ICML
International Conference
On Machine Learning

北京航空航天大學
BEIHANG UNIVERSITY

RT-Lynx: Putting GEMM Sparsity in the Right Place for Diffusion Models

Xing Cong[1], Hanlin Tang[2], Kan Liu[2], Lan Tao[2], Lin Qu[2], **Chenhao Xie**[1*]

[1]*School of Computer Science and Engineering, Beihang University, Beijing, China*

[2]*Independent Researcher, Beijing, China*

Seoul, South Korea **ICML'26 July 6-11, 2026**

OUTLINE

北京航空航天大學
BEIHANG UNIVERSITY

OUTLINE

1.1 Diffusion Has Become Mainstream for Image Generation

Table 8. Tensor Shapes of Weights and Activations in the Image and Text Modules of Qwen-Image (where X denotes activations, W denotes weights, N_{txt} is the text length, and N_{img} is the image length). Text in brackets([abbr.]) indicates the abbreviation used in this paper for this type of layer in the model.

Image Module	Image X shape	Image Weight shape	Text Module	Text X shape	Text Weight shape
img_mod.1	(3072)	$(3072, 18432)$	txt_mod.1	(3072)	$(3072, 18432)$
attn.to_q [Q]	$(N_{\text{img}}, 3072)$	$(3072, 3072)$	attn.add_q_proj	$(N_{\text{txt}}, 3072)$	$(3072, 3072)$
attn.to_k [K]	$(N_{\text{img}}, 3072)$	$(3072, 3072)$	attn.add_k_proj	$(N_{\text{txt}}, 3072)$	$(3072, 3072)$
attn.to_v [V]	$(N_{\text{img}}, 3072)$	$(3072, 3072)$	attn.add_v_proj	$(N_{\text{txt}}, 3072)$	$(3072, 3072)$
attn.to_out.0 [Out]	$(N_{\text{img}}, 3072)$	$(3072, 3072)$	attn.to_add_out	$(N_{\text{txt}}, 3072)$	$(3072, 3072)$
img_mlp.net.0.proj [Up]	$(N_{\text{img}}, 3072)$	$(3072, 12288)$	txt_mlp.net.0.proj	$(N_{\text{txt}}, 3072)$	$(3072, 12288)$
img_mlp.net.2 [Down]	$(N_{\text{img}}, 12288)$	$(12288, 3072)$	txt_mlp.net.2	$(N_{\text{txt}}, 12288)$	$(12288, 3072)$

Table 10. Tensor Shapes of Weights and Activations in the Image and Text Streams of FLUX.1-dev (where X denotes activations, W denotes weights, N_{txt} is the text length, and N_{img} is the image length). Text in brackets([abbr.]) indicates the abbreviation used in this paper for this type of layer in the model.

Img Module	Img X shape	Img W shape	Txt Module	Txt X shape	Txt W shape
norm1_a.linear	(3072)	$(3072, 18432)$	norm1_b.linear	(3072)	$(3072, 18432)$
attn.a_to_qkv [QKV]	$(N_{\text{img}}, 3072)$	$(3072, 9216)$	attn.b_to_qkv	$(N_{\text{txt}}, 3072)$	$(3072, 9216)$
attn.a_to_out [Out]	$(N_{\text{img}}, 3072)$	$(3072, 3072)$	attn.b_to_out	$(N_{\text{txt}}, 3072)$	$(3072, 3072)$
ff_a.0 [Up]	$(N_{\text{img}}, 3072)$	$(3072, 12288)$	ff_b.0	$(N_{\text{txt}}, 3072)$	$(3072, 12288)$
ff_a.2 [Down]	$(N_{\text{img}}, 12288)$	$(12288, 3072)$	ff_b.2	$(N_{\text{txt}}, 12288)$	$(12288, 3072)$

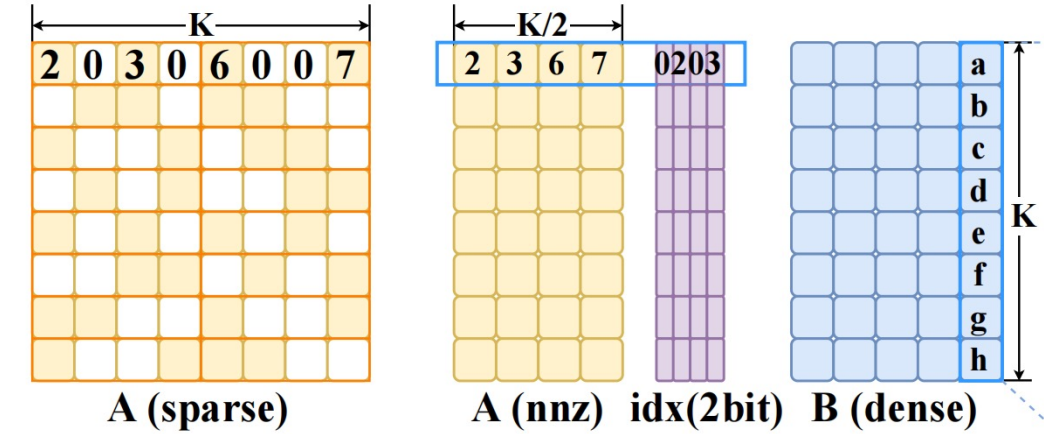
➢ DiT models are mainly constrained by two factors

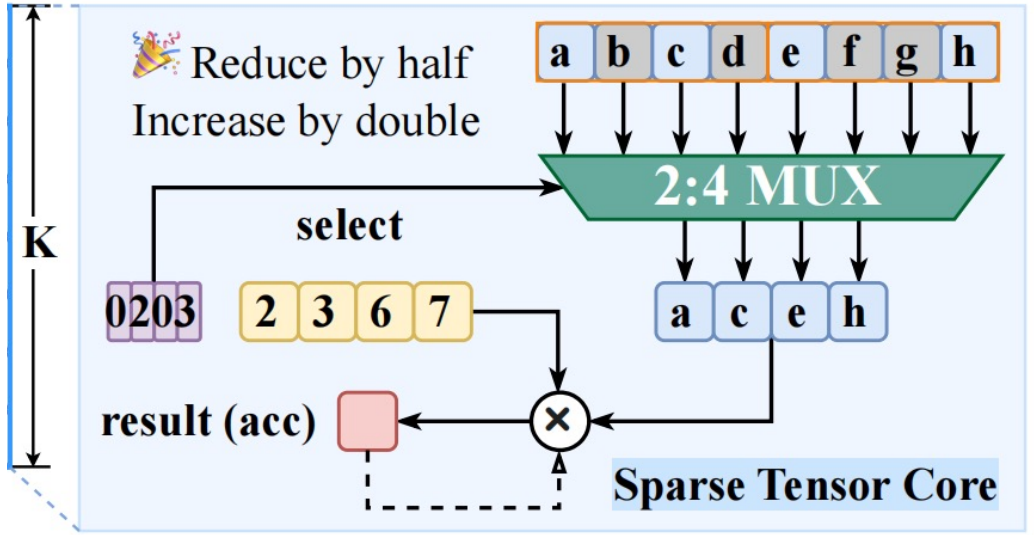
- Generating one image requires **several or even dozens of denoising steps**
- Each denoising step is usually **more compute-intensive** than conventional LLM inference

1.2 Mainstream Acceleration Methods for Diffusion

- **Distillation**: distills multi-step generation into fewer steps, reducing denoising steps, e.g., from 50 to 8. [1]
- **Quantization:** lowers weight and activation precision, e.g., from BF16/FP16 to FP8 or even FP4, to exploit high-FLOPS low-precision cores. [2]
- **Caching:** uses similarity between adjacent steps to reduce computation, e.g., merging two steps into one. [3]
- **Attention:** reduces redundant computation in attention to lower overall cost. [4]

However, sparsity, the most common acceleration method in LLMs (up to 2x), remains unused here...





[1] Zheng, K., Wang, Y., Ma, Q., Chen, H., Zhang, J., Balaji, Y., Chen, J., Liu, M., Zhu, J., & Zhang, Q. (2025). Large Scale Diffusion Distillation via Score-Regularized Continuous-Time Consistency. ArXiv, abs/2510.08431.

[2] Li, M., Lin, Y., Zhang, Z., Cai, T., Li, X., Guo, J., Xie, E., Meng, C., Zhu, J., & Han, S. (2024). SVDQuant: Absorbing Outliers by Low-Rank Components for 4-Bit Diffusion Models. ArXiv, abs/2411.05007.

[3] Liu, F., Zhang, S., Wang, X., Wei, Y., Qiu, H., Zhao, Y., Zhang, Y., Ye, Q., & Wan, F. (2024). Timestep Embedding Tells: It's Time to Cache for Video Diffusion Model. 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

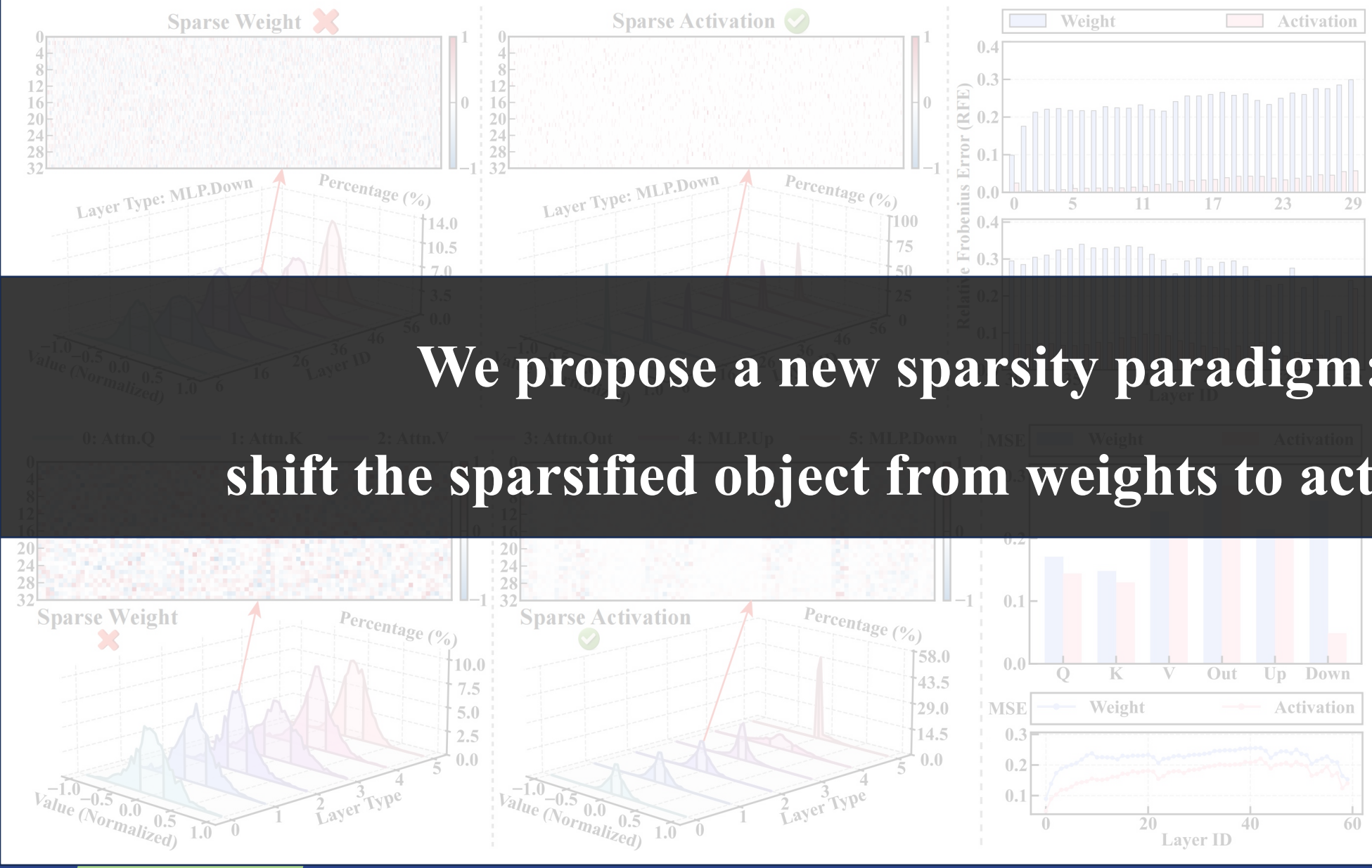
[4] Zhang, J., Xiang, C., Huang, H., Wei, J., Xi, H., Zhu, J., & Chen, J. (2025). SpargeAttn: Accurate Sparse Attention Accelerating Any Model Inference. ArXiv, abs/2502.18137.

1.3 Is Diffusion Unsuitable for Sparsity?

Experiment: Comparison of sparsification strategies on Qwen-Image. Both weight and activation sparsity use Top-K selection to impose 2:4 semi-structured sparsity.

- **Sparse Weight** severely damages the original model's image-generation capability
- **Sparse Activation** shows some robustness to sparse operations, preserving the basic elements and visual quality of generated images

1.4 Why Sparse Activation is Better?



We propose a new sparsity paradigm:

shift the sparsified object from weights to activations

- **Experiment-1:** Distribution and error comparison of weight vs. activation sparsity in Qwen-Image MLP-down layers under Top-K 2:4 sparsification.
- **Experiment-2:** Distribution and error comparison of weight vs. activation sparsity across Qwen-Image Linear layers under Top-K 2:4 sparsification.

So: Activations in DiT are numerically sparser and less sensitive to errors from sparse operations!

➢ Accuracy

- Sparse Activation still introduces large **errors on some prompts.**

➢ Performance

Table 5. Proportion of online activation sparsification overhead in total sparse execution for a typical DiT inference pipeline across methods on H20 GPUs. Gray rows indicate matrix sizes used in Qwen-Image.

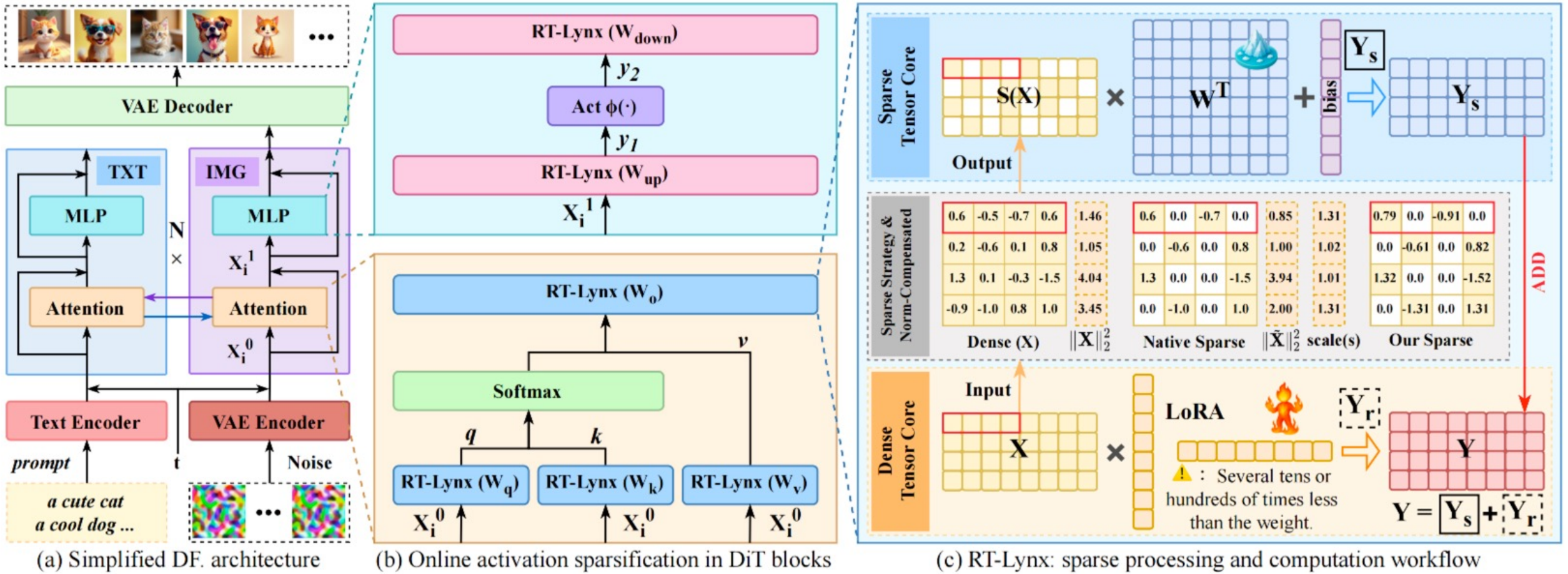
$M=N$	K	PyTorch*	CUTLASS	cuSparseLt
4096	3072	35.08%	26.98%	44.61%
8192	3072	24.63%	16.02%	27.80%
4096	12288	39.29%	28.10%	43.37%
8192	12288	23.14%	17.01%	26.99%

- Sparse Activation requires online sparsification and format conversion while also enabling efficient pipelined GEMM execution. This remains largely unexplored, **making speedup difficult to achieve.**

Accuracy compensation and an efficient execution pipeline are needed to preserve quality and achieve speedup

北京航空航天大學
BEIHANG UNIVERSITY

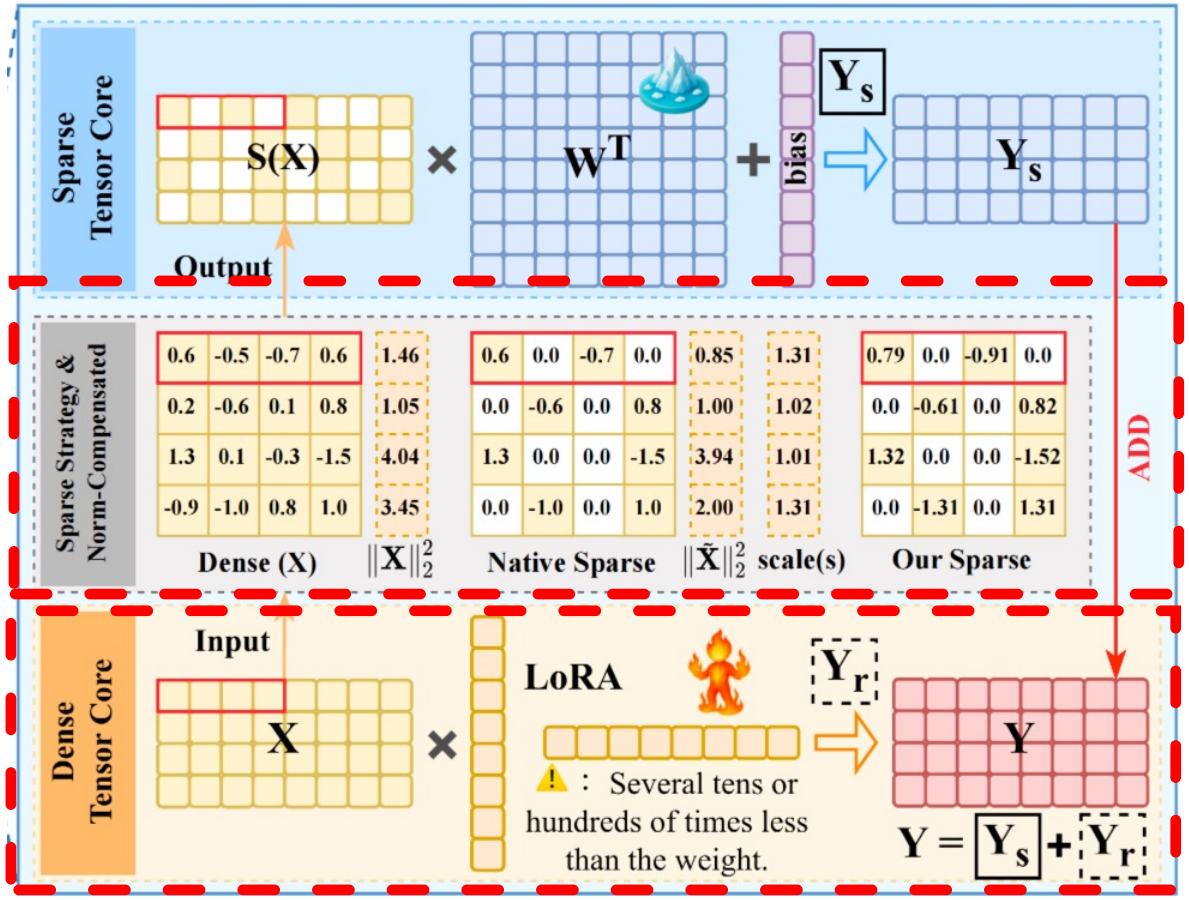
OUTLINE



(a) Simplified DF. architecture (b) Online activation sparsification in DiT blocks (c) RT-Lynx: sparse processing and computation workflow

Replace all linear layers in the model with RT-Lynx (Plug-and-Play).

2.2 Error Compensation



- Pruning **reduces the overall magnitude**.

$$S(\mathbf{X}) = s \cdot \tilde{\mathbf{X}}, \quad s = \sqrt{\frac{\|\mathbf{X}\|_2^2}{\|\tilde{\mathbf{X}}\|_2^2 + \epsilon}}.$$

✓ **Rescale sparse activations:** so their L2 norm aligns with the original dense activations.

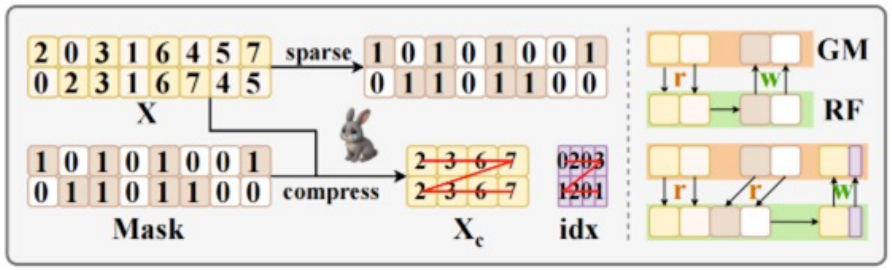
- Although activations are close to zero, they still **contain fine-grained information**;

$$\mathbf{Y} = \mathbf{Y_s} + \mathbf{Y_r} = S(\mathbf{X}) \cdot \mathbf{W}^\top + \mathbf{X} \cdot (\mathbf{L_A L_B})^\top$$

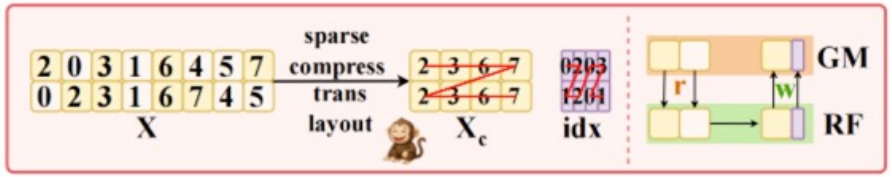
$$Loss = \left\| \mathbf{X} \cdot \mathbf{W}^\top - \left(S(\mathbf{X}) \cdot \mathbf{W}^\top + \mathbf{X} \cdot (\mathbf{L_A L_B})^\top \right) \right\|^2.$$

✓ A lightweight **LoRA branch** recovers sparsification residuals by aligning sparse outputs with dense outputs through self-distillation.

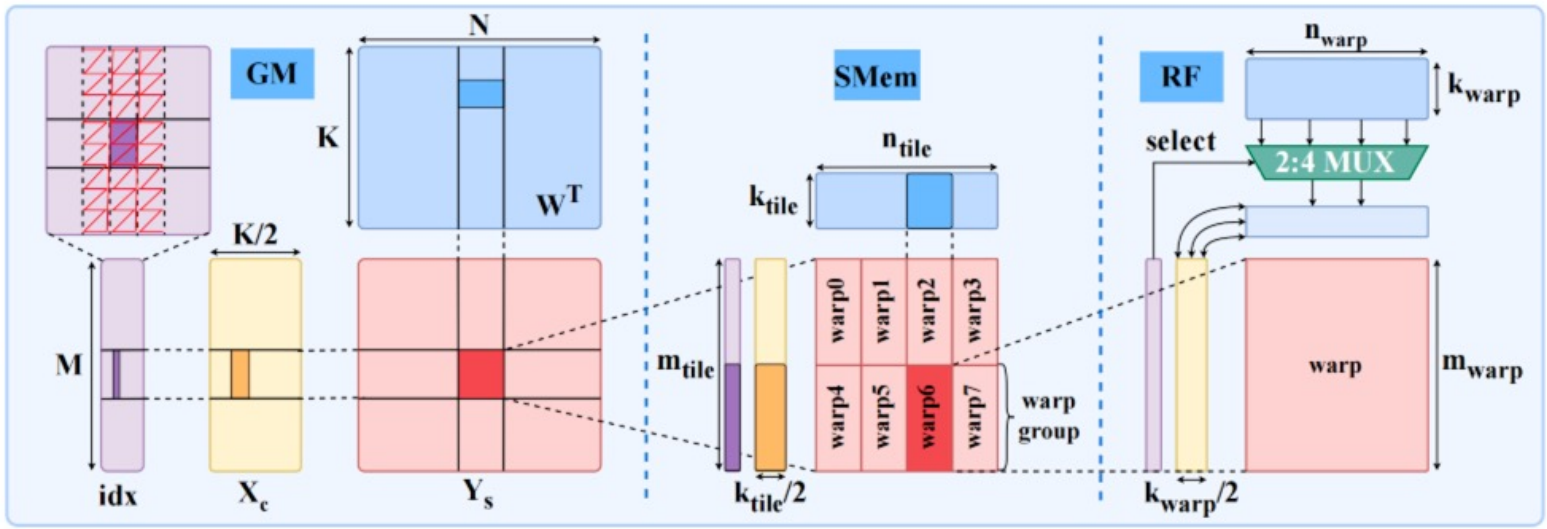
✓ For sensitive layers that cannot be fully compensated, sparsification is **skipped** to balance accuracy and speed.



(a) Other : sparse kernel and compress kernel



(b) RT-Lynx : fused sparse kernel



(c) RT-Lynx : compute sparse-dense matrix multiplication with tiling and pipeline

- The online computation of sparse matrix multiplication using existing kernels **takes too long**.

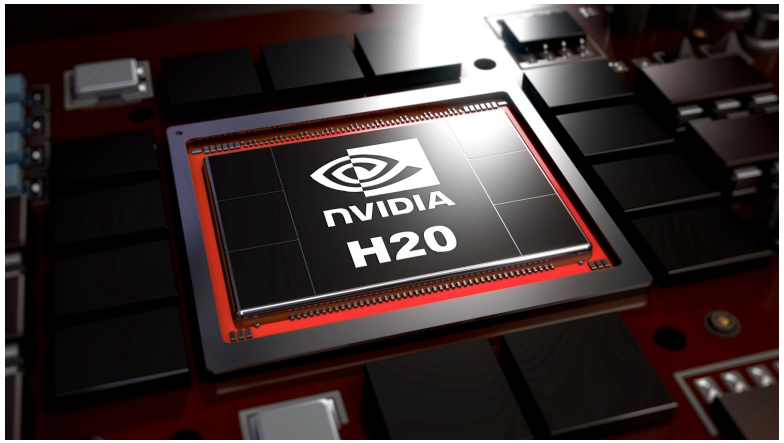
- ✓ **Fuse pattern** determination, Top-K, and compression into a single CUDA kernel, directly generating a 2:4 SpTC-compatible layout at the register level.
- ✓ Build a **execution pipeline** for sparse computation + dense LoRA computation, combining Yr and Ys on chip. Avoid materializing intermediate LoRA results, reducing synchronization and memory access overhead.

北京航空航天大學
BEIHANG UNIVERSITY

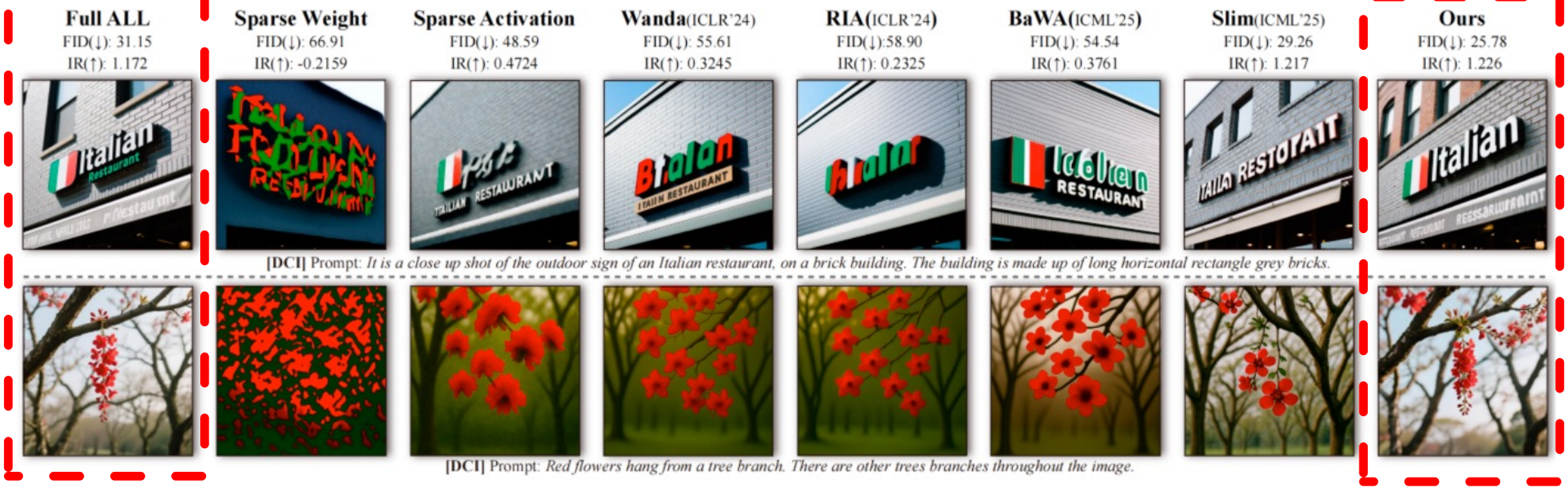
OUTLINE

3.1 Experimental Setup

- **Model**
  - Qwen-Image
  - FLUX.1-Dev
  - Z-Image
  - Qwen-Image-2512
- **Datasets (5000 prompt)**
  - MJHQ
  - sDCI
- **Baseline**
  - Wanda
  - RIA
  - BaWA
  - Slim
- **Metrics**
  - FID
  - Image Reward
  - CLIP-IQA
  - CLIP-Score
- **Details**
  - H20 GPUs
  - Driver: 580.82.07
  - CUDA 13.0



Method	MJHQ FID(↓)	MJHQ IR(↑)	MJHQ C.SCR(↑)	MJHQ C.IQA(↑)	sDCI FID(↓)	sDCI IR(↑)	sDCI C.SCR(↑)	sDCI C.IQA(↑)
Full	21.98	1.219	27.12	0.9317	31.15	1.172	26.49	0.9492
Sparse Weight	51.63	-0.1605	24.22	0.5783	66.91	-0.2159	24.93	0.5792
Sparse Activation	**35.85**	**0.5994**	**26.28**	**0.7473**	**48.59**	**0.4724**	**26.13**	**0.7550**
Wanda (ICLR'24)	40.81	0.5356	26.06	0.7344	55.61	0.3245	26.01	0.7268
RIA (ICLR'24)	43.02	0.4627	25.90	0.7150	58.90	0.2325	25.90	0.7124
BaWA (ICML'25)	39.68	0.5888	26.15	0.7660	54.54	0.3761	26.15	0.7692
Slim (ICML'25)	22.25	1.278	27.18	0.9242	29.26	1.217	26.23	0.9350
RT-Lynx (**Ours**)	**21.25**	**1.304**	**27.33**	**0.9263**	**25.78**	**1.226**	**26.61**	**0.9366**

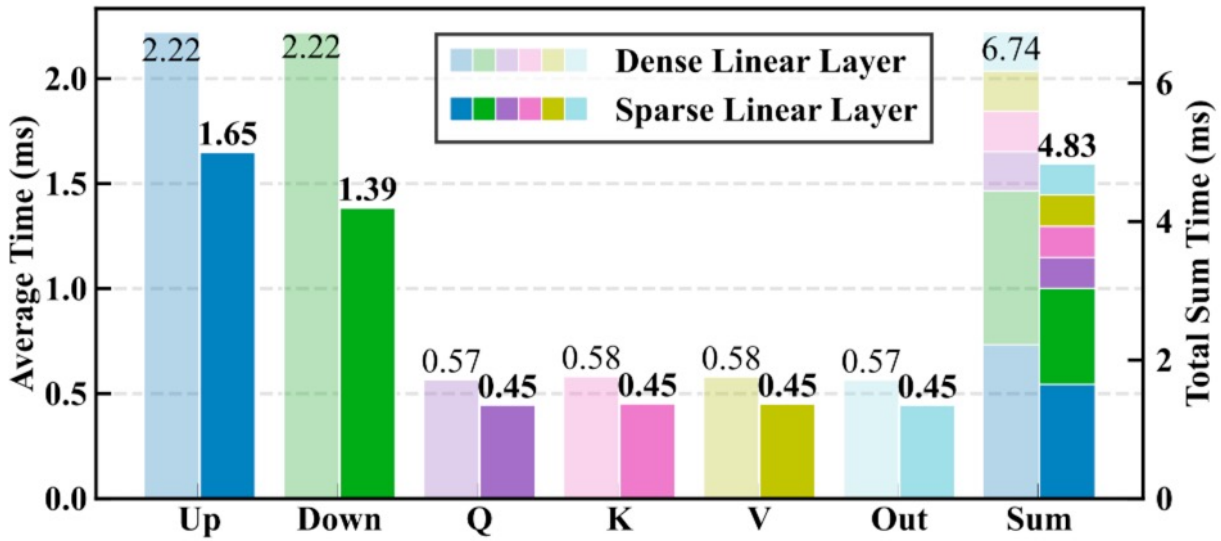


(1) Naive weight sparsity severely degrades image quality, while **activation sparsity provides a stronger starting point**.

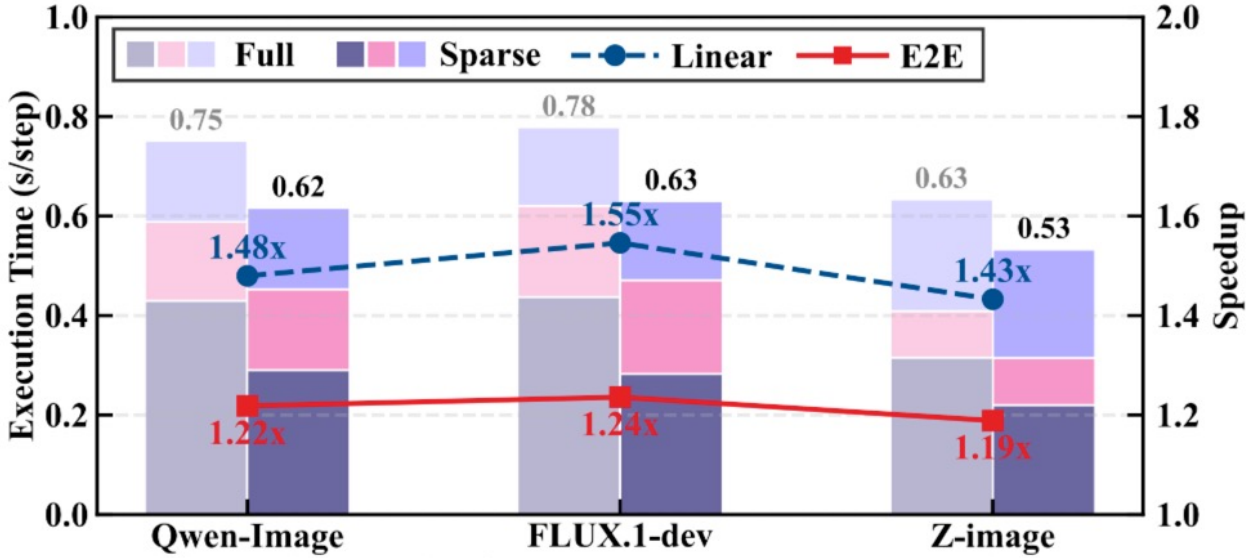
(2) RT-Lynx further restores quality, achieving the best FID and IR among sparse methods on both MJHQ and sDCI.

3.3 Performance

$M=N$	K	Pytorch-GEMM	Pytorch-SpMM	CUTLASS	cuSparseLt	RT-Lynx-Kernel	Sparse-Cost(%)
2048	3072	0.199	0.221 (0.90×)	0.216 (0.92×)	0.282 (0.71×)	0.135 (**1.47**×)	**8.33%**
4096	3072	0.781	0.715 (1.09×)	0.616 (1.26×)	0.709 (1.10×)	0.465 (**1.68**×)	**4.60%**
8192	3072	2.994	2.087 (1.43×)	2.044 (1.46×)	2.066 (1.50×)	1.802 (**1.66**×)	**2.28%**
2048	12288	0.781	1.259 (0.62×)	0.801 (0.97×)	1.137 (0.69×)	0.454 (**1.72**×)	**9.39%**
4096	12288	3.099	2.648 (1.17×)	2.287 (1.35×)	2.669 (1.16×)	1.652 (**1.88**×)	**4.83%**
8192	12288	11.95	8.217 (1.45×)	7.497 (1.59×)	8.202 (1.45×)	6.754 (**1.77**×)	**2.37%**



(a) Single-Step Inference Average Latency of Different Linear Layers



(b) End-to-End Inference Average Latency Analysis

Kernel: RT-Lynx accelerates GEMM with low sparsification overhead, achieving up to **1.88× kernel speedup.**
E2E: The optimized sparse Linear layers reduce single-step latency and bring consistent end-to-end gains.

3.4 Ablation Study

Model	Method	MJHQ FID(↓)	MJHQ IR(↑)	MJHQ C.SCR(↑)	MJHQ C.IQA(↑)	sDCI FID(↓)	sDCI IR(↑)	sDCI C.SCR(↑)	sDCI C.IQA(↑)
Qwen-Image	Full	21.98	1.219	27.12	0.9317	31.15	1.172	26.49	0.9492
	SA-Native	35.85	0.5994	26.28	0.7473	48.59	0.4724	26.13	0.7550
	SA-NC	25.28	0.9393	26.76	0.8349	37.56	0.8399	26.34	0.8321
	SA-NC-LoRA	**21.25**	**1.304**	**27.33**	**0.9263**	**25.78**	**1.226**	**26.61**	**0.9366**
Qwen-Image (2512)	Full	20.45	1.290	26.45	0.9569	24.42	1.130	25.91	0.9435
	SA-Native	39.13	0.6575	25.60	0.7379	51.41	0.3205	**26.48**	0.6593
	SA-NC	26.46	1.064	26.08	0.8891	33.29	0.8747	26.32	0.8251
	SA-NC-LoRA	**20.84**	**1.302**	**26.54**	**0.9484**	**24.10**	**1.132**	26.07	**0.9285**
Flux.1-dev	Full	22.05	1.010	26.51	0.9401	25.96	1.074	26.01	0.9487
	SA-Native	49.87	0.4520	24.83	0.7881	52.88	0.590	25.74	0.7471
	SA-NC	38.67	0.7823	25.56	0.8565	39.67	0.8704	25.86	0.8340
	SA-NC-LoRA	22.61	0.9783	26.30	0.9341	26.35	1.050	25.90	0.9415
	SA-NC-LoRA-SL	**21.17**	**1.011**	**26.42**	**0.9381**	**24.41**	**1.091**	**25.90**	**0.9445**
Z-image	Full	25.70	0.9928	25.57	0.9307	25.40	0.9974	25.91	0.9467
	SA-Native	42.65	0.5982	25.42	0.7917	36.18	0.7306	**26.72**	0.7508
	SA-NC	31.65	0.8054	**25.61**	0.8525	28.48	0.8818	26.45	0.8504
	SA-NC-LoRA	27.39	0.9292	25.48	0.9016	25.98	0.9788	26.08	0.9137
	SA-NC-LoRA-SL	**26.17**	**0.9673**	25.39	**0.9250**	**24.93**	**0.9803**	26.22	**0.9396**

Each component contributes to stable sparse inference: norm compensation reduces error, LoRA restores output quality, and sensitive-layer skipping protects fragile layers.

Together, they deliver the best visual quality and metric performance across models.

3.5 Compatibility

Model	Method	MJHQ FID(↓)	MJHQ IR(↑)	MJHQ C.SCR(↑)	MJHQ C.IQA(↑)	sDCI FID(↓)	sDCI IR(↑)	sDCI C.SCR(↑)	sDCI C.IQA(↑)
Distillation (Z-Image)	Baseline(step-50)	22.31	0.8790	26.25	0.9046	18.60	0.9545	26.23	0.9196
	RT-Lynx	*22.68*	*0.9301*	*26.25*	*0.9058*	*18.80*	*1.006*	*26.25*	*0.9167*
	Turbo(step-8)	25.70	0.9928	25.57	0.9307	25.40	0.9974	25.91	0.9467
	RT-Lynx+Turbo	**26.17**	**0.9673**	**25.39**	**0.9250**	**24.93**	**0.9803**	**26.22**	**0.9396**
Quant (Qwen-Image)	Baseline	21.98	1.219	27.12	0.9317	31.15	1.172	26.49	0.9492
	RT-Lynx	*21.25*	*1.304*	*27.33*	*0.9263*	*25.78*	*1.226*	*26.61*	*0.9366*
	W8A8	21.78	1.220	27.10	0.9298	30.59	1.173	26.48	0.9474
	RT-Lynx+W8A8	**21.43**	**1.299**	**27.33**	**0.9250**	**25.92**	**1.227**	**26.65**	**0.9341**
Cache (FLUX.1-dev)	Baseline	22.05	1.010	26.51	0.9401	25.96	1.074	26.01	0.9487
	RT-Lynx	*21.17*	*1.011*	*26.42*	*0.9381*	*24.41*	*1.091*	*25.90*	*0.9445*
	Teacache(l=0.6)	20.67	0.937	25.83	0.9373	25.76	1.024	25.46	0.9487
	RT-Lynx+Teacache	**20.57**	**0.941**	**25.73**	**0.9414**	**24.98**	**1.026**	**25.32**	**0.9443**
Attention (Qwen-Image-2512)	Baseline	20.45	1.290	26.45	0.9569	24.42	1.130	25.91	0.9435
	RT-Lynx	*20.84*	*1.302*	*26.54*	*0.9484*	*24.10*	*1.132*	*26.07*	*0.9285*
	SpargeAttn(k=0.5)	21.17	1.192	26.18	0.9557	25.46	1.067	26.16	0.9390
	RT-Lynx+SpargeAttn	**21.39**	**1.212**	**26.37**	**0.9420**	**25.22**	**1.075**	**26.42**	**0.9267**

Model	Method	End2End(s)
Distillation (Z-Image)	Baseline(step-50)	49.44
	RT-Lynx	*40.76 (1.21×)*
	Turbo(step-8)	4.99 (9.91×)
	RT-Lynx+Turbo	**4.17 (11.86×)**
Quant (Qwen-Image)	Baseline	60.66
	RT-Lynx	*50.60 (1.20×)*
	W8A8	54.45 (1.11×)
	RT-Lynx+W8A8	**46.04 (1.32×)**
Cache (FLUX.1-dev)	Baseline	77.99
	RT-Lynx	*63.00 (1.24×)*
	TeaCache(l=0.6)	29.56 (2.64×)
	RT-Lynx+TeaCache	**24.89 (3.13×)**
Attention (Qwen-Image-2512)	Baseline	60.42
	RT-Lynx	*49.35 (1.22×)*
	SpargeAttn(k=0.5)	54.49 (1.11×)
	RT-Lynx+SpargeAttn	**44.35 (1.36×)**

(1) RT-Lynx can be seamlessly combined with Distillation, Quantization, Cache, and Sparse Attention.

(2) These combinations further **improve end-to-end speedup**, reaching up to **11.86×**, **while preserving competitive** FID, IR, C.SCR, and C.IQA scores.

北京航空航天大學
BEIHANG UNIVERSITY

OUTLINE

- We explore **shifting the sparsity paradigm from weights to activations** to enable efficient inference acceleration for DiT models.
- We **reveal that activations exhibit inherent sparsity** and are more robust than weights under N:M sparsity.
- We propose RT-Lynx, introducing norm compensation, a LoRA branch, and a layer-skipping mechanism to **reduce sparsification error**.
- We design **an efficient CUDA kernel** that fuses online activation sparsity generation with sparse computation.
- Across multiple diffusion models, RT-Lynx achieves lossless acceleration with significant inference speedups.

ICML
International Conference
On Machine Learning

北京航空航天大學
BEIHANG UNIVERSITY

Thanks!

Thank you, and have a great day!

Email: congxingcs@163.com