SVDQuant: Absorbing Outliers by Low-Rank Components for 4-Bit Diffusion Models

ICLR 2025

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, Song Han

Presented by Changhwan Sung[1], Jungyu Jin[1], Junho Jeong[2]

[1]Graduate School of AI, [2]Electrical Engineering, POSTECH

May 21, 2025

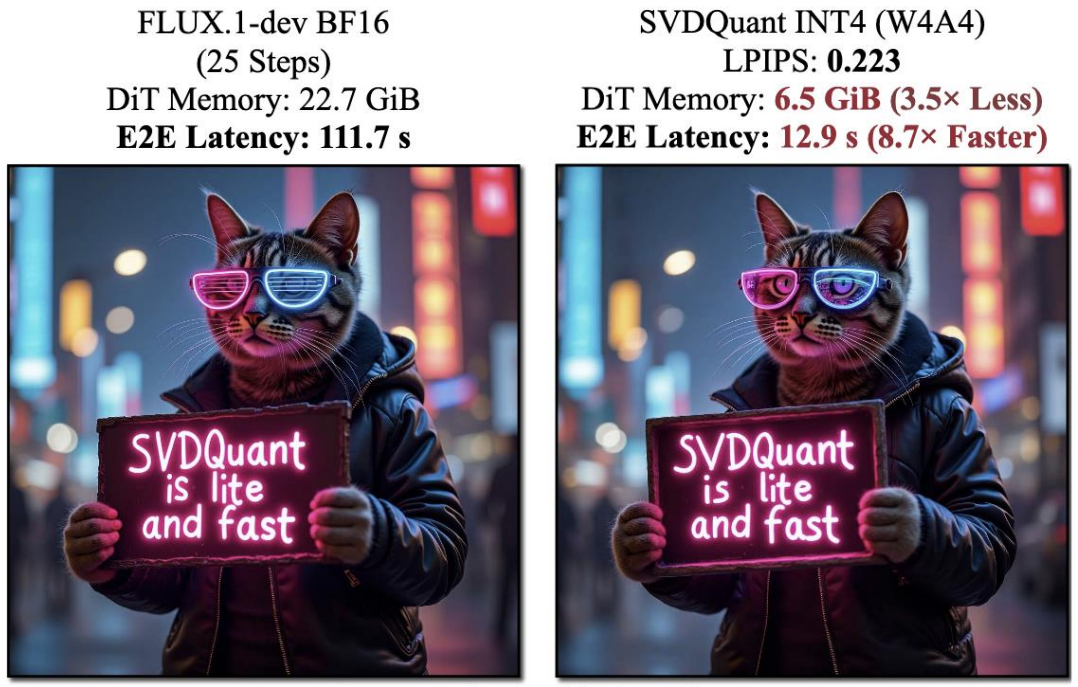
Contents

Motivation

- Diffusion models generate high-quality images but are computationally intensive.

- As model size scales (e.g., SD 1.4 (0.8B) → SDXL (2.6B) → FLUX.1 (12B)):
  Better quality.
  Heavier memory.
  Higher inference latency.

- Low-bit quantization (4-bit) is promising
  Goal: W4A4 Quantization.



Prompt: *a cyberpunk cat holding a huge neon sign that says "SVDQuant is lite and fast", wearing fancy goggles and a black leather jacket.*

Limits of Standard PTQ in Diffusion Models

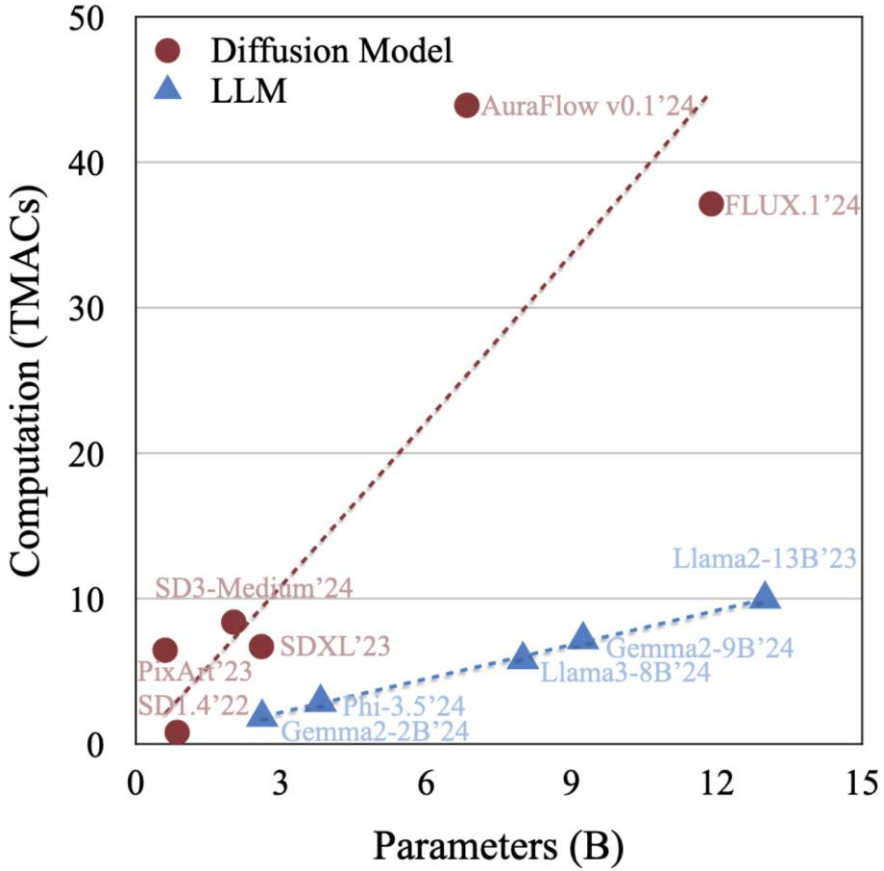
- As model size scales (e.g., SD 1.4 (0.8B) → SDXL (2.6B) → FLUX.1 (12B)):

  Better quality.

  Heavier memory.

  Higher inference latency.

- Unlike LLMs:
  - LLMs are bandwidth-bound
  - Diffusion models are compute-bound
  - FLUX.1 (12B) vs Llama2 (13B):
    FLUX.1 is over 3x heavier than Llama2



Prior Works

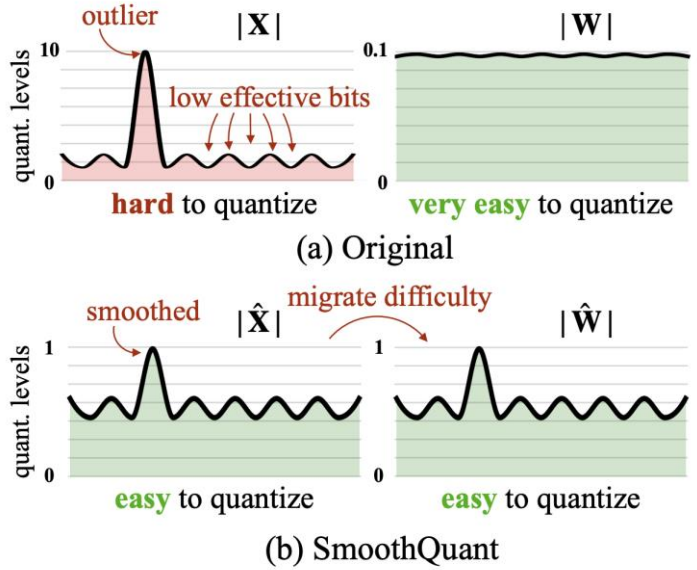
- Q-Diffusion, PTQ4DM: 8-bit quantization.
  - For U-Net backbone.

- MixDQ, ViDiT-Q:
  - Use low-bit inference engines.
  - Show actual speedups on GPUs.
  - ViDiT-Q: W8A8, MixDQ: W4A8

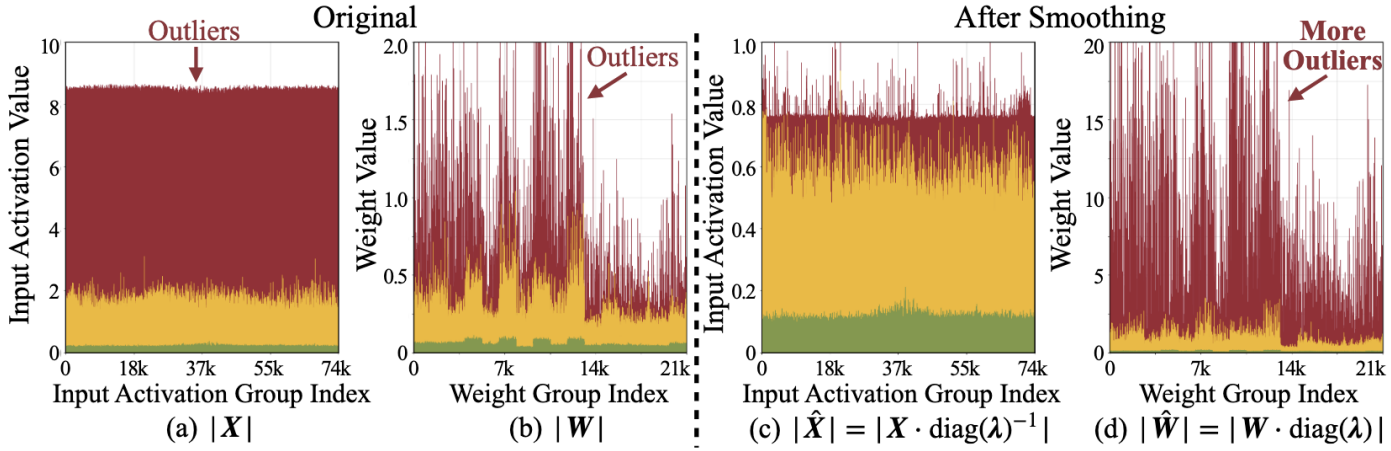
- Most prior work:
  - Targets only weights (W4A16)
  - No support for both weights & activations in 4-bit

Existing Strategy: Smoothing

- Shift outliers from activations to weights
  - Scaling each channel in X by smoothing factor λ.
  - $\hat{X} = X \cdot \mathrm{diag}(\lambda)^{-1},\ \hat{W} = W \cdot \mathrm{diag}(\lambda)$.
  - $Y = (X \cdot \mathrm{diag}(\lambda)^{-1}) \cdot \left(W \cdot \mathrm{diag}(\lambda)\right)$

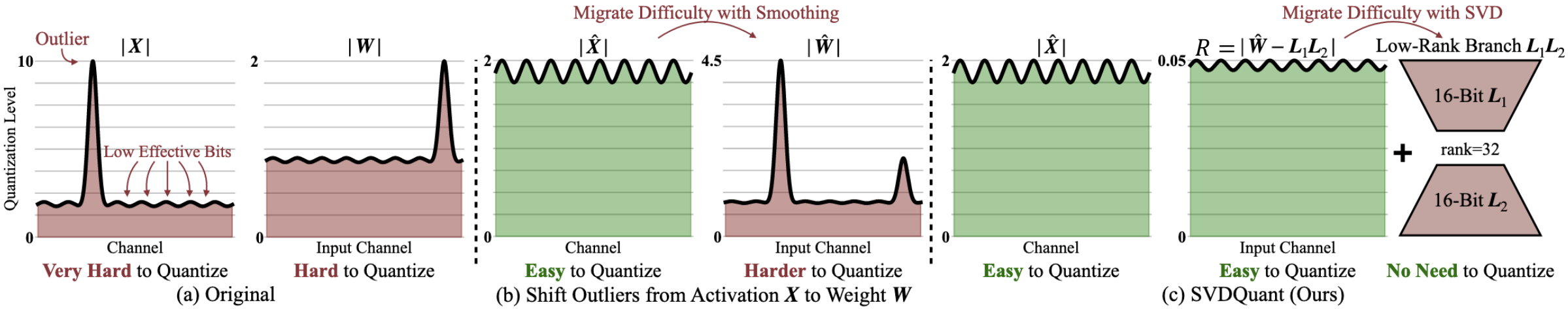


- For Diffusion Models, weight outlier is also critical.
  - Looks like this:

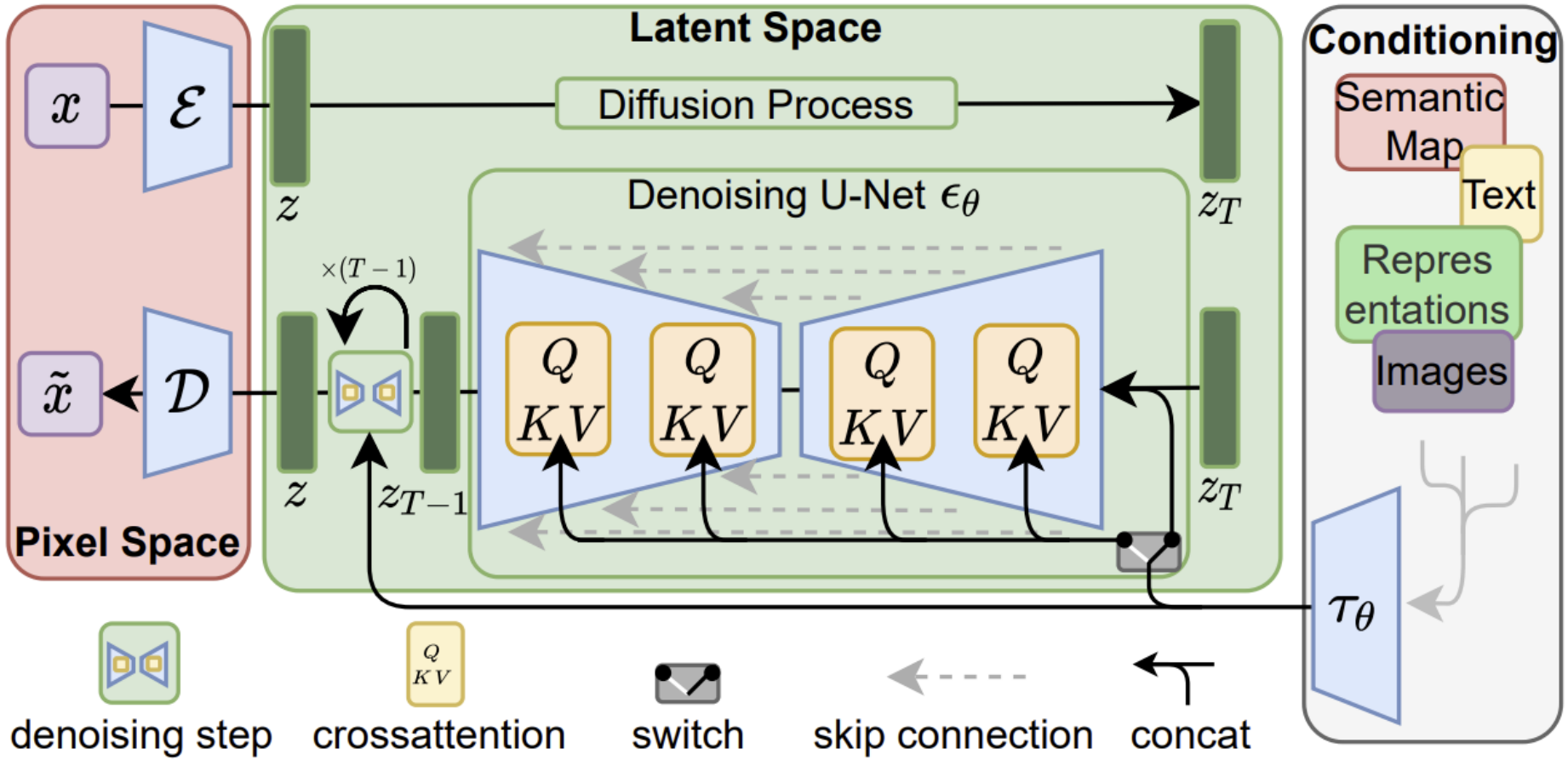


4bit Quantization + Low-Rank Branch

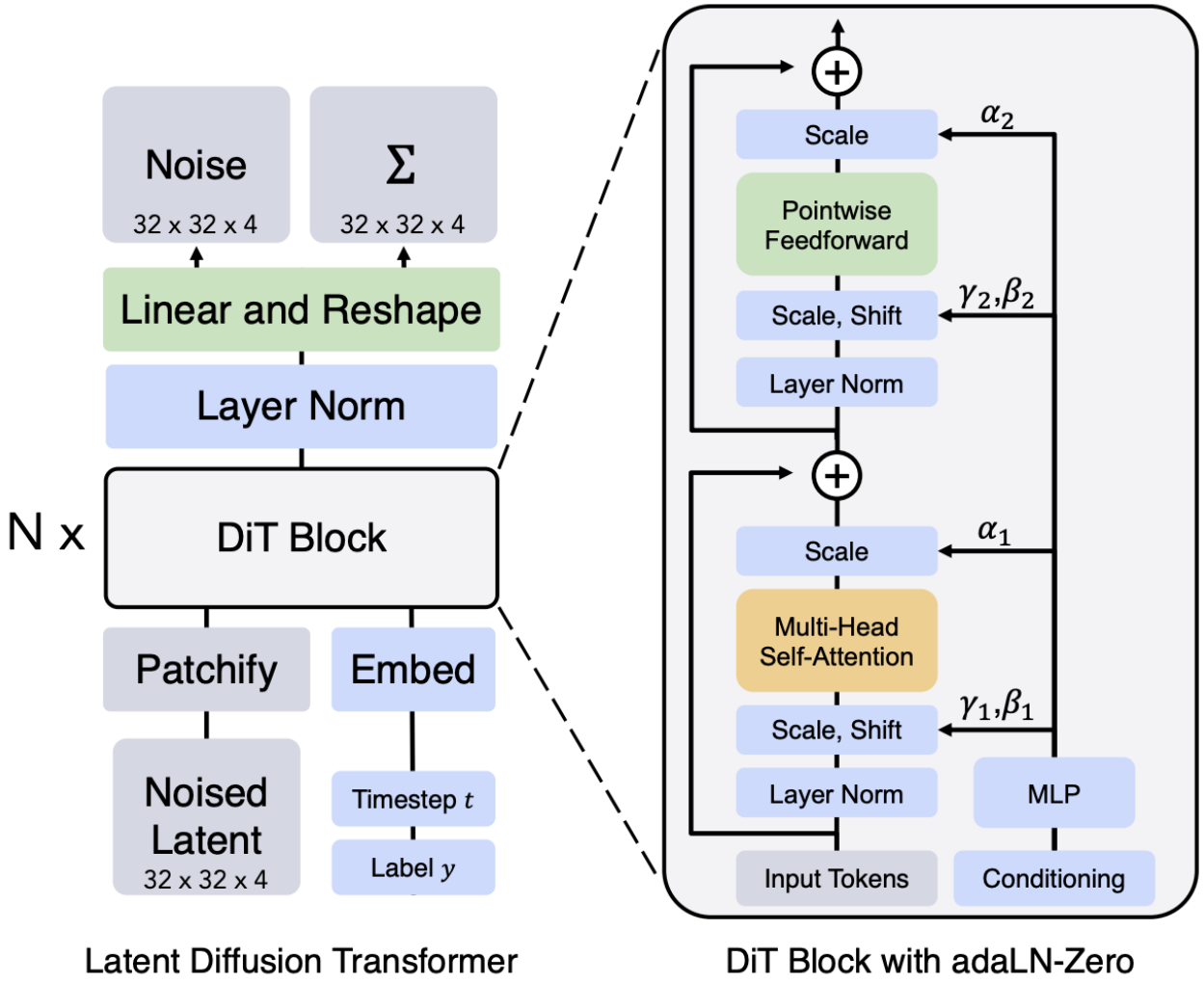
- High-precision branch for outliers.
  - 16-bit low rank branch + 4-bit residual branch.
  - $Y = (X \cdot \mathrm{diag}(\lambda)^{-1}) \cdot \left(W \cdot \mathrm{diag}(\lambda)\right) = \hat{X} \cdot \hat{W} = \hat{X} \cdot (L_1 L_2 + R) = \hat{X} L_1 L_2 + \hat{X} R$
  - $Y \approx \hat{X} L_1 L_2 + s_X s_R Q(\hat{X}) Q(R)$



Diffusion model (U-Net)



Diffusion model (DiT)



Quantization Preliminary

- Given a tensor X, the quantization process is defined as:

$$Q_X = round\left(\frac{X}{s_X}\right), s_X = \frac{\max(|X|)}{q_{max}}$$

  - For signed k-bit integer quantization, $q_{max} = 2^{k-1} - 1$
  - For 4-bit floating-point quantization (E2M1), $q_{max} = 6\ (1.1_2\ * 2^{3-1})$
  - Dequantized tensor $Q(X) = s_X \cdot Q_X$

Q_X: Low-bit representation of X
s_X: Scaling factor
q_{max}: maximum quantized value

- For a linear layer with input X and weight W, its computation:

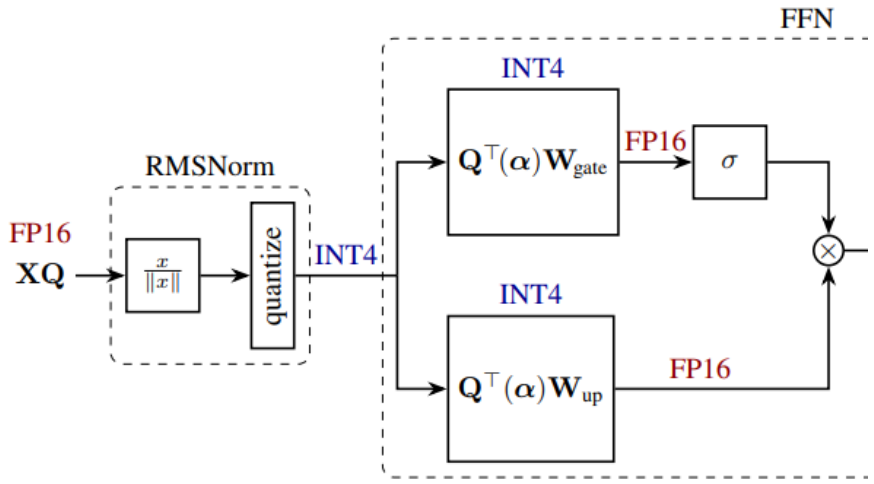
$$\boldsymbol{XW \approx Q(X)Q(W) = s_X s_W \cdot Q_X Q_W}$$

- we denote x-bit weight, y-bit activation as $W_x A_y$.
  - This work focus on **W4A4** quantization for acceleration.

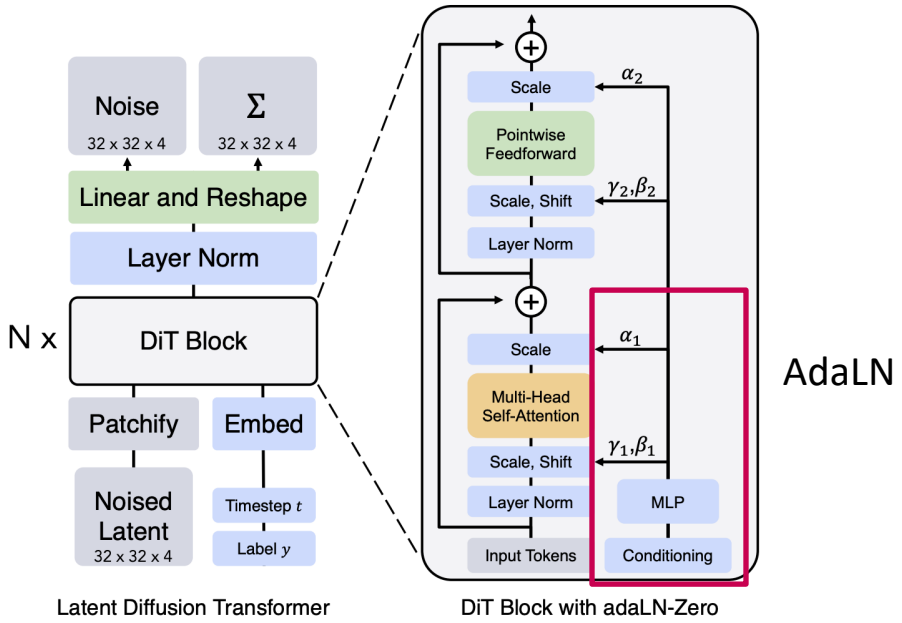
Quantization Preliminary

- Traditional methods to suppress outliers include QAT and rotation.
  - QAT requires massive computing resources.

- Rotation is inapplicable due to the usage of **adaptive normalization layers.**
  - Online rotation of both activations and weights incurs significant runtime overhead.



Rotation applied to a LLaMa-style FFN. The RMSNorm scaling (α) has been absorbedinto the weight matrices



Problem Formulation

- Consider a linear layer with input $X \in R^{b \times m}$ and weight $W \in R^{m \times n}$, Quantization error can be defined as:

$$E(X, W) = \|XW - Q(X)Q(W)\|_F$$

- Proposition 4.1 (Error decomposition).
  The quantization error can be decomposed as follows:

$$E(X, W) \leq \|X\|_F \|W - Q(W)\|_F + \|X - Q(X)\|_F (\|W\|_F + \|W - Q(W)\|_F)$$

- To minimize the overall quantization error, we aim to optimize four terms.

Proof of Proposition 4.1

- The proof of proposition use two inequalities.
  - 1) Triangle inequality: $\|\boldsymbol{A}+\boldsymbol{B}\|_F \leq \|\boldsymbol{A}\|_F + \|\boldsymbol{B}\|_F$
  - 2) Cauchy-Schwarz Inequality: $\|\boldsymbol{AB}\|_F \leq \|\boldsymbol{A}\|_F \|\boldsymbol{B}\|_F$

$$
\begin{aligned}
&\|\boldsymbol{X}\boldsymbol{W} - Q(\boldsymbol{X})Q(\boldsymbol{W})\|_F \\
=&\|\boldsymbol{X}\boldsymbol{W} - \boldsymbol{X}Q(\boldsymbol{W}) + \boldsymbol{X}Q(\boldsymbol{W}) - Q(\boldsymbol{X})Q(\boldsymbol{W})\|_F \\
\leq&\|\boldsymbol{X}(\boldsymbol{W} - Q(\boldsymbol{W}))\|_F + \|(\boldsymbol{X} - Q(\boldsymbol{X}))Q(\boldsymbol{W})\|_F \\
\leq&\|\boldsymbol{X}\|_F \|\boldsymbol{W} - Q(\boldsymbol{W})\|_F + \|\boldsymbol{X} - Q(\boldsymbol{X})\|_F \|Q(\boldsymbol{W})\|_F \\
\leq&\|\boldsymbol{X}\|_F \|\boldsymbol{W} - Q(\boldsymbol{W})\|_F + \|\boldsymbol{X} - Q(\boldsymbol{X})\|_F \|\boldsymbol{W} - (\boldsymbol{W} - Q(\boldsymbol{W}))\|_F \\
\leq&\|\boldsymbol{X}\|_F \|\boldsymbol{W} - Q(\boldsymbol{W})\|_F + \|\boldsymbol{X} - Q(\boldsymbol{X})\|_F (\|\boldsymbol{W}\|_F + \|\boldsymbol{W} - Q(\boldsymbol{W})\|_F)
\end{aligned}
$$

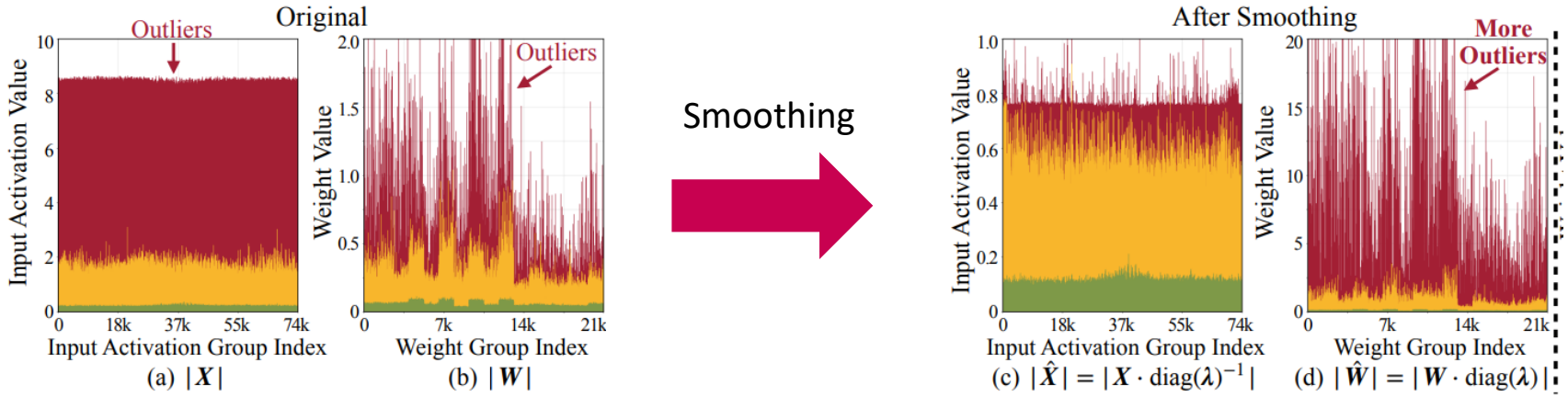
SVDQuant

- SVDQuant 1) first migrates outliers from activation to weight by **smoothing**, and then 2) absorbs the migrated weight outliers into the **low-rank branch.**

- Smooth outliers in activations using a per-channel smoothing factor $\lambda \in R^m$:

$$\hat{X} = X \cdot diag(\lambda)^{-1}, \qquad \widehat{W} = diag(\lambda) \cdot W$$

$$\lambda_i = \frac{max(|X_{:,i}|)^{\alpha}}{max(|W_{i,:}|)^{1-\alpha}}$$



- While the smoothed input $\hat{X}$ exhibits fewer outliers, $\widehat{W}$ has a significant increase in both magnitude and the presence of outliers. → raise $\|W - Q(W)\|_F$

SVDQuant

- Core insight is to introduce a 16-bit low-rank branch to further migrate the weight quantization difficulty.

- Specifically, we decompose $\widehat{W} = L_1 L_2 + R, where\ L_1 \in R^{m \times r}, L_2 \in R^{r \times n}$. Then XW can be approximated as:

$$XW = \hat{X}\widehat{W} = \hat{X}L_1 L_2 + \hat{X}R \approx \hat{X}L_1 L_2 + Q(\hat{X})Q(R)$$

16-bit low-rank path 4-bit residual path

- r is typically set to 16 or 32, so additional parameters and computation for the low-rank branch are negligible.
  - However, it still requires careful system design to eliminate redundant memory access

SVDQuant

- From reformulated equation, quantization error can be expressed as

$$E(X,W) = \|XW - Q(X)Q(W)\|_F$$
$$= \left\|\hat{X}\widehat{W} - (\hat{X}L_1L_2 + Q(\hat{X})Q(R))\right\|_F = \left\|\hat{X}R - Q(\hat{X})Q(R)\right\|_F = E(\hat{X},R)$$

- Since $\hat{X}$ is already free from outliers, we only need to focus on optimizing the $\|\boldsymbol{R}\|_F$ **and** $\|\boldsymbol{R} - \boldsymbol{Q}(\boldsymbol{R})\|_F$.

- Then, how to minimize terms related to $R = \widehat{W} - L_1L_2$?

SVDQuant

- Proposition 4.2 (Quantization error bound).
  For any tensor $\boldsymbol{R}$ and quantization method $Q(\boldsymbol{R}) = s_{\boldsymbol{R}} Q_{\boldsymbol{R}}$. Assuming the elements of $\boldsymbol{R}$ follow a distribution that satisfies the following regularity condition: There exists a constant c such that

$$E[max(|\boldsymbol{R}|)] \leq c \cdot E[\|\boldsymbol{R}\|_F].$$

  Then, we have

$$E[\|\boldsymbol{R} - Q(\boldsymbol{R})\|_F] \leq \frac{c\sqrt{size(R)}}{q_{max}} \cdot E[\|\boldsymbol{R}\|_F].$$

- From this proposition, the quantization error is bounded by the magnitude of $\boldsymbol{R}$.
  - Thus, our goal is to find the optimal $L_1 L_2$ that minimizes $\|\boldsymbol{R}\|_F = \left\|\widehat{W} - L_1 L_2\right\|_F$.

Proof of Proposition 4.2

Proposition 4.2 (Quantization error bound). *For any tensor $\boldsymbol{R}$ and quantization method described in Equation 1 as $Q(\boldsymbol{R}) = s_{\boldsymbol{R}} \cdot \boldsymbol{Q}_{\boldsymbol{R}}$. Assuming the elements of $\boldsymbol{R}$ follow a distribution that satisfies the following regularity condition: There exists a constant c such that*

$$\mathbb{E}\left[\max(|\boldsymbol{R}|)\right] \leq c \cdot \mathbb{E}\left[\|\boldsymbol{R}\|_F\right]. \tag{7}$$

Then, we have

$$\mathbb{E}\left[\|\boldsymbol{R} - Q(\boldsymbol{R})\|_F\right] \leq \frac{c\sqrt{size(\boldsymbol{R})}}{q_{\max}} \cdot \mathbb{E}\left[\|\boldsymbol{R}\|_F\right] \tag{8}$$

where $size(\boldsymbol{R})$ denotes the number of elements in $\boldsymbol{R}$. Especially if the elements of $\boldsymbol{R}$ follow a normal distribution, Equation 7 holds for $c = \sqrt{\frac{\log(size(\boldsymbol{R}))\pi}{size(\boldsymbol{R})}}$.

Especially, if the elements of $\boldsymbol{R}$ follows a normal distribution, we have

$$\mathbb{E}\left[\max(|\boldsymbol{R}|)\right] \leq \sigma\sqrt{2\log\left(\text{size}(\boldsymbol{R})\right)}$$

$$\begin{aligned}
&\mathbb{E}\left[\max(|\boldsymbol{R}|)\right] \\
\leq &\sigma\sqrt{2\log\left(\text{size}(\boldsymbol{R})\right)} \\
\leq &\sqrt{\frac{\log\left(\text{size}(\boldsymbol{R})\right)\pi}{\text{size}(\boldsymbol{R})}}\mathbb{E}\left[\|\boldsymbol{R}\|_F\right].
\end{aligned}$$

$$\begin{aligned}
&\mathbb{E}\left[\|\boldsymbol{R}\|_F\right] \\
=&\mathbb{E}\left[\sqrt{\sum_{x\in\boldsymbol{R}} x^2}\right] \\
\geq&\mathbb{E}\left[\frac{\sum_{x\in\boldsymbol{R}}|x|}{\sqrt{\text{size}(\boldsymbol{R})}}\right] \\
=&\sigma\sqrt{\frac{2\text{size}(\boldsymbol{R})}{\pi}},
\end{aligned}$$

$$\begin{aligned}
&\|\boldsymbol{R} - Q(\boldsymbol{R})\|_F \\
=&\|\boldsymbol{R} - s_{\boldsymbol{R}} \cdot \boldsymbol{Q}_{\boldsymbol{R}}\|_F \\
=&\left\|s_{\boldsymbol{R}} \cdot \frac{\boldsymbol{R}}{s_{\boldsymbol{R}}} - s_{\boldsymbol{R}} \cdot \text{round}\left(\frac{\boldsymbol{R}}{s_{\boldsymbol{R}}}\right)\right\|_F \\
=&|s_{\boldsymbol{R}}|\left\|\frac{\boldsymbol{R}}{s_{\boldsymbol{R}}} - \text{round}\left(\frac{\boldsymbol{R}}{s_{\boldsymbol{R}}}\right)\right\|_F.
\end{aligned}$$

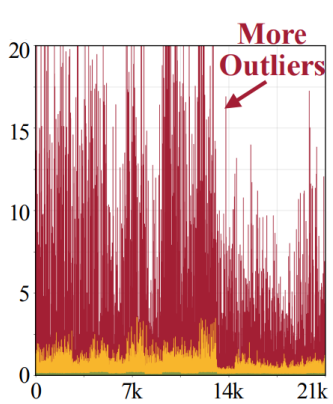
$$\begin{aligned}
&\mathbb{E}\left[\|\boldsymbol{R} - Q(\boldsymbol{R})\|_F\right] \\
\leq&\mathbb{E}\left[|s_{\boldsymbol{R}}|\right]\sqrt{\text{size}(\boldsymbol{R})} \\
=&\frac{\sqrt{\text{size}(\boldsymbol{R})}}{q_{\max}} \cdot \mathbb{E}\left[\max(|\boldsymbol{R}|)\right] \\
\leq&\frac{c\sqrt{\text{size}(\boldsymbol{R})}}{q_{\max}} \cdot \mathbb{E}\left[\|\boldsymbol{R}\|_F\right]
\end{aligned}$$

SVDQuant

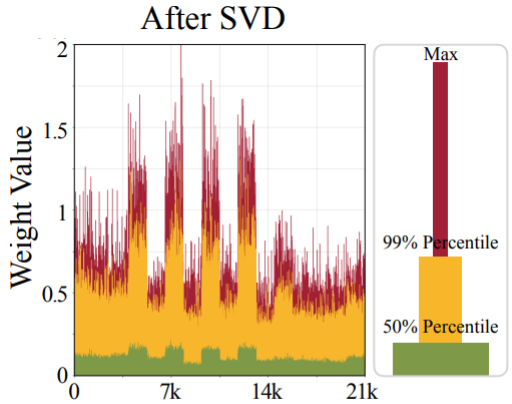
- The problem of minimizing $\left\|\widehat{W} - L_1 L_2\right\|_F$ can be solved by Singular Value Decomposition (SVD).
  - Minimize $\left\|\widehat{W} - L_1 L_2\right\|_F$ => Maximize $\|L_1 L_2\|_F$

- $\|\boldsymbol{A}\|_F = \sqrt{\sum_i^m \sum_j^n \boldsymbol{A_{ij}^2}} = \sqrt{trace(A^T A)} = \sqrt{\sum_{i=1}^{\min\{m,n\}} \boldsymbol{\sigma}_i^2(\boldsymbol{A})}$, where $\sigma_i(A)$ is i-th singular value of A.

- Given SVD of $\widehat{W} = U\Sigma V$, the optimal solution is $\boldsymbol{L_1 = U\Sigma_{:,:r}}$ **and** $\boldsymbol{L_2 = V_{:r,:}}$
  - By removing dominant values, the magnitude of the residual R is dramatically reduced, as $\|\boldsymbol{R}\|_F = \sqrt{\sum_{i=r+1}^{\min\{m,n\}} \boldsymbol{\sigma}_i^2(\boldsymbol{A})}$, compared to $\left\|\widehat{W}\right\|_F = \sqrt{\sum_{i=1}^{\min\{m,n\}} \boldsymbol{\sigma}_i^2(\boldsymbol{A})}$



(d) $|\hat{\boldsymbol{W}}| = |\boldsymbol{W} \cdot \mathrm{diag}(\boldsymbol{\lambda})|$



(e) $|\boldsymbol{R}| = |\hat{\boldsymbol{W}} - \boldsymbol{L_1 L_2}|$

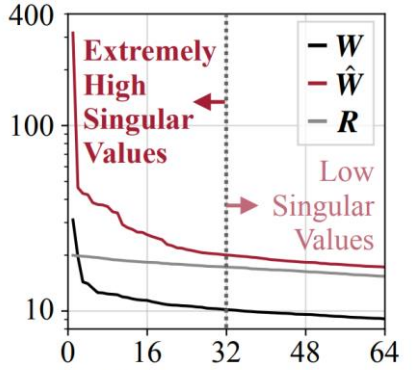
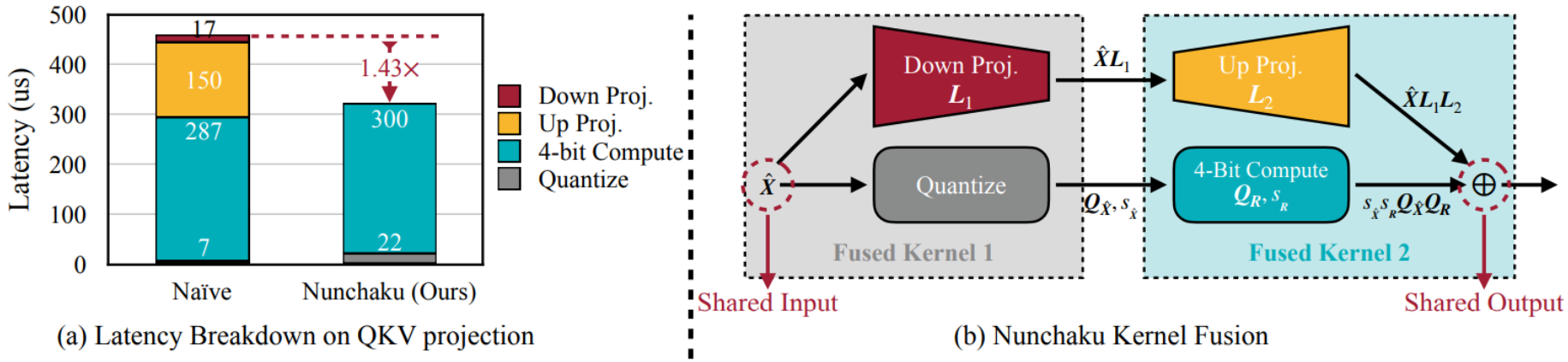


Figure 5: First 64 singular values of $\boldsymbol{W}$, $\hat{\boldsymbol{W}}$, and $\boldsymbol{R}$.

Nunchaku:
Fusing Low-rank And Low-bit Branch Kernels

- Although the low-rank branch introduces negligible computation **in theory**, running it as a separate branch incurs large latency overhead.



(a) Latency Breakdown on QKV projection

(b) Nunchaku Kernel Fusion

Figure 6: (a) Naïvely running low-rank branch with rank 32 will introduce 57% latency overhead due to extra read of 16-bit inputs in *Down Projection* and extra write of 16-bit outputs in *Up Projection*.

- Because the input and output activation sizes remain unchanged, shifting **bottleneck** from computation to **memory access.**
- For example, Up projection for QKV projection is much slower since its output exceeds the available L2 cache, resulting in the **extra DRAM load and store operations.**

Nunchaku:
Fusing Low-rank And Low-bit Branch Kernels

- Author eliminate these extra memory access and halving the number of kernel calls by fusing.
  - 1. Down projection(L_1) shares the **same input** as quantization kernel => Fusing
  - 2. Up projection(L_2) shares the **same output** as the 4-bit computational kernel => Fusing

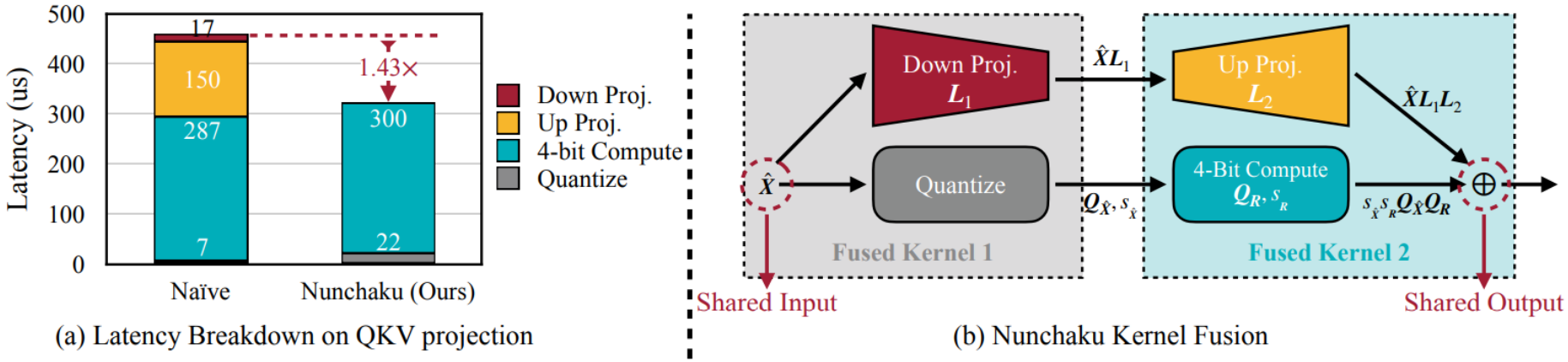


Figure 6: (a) Naïvely running low-rank branch with rank 32 will introduce 57% latency overhead due to extra read of 16-bit inputs in *Down Projection* and extra write of 16-bit outputs in *Up Projection*.

- As a result, low-rank branch adds only 5~10% latency.

Experimental Setup

- Models : **FLUX.1** | **PixArt-Σ** | **SANA** | **SDXL** | **SDXL-Turbo** |

- Datasets : **COCO Captions 2024** → for Calibration
  MJHQ-30K → 5K prompts for benchmarking
  sDCI → 5K prompts for benchmarking

- Comparison : **NF4** | **ViDiT-Q** | **MixDQ** | **TensorRT** | **SVDQuant (ours)**

Baselines Comparison

- PTQ-based Quantization Methods

Category	NF4	ViDiT-Q	MixDQ	TensorRT
Precision	W4	W8A8	W4A8	W8A8
Core Technique	NormalFloat: nonlinear normal distribution-based 4-bit	Token-level quantization + smoothing	16-bit for initial tokens + 4-bit for the rest	Percentile-based calibration + smoothing
Strengths	Very low memory usage, strong weight preservation	Strong for text alignment and token outliers	Minimal performance loss, maximizing compression efficiency	Optimized for production, fast inference, stable
Weaknesses	Fixed activation precision (needs FP16 etc.)	Limited memory savings due to 8-bit usage	Performance varies by sentence length, complex logic	Hard to customize, only activation is quantized
Use Cases	FLUX.1	PixArt-Σ	SDXL-Turbo	SDXL, LLaMA

Models Comparison

- Diffusion Transformer & UNet based Models

Model	Parameters (B)	Architecture	Step Type	Estimated Memory (FP16)	Use Case
FLUX.1	12B	DiT (Large)	multi-step	22.2 GiB	High-fidelity generation, LoRA support
PixArt-Σ	0.6B	DiT (Compact PixArt variant)	multi-step	2~3 GiB	Compact model, alignment-sensitive
SANA	0.6B	Linear DiT	1-step	<4 GiB	Real-time generation, low-latency
SDXL	2.6B	LDM (multi-step)	multi-step (~30)	6~7 GiB	High-quality baseline diffusion
SDXL-Turbo	~1B	Optimized SDXL (1~2 steps)	1~2 steps	4~5 GiB	Latency-critical, real-time image gen

Visual Quality Results

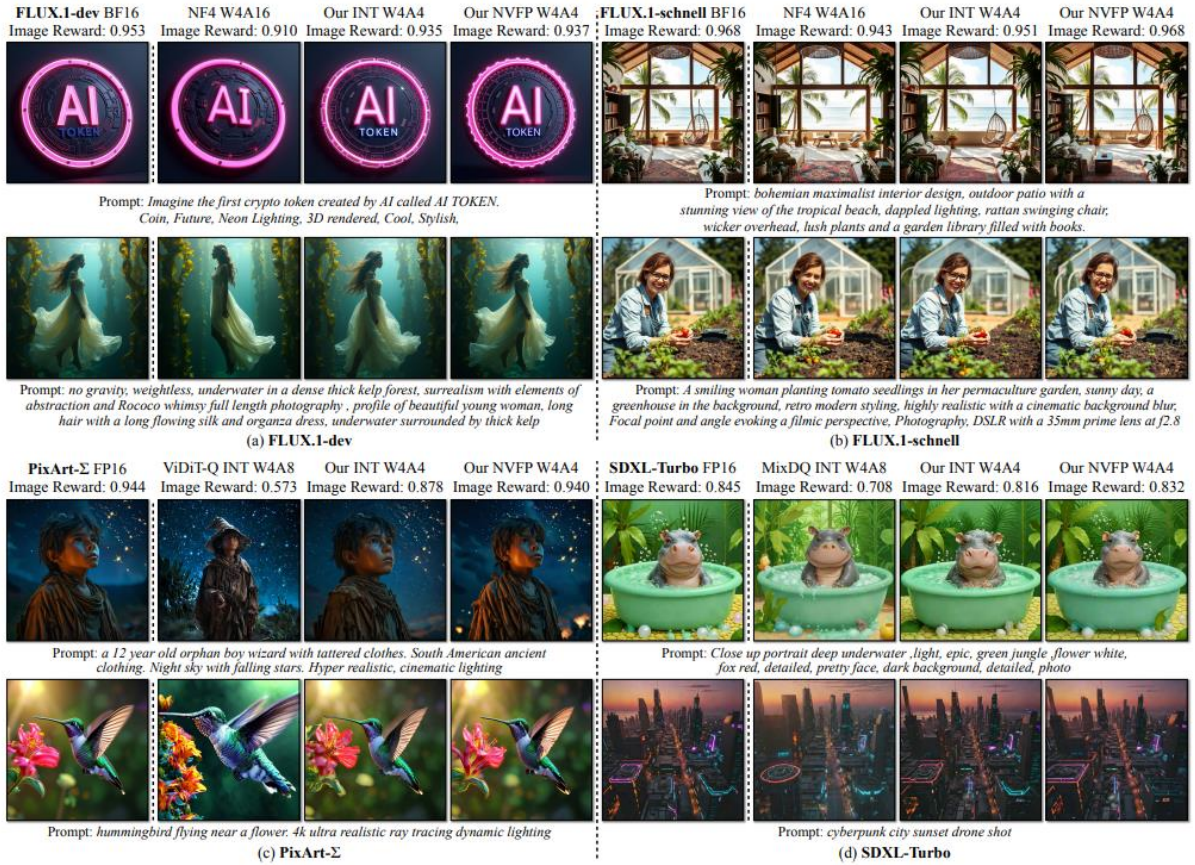


Figure 7: Qualitative visual results on MJHQ. Image Reward is calculated over the entire dataset. On FLUX.1 models, our 4-bit models outperform the NF4 W4A16 baselines, demonstrating superior text alignment and closer similarity to the 16-bit models. For instance, NF4 misses the swinging chair in the top right example. On PixArt-Σ and SDXL-Turbo, our 4-bit results demonstrate noticeably better visual quality than ViDiT-Q's and MixDQ's W4A8 results.

Backbone	Model	Precision	Method	MJHQ				sDCI			
				Quality		Similarity		Quality		Similarity	
				FID (↓)	IR (↑)	LPIPS (↓)	PSNR(↑)	FID (↓)	IR (↑)	LPIPS (↓)	PSNR (↑)
DiT	FLUX.1 -dev (50 Steps)	BF16	–	20.3	0.953	–	–	24.8	1.02	–	–
		INT W8A8	Ours	20.4	0.948	0.089	27.0	24.7	1.02	0.106	24.9
		W4A16	NF4	20.6	0.910	0.272	19.5	24.9	0.986	0.292	18.2
		INT W4A4	Ours	**19.9**	0.935	0.223	21.0	**24.2**	**1.01**	0.240	19.7
		NVFP W4A4	Ours	20.4	**0.937**	**0.208**	**21.4**	24.7	**1.01**	**0.218**	**20.2**
	FLUX.1 -schnell (4 Steps)	BF16	–	19.2	0.938	–	–	20.8	0.932	–	–
		INT W8A8	Ours	19.2	0.966	0.120	22.9	20.7	0.975	0.133	21.3
		W4A16	NF4	18.9	0.943	0.257	18.2	20.7	0.953	0.263	17.1
		INT W4A4	Ours	**18.3**	0.951	0.258	18.3	**20.1**	**0.979**	0.260	17.2
		NVFP W4A4	Ours	19.0	**0.968**	**0.227**	**19.0**	20.5	**0.979**	**0.226**	**18.1**
	PixArt-Σ (20 Steps)	FP16	–	16.6	0.944	–	–	24.8	0.966		
		INT W8A8	ViDiT-Q	**15.7**	0.944	0.137	22.5	**23.5**	**0.974**	0.163	20.4
		INT W8A8	Ours	16.3	**0.955**	**0.109**	**23.7**	24.2	0.969	**0.129**	**21.8**
		INT W4A8	ViDiT-Q	37.3	0.573	0.611	12.0	40.6	0.600	0.629	11.2
		INT W4A4	ViDiT-Q	412	-2.27	0.854	6.44	425	-2.28	0.838	6.70
		INT W4A4	Ours	19.2	0.878	0.323	17.6	25.9	0.918	0.352	16.5
		NVFP W4A4	Ours	**16.6**	**0.940**	**0.271**	**18.5**	**22.9**	**0.971**	**0.298**	**17.2**
	SANA -1.6B (20 Steps)	BF16	–	20.6	0.952	–	–	29.9	0.847	–	–
		INT W4A4	RTN	20.5	0.894	0.339	15.3	28.6	0.807	0.371	13.8
		INT W4A4	Ours	**19.3**	0.935	0.220	17.8	**28.1**	**0.846**	0.242	16.2
		NVFP W4A4	RTN	19.7	0.932	0.237	17.3	29.0	0.829	0.265	15.6
		NVFP W4A4	Ours	20.0	**0.955**	**0.177**	**19.0**	29.3	**0.846**	**0.196**	**17.3**
UNet	SDXL -Turbo (4 Steps)	FP16	–	24.3	0.845	–	–	24.7	0.705	–	–
		INT W8A8	MixDQ	**24.1**	0.834	0.147	21.7	25.0	0.690	0.157	21.6
		INT W8A8	Ours	24.3	**0.845**	**0.100**	**24.0**	**24.8**	**0.701**	**0.110**	**23.7**
		INT W4A8	MixDQ	27.7	0.708	0.402	15.7	25.9	0.610	0.415	15.7
		INT W4A4	MixDQ	353	-2.26	0.685	11.0	373	-2.28	0.686	11.3
		INT W4A4	Ours	24.6	0.816	0.262	18.1	26.0	0.671	0.272	18.0
		NVFP W4A4	Ours	**24.4**	**0.832**	**0.231**	**18.9**	**25.2**	**0.688**	**0.238**	**18.9**
	SDXL (30 Steps)	FP16	–	16.6	0.729	–	–	22.5	0.573	–	–
		INT W8A8	TensorRT	20.2	0.591	0.247	22.0	25.4	0.453	0.265	21.7
		INT W8A8	Ours	**16.6**	**0.718**	**0.119**	**26.4**	**22.4**	**0.574**	**0.129**	**25.9**
		INT W4A4	Ours	20.6	0.601	0.288	21.0	26.2	0.477	0.307	20.7
		NVFP W4A4	Ours	**18.3**	**0.640**	**0.250**	**21.8**	**23.9**	**0.502**	**0.261**	**21.7**

Integrate with LoRA

Integrate with LoRA (Low-Rank Adaptation)

- No re-quantization required for **LoRA** integration.
- Maintains 16-bit image quality across five LoRA styles, from Realism to Anime and Yarn Art.

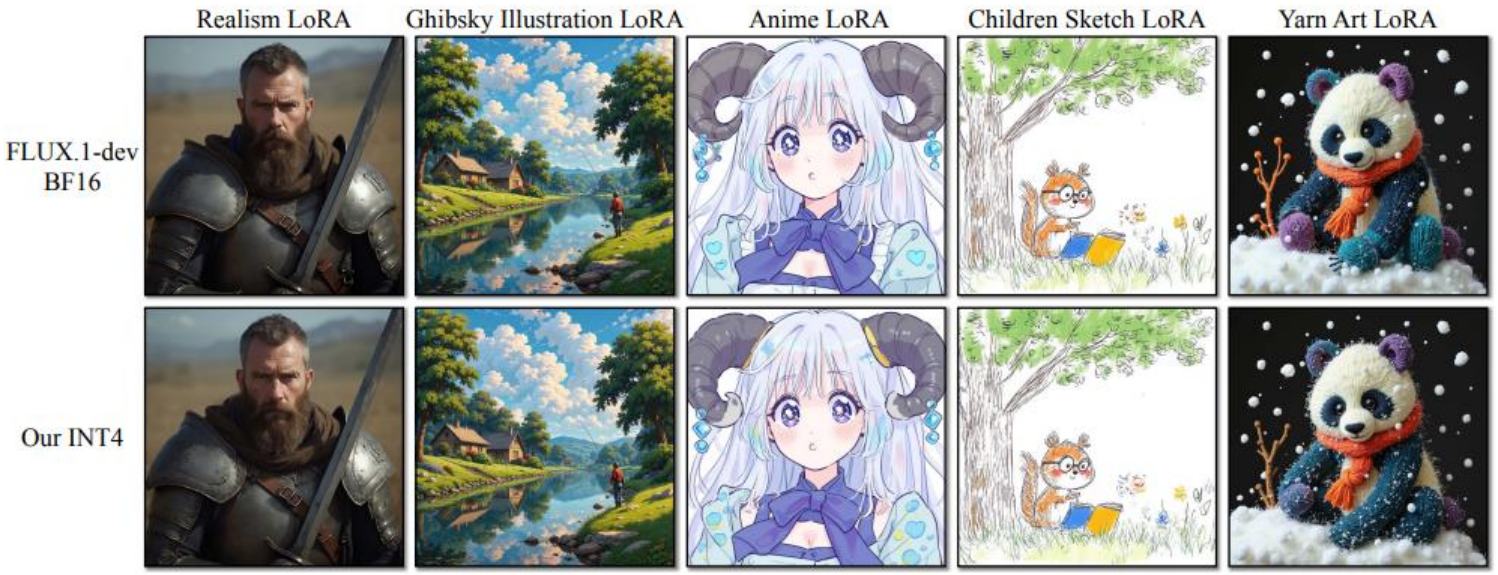


Figure 9: Our 4-bit model seamlessly integrates with off-the-shelf LoRAs without requiring requantization. When applying LoRAs, it matches the image quality of the original 16-bit FLUX.1-dev. See Appendix F for the text prompts.

$W' = W + \Delta W = W + AB$ (LoRA)

$A \in R^{m \times r_{LoRA}}, B \in R^{r_{LoRA} \times n} \cdots r_{LoRA} \ll \min(m, n)$

$\widehat{W} = L_1 L_2 + R$ (SVDQuant)

$\cdots A \in L_1^{m \times r}, L_2 \in R^{r \times n}$, R = residual

$XW \approx XL_1L_2 + Q(X)Q(R)$

$\rightarrow XW \approx X(L_1L_2 + AB) + Q(X)Q(R)$

$\rightarrow rank\ r' = r + r_{LoRA}$

Ablation Study

Ablation Study

- **Naïve quantization** and **SVD only** show severe degradation in 4-bit.
- **SVDQuant** improves quality by decomposing weights and quantizing only the residual Smoothing
- Low-rank branch absorbs outliers, achieving near-FP16 quality

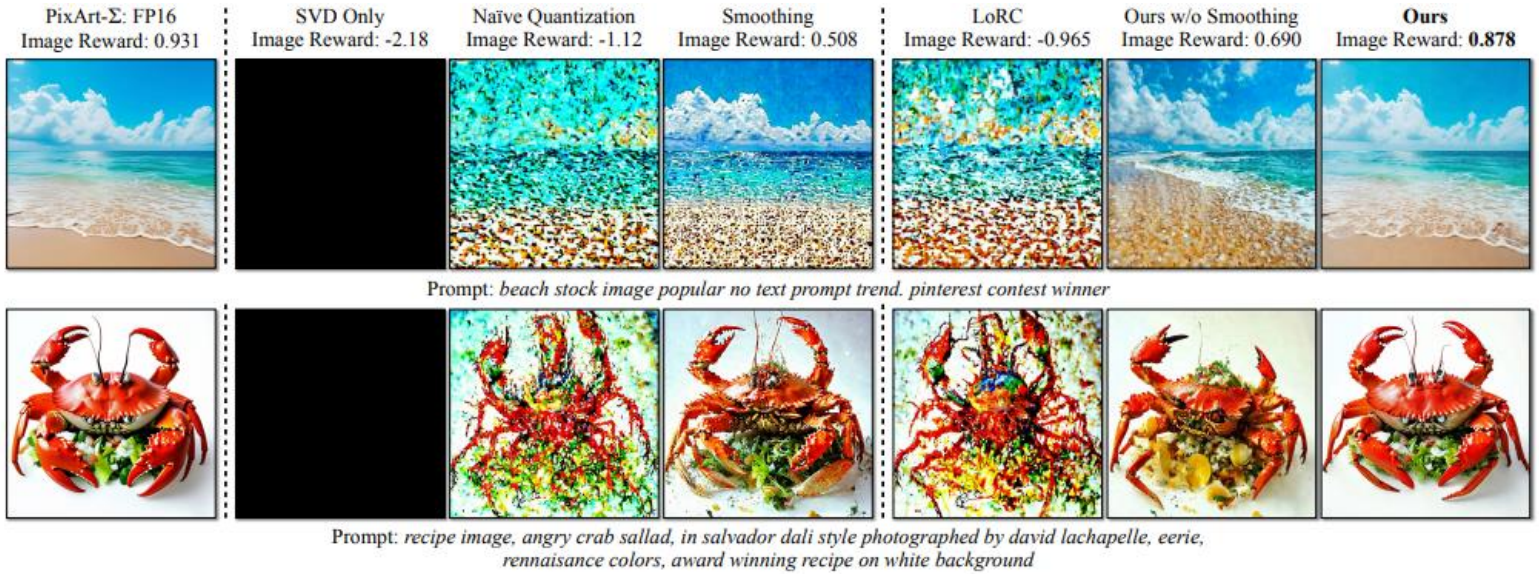


Figure 10: Ablation study of SVDQuant on PixArt-Σ. The rank of the low-rank branch is 64. Image Reward is measured over 1K samples from MJHQ. Our results significantly outperform the others, achieving the highest image quality by a wide margin.

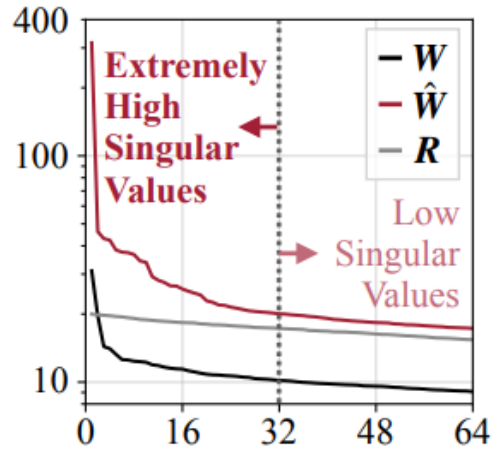


Figure 5: First 64 singular values of $\boldsymbol{W}$, $\hat{\boldsymbol{W}}$, and $\boldsymbol{R}$. The first 32 singular values of $\hat{\boldsymbol{W}}$ exhibit a steep drop, while the remaining values are much more gradual.

Increasing Rank

Increasing Rank

- **Higher rank(r)** improves image quality, with Image Reward increasing from 0.787(r=16) to 0.859(r=64).
- However, model size and latency overhead grow significantly, reaching 11.3% size and 8.8% latency at rank 64.
- **Rank 32** was chosen as a middle-ground trade-off in the paper

Figure 18: Increasing the rank r of the low-rank branch in SVDQuant can enhance image quality, but it also leads to higher parameter and latency overhead.

Memory save and speedup

Memory save and speedup

- Up to **3.6x model size** and **3.5x memory reduction** compared to BF16 models.
- **3.0~10.1x latency speedup** across RTX 4090 and RTX 5090.
- Outperforms NF4 (W4A16) in both efficiency and speed, while preserving quality.

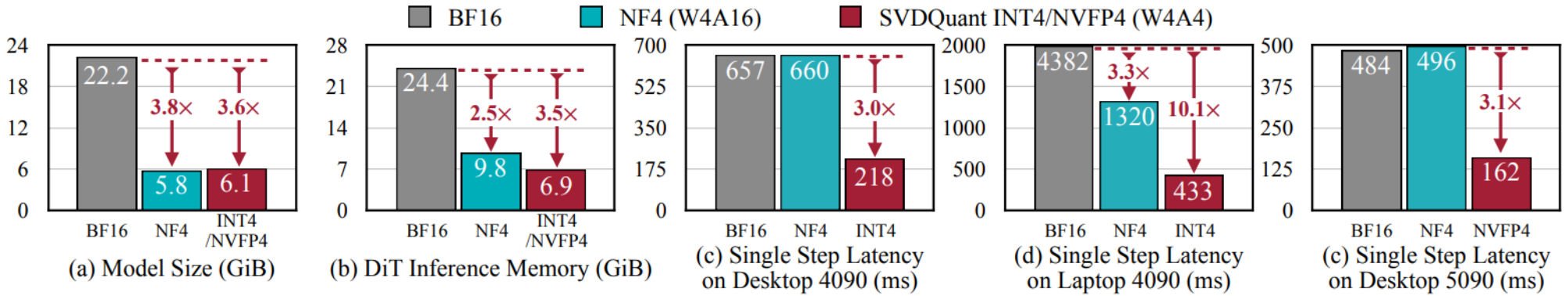


Figure 8: SVDQuant reduces the 12B FLUX.1 model size by 3.6× and cuts the 16-bit model's memory usage by 3.5×. With Nunchaku, our INT4 model runs 3.0× faster than the NF4 W4A16 baseline on both desktop and laptop NVIDIA RTX 4090 GPUs. Notably, on the laptop 4090, it achieves a total 10.1× speedup by eliminating CPU offloading. Our NVFP4 model is also 3.1× faster than both BF16 and NF4 on the RTX 5090 GPU.

Conclusion

- **SVDQuant** : A 4-bit PTQ method for diffusion models
- Introduces a low-rank branch to absorb outliers in weight and activations
- The Nunchaku engine fuses low-rank and 4-bit branches to reduce memory and overhead
- Achieves 3.5x memory reduction and 3.0 speedup on RTX4090/5090

Limitation

- Lack of fine-grained control over quantization granularity
- Limited generality and stability of activation quantization

Thank you