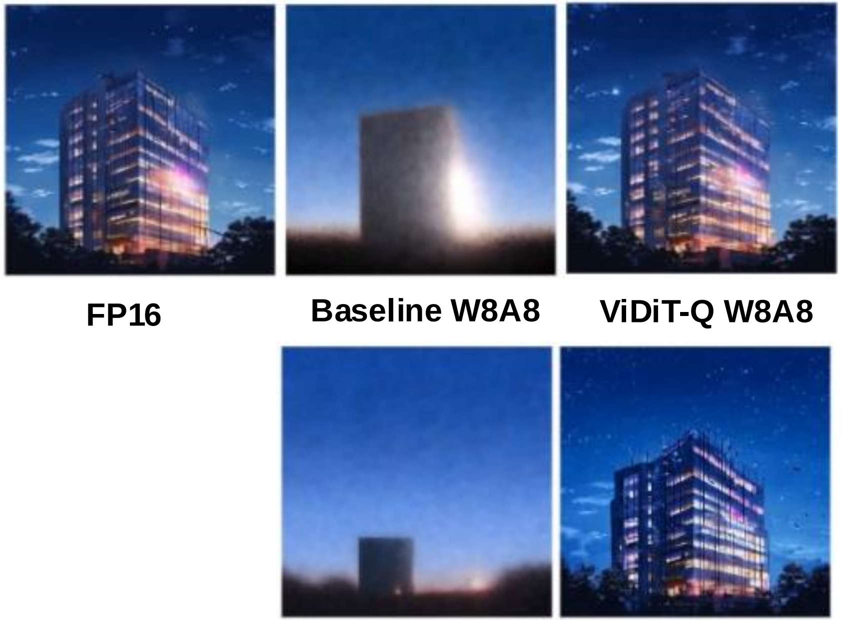
[1] Zhao, Tianchen, **Ning, Xuefei**, et al. "MixDQ: Memory-Efficient Few-Step Text-to-Image Diffusion Models with Metric-Decoupled Mixed Precision Quantization." ECCV 2024.

Video and Image DiT Quantization (ViDiT-Q)

INFINIGENCE 无问芯穹

DiT Quantization for Image and Video Generation





[1] Zhao, Tianchen, …, **Ning, Xuefei**, et al. "ViDiT-Q: Efficient and Accurate Quantization of Diffusion Transformers for Image and Video Generation" ICLR 2025 Submission

Video and Image DiT Quantization (ViDiT-Q)

Motivation: DiT (Diffusion Transformers) have unique properties for quantization

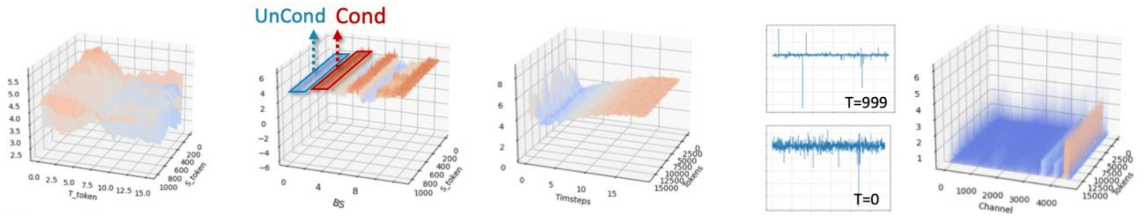
Solution: Quantization scheme tailored for DiTs

Unique challenges for quantizing DiT

- “highly variant activation along different levels”
- “Time-varying” Channel Imbalance

Static-Dynamic Channel Balance

- Combine the advantage of current scale-based (AWQ) and rotation-based (Quarot) channel balancing methods



Token-wise Variation

CFG-wise Variation

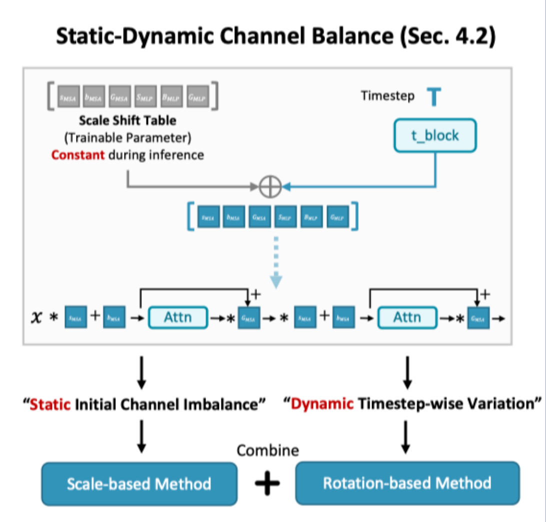
Timestep-wise Variation

Time-varying Channel-wise Variation

Resolve

Fine-grained grouping and Dynamic Quantization (Sec 4.1)

Static-Dynamic Channel Balance (Sec 4.2)



[1] Zhao, Tianchen, …, **Ning, Xuefei**, et al. “ViDiT-Q: Efficient and Accurate Quantization of Diffusion Transformers for Image and Video Generation” ICLR 2025 Submission

Video and Image DiT Quantization (ViDiT-Q)

Motivation: Video generation task have unique properties for quantization

Solution: Quantization scheme tailored for visual generation task

Quantization has effects on **multiple aspects** of visual generation

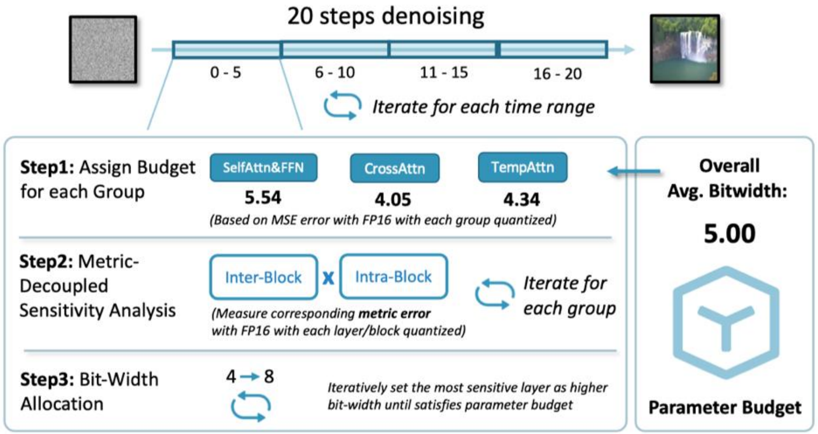
Text Alignment

Visual Quality (Fidelity)

Time Consistency

Decouple the quantization's effect on multiple aspects
To preserve performance for multiple aspects

Metric-Decoupled Mixed Precision (Sec. 4.3)



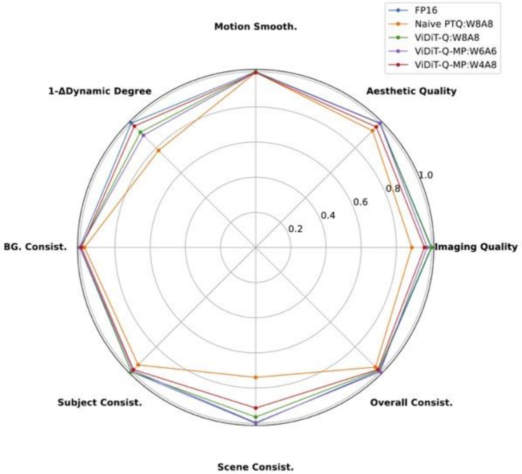
[1] Zhao, Tianchen, …, **Ning, Xuefei**, et al. "ViDiT-Q: Efficient and Accurate Quantization of Diffusion Transformers for Image and Video Generation" ICLR 2025 Submission

Video and Image DiT Quantization (ViDiT-Q)

Achieve superior performance for multiple aspects

Comprehensive Benchmark

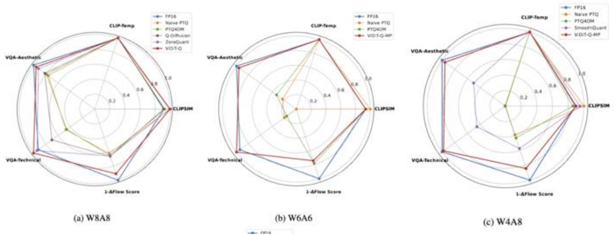
VBENCH
Comprehensive Benchmark Suite for Video Generative Models



Similar performance with FP16

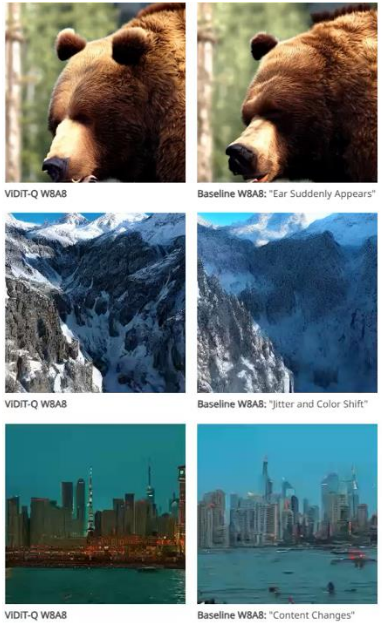
Multiple Metrics

Method	Bit-width (W/A)	CLIPSIM	CLIP-Temp	VQA-Aesthetic	VQA-Technical	Δ Flow Score. (↓)
-	16/16	0.1797	0.9988	63.40	50.46	-
Q-Diffusion	8/8	0.1781	0.9987	51.68	38.27	0.328
Q-DiT	8/8	0.1788	0.9977	61.03	34.97	0.473
PTQ4DiT	8/8	0.1836	0.9991	54.56	53.33	0.440
SmoothQuant	8/8	0.1951	0.9986	59.78	51.53	0.331
Quarot	8/8	0.1949	0.9976	58.73	52.28	0.215
ViDiT-Q	8/8	0.1950	0.9991	60.70	54.64	0.089
Q-DiT	6/6	0.1710	0.9943	11.04	1.869	41.10
PTQ4DiT	6/6	0.1799	0.9976	59.97	43.89	0.997
SmoothQuant	6/6	0.1807	0.9985	56.45	48.21	29.26
Quarot	6/6	0.1820	0.9975	61.47	53.06	0.146
ViDiT-Q	6/6	0.1791	0.9984	64.45	51.58	0.625
Q-DiT	4/8	0.1687	0.9833	0.007	0.018	3.013
PTQ4DiT	4/8	0.1735	0.9973	2.210	0.318	0.108
SmoothQuant	4/8	0.1832	0.9983	31.96	22.85	0.415
Quarot	4/8	0.1817	0.9965	47.36	33.13	0.326
ViDiT-Q	4/8	0.1809	0.9989	60.62	49.38	0.153



Outperform baseline quantization methods

Qualitative Examples



[1] Zhao, Tianchen, …, **Ning, Xuefei**, et al. "ViDiT-Q: Efficient and Accurate Quantization of Diffusion Transformers for Image and Video Generation" ICLR 2025 Submission

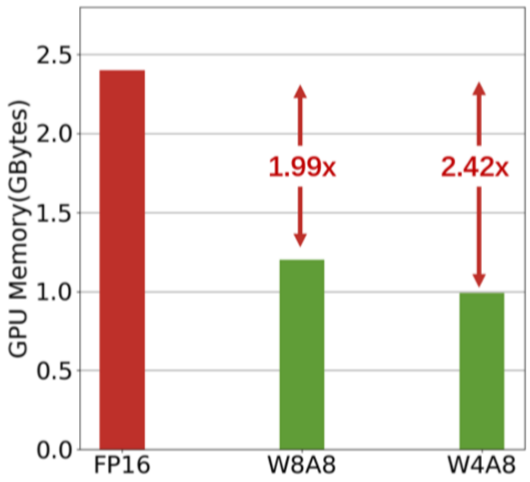
Video and Image DiT Quantization (ViDiT-Q)

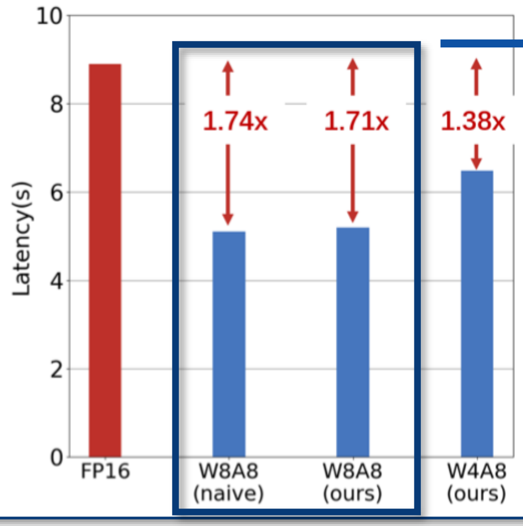
INFINIGENCE 无问芯穹

Achieve Efficiency Improvement with CUDA kernels

Practical Hardware Resource Savings:

- **W8A8:** 1.99x Memory, 1.71x Latency Speedup
- **W4A8:** 2.42x Memory, 1.38x Latency Speedup





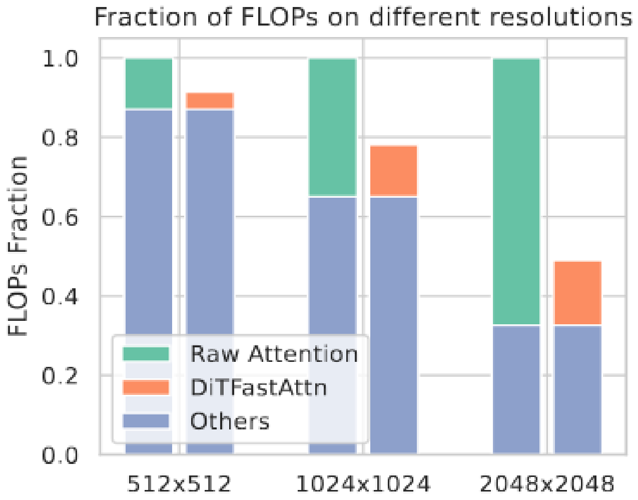
ViDiT-Q improved quantization technique introduces negligible overhead while improving performance, It achieves similar speedup compared with naïve quantization scheme

[1] Zhao, Tianchen, …, **Ning, Xuefei**, et al. "ViDiT-Q: Efficient and Accurate Quantization of Diffusion Transformers for Image and Video Generation" ICLR 2025 Submission

Attention Compression (DiTFastAttn)

INFINIGENCE 无问芯穹

Reduces up to 76% of the attention FLOPs.
Achieve up to 1.8x speedup of DiT models on 2Kx2K generation.
Support both image generation and video generation.



Motivation:

1. Attention can be computationally costly, especially when processing a large number of tokens, as is the case in high-resolution image generation and long-form video generation tasks.

2. Diffusion Transformer (DiT) are emerging as a popular model for image and video generation tasks. There are some redundant computations involved in the generation process when using DiT models.

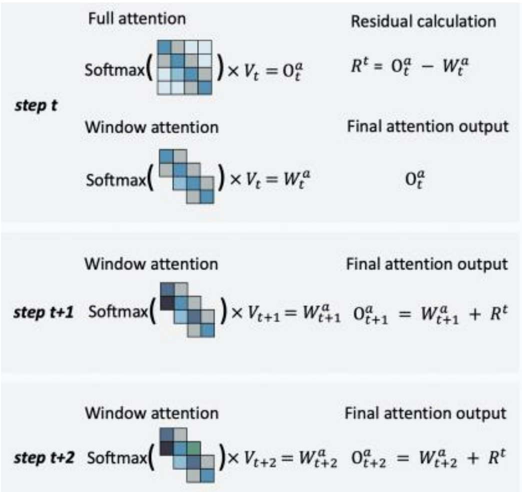
[1] Yuan Zhihang, …, Ning Xuefei et al. "DiTFastAttn: Attention Compression for Diffusion Transformer Models". NeurIPS 2024.

Attention Compression (DiTFastAttn)

INFINIGENCE 无问芯穹

Method 1: Window attention with residual share

The changes in the attention output across different timesteps are primarily driven by a local attention window.

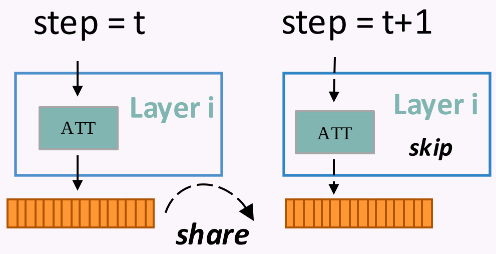


In each timestep, we only compute the local attention and then add the residual of previous global attention, without the need to recompute the full global attention.

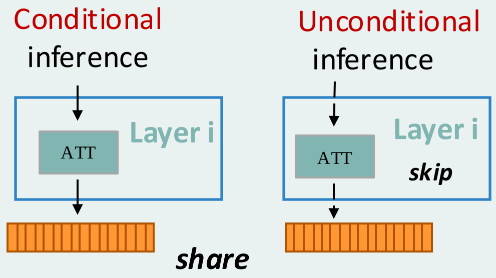
Method 2 & Method 3:

Attention sharing across steps & CFG

Attention Sharing across Step



Attention Sharing across CFG



[1] Yuan Zhihang, …, Ning Xuefei et al. "DiTFastAttn: Attention Compression for Diffusion Transformer Models". NeurIPS 2024.

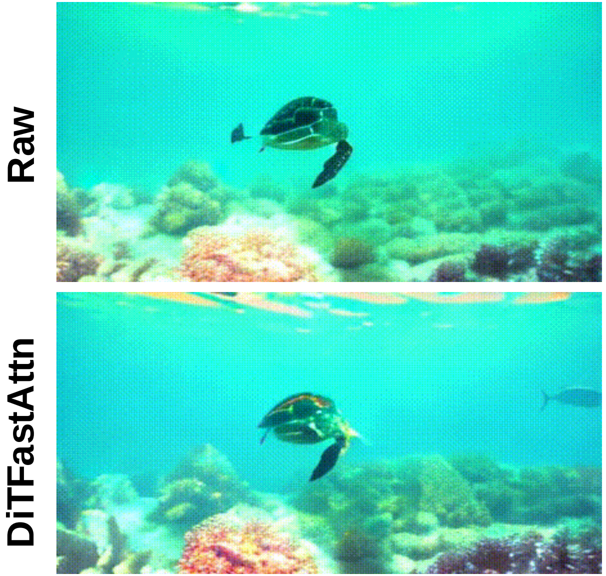
Attention Compression (DiTFastAttn)

Apply to 2K Image Generation

Experiments on PixArt-Sigma

Apply to Video Generation

Experiments on OpenSORA

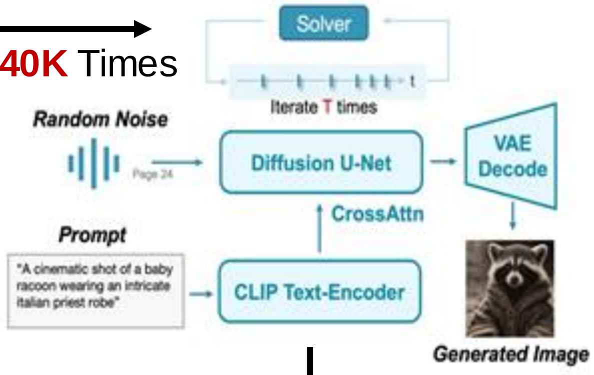


[1] Yuan Zhihang, …, Ning Xuefei et al. “DiTFastAttn: Attention Compression for Diffusion Transformer Models”. NeurIPS 2024.

Faster Evaluation (FlashEval)

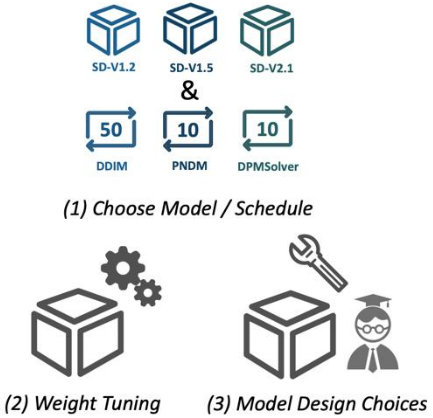
Motivation: Text-to-image Diffusion Evaluation is **slow,**
Many applications requires repeated evaluation

Iterative **40K** Times



Prompt Set
(~40K)

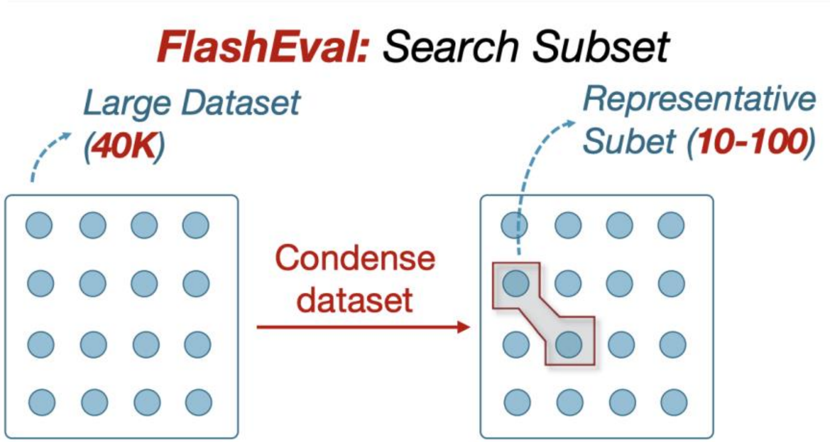
Evaluating SD v1.5 50 steps on complete COCO cost **~50 GPU hours**

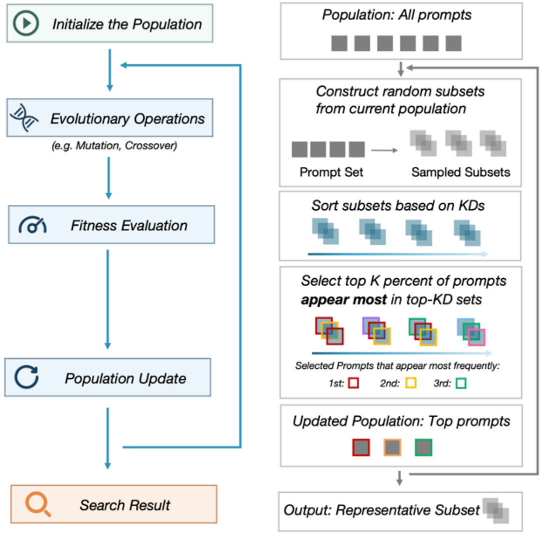


Many applications require
Repeated Evaluation

[1] Zhao, Lin, … **Ning, Xuefei**, et al. "FlashEval: Towards Fast and Accurate Evaluation of Text-to-image Diffusion Generative Models." CVPR 2024.

Methodology: Find "Representative Subset", by Evolutionary-inspired searching





[1] Zhao, Lin, … **Ning, Xuefei**, et al. "FlashEval: Towards Fast and Accurate Evaluation of Text-to-image Diffusion Generative Models." CVPR 2024.

Faster Evaluation (FlashEval)

INFINIGENCE 无问芯穹

12 Model Variants
(Dreamlike, SD v1.2/1.5/2.1 and their 6/8 bit Quantized version)

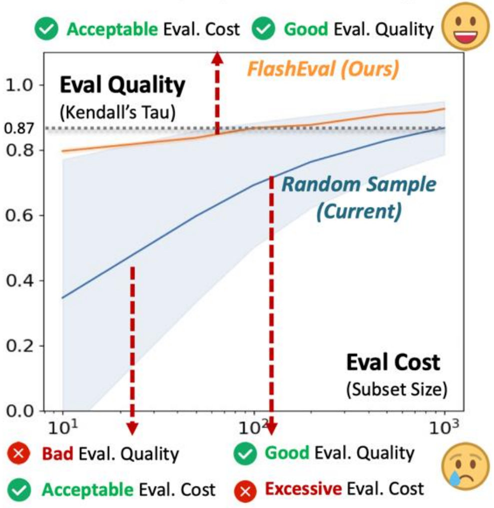
8 Schedules
(DDIM, DPMSolver, PNDM 10/20/50 Steps)

4 Metrics
(FID, ImageReward CLIPScore, HPS)

Diverse Evaluation Settings

models	item size	N'=50			N'=500		
	methods \sub-tasks	random	model variants	schedulers	random	model variants	schedulers
Train	RS	0.607±0.000	0.594±0.000	0.632±0.000	0.857±0.000	0.858±0.000	0.857±0.000
	B3-prompt	0.900	0.909	0.862	0.895	0.917	0.872
	B3-set	0.895±0.002	0.912±0.002	0.894±0.002	0.971±0.002	0.970±0.001	0.966±0.002
	Ours	**0.956±0.004**	**0.969±0.004**	**0.960±0.004**	**0.984±0.003**	**0.986±0.003**	**0.978±0.003**
Test	RS	0.597±0.000	0.588±0.000	0.560±0.000	0.829±0.000	0.826±0.000	0.827±0.000
	B3-prompt	0.729	0.784	0.810	0.805	0.822	0.851
	B3-set	0.750±0.014	0.680±0.021	0.721±0.013	0.875±0.007	0.836±0.008	0.863±0.008
	Ours	**0.851±0.004**	**0.800±0.008**	**0.850±0.005**	**0.906±0.003**	**0.899±0.004**	**0.909±0.003**

Our Searched **50-item** Subset have comparable evaluation quality with **Random-sampled 500**



Better Evaluation Eff-Perf Trade-off

[1] Zhao, Lin, … **Ning, Xuefei**, et al. "FlashEval: Towards Fast and Accurate Evaluation of Text-to-image Diffusion Generative Models." CVPR 2024.

Efficient Techniques for DMs

INFINIGENCE 无问芯穹

Overall Cost
(for each iter)

Total Latency: $t_{model} * N_{timestep}$

Total Memory: $M_{weight} + M_{activation}$

LCSC & OMS-DPM & USF & DD
(Schedule Optimization)

Total Latency: $t_{model} * N_{timestep} \downarrow$

Total Memory: $M_{weight} + M_{activation}$

MixDQ & ViDiT-Q
(Mixed-precision quantization)

Total Latency: $t_{model} \downarrow * N_{timestep}$

Total Memory: $M_{weight} \downarrow + M_{activation} \downarrow$

DiTFastAttn
(Attention Compression)

Total Latency: $t_{model} \downarrow * N_{timestep}$

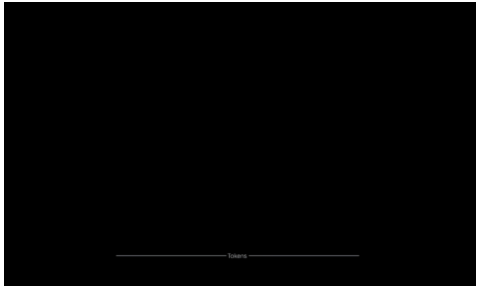
Total Memory: $M_{weight} + M_{activation} \downarrow$

目录 Contents

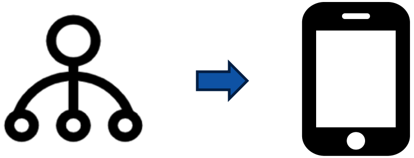
Future Scenarios

Language Generation

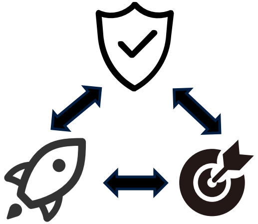
Agent and Multi-model Framework



Long Context LLMs

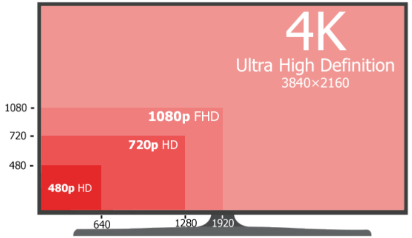


Edge Scenario Deployment



Security-Efficiency Synergy

Visual Generation



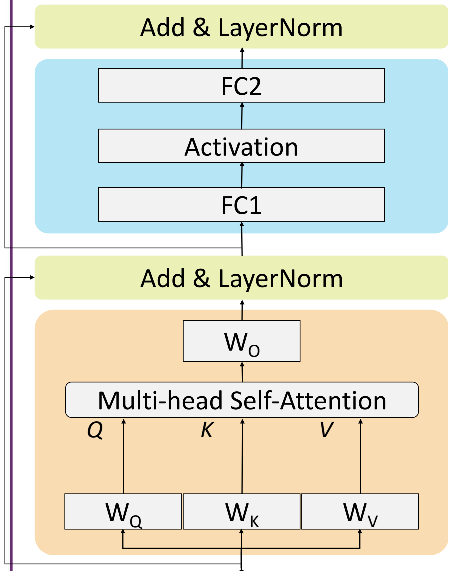
Spatial-Dimension:
High-resolution Generation

Temporal-Dimension:
Long Video Generation

Future Directions

Goal: higher generation quality + better controllability and interactivity

Unified Model Architecture



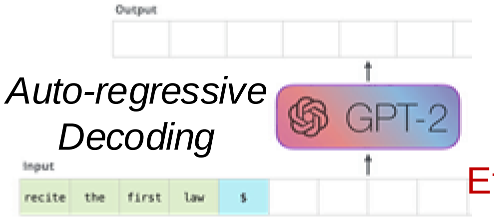
Transformer

Efficiency Challenge:

Quadratic complexity in context length

Unified Generation Approach

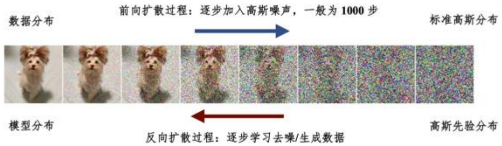
Auto-regressive Decoding



Efficiency Challenge:

Multiple generation steps

Diffusion



Research Summary

Overview

Survey
[CSUR Submission]

Survey on efficient LLM inference techniques

Algorithm-level

SoT
[ICLR'24]

Parallel generation via prompting.
1.91~2.39x speed-up

Model-level

Sparse Attention

MoA
[ICLR Submission]

Decide the heterogeneous elastic rule of the attention span for each head.
5.5~6.7x throughput improvement

Pruning

EEP
[ICLR Submission]

Search the pruning pattern for MoE and use expert merging for finetuning.
48%~71% memory reduction,
1.11~1.40x speed-up,
better performance

Quantization

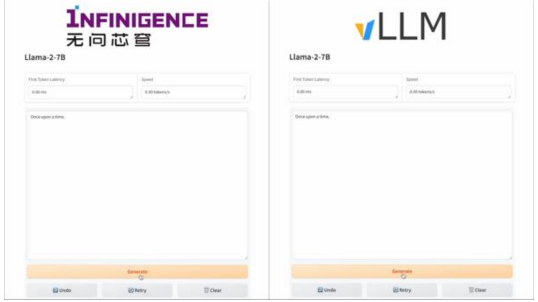
LLM-MQ
[NeurIPS'23 Workshop]

Mixed-precision quantization.
2.8-bit quantization

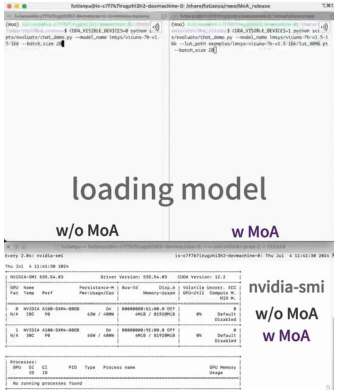
QLLM-Eval
[ICML'24]

Evaluating the effect of quantization.
Providing knowledge and practical suggestions

Efficient Large Language Models



LLaMA-2-7B on AMD MI210
2× throughput improvement



Vicuna-7B on Nvidia-A100
batch size 20
end-to-end latency 2.3x

Research Summary

Algorithm-level
Time Step Compression

LCSC [ICLR Submission]
Linear combination of checkpoints.
15~23x training acceleration,
1.25~2x timestep compression

USF [ICLR'24]
OMS-DPM [ICML'23]
Search for optimal diffusion schedulers.
1.5~2x speed-up

DD [ICLR Submission]
generates image in **0.01s** and can achieve **>100x** speedup for Image AR model

Fast Compression

FlashEval [CVPR'24]
10x evaluation acceleration

Model-level
Quantization

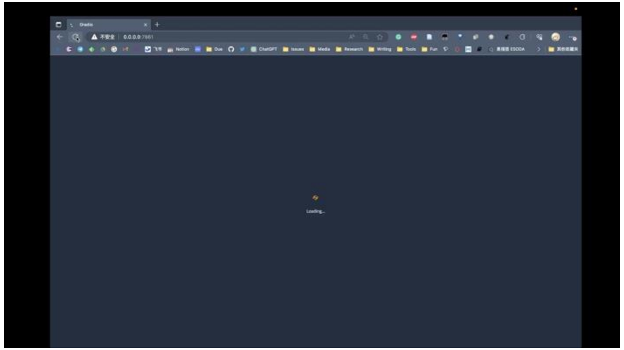
MixDQ [ECCV'24]
Mixed-precision quantization.
3x memory decrease,
1.5x speed-up

ViDiT-Q [ICLR Submission]
Quantization for DiT.
2.5x memory improvement,
1.5x speed-up

Pruning & Sparse Attention

DiTFastAttn [NeurIPS'24]
Window & reused attention for DiT.
1.6x speed-up

Efficient Diffusion Models



Stable Diffusion on a single NVIDIA A100 GPU, Achieving 6.9× speed-up and reducing 1.5× memory

Pixart-Sigma, 2K generation on NVIDIA A100 GPU
1.8x latency speedup

OpenSORA, 512x512x16 Frames, on NVIDIA A100 GPU,
2x Memory Savings, **1.7x** latency speedup

清華大學电子工程系
Department of Electronic Engineering, Tsinghua University

INFINIGENCE 无问芯穹

Thank You !

新书：《高效深度学习：模型压缩与设计（全彩）》

小组网站

Xuefei Ning foxdoraame@gmail.com

Affiliated with: Department of Electronic Engineering, Tsinghua University

Group Leader: Prof. Yu Wang yu-wang@tsinghua.edu.cn

NICS EFC