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

Anonymous authors
Paper under double-blind review

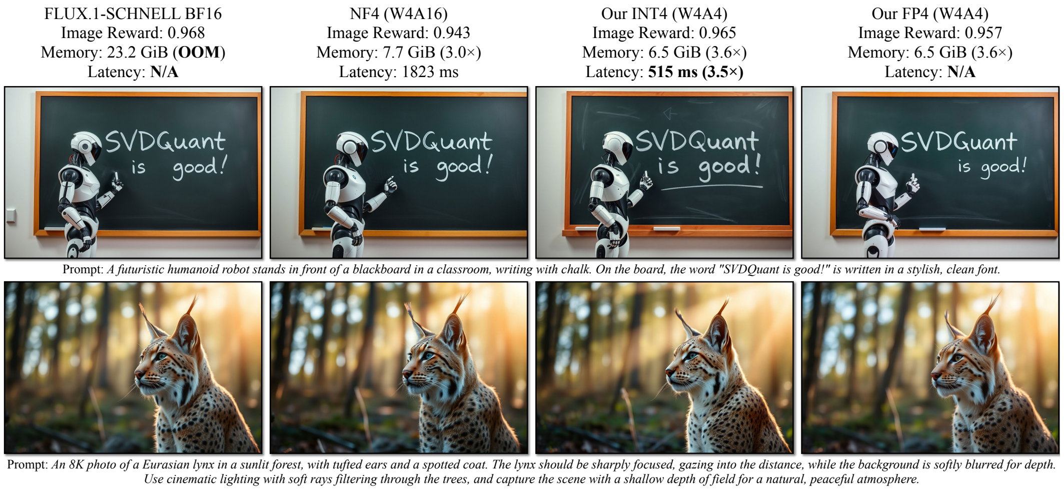

FLUX.1-SCHNELL BF16
Image Reward: 0.968
Memory: 23.2 GiB (**OOM**)
Latency: **N/A**
NF4 (W4A16)
Image Reward: 0.943
Memory: 7.7 GiB (3.0×)
Latency: 1823 ms
Our INT4 (W4A4)
Image Reward: 0.965
Memory: 6.5 GiB (3.6×)
Latency: **515 ms (3.5×)**
Our FP4 (W4A4)
Image Reward: 0.957
Memory: 6.5 GiB (3.6×)
Latency: **N/A**
SVDQuant is good!
Prompt: *A futuristic humanoid robot stands in front of a blackboard in a classroom, writing with chalk. On the board, the word "SVDQuant is good!" is written in a stylish, clean font.*
Prompt: *An 8K photo of a Eurasian lynx in a sunlit forest, with tufted ears and a spotted coat. The lynx should be sharply focused, gazing into the distance, while the background is softly blurred for depth. Use cinematic lighting with soft rays filtering through the trees, and capture the scene with a shallow depth of field for a natural, peaceful atmosphere.*

Figure 1: We present SVDQuant, a post-training quantization method for diffusion models. It effectively quantizes both weights and activations of 12B FLUX.1-schnell to 4 bits without compromising visual quality, as assessed by ImageReward (Xu et al., 2024a). Compared to the BF16 model, we reduce the memory usage by 3.6×. Compared to the NF4 W4A16 baseline, we achieve 3.6× speedup on a laptop with RTX-4090 GPU.

Abstract

Diffusion models have been proven highly effective at generating high-quality images. However, as these models grow larger, they require significantly more memory and suffer from higher latency, posing substantial challenges for deployment. In this work, we aim to accelerate diffusion models by quantizing their weights and activations to 4 bits. At such an aggressive level, both weights and activations are highly sensitive, where conventional post-training quantization methods for large language models like smoothing become insufficient. To overcome this limitation, we propose *SVDQuant*, a new 4-bit quantization paradigm. Different from smoothing which redistributes outliers between weights and activations, our approach *absorbs* these outliers using a low-rank branch. We first consolidate the outliers by shifting them from activations to weights, then employ a high-precision low-rank branch to take in the weight outliers with Singular Value Decomposition (SVD). This process eases the quantization on both sides. However, naïvely running the low-rank branch independently incurs significant overhead due to extra data movement of activations, negating the quantization speedup. To address this, we co-design an inference engine *LoRunner* that fuses the kernels of the low-rank branch into those of the low-bit branch to cut off redundant memory access. It can also seamlessly support off-the-shelf low-rank adapters (LoRAs) without the need for re-quantization. Extensive experiments on SDXL, PixArt-Σ, and FLUX.1 validate the effectiveness of SVDQuant in preserving image quality. We reduce the memory usage for the 12B FLUX.1 models by 3.6×, achieving 3.5× speedup over the 4-bit weight-only quantized baseline on a 16GB RTX-4090 GPU, paving the way for more interactive applications on PCs. We will release the code and models upon publication.

1 INTRODUCTION

Diffusion models have shown remarkable capabilities in generating high-quality images (Ho et al., 2020), with recent advances further enhancing user control over the generation process. Trained on vast data, these models can create stunning images from simple text prompts, unlocking diverse image editing and synthesis applications (Meng et al., 2022b; Ruiz et al., 2023; Zhang et al., 2023).

To pursue higher image quality and more precise text-to-image alignment, researchers are increasingly scaling up diffusion models. As shown in Fig. 2, Stable Diffusion (SD) (Rombach et al., 2022) 1.4 only has 800M parameters, while SDXL (Podell et al., 2024) scales this up to 2.6B parameters. AuraFlow v0.1 (fal.ai, 2024) extends this further to 6B parameters, with the latest model, FLUX.1 (Black-Forest-Labs, 2024), pushing the boundary to 12B parameters. Compared to large language models (LLMs), diffusion models are significantly more computationally intensive. Their computational costs[*] increase more rapidly with model size, posing a prohibitive memory and latency barrier for real-world model deployment, particularly for interactive use cases that demand low latency.

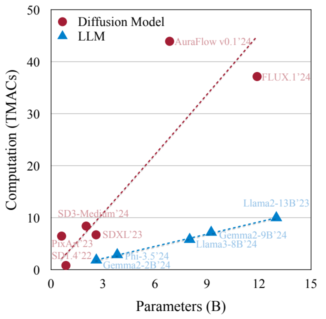


Figure 2: Computation *vs*. parameters for LLMs and diffusion models. LLMs' computation is measured with 512 context and 256 output tokens, and diffusion models' computation is for a single step. Dashed lines show trends.

As Moore's law slows down, hardware vendors are turning to low-precision inference to sustain performance improvements. For instance, NVIDIA's Blackwell Tensor Cores introduce a new 4-bit floating point (FP4) precision, doubling the performance compared to FP8 (NVIDIA, 2024). Therefore, using 4-bit inference to accelerate diffusion models is appealing. In the realm of LLMs, researchers have leveraged quantization to compress model sizes and boost inference speed (Dettmers et al., 2022; Xiao et al., 2023). However, unlike LLMs–where latency is primarily constrained by loading model weights on modern GPUs, especially with small batch sizes–diffusion models are heavily computationally bound, even with a single batch. As a result, weight-only quantization cannot accelerate diffusion models on GPUs. To achieve speedup on these devices, both weights and activations must be quantized to the same bit width; otherwise, the lower-precision weight will be upcast during computation, negating potential performance enhancements.

In this work, we focus on quantizing both the weights and activations of diffusion models to 4 bits. This challenging and aggressive scheme is often prone to severe quality degradation. Existing methods like smoothing (Xiao et al., 2023; Lin et al., 2024a), which attempt to transfer the outliers between the weights and activations, are less effective since both sides are highly vulnerable to outliers. To address this issue, we propose a new general-purpose quantization paradigm, *SVDQuant*. Our core idea is to introduce a low-cost branch to absorb outliers on both sides. To achieve this, as illustrated in Fig. 3, we first aggregate the outliers by migrating them from activation $\boldsymbol{X}$ to weight $\boldsymbol{W}$ via smoothing. Then we apply Singular Value Decomposition (SVD) to the updated weight, $\hat{\boldsymbol{W}}$, decomposing it into a low-rank branch $\boldsymbol{L}_1\boldsymbol{L}_2$ and a residual $\hat{\boldsymbol{W}} - \boldsymbol{L}_1\boldsymbol{L}_2$. The low-rank branch operates at 16 bits, allowing us to quantize only the residual to 4 bits, which has significantly reduced outliers and magnitude. However, naively running the low-rank branch separately incurs substantial memory access overhead, offsetting the speedup of 4-bit inference. To overcome this, we co-design a specialized inference engine *LoRunner*, which fuses the low-rank branch computation into the 4-bit quantization and computation kernels. This design enables us to achieve measured inference speedup even with additional branches.

SVDQuant can quantize various text-to-image diffusion architectures, including both UNet (Ho et al., 2020; Ronneberger et al., 2015) and DiT (Peebles & Xie, 2023) backbones, into 4 bits, while maintaining visual quality. It supports both INT4 and FP4 data types, and integrates seamlessly with pre-trained low-rank adapters (LoRA) (Hsu et al., 2022) without requiring re-quantization. To our knowledge, we are the first to successfully apply 4-bit post-training quantization to both the weights and activations of diffusion models, and achieve measured speedup on NVIDIA GPUs. On the latest 12B FLUX.1, we largely preserve the image quality and reduce the memory footprint of the original BF16 model by 3.6× and deliver a 3.5× speedup over the NF4 weight-only quantized baseline, measured on a 16GB laptop-level RTX4090 GPU. See Fig. 1 for visual examples.

[*]Computational cost is measured by number of Multiply-Accumulate operations (MACs). 1 MAC=2 FLOPs.

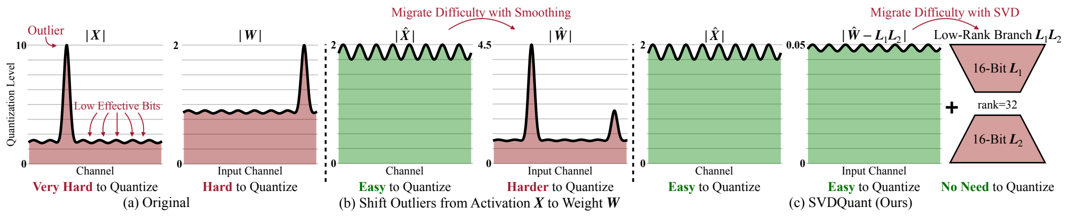


Figure 3: Overview of SVDQuant. (a) Originally, both the activation $\boldsymbol{X}$ and weight $\boldsymbol{W}$ contain outliers, making 4-bit quantization challenging. (b) We migrate the outliers from the activation to weight, resulting in the updated activation $\hat{\boldsymbol{X}}$ and weight $\hat{\boldsymbol{W}}$. While $\hat{\boldsymbol{X}}$ becomes easier to quantize, $\hat{\boldsymbol{W}}$ now becomes more difficult. (c) SVDQuant further decomposes $\hat{\boldsymbol{W}}$ into a low-rank component $\boldsymbol{L}_1\boldsymbol{L}_2$ and a residual $\hat{\boldsymbol{W}} - \boldsymbol{L}_1\boldsymbol{L}_2$ with SVD. Thus, the quantization difficulty is alleviated by the low-rank branch, which runs at 16-bit precision.

2 Related Work

Diffusion models. Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) have emerged as a powerful class of generative models, known for their ability to generate high-quality samples by modeling the data distribution through an iterative denoising process. Recent advancements in text-to-image diffusion models (Balaji et al., 2022; Rombach et al., 2022; Podell et al., 2024) have already revolutionized content generation. Researchers further shifted from convolution-based UNet architectures (Ronneberger et al., 2015; Ho et al., 2020) to transformers (*e.g.*, DiT (Peebles & Xie, 2023) and U-ViT (Bao et al., 2023)) and scaled up the model size (Esser et al., 2024). However, diffusion models suffer from extremely slow inference speed due to their long denoising sequences and intense computation. To address this, various approaches have been proposed, including few-step samplers (Zhang & Chen, 2022; Zhang et al., 2022; Lu et al., 2022) or distilling fewer-step models from pre-trained ones (Salimans & Ho, 2021; Meng et al., 2022a; Song et al., 2023; Luo et al., 2023; Sauer et al., 2023; Yin et al., 2024b;a; Kang et al., 2024). Another line of works choose to optimize or accelerate computation via efficient architecture design (Li et al., 2023b; 2020; Cai et al., 2024; Liu et al., 2024b), quantization (Shang et al., 2023; Li et al., 2023a), sparse inference (Li et al., 2022; Ma et al., 2024c;b), and distributed inference (Li et al., 2024b; Wang et al., 2024c; Chen et al., 2024c). This work focuses on quantizing the diffusion models to 4 bits to reduce the computation complexity. Our method can also be applied to the few-step diffusion models to further reduce the latency (see Sec. 5.2).

Quantization. Quantization has been recognized as an effective approach for LLMs to reduce the model size and accelerate inference (Dettmers et al., 2022; Frantar et al., 2023; Xiao et al., 2023; Lin et al., 2024b;a; Kim et al., 2024; Zhao et al., 2024d). For diffusion models, Q-Diffusion (Li et al., 2023a) and PTQ4DM (Shang et al., 2023) first achieved 8-bit quantization. Subsequent works refined these techniques with approaches like sensitivity analysis (Yang et al., 2023) and timestep-aware quantization (He et al., 2023; Huang et al., 2024; Liu et al., 2024c; Wang et al., 2024a). Some recent works extended the setting to text-to-image models (Tang et al., 2023; Zhao et al., 2024c), DiT backbones (Wu et al., 2024), quantization-aware training (He et al., 2024; Zheng et al., 2024; Wang et al., 2024b; Sui et al., 2024), video generation (Zhao et al., 2024b), and different data types (Liu & Zhang, 2024). Among these works, only MixDQ (Zhao et al., 2024c) and ViDiT-Q (Zhao et al., 2024b) implement low-bit inference engines and report measured 8-bit speedup on GPUs. In this work, we push the boundary further by quantizing diffusion models to 4 bits, supporting both the integer or floating-point data types, compatible with the UNet backbone (Ho et al., 2020) and recent DiT architecture (Peebles & Xie, 2023). Our co-designed inference engine, LoRunner, further ensures on-hardware speedup. Additionally, when applying LoRA to the model, existing methods require fusing the LoRA branch to the main branch and re-quantizing the model to avoid tremendous memory-access overhead in the LoRA branch. LoRunner cuts off this overhead via kernel fusion, allowing the low-rank branch to run efficiently as a separate branch, eliminating the need for re-quantization.

Low-rank decomposition. Low-rank decomposition has gained significant attention in deep learning for enhancing computational and memory efficiency (Hu et al., 2022; Zhao et al., 2024a; Jaiswal et al., 2024). While directly applying this approach to model weights can reduce the compute and memory demands (Hsu et al., 2022; Yuan et al., 2023; Li et al., 2023c), it often leads to performance degradation. Instead, Yao et al. (2023) combined it with quantization for model compression, employing a low-rank branch to compensate for the quantization error. Low-Rank Adaptation (LoRA) (Hu et al., 2022) introduces another important line of research by using low-rank matrices to adjust a subset of pre-trained weights for efficient fine-tuning. This has sparked numerous advancements (Dettmers et al., 2023; Guo et al., 2024; Li et al., 2024c; Xu et al., 2024b; Meng et al., 2024), which combines quantized models with low-rank adapters to reduce memory usage during model fine-tuning. However,

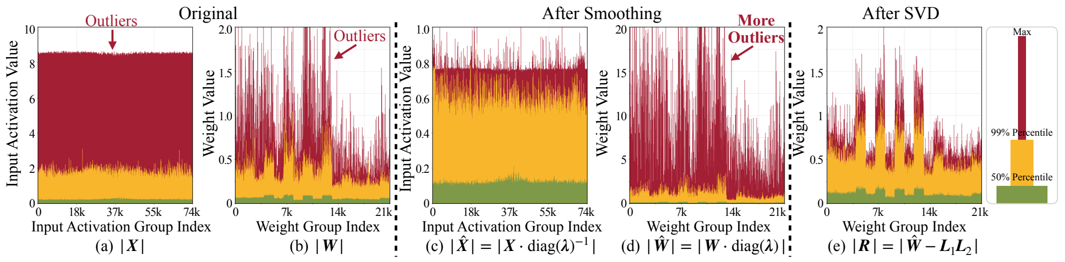


Figure 4: Example value distribution of inputs and weights in PixArt-Σ. $\boldsymbol{\lambda}$ is the smooth factor. Red indicates the outliers. Initially, both the input $\boldsymbol{X}$ and weight $\boldsymbol{W}$ contain significant outliers. After smoothing, the range of $\hat{\boldsymbol{X}}$ is reduced with much fewer outliers, while $\hat{\boldsymbol{W}}$ shows more outliers. Once the SVD low-rank branch $\boldsymbol{L}_1\boldsymbol{L}_2$ is subtracted, the residual $\boldsymbol{R}$ has a narrower range and is free from outliers.

our work differs in two major aspects compared to this line of work that fine-tunes LoRA branches on a quantized base model. Firstly, our goal is different, as we aim to accelerate model inference through quantization, while previous works focus on model compression or efficient fine-tuning. Thus, they primarily consider weight-only quantization, resulting in no speedup. Secondly, as shown in our experiments (Fig. 6 and ablation study in Sec. 5.2), directly applying these methods not only degrades the image quality but also introduces significant overhead. In contrast, our method yields much better performance due to our joint quantization of weights and activations and our inference engine LoRunner minimizes the overhead by fusing the low-rank branch kernels into the low-bit computation ones.

3 Quantization Preliminary

Quantization is an effective approach to accelerate linear layers in networks. Given a tensor $\boldsymbol{X}$, the quantization process is defined as:

$$\boldsymbol{Q}_{\boldsymbol{X}} = \text{round}\left(\frac{\boldsymbol{X}}{s_{\boldsymbol{X}}}\right), s_{\boldsymbol{X}} = \frac{\max(|\boldsymbol{X}|)}{q_{\max}}. \tag{1}$$

Here, $\boldsymbol{Q}_{\boldsymbol{X}}$ is the low-bit representation of $\boldsymbol{X}$, $s_{\boldsymbol{X}}$ is the scaling factor, and $q_{\max}$ is the maximum quantized value. For signed b-bit integer quantization, $q_{\max} = 2^{b-1} - 1$. For 4-bit floating-point quantization with 1-bit mantissa and 2-bit exponent, $q_{\max} = 6$. Thus, the dequantized tensor can be formulated as $Q(\boldsymbol{X}) = s_{\boldsymbol{X}} \cdot \boldsymbol{Q}_{\boldsymbol{X}}$. For a linear layer with input $\boldsymbol{X}$ and weight $\boldsymbol{W}$, its computation can be approximated by

$$\boldsymbol{X}\boldsymbol{W} \approx Q(\boldsymbol{X})Q(\boldsymbol{W}) = s_{\boldsymbol{X}}s_{\boldsymbol{W}} \cdot \boldsymbol{Q}_{\boldsymbol{X}}\boldsymbol{Q}_{\boldsymbol{W}}. \tag{2}$$

The same approximation applies to convolutional layers. To speed up computation, modern commercial GPUs require both $\boldsymbol{Q}_{\boldsymbol{X}}$ and $\boldsymbol{Q}_{\boldsymbol{W}}$ using the same bit width. Otherwise, the low-bit weights need to be upcast to match the higher bit width of activations, or vice versa, negating the speed advantage. Following the notation in QServe (Lin et al., 2024b), we denote x-bit weight, y-bit activation as WxAy. “INT” and “FP” refer to the integer and floating-point data types, respectively.

In this work, we focus on W4A4 quantization for acceleration, where outliers in both weights and activations place substantial obstacles. Traditional methods to suppress these outliers include quantization-aware training (QAT) (He et al., 2024) and rotation (Ashkboos et al., 2024; Liu et al., 2024d; Lin et al., 2024b). QAT requires massive computing resources, especially for tuning models with more than 10B parameters (*e.g.*, FLUX.1). Rotation is inapplicable due to the usage of adaptive normalization layers (Peebles & Xie, 2023) in diffusion models. The runtime-generated normalization weights preclude the offline integration of the rotation matrix with the weights of projection layers. Consequently, online rotation of both activations and weights incurs significant runtime overhead.

4 Method

In this section, we first formulate our problem and discuss where the quantization error comes from. Next, we present SVDQuant, a new W4A4 quantization paradigm for diffusion models. Our key idea is to introduce an additional low-rank branch that can absorb quantization difficulties in both weights and activations. Finally, we provide a co-designed inference engine LoRunner with kernel fusion to minimize the overhead of the low-rank branches in the 4-bit model.

4.1 Problem Formulation

Consider a linear layer with input $\boldsymbol{X} \in \mathbb{R}^{b\times m}$ and weight $\boldsymbol{W} \in \mathbb{R}^{m\times n}$. The quantization error can be defined as

$$E(\boldsymbol{X}, \boldsymbol{W}) = \|\boldsymbol{X}\boldsymbol{W} - Q(\boldsymbol{X})Q(\boldsymbol{W})\|_F, \tag{3}$$

where $\|\cdot\|_F$ denotes Frobenius Norm.

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

$$E(\boldsymbol{X}, \boldsymbol{W}) \le \|\boldsymbol{X}\|_F \|\boldsymbol{W} - Q(\boldsymbol{W})\|_F + \|\boldsymbol{X} - Q(\boldsymbol{X})\|_F (\|\boldsymbol{W}\|_F + \|\boldsymbol{W} - Q(\boldsymbol{W})\|_F). \tag{4}$$

See App. A.1 for the proof. From the proposition, we can see that the error is bounded by four elements – the magnitude of the weight and input, $\|\boldsymbol{W}\|_F$ and $\|\boldsymbol{X}\|_F$, and their respective quantization errors, $\|\boldsymbol{W} - Q(\boldsymbol{W})\|_F$ and $\|\boldsymbol{X} - Q(\boldsymbol{X})\|_F$. To minimize the overall quantization error, we aim to optimize these four terms.

4.2 SVDQuant: Absorbing Outliers via Low-Rank Branch

Migrate outliers from activation to weight. Smoothing (Xiao et al., 2023; Lin et al., 2024a) is an effective approach for reducing outliers. We can smooth outliers in activations by scaling down the input $\boldsymbol{X}$ and adjusting the weight matrix $\boldsymbol{W}$ correspondingly using a per-channel smoothing factor $\boldsymbol{\lambda} \in \mathbb{R}^m$. As shown in Fig. 4(a)(c), the smoothed input $\hat{\boldsymbol{X}} = \boldsymbol{X} \cdot \text{diag}(\boldsymbol{\lambda})^{-1}$ exhibits reduced magnitude and fewer outliers, resulting in lower input quantization error. However, in Fig. 4(b)(d), the transformed weight $\hat{\boldsymbol{W}} = \boldsymbol{W} \cdot \text{diag}(\boldsymbol{\lambda})$ has a significant increase in both magnitude and the presence of outliers, which in turn raises the weight quantization error. Consequently, the overall error reduction is limited.

Absorb magnified weight outliers with a low-rank branch. Our core insight is to introduce a 16-bit low-rank branch and further migrate the weight quantization difficulty to this branch. Specifically, we decompose the transformed weight as $\hat{\boldsymbol{W}} = \boldsymbol{L}_1\boldsymbol{L}_2 + \boldsymbol{R}$, where $\boldsymbol{L}_1 \in \mathbb{R}^{m\times r}$ and $\boldsymbol{L}_2 \in \mathbb{R}^{r\times n}$ are two low-rank factors of rank r, and $\boldsymbol{R}$ is the residual. Then $\boldsymbol{X}\boldsymbol{W}$ can be approximated as

$$\boldsymbol{X}\boldsymbol{W} = \hat{\boldsymbol{X}}\hat{\boldsymbol{W}} = \hat{\boldsymbol{X}}\boldsymbol{L}_1\boldsymbol{L}_2 + \hat{\boldsymbol{X}}\boldsymbol{R} \approx \underbrace{\hat{\boldsymbol{X}}\boldsymbol{L}_1\boldsymbol{L}_2}_{\text{16-bit low-rank branch}} + \underbrace{Q(\hat{\boldsymbol{X}})Q(\boldsymbol{R})}_{\text{4-bit residual}}. \tag{5}$$

Compared to direct 4-bit quantization, i.e., $Q(\hat{\boldsymbol{X}})Q(\boldsymbol{W})$, our method first computes the low-rank branch $\hat{\boldsymbol{X}}\boldsymbol{L}_1\boldsymbol{L}_2$ in 16-bit precision, and then approximates the residual $\hat{\boldsymbol{X}}\boldsymbol{R}$ with 4-bit quantization. Empirically, $r \ll \min(m, n)$, and is typically set to 16 or 32. As a result, the additional parameters and computation for the low-rank branch are negligible, contributing only $\frac{mr+nr}{mn}$ to the overall costs. However, it still requires careful system design to eliminate redundant memory access, which we will discuss in Sec. 4.3.

From Eq. 5, the quantization error can be expressed as

$$\left\|\hat{\boldsymbol{X}}\hat{\boldsymbol{W}} - (\hat{\boldsymbol{X}}\boldsymbol{L}_1\boldsymbol{L}_2 + Q(\hat{\boldsymbol{X}})Q(\boldsymbol{R}))\right\|_F = \left\|\hat{\boldsymbol{X}}\boldsymbol{R} - Q(\hat{\boldsymbol{X}})Q(\boldsymbol{R})\right\|_F = E(\hat{\boldsymbol{X}}, \boldsymbol{R}), \tag{6}$$

where $\boldsymbol{R} = \hat{\boldsymbol{W}} - \boldsymbol{L}_1\boldsymbol{L}_2$. According to Proposition 4.1, since $\hat{\boldsymbol{X}}$ is already free from outliers, we only need to focus on optimizing the magnitude of $\boldsymbol{R}$, $\|\boldsymbol{R}\|_F$ and its quantization error, $\|\boldsymbol{R} - Q(\boldsymbol{R})\|_F$.

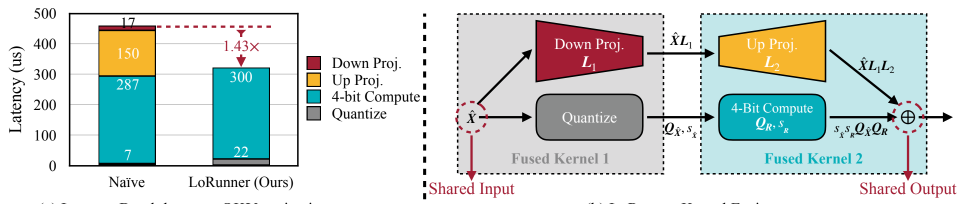
Proposition 4.2 (Quantization error bound)**.** *For any tensor $\boldsymbol{R}$ and quantization method described in Eq. 1 as $Q(\boldsymbol{R}) = s_{\boldsymbol{R}} \cdot \boldsymbol{Q}_{\boldsymbol{R}}$. Assuming the elements of $\boldsymbol{R}$ follow a distribution that satisfies the following regularity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

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

Then, we have

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

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



(a) Latency Breakdown on QKV projection (b) LoRunner 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*. Our LoRunner engine optimizes this overhead with kernel fusion. (b) *Down Projection* and *Quantize* kernels use the same input, while *Up Projection* and *4-Bit Compute* kernels share the same output. To reduce data movement overhead, we fuse the first two and the latter two kernels together.

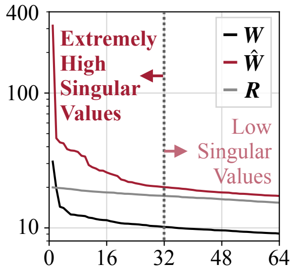


Figure 5: First 64 singular values of $\boldsymbol{W}$, $\hat{\boldsymbol{W}}$, and $\boldsymbol{R}$. The first 32 singular values of $\hat{\boldsymbol{W}}$ exhibit a steep drop, while the remaining values are much more gradual.

See App. A.2 for the proof. From this proposition, we obtain the intuition that the quantization error $\|\boldsymbol{R} - Q(\boldsymbol{R})\|_F$ is bounded by the magnitude of the residual $\|\boldsymbol{R}\|_F$. Thus, our goal is to find the optimal $\boldsymbol{L}_1\boldsymbol{L}_2$ that minimizes $\|\boldsymbol{R}\|_F = \left\|\hat{\boldsymbol{W}} - \boldsymbol{L}_1\boldsymbol{L}_2\right\|_F$, which can be solved by simple Singular Value Decomposition (SVD). Given the SVD of $\hat{\boldsymbol{W}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$, the optimal solution is $\boldsymbol{L}_1 = \boldsymbol{U}\boldsymbol{\Sigma}_{:,:r}$ and $\boldsymbol{L}_2 = \boldsymbol{V}_{:r,:}$. Fig. 5 illustrates the singular value distribution of the original weight $\boldsymbol{W}$, transformed weight $\hat{\boldsymbol{W}}$ and residual $\boldsymbol{R}$. The singular values of the original weight $\boldsymbol{W}$ are highly imbalanced. After smoothing, the singular value distribution of $\hat{\boldsymbol{W}}$ becomes even sharper, with only the first several values being significantly larger. By removing these dominant values, Eckart–Young–Mirsky theorem[†] suggests that the magnitude of the residual $\boldsymbol{R}$ is dramatically reduced, as $\|\boldsymbol{R}\|_F = \sqrt{\sum_{i=r+1}^{\min(m,n)} \sigma_i^2}$, compared to the original magnitude $\left\|\hat{\boldsymbol{W}}\right\|_F = \sqrt{\sum_{i=1}^{\min(m,n)} \sigma_i^2}$, where σ_i is the i-th singular value of $\hat{\boldsymbol{W}}$. Furthermore, empirical observations reveal that $\boldsymbol{R}$ exhibits fewer outliers with a substantially compressed value range compared to $\hat{\boldsymbol{W}}$, as shown in Fig. 4(d)(e). In practice, we can further reduce quantization errors by iteratively updating the low-rank branch through decomposing $\boldsymbol{W} - Q(\boldsymbol{R})$ and adjusting $\boldsymbol{R}$ accordingly for several iterations, and then picking the result with the smallest error.

4.3 LoRunner: Fusing Low-Rank and Low-Bit Branch Kernels

Although the low-rank branch introduces theoretically negligible computation, running it as a separate branch would incur significant latency overhead—approximately 50% of the 4-bit branch latency, as shown in Fig. 6(a). This is because, for a small rank r, even though the computational cost is greatly reduced, the data sizes of input and output activations remain unchanged, shifting the bottleneck from computation to memory access. The situation deteriorates, especially when the activation cannot fit into the GPU L2 cache. For example, in the diffusion transformer block, the up projection in the low-rank branch for QKV projection is much slower since its output exceeds the available L2 cache and results in the extra load and store operations to DRAM. Fortunately, we observe that the down projection $\boldsymbol{L}_1$ in the low-rank branch shares the same input as the quantization kernel in the low-bit branch, while the up projection $\boldsymbol{L}_2$ shares the same output as the 4-bit computation kernel, as illustrated in Fig. 6(b). By fusing the down projection with the quantization kernel and the up projection with the 4-bit computation kernel, the low-rank branch can share the activations with the low-bit branch, eliminating the extra memory access and also halving the number of kernel calls. As a result, the low-rank branch adds only 5∼10% latency, making it nearly cost-free.

5 Experiments

5.1 Setups

Models. We benchmark our methods using the following text-to-image models, including both the UNet (Ronneberger et al., 2015; Ho et al., 2020) and DiT (Peebles & Xie, 2023) backbones:

[†]`https://en.wikipedia.org/wiki/Low-rank_approximation`

Table 1: Quantitative quality comparisons across different models. IR means ImageReward. Our 8-bit results closely match the quality of the 16-bit models. Moreover, our 4-bit results outperform other 4-bit baselines, effectively preserving the visual quality of 16-bit models.

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	**20.0**	0.924	0.259	20.0	**24.6**	0.992	0.275	**18.8**
		FP W4A4	Ours	20.9	**0.932**	**0.245**	**20.2**	25.6	**0.998**	**0.269**	18.7
	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	**18.1**	**0.965**	0.292	17.5	**19.8**	**0.986**	0.298	16.4
		FP W4A4	Ours	20.1	0.957	0.281	17.4	21.7	0.971	0.280	16.6
	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	20.1	0.898	0.394	16.2	25.1	0.922	0.434	14.9
		FP W4A4	Ours	**18.3**	**0.946**	**0.326**	**17.4**	**23.7**	**0.978**	**0.357**	**16.1**
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	24.2	0.796	0.279	17.7	25.7	0.657	0.289	17.6
		FP W4A4	Ours	**24.1**	**0.822**	**0.250**	**18.5**	**24.7**	**0.699**	**0.261**	**18.4**
	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	21.4	0.591	0.306	20.4	26.8	0.470	0.320	20.3
		FP W4A4	Ours	**19.0**	**0.607**	**0.294**	**21.0**	**25.4**	**0.480**	**0.312**	**20.7**

- FLUX.1 (Black-Forest-Labs, 2024) is the SoTA open-sourced DiT-based diffusion model. It consists of 19 joint attention blocks (Esser et al., 2024) and 38 parallel attention blocks (Dehghani et al., 2023), totaling 12B parameters. We evaluate on both the 50-step guidance-distilled (FLUX.1-dev) and 4-step timestep-distilled (FLUX.1-schnell) variants.
- PixArt-Σ (Chen et al., 2024a) is another DiT-based model. Instead of using joint attention, it stacks 28 attention blocks composed of self-attention, cross-attention, and feed-forward layers, amounting to 600M parameters. We evaluate it on the default 20-step setting.
- Stable Diffusion XL (SDXL) is a widely-used UNet-based model with 2.6B parameters (Podell et al., 2024). It predicts noise with three resolution scales. The highest-resolution stage is processed entirely by ResBlocks (He et al., 2016), while the other two stages jointly use ResBlocks and attention layers. Like PixArt-Σ, SDXL employs cross-attention layers for text conditioning. We evaluate it in the 30-step setting, along with its 4-step distilled variant, SDXL-Turbo (Sauer et al., 2023).

Datasets. Following previous works (Li et al., 2023a; Zhao et al., 2024c;b), we randomly sample the prompts in COCO Captions 2024 (Chen et al., 2015) for calibration. To assess the generalization capability of our method, we adopt two distinct prompt sets with varying styles for benchmarking:

- MJHQ-30K (Li et al., 2024a) consists of 30K samples from Midjourney with 10 common categories, 3K samples each. We uniformly select 5K prompts from this dataset to evaluate model performance on artistic image generation.
- Densely Captioned Images (DCI) (Urbanek et al., 2024) is a dataset containing ∼8K images with detailed human-annotated captions, averaging over 1,000 words. For our experiments, we use the summarized version (sDCI), where captions are condensed to 77 tokens using large language models (LLMs) to accommodate diffusion models. Similarly, we randomly sample 5K prompts for efficient evaluation of realistic image generation.

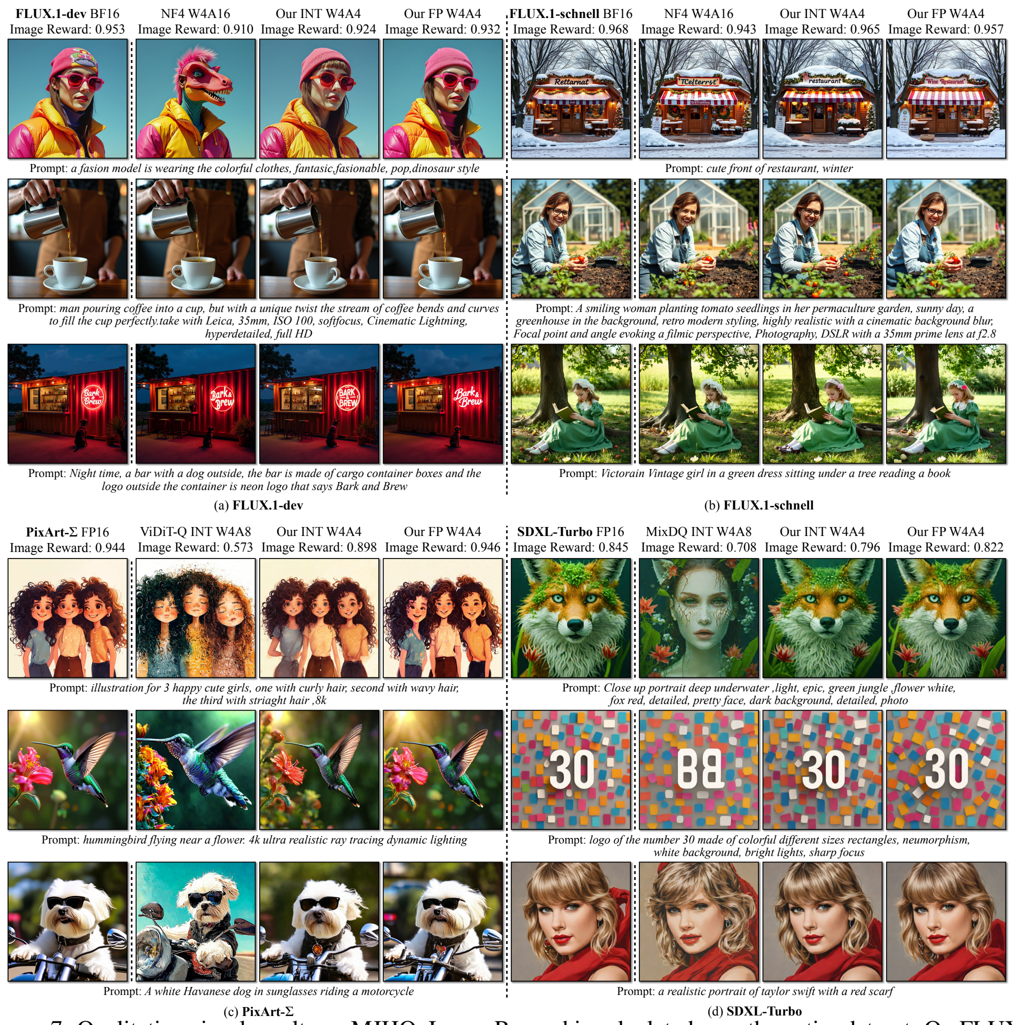

FLUX.1-dev BF16
Image Reward: 0.953
NF4 W4A16
Image Reward: 0.910
Our INT W4A4
Image Reward: 0.924
Our FP W4A4
Image Reward: 0.932
FLUX.1-schnell BF16
Image Reward: 0.968
NF4 W4A16
Image Reward: 0.943
Our INT W4A4
Image Reward: 0.965
Our FP W4A4
Image Reward: 0.957
Prompt: a fasion model is wearing the colorful clothes, fantasic,fasionable, pop,dinosaur style
Prompt: cute front of restaurant, winter
Prompt: man pouring coffee into a cup, but with a unique twist the stream of coffee bends and curves to fill the cup perfectly.take with Leica, 35mm, ISO 100, softfocus, Cinematic Lightning, hyperdetailed, full HD
Prompt: A smiling woman planting tomato seedlings in her permaculture garden, sunny day, a greenhouse in the background, retro modern styling, highly realistic with a cinematic background blur, Focal point and angle evoking a filmic perspective, Photography, DSLR with a 35mm prime lens at f2.8
Prompt: Night time, a bar with a dog outside, the bar is made of cargo container boxes and the logo outside the container is neon logo that says Bark and Brew
Prompt: Victorain Vintage girl in a green dress sitting under a tree reading a book
(a) FLUX.1-dev
(b) FLUX.1-schnell
PixArt-Σ FP16
Image Reward: 0.944
ViDiT-Q INT W4A8
Image Reward: 0.573
Our INT W4A4
Image Reward: 0.898
Our FP W4A4
Image Reward: 0.946
SDXL-Turbo FP16
Image Reward: 0.845
MixDQ INT W4A8
Image Reward: 0.708
Our INT W4A4
Image Reward: 0.796
Our FP W4A4
Image Reward: 0.822
Prompt: illustration for 3 happy cute girls, one with curly hair, second with wavy hair, the third with striaght hair ,8k
Prompt: Close up portrait deep underwater ,light, epic, green jungle ,flower white, fox red, detailed, pretty face, dark background, detailed, photo
Prompt: hummingbird flying near a flower. 4k ultra realistic ray tracing dynamic lighting
Prompt: logo of the number 30 made of colorful different sizes rectangles, neumorphism, white background, bright lights, sharp focus
Prompt: A white Havanese dog in sunglasses riding a motorcycle
Prompt: a realistic portrait of taylor swift with a red scarf
(c) PixArt-Σ
(d) SDXL-Turbo

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. On PixArt-Σ and SDXL-Turbo, our 4-bit results demonstrate better visual quality than ViDiT-Q's and MixDQ's W4A8 results.

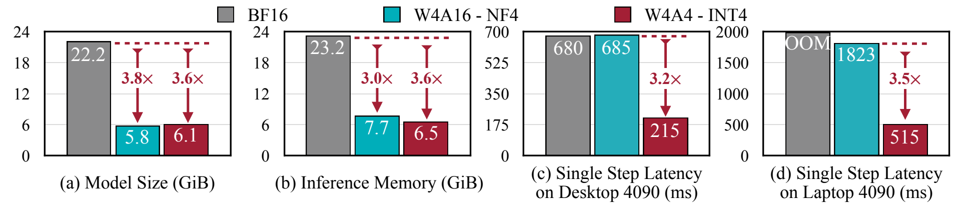


Figure 8: SVDQuant reduces the model size of the 12B FLUX.1 by 3.6×. Additionally, our engine, LoRunner, further reduces memory usage by another 3.6×, resulting in speedups of 3.2× and 3.5× on desktop and laptop version NVIDIA RTX 4090 GPUs, respectively.

Baselines. We compare SVDQuant against the following post-training quantization (PTQ) methods:

- 4-bit NormalFloat (NF4) is a data type for weight-only quantization (Dettmers et al., 2023). It assumes that weights follow a normal distribution and is the information-theoretically optimal 4-bit representation. We use the community-quantized NF4 FLUX.1 models (Lllyasviel) as the baselines.
- ViDiT-Q (Zhao et al., 2024b) uses per-token quantization and smoothing (Xiao et al., 2023) to alleviate the outliers across different batches and tokens and achieves lossless 8-bit quantization on PixArt-Σ.
- MixDQ (Zhao et al., 2024c) identifies the outliers in the begin-of-sentence token of text embedding and protects them with 16-bit pre-computation. This method enables up to W4A8 quantization with negligible performance degradation on SDXL-Turbo.

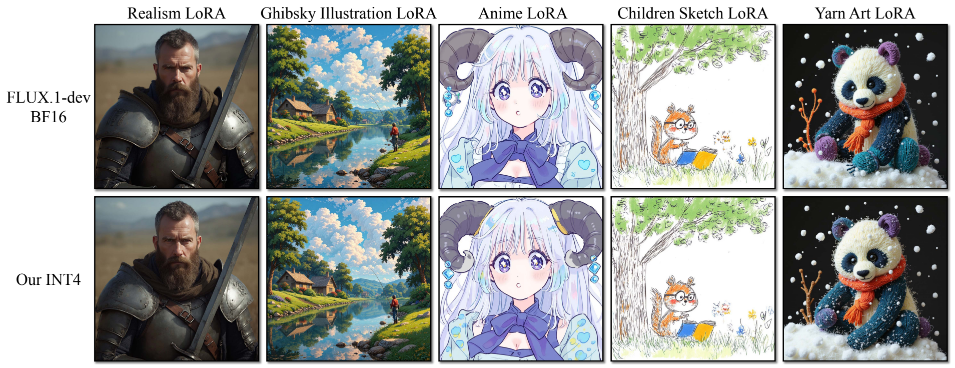


Figure 9: Our INT4 model seamlessly integrates with off-the-shelf LoRAs without requiring requantization. When applying LoRAs, it matches the image quality of the original 16-bit FLUX.1-dev. See App. F for the text prompts.

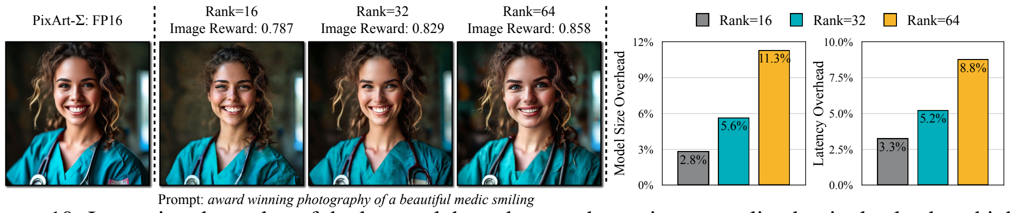


Figure 10: Increasing the rank r of the low-rank branch can enhance image quality, but it also leads to higher parameter and latency overhead.

- TensorRT contains an industry-level PTQ toolkit to quantize the diffusion models to 8 bits. It uses smoothing and only calibrates activations over a selected timestep range with a percentile scheme.

Metrics. Following previous work (Li et al., 2022; 2024b), we mainly benchmark image quality and similarity to the results produced by the original 16-bit models. For the image quality assessment, we use Fréchet Inception Distance (FID, lower is better) to measure the distribution distance between the generated images and the ground-truth images (Heusel et al., 2017; Parmar et al., 2022). Besides, we employ Image Reward (higher is better) to approximate the human rating of the generated images (Xu et al., 2024a). We use LPIPS (lower is better) to measure the perceptual similarity (Zhang et al., 2018) and Peak Signal Noise Ratio (PSNR, higher is better) to measure the numerical similarity of the images from the 16-bit models. Please refer to our App. D.1 for more metrics (CLIP IQA (Wang et al., 2023b), CLIP Score (Hessel et al., 2021) and SSIM[‡]).

Implementation details. Please refer to App. B.

5.2 Results

Quality results. We report the quantitative quality results in Tab. 1 across various models and precision levels, and show some corresponding 4-bit qualitative comparisons in Fig. 7. Among all models, our 8-bit results can perfectly mirror the 16-bit results, achieving PSNR higher than 21, beating all other 8-bit baselines. On FLUX.1-dev, our INT8 PSNR even reaches 27 on MJHQ.

For 4-bit quantization, on FLUX.1, our SVDQuant surpasses the NF4 W4A16 baseline regarding Image Reward. On the schnell variant, our Image Reward even exceeds that of the original BF16 model, suggesting stronger human preference. On PixArt-Σ, while our INT4 Image Reward shows slight degradation, our FP4 model achieves an even higher score than the FP16 model. This is likely due to PixArt-Σ's small model size (600M parameters), which is already highly compact and benefits from a smaller group size. Remarkably, both our INT4 and FP4 results consistently outperform ViDiT-Q's[§] W4A8 results by a large margin across all metrics. For UNet-based models, on SDXL-Turbo, our 4-bit models significantly outperform MixDQ's W4A8 results, and our FID scores are on par with the FP16 models, indicating no loss in performance. On SDXL, both our INT4 and FP4 results achieve comparable quality to TensorRT's W8A8 performance, which represents the 8-bit SoTA. As shown in Fig. 15 in the Appendix, our visual quality only shows minor degradation.

[‡]`https://en.wikipedia.org/wiki/Structural_similarity_index_measure`

[§]Our FP16 PixArt-Σ model is slightly different from ViDiT's, though both offer the same quality. For fair comparisons, ViDiT-Q's similarity results are calculated using their FP16 results.

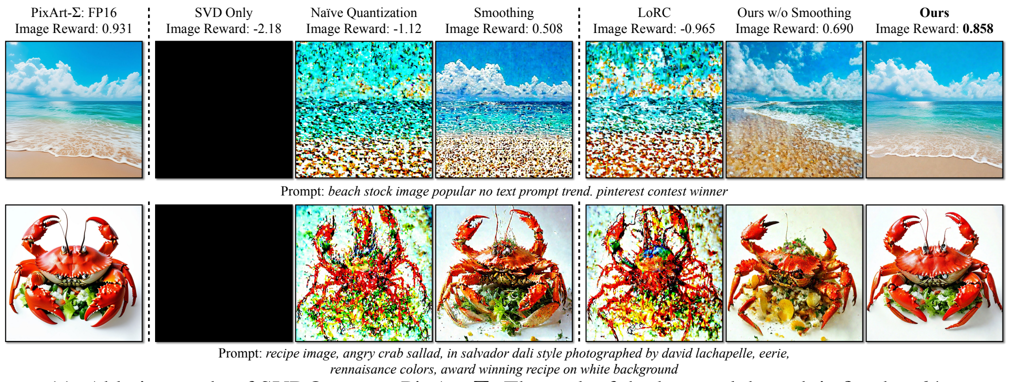

PixArt-Σ: FP16
Image Reward: 0.931
SVD Only
Image Reward: -2.18
Naïve Quantization
Image Reward: -1.12
Smoothing
Image Reward: 0.508
LoRC
Image Reward: -0.965
Ours w/o Smoothing
Image Reward: 0.690
Ours
Image Reward: 0.858
Prompt: beach stock image popular no text prompt trend. pinterest contest winner
Prompt: recipe image, angry crab sallad, in salvador dali style photographed by david lachapelle, eerie, rennaisance colors, award winning recipe on white background

Figure 11: Ablation study of SVDQuant on PixArt-Σ. The rank of the low-rank branch is fixed at 64 across all experiments. ImageReward is measured over the 1K samples from MJHQ. Our results significantly outperform the other, achieving the highest image quality by a wide margin.

Memory save & speedup. In Fig. 8, we report measured model size, memory savings and speedup for FLUX.1. Our INT4 quantization reduces the original transformer size from 22.2 GiB to 6.1 GiB, including a 0.3 GiB overhead due to the low-rank branch, resulting in an overall 3.6× reduction. Since both weights and activations are quantized, compared to NF4 weight-only-quantized variant, our inference engine LoRunner even saves 1.2× memory footprint, and attains a speedup of 3.2× on desktop-level and 3.5× on laptop-level NVIDIA RTX 4090 systems, respectively. We anticipate even greater speedups for FP4-quantized models on NVIDIA's next-generation Blackwell GPUs, as they inherently support microscaling for group quantization without the need for specialized GPU kernels.

Integrate with LoRA. Previous quantization methods require fusing the LoRA branches and re-quantizing the model when integrating LoRAs. In contrast, our LoRunner eliminates redundant memory access, allowing adding a separate LoRA branch. In practice, we can fuse the LoRA branch into our low-rank branch by slightly increasing the rank, further enhancing efficiency. In Fig. 9, we exhibit some visual examples of applying LoRAs of five different styles (Realism, Ghibsky Illustration, Anime, Children Sketch, and Yarn Art) to our INT4 FLUX.1-dev model. Our INT4 model successfully adapts to each style while preserving the image quality of the 16-bit version. For more visual examples, see App. D.2.

Ablation study. In Fig. 11, we present several ablation studies of SVDQuant on PixArt-Σ. First, both SVD-only and naïve quantization perform poorly in the 4-bit setting, resulting in a severe degradation of image quality. While applying smoothing to the quantization slightly improves image quality compared to naïve quantization, the overall results remain unsatisfactory. LoRC (Yao et al., 2023) introduces a low-rank branch to compensate for quantization errors, but this approach is suboptimal. Quantization errors exhibit a smooth singular value distribution. Consequently, low-rank compensation fails to effectively mitigate these errors, as discussed in Sec. 4.2. In contrast, we first decompose the weights and quantize only the residual. As demonstrated in Fig. 5, the first several singular values are significantly larger than the rest, allowing us to shift them to the low-rank branch to effectively reduce weight magnitude. Finally, incorporating smoothing further enables the low-rank branch to absorb outliers from the activations, substantially improving in image quality.

Trade-off of increasing rank. Fig. 10 presents the results of different rank r in SVDQuant on PixArt-Σ. Increasing the rank from 16 to 64 significantly enhances image quality but increases parameter and latency overhead. In our experiments, we select a rank of 32, which offers a decent quality with minor overhead.

6 Conclusion

In this work, we introduce a novel 4-bit post-training quantization paradigm SVDQuant for diffusion models. It adopts a low-rank branch to absorb the outliers in both the weights and activations, easing the process of quantization. Our inference engine LoRunner further fuses the low-rank and low-bit branch kernels, reducing memory usage and eliminating redundant data movement overhead. Extensive experiments demonstrate that SVDQuant preserves image quality. Our LoRunner further achieves a 3.5× reduction in memory usage and latency on an NVIDIA RTX-4090 laptop. This advancement enables the efficient deployment of large-scale diffusion models on edge devices, unlocking broader potential for interactive AI applications.

REFERENCES

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024. 4, 25

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022. 3

Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *CVPR*, 2023. 3

Black-Forest-Labs. Flux.1, 2024. URL `https://blackforestlabs.ai/`. 2, 7

Han Cai, Muyang Li, Qinsheng Zhang, Ming-Yu Liu, and Song Han. Condition-aware neural network for controlled image generation. In *CVPR*, 2024. 3

Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-sigma: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. *arXiv preprint arXiv:2403.04692*, 2024a. 7

Lei Chen, Yuan Meng, Chen Tang, Xinzhu Ma, Jingyan Jiang, Xin Wang, Zhi Wang, and Wenwu Zhu. Q-dit: Accurate post-training quantization for diffusion transformers. *CoRR*, 2024b. 19

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 7

Zigeng Chen, Xinyin Ma, Gongfan Fang, Zhenxiong Tan, and Xinchao Wang. Asyncdiff: Parallelizing diffusion models by asynchronous denoising. *arXiv preprint arXiv:2406.06911*, 2024c. 3

Huanpeng Chu, Wei Wu, Chengjie Zang, and Kun Yuan. Qncd: Quantization noise correction for diffusion models. *arXiv preprint arXiv:2403.19140*, 2024. 19

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *ICML*. PMLR, 2023. 7

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *NeurIPS*, 2022. 2, 3, 25

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 3, 8, 19

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. 3, 7

fal.ai. Auraflow v0.1, 2024. URL `https://blog.fal.ai/auraflow/`. 2

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training compression for generative pretrained transformers. *ICLR*, 2023. 3

Han Guo, Philip Greengard, Eric Xing, and Yoon Kim. Lq-lora: Low-rank plus quantized matrix decomposition for efficient language model finetuning. *ICLR*, 2024. 3

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 7

Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptqd: Accurate post-training quantization for diffusion models. *NeurIPS*, 2023. 3

Yefei He, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Efficientdm: Efficient quantization-aware fine-tuning of low-bit diffusion models. In *ICLR*, 2024. 3, 4, 19

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021. 9, 20

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*, 2017. 9

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 2, 3, 6

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. In *The Tenth International Conference on Learning Representations*, 2022. 2, 3

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*, 2022. 3

Yushi Huang, Ruihao Gong, Jing Liu, Tianlong Chen, and Xianglong Liu. Tfmq-dm: Temporal feature maintenance quantization for diffusion models. In *CVPR*, 2024. 3

Ajay Jaiswal, Lu Yin, Zhenyu Zhang, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. From galore to welore: How low-rank weights non-uniformly emerge from low-rank gradients. *arXiv preprint arXiv: 2407.11239*, 2024. 3

Minguk Kang, Richard Zhang, Connelly Barnes, Sylvain Paris, Suha Kwak, Jaesik Park, Eli Shechtman, Jun-Yan Zhu, and Taesung Park. Distilling diffusion models into conditional gans. *arXiv preprint arXiv:2405.05967*, 2024. 3

Sehoon Kim, Coleman Richard Charles Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *ICML*, 2024. 3

Daiqing Li, Aleks Kamko, Ehsan Akhgari, Ali Sabet, Linmiao Xu, and Suhail Doshi. Playground v2.5: Three insights towards enhancing aesthetic quality in text-to-image generation, 2024a. 7

Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In *CVPR*, 2020. 3

Muyang Li, Ji Lin, Chenlin Meng, Stefano Ermon, Song Han, and Jun-Yan Zhu. Efficient spatially sparse inference for conditional gans and diffusion models. In *NeurIPS*, 2022. 3, 9

Muyang Li, Tianle Cai, Jiaxin Cao, Qinsheng Zhang, Han Cai, Junjie Bai, Yangqing Jia, Ming-Yu Liu, Kai Li, and Song Han. Distrifusion: Distributed parallel inference for high-resolution diffusion models. In *CVPR*, 2024b. 3, 9

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *ICCV*, 2023a. 3, 7, 19, 25, 26

Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds. *NeurIPS*, 2023b. 3

Yixiao Li, Yifan Yu, Qingru Zhang, Chen Liang, Pengcheng He, Weizhu Chen, and Tuo Zhao. LoSparse: Structured compression of large language models based on low-rank and sparse approximation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 20336–20350. PMLR, 2023c. 3

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024c. 3

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. In *MLSys*, 2024a. 2, 3, 5, 25

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv preprint arXiv:2405.04532*, 2024b. 3, 4, 25

Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. Qllm: Accurate and efficient low-bitwidth quantization for large language models. In *ICLR*, 2024a. 25

Songhua Liu, Weihao Yu, Zhenxiong Tan, and Xinchao Wang. Linfusion: 1 gpu, 1 minute, 16k image. *arXiv preprint arXiv:2409.02097*, 2024b. 3

Wenxuan Liu and Saiqian Zhang. Hq-dit: Efficient diffusion transformer with fp4 hybrid quantization. *arXiv preprint arXiv:2405.19751*, 2024. 3

Xuewen Liu, Zhikai Li, Junrui Xiao, and Qingyi Gu. Enhanced distribution alignment for post-training quantization of diffusion models. *arXiv preprint arXiv:2401.04585*, 2024c. 3

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant–llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024d. 4

Lllyasviel. [major update] bitsandbytes guidelines and flux · lllyasviel stable-diffusion-webui-forge · discussion #981. URL `https://github.com/lllyasviel/stable-diffusion-webui-forge/discussions/981`. 8

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *NeurIPS*, 2022. 3

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv: 2310.04378*, 2023. 3

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024a. 25

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. *arXiv preprint arXiv:2406.01733*, 2024b. 3

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *CVPR*, 2024c. 3

Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. Affinequant: Affine transformation quantization for large language models. In *ICLR*, 2024d. 25

Pascal Massart. *Concentration inequalities and model selection: Ecole d'Eté de Probabilités de Saint-Flour XXXIII-2003*. Springer, 2007. 17

Chenlin Meng, Ruiqi Gao, Diederik P Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. *arXiv preprint arXiv:2210.03142*, 2022a. 3

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022b. 2

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*, 2024. 3

NVIDIA. Nvidia blackwell architecture technical brief, 2024. URL `https://resources.nvidia.com/en-us-blackwell-architecture`. 2

Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022. 9

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 2, 3, 4, 6

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. In *ICLR*, 2024. 2, 3, 7

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 20

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2, 3, 26

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015. 2, 3, 6

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023. 18

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, 2023. 2

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2021. 3

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023. 3, 7

Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *CVPR*, 2023. 3

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 3

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, 2023. 3

Yang Sui, Yanyu Li, Anil Kag, Yerlan Idelbayev, Junli Cao, Ju Hu, Dhritiman Sagar, Bo Yuan, Sergey Tulyakov, and Jian Ren. Bitsfusion: 1.99 bits weight quantization of diffusion model. *arXiv preprint arXiv:2406.04333*, 2024. 3

Siao Tang, Xin Wang, Hong Chen, Chaoyu Guan, Zewen Wu, Yansong Tang, and Wenwu Zhu. Post-training quantization with progressive calibration and activation relaxing for text-to-image diffusion models. *arXiv preprint arXiv:2311.06322*, 2023. 3

Jack Urbanek, Florian Bordes, Pietro Astolfi, Mary Williamson, Vasu Sharma, and Adriana Romero-Soriano. A picture is worth more than 77 text tokens: Evaluating clip-style models on dense captions. In *CVPR*, 2024. 7

Changyuan Wang, Ziwei Wang, Xiuwei Xu, Yansong Tang, Jie Zhou, and Jiwen Lu. Towards accurate post-training quantization for diffusion models. In *CVPR*, 2024a. 3

Haoxuan Wang, Yuzhang Shang, Zhihang Yuan, Junyi Wu, and Yan Yan. Quest: Low-bit diffusion model quantization via efficient selective finetuning. *arXiv preprint arXiv:2402.03666*, 2024b. 3, 26

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*, 2023a. 25

Jiannan Wang, Jiarui Fang, Aoyu Li, and PengCheng Yang. Pipefusion: Displaced patch pipeline parallelism for inference of diffusion transformer models. *arXiv preprint arXiv:2405.14430*, 2024c. 3

Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *AAAI*, 2023b. 9, 20

Junyi Wu, Haoxuan Wang, Yuzhang Shang, Mubarak Shah, and Yan Yan. Ptq4dit: Post-training quantization for diffusion transformers. *arXiv preprint arXiv:2405.16005*, 2024. 3

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *ICML*, 2023. 2, 3, 5, 8, 18, 19, 25

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *NeurIPS*, 2024a. 1, 9

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024b. 3

Yuewei Yang, Xiaoliang Dai, Jialiang Wang, Peizhao Zhang, and Hongbo Zhang. Efficient quantization strategies for latent diffusion models. *arXiv preprint arXiv:2312.05431*, 2023. 3

Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, and Yuxiong He. Zeroquant-v2: Exploring post-training quantization in llms from comprehensive study to low rank compensation. *arXiv preprint arXiv:2303.08302*, 2023. 3, 10, 19

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *arXiv preprint arXiv:2405.14867*, 2024a. 3

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024b. 3

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv: 2312.05821*, 2023. 3

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023. 2

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *ICLR*, 2022. 3

Qinsheng Zhang, Molei Tao, and Yongxin Chen. gddim: Generalized denoising diffusion implicit models. In *ICLR*, 2022. 3

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 9

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 61121–61143. PMLR, 2024a. 3

Tianchen Zhao, Tongcheng Fang, Enshu Liu, Wan Rui, Widyadewi Soedarmadji, Shiyao Li, Zinan Lin, Guohao Dai, Shengen Yan, Huazhong Yang, et al. Vidit-q: Efficient and accurate quantization of diffusion transformers for image and video generation. *arXiv preprint arXiv:2406.02540*, 2024b. 3, 7, 8, 19

Tianchen Zhao, Xuefei Ning, Tongcheng Fang, Enshu Liu, Guyue Huang, Zinan Lin, Shengen Yan, Guohao Dai, and Yu Wang. Mixdq: Memory-efficient few-step text-to-image diffusion models with metric-decoupled mixed precision quantization. *arXiv preprint arXiv:2405.17873*, 2024c. 3, 7, 8, 19

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *MLSys*, 2024d. 3, 25

Xingyu Zheng, Haotong Qin, Xudong Ma, Mingyuan Zhang, Haojie Hao, Jiakai Wang, Zixiang Zhao, Jinyang Guo, and Xianglong Liu. Binarydm: Towards accurate binarization of diffusion model. *arXiv preprint arXiv:2404.05662*, 2024. 3

A Missing Proofs

A.1 Proof of Proposition 4.1

Proof.

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

□

A.2 Proof of Proposition 4.2

Proof.

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

So,

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

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

$$\mathbb{E}\left[\max(|\boldsymbol{R}|)\right] \leq \sigma\sqrt{2\log\left(\text{size}(\boldsymbol{R})\right)} \tag{9}$$

where σ is the std deviation of the normal distribution. Eq. 9 comes from the maximal inequality of Gaussian variables (Lemma 2.3 in Massart (2007)).

On the other hand,

$$\begin{aligned}
&\mathbb{E}\left[\|\boldsymbol{R}\|_F\right] \\
=&\mathbb{E}\left[\sqrt{\sum_{x\in\boldsymbol{R}} x^2}\right] \\
\geq&\mathbb{E}\left[\frac{\sum_{x\in\boldsymbol{R}}|x|}{\sqrt{\text{size}(\boldsymbol{R})}}\right] && (10)\\
=&\sigma\sqrt{\frac{2\text{size}(\boldsymbol{R})}{\pi}}, && (11)
\end{aligned}$$

where Eq. 10 comes from Cauchy-Schwartz inequality and Eq. 11 comes from the expectation of half-normal distribution.

Together, we have that for normal distribution,

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

In other words, Eq. 7 holds for $c = \sqrt{\frac{\log(\text{size}(\boldsymbol{R}))\pi}{\text{size}(\boldsymbol{R})}}$. □

B Implementation Details

For the 8-bit setting, we use per-token dynamic activation quantization and per-channel weight quantization with a low-rank branch of rank 16. For the 4-bit setting, we adopt per-group symmetric quantization for both activations and weights, along with a low-rank branch of rank 32. INT4 quantization uses a group size of 64 with 16-bit scales. FP4 quantization uses a group size of 32 with FP8 scales (Rouhani et al., 2023). For FLUX.1 models, the inputs of linear layers in adaptive normalization are kept in 16 bits (*i.e.*, W4A16). For other models, key and value projections in the cross-attention are retained at 16 bits since their latency only covers less than 5% of total runtime.

The smoothing factor $\lambda \in \mathbb{R}^m$ is a per-channel vector whose i-th element is computed as $\lambda_i = \max(|\boldsymbol{X}_{:,i}|)^\alpha / \max(|\boldsymbol{W}_{i,:}|)^{1-\alpha}$ following SmoothQuant(Xiao et al., 2023) Here, $\boldsymbol{X} \in \mathbb{R}^{b\times m}$ and $\boldsymbol{W} \in \mathbb{R}^{m\times n}$. It is decided offline by searching for the best migration strength α for each layer to minimize the layer output mean squared error (MSE) after SVD on the calibration dataset.

C Detailed Discussion with Related Work

We compare the similarities and differences between our SVDQuant and some related works in Tab. 2:

Table 2: Similarity and difference comparisons between SVDQuant and related works.

Method	**Similarity**	**Difference**
QLoRA (Dettmers et al., 2023)	Both have low-rank branches and a quantized-based model.	QLoRA focuses on PeFE on LLM and only quantizes weights, while SVDQuant is a PTQ method that quantizes both weights and activations for diffusion models.
LoRC (Yao et al., 2023)	Both are PTQ methods with low-rank branches.	LoRC focuses on LLM quantization and uses low-rank decomposition to compensate for quantization errors. In contrast, we focus on diffusion model quantization by first decomposing weights and then quantizing the residuals. Additionally, we ease activation quantization and achieve significantly better results than LoRC.
QNCD (Chu et al., 2024)	Both are PTQ methods for diffusion models.	The methods differ entirely. QNCD is only applied to U-Net backbones, while our approach supports both U-Net and DiT. Additionally, we push the boundary quantization from W4A8 to W4A4 and demonstrate speedups on GPUs.
Q-DiT (Chen et al., 2024b)	Both are PTQ methods for diffusion models.	The methods differ entirely. Besides, Q-DiT is only applied to class-conditioned models, while we can work on large text-to-image models. We also push the quantization boundary from their W4A8 to W4A4 and demonstrate speedups on GPUs.
Q-Diffusion (Li et al., 2023a)	Both are PTQ methods for diffusion models.	The methods differ entirely. Besides, their work is only applied to U-Net models, while we can work on both the U-Net and DiT backbones. We also push the quantization boundary from W4A8 to W4A4 and demonstrate measured speedups on GPUs.
MixDQ (Zhao et al., 2024c)	Both are PTQ methods for diffusion models.	The methods differ entirely. Besides, MixDQ is only applied U-Net models, while we can work on both the U-Net and DiT backbones. We also push the quantization boundary from W4A8 to W4A4.
ViDiT-Q (Zhao et al., 2024b)	Both are PTQ methods for diffusion models.	The methods are completely different. We also push the quantization boundary from their W4A8 to W4A4.
EfficientDM (He et al., 2024)	Both are diffusion quantization methods with low-rank branches.	They use low-rank branches to reduce the cost of quantization-aware training, requiring fusion after tuning. In contrast, our method doesn't need training and preserves the low-rank branches during inference. Additionally, EfficientDP is only applied to class-conditioned U-Nets. We support large text-to-image models and demonstrate speedups on GPUs.
SmoothQuant (Xiao et al., 2023)	Both are quantization methods.	SmoothQuant is for LLM quantization, while we focus on quantizing diffusion models. The ideas are also different: Smoothing in our method is a tool to aggregate outliers.

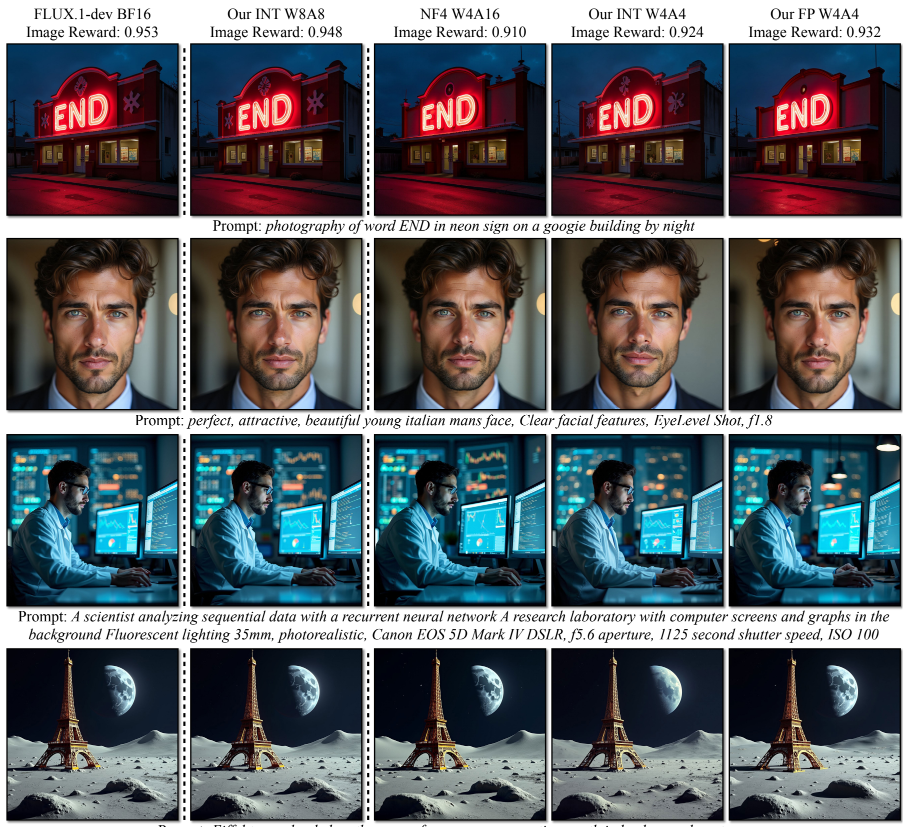
D Additional Results

D.1 Quality Results

We report extra quantitative quality results with additional metrics in Tab. 3. Specifically, CLIP IQA (Wang et al., 2023b) and CLIP Score (Hessel et al., 2021) assesses the image quality and text-image alignment with CLIP (Radford et al., 2021), respectively. Structural Similarity Index Measure (SSIM) is used to measure the luminance, contrast, and structure similarity of images produced by our 4-bit model against the original 16-bit model. We also visualize more qualitative comparsions in Fig. 12, 13, 14, 15 and 16.

Table 3: Additional quantitative quality comparisons across different models. C.IQA means CLIP IQA, and C.SCR means CLIP Score.

Backbone	Model	Precision	Method	MJHQ			sDCI		
				Quality		Similarity	Quality		Similarity
				C.IQA ($\uparrow$)	C.SCR ($\uparrow$)	SSIM($\uparrow$)	C.IQA ($\uparrow$)	C.SCR ($\uparrow$)	SSIM ($\uparrow$)
DiT	FLUX.1 -dev (50 Steps)	BF16	–	0.952	26.0	–	0.955	25.4	–
		INT W8A8	Ours	0.953	26.0	0.748	0.955	25.4	0.697
		W4A16	NF4	0.947	**25.8**	0.748	0.951	**25.4**	0.697
		INT W4A4	Ours	**0.950**	**25.8**	0.780	0.952	25.3	0.720
		FP W4A4	Ours	**0.950**	**25.8**	**0.781**	**0.953**	25.3	**0.726**
	FLUX.1 -schnell (4 Steps)	BF16	–	0.938	26.6	–	0.932	26.2	–
		INT W8A8	Ours	0.938	26.6	0.844	0.932	26.2	0.811
		W4A16	NF4	**0.941**	**26.6**	**0.713**	**0.933**	**26.2**	**0.674**
		INT W4A4	Ours	0.938	26.5	0.691	0.931	**26.2**	0.647
		FP W4A4	Ours	0.938	26.5	0.691	0.931	**26.2**	0.647
	PixArt-Σ (20 Steps)	FP16	–	0.944	26.8	–	0.966	26.1	–
		INT W8A8	ViDiT-Q	**0.948**	26.7	0.815	0.966	**26.1**	0.756
		INT W8A8	Ours	0.947	**26.8**	**0.849**	**0.967**	26.0	**0.800**
		INT W4A8	ViDiT-Q	0.912	25.7	0.356	0.917	25.4	0.295
		INT W4A4	ViDiT-Q	0.185	13.3	0.077	0.176	13.3	0.080
		INT W4A4	Ours	0.927	26.6	0.602	0.952	**26.1**	0.519
		FP W4A4	Ours	**0.935**	**26.7**	**0.652**	**0.957**	**26.1**	**0.574**
UNet	SDXL -Turbo (4 Steps)	FP16	–	0.926	26.5	–	0.913	26.5	–
		INT W8A8	MixDQ	0.922	26.5	0.763	0.907	26.5	0.750
		INT W8A8	Ours	**0.925**	**26.5**	**0.821**	**0.912**	**26.5**	**0.808**
		INT W4A8	MixDQ	0.893	25.9	0.512	0.895	26.1	0.493
		INT W4A4	MixDQ	0.556	13.1	0.289	0.548	11.9	0.296
		INT W4A4	Ours	0.913	**26.4**	0.618	0.888	**26.8**	0.600
		FP W4A4	Ours	**0.919**	**26.4**	**0.640**	**0.901**	26.7	**0.620**
	SDXL (30 Steps)	FP16	–	0.907	27.2	–	0.911	26.5	–
		INT W8A8	TensorRT	0.905	26.7	0.733	0.901	26.1	0.697
		INT W8A8	Ours	**0.912**	**27.0**	**0.843**	**0.910**	**26.3**	**0.814**
		INT W4A4	Ours	0.878	26.6	**0.709**	**0.869**	26.2	0.666
		FP W4A4	Ours	**0.883**	**26.8**	0.707	0.860	**26.4**	**0.661**



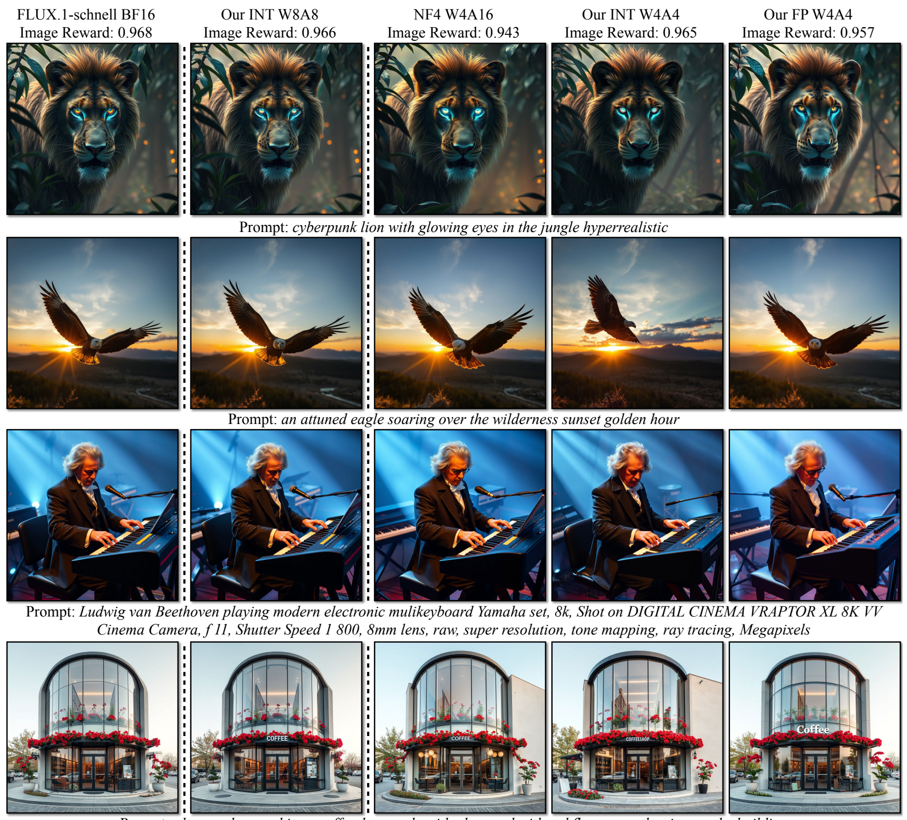
Prompt: *photography of word END in neon sign on a googie building by night*

Prompt: *perfect, attractive, beautiful young italian mans face, Clear facial features, EyeLevel Shot, f1.8*

Prompt: *A scientist analyzing sequential data with a recurrent neural network A research laboratory with computer screens and graphs in the background Fluorescent lighting 35mm, photorealistic, Canon EOS 5D Mark IV DSLR, f5.6 aperture, 1125 second shutter speed, ISO 100*

Prompt: *Eiffel tower, landed on the moon, from moon perspective, earth in background, no town*

Figure 12: Qualitative visual results of FLUX.1-dev on MJHQ.



Prompt: *cyberpunk lion with glowing eyes in the jungle hyperrealistic*

Prompt: *an attuned eagle soaring over the wilderness sunset golden hour*

Prompt: *Ludwig van Beethoven playing modern electronic mulikeyboard Yamaha set, 8k, Shot on DIGITAL CINEMA VRAPTOR XL 8K VV Cinema Camera, f 11, Shutter Speed 1 800, 8mm lens, raw, super resolution, tone mapping, ray tracing, Megapixels*

Prompt: *ultra modern architure coffeeshop made with glass and with red flowers on the rims on the building*

Figure 13: Qualitative visual results of FLUX.1-schnell on MJHQ.

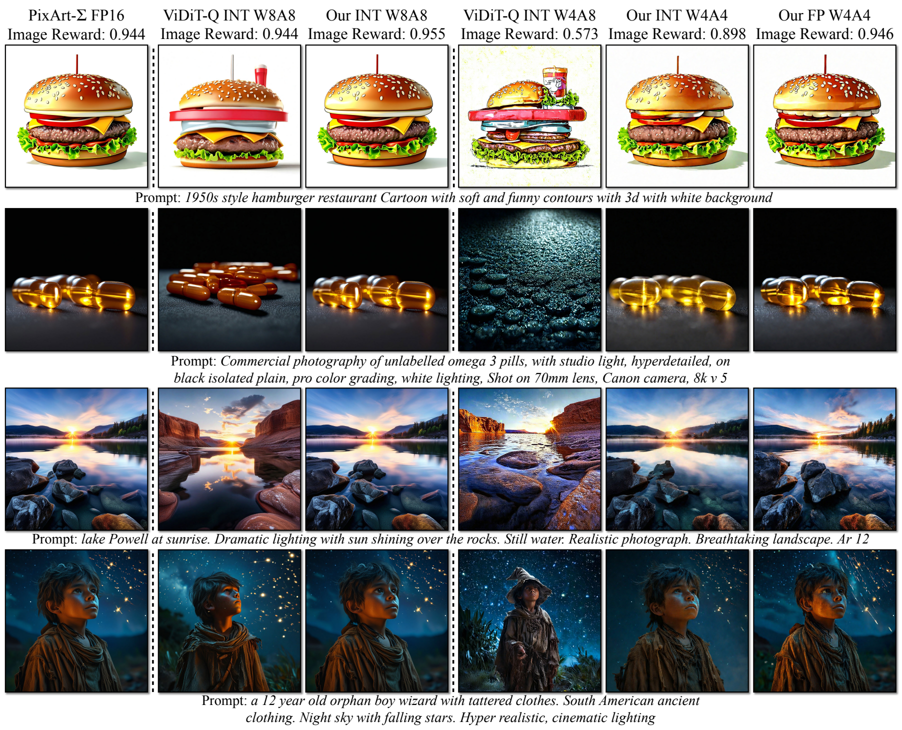

PixArt-Σ FP16
Image Reward: 0.944
ViDiT-Q INT W8A8
Image Reward: 0.944
Our INT W8A8
Image Reward: 0.955
ViDiT-Q INT W4A8
Image Reward: 0.573
Our INT W4A4
Image Reward: 0.898
Our FP W4A4
Image Reward: 0.946
Prompt: 1950s style hamburger restaurant Cartoon with soft and funny contours with 3d with white background
Prompt: Commercial photography of unlabelled omega 3 pills, with studio light, hyperdetailed, on black isolated plain, pro color grading, white lighting, Shot on 70mm lens, Canon camera, 8k v 5
Prompt: lake Powell at sunrise. Dramatic lighting with sun shining over the rocks. Still water. Realistic photograph. Breathtaking landscape. Ar 12
Prompt: a 12 year old orphan boy wizard with tattered clothes. South American ancient clothing. Night sky with falling stars. Hyper realistic, cinematic lighting

Figure 14: Qualitative visual results of PixArt-Σ on MJHQ.

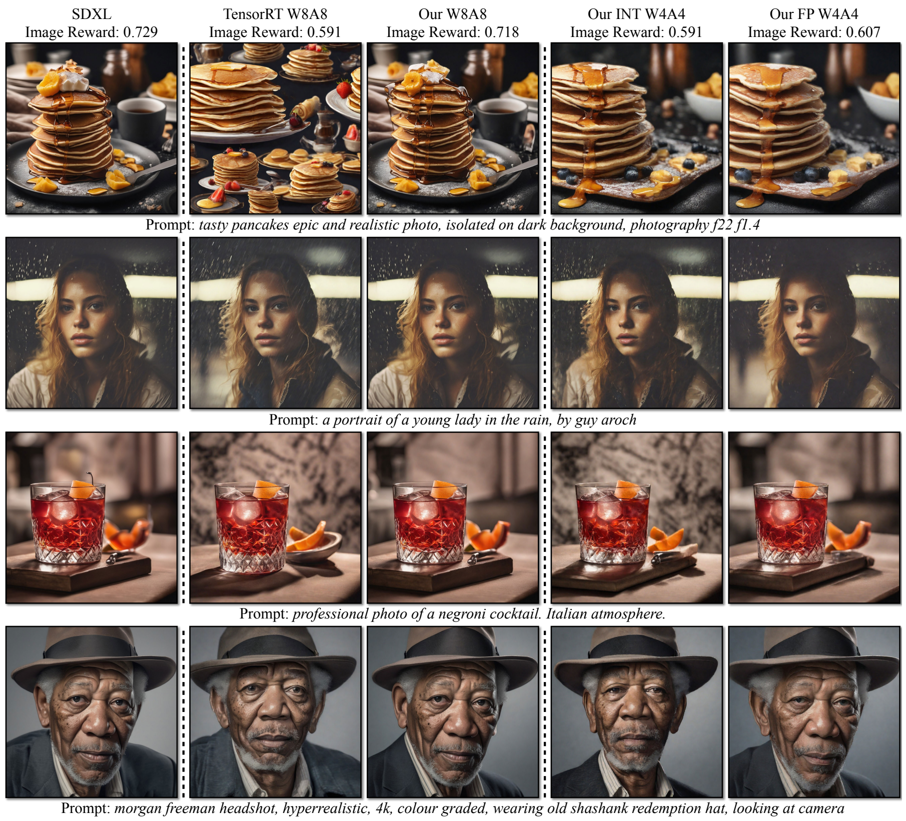

SDXL
Image Reward: 0.729
TensorRT W8A8
Image Reward: 0.591
Our W8A8
Image Reward: 0.718
Our INT W4A4
Image Reward: 0.591
Our FP W4A4
Image Reward: 0.607
Prompt: tasty pancakes epic and realistic photo, isolated on dark background, photography f22 f1.4
Prompt: a portrait of a young lady in the rain, by guy aroch
Prompt: professional photo of a negroni cocktail. Italian atmosphere.
Prompt: morgan freeman headshot, hyperrealistic, 4k, colour graded, wearing old shashank redemption hat, looking at camera

Figure 15: Qualitative visual results of SDXL on MJHQ.

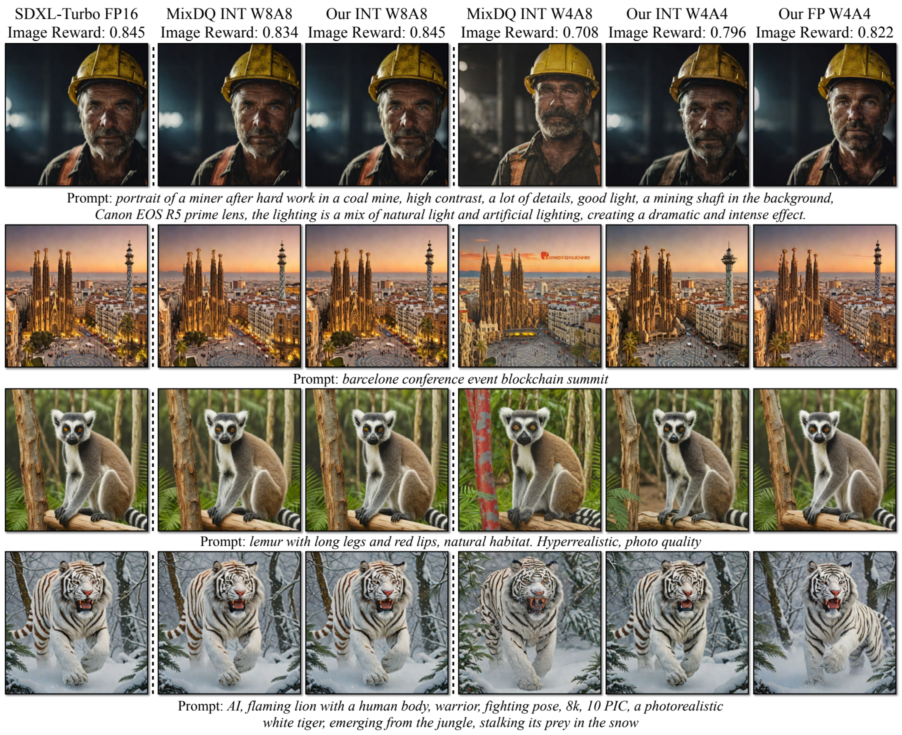

SDXL-Turbo FP16
Image Reward: 0.845
MixDQ INT W8A8
Image Reward: 0.834
Our INT W8A8
Image Reward: 0.845
MixDQ INT W4A8
Image Reward: 0.708
Our INT W4A4
Image Reward: 0.796
Our FP W4A4
Image Reward: 0.822
Prompt: portrait of a miner after hard work in a coal mine, high contrast, a lot of details, good light, a mining shaft in the background, Canon EOS R5 prime lens, the lighting is a mix of natural light and artificial lighting, creating a dramatic and intense effect.
Prompt: barcelone conference event blockchain summit
Prompt: lemur with long legs and red lips, natural habitat. Hyperrealistic, photo quality
Prompt: AI, flaming lion with a human body, warrior, fighting pose, 8k, 10 PIC, a photorealistic white tiger, emerging from the jungle, stalking its prey in the snow

Figure 16: Qualitative visual results of SDXL-Turbo on MJHQ.

D.2 LoRA Results

In Fig. 17, we showcase more visual results of applying the aforementioned five community-contributed LoRAs of different styles (Realism, Ghibsky Illustration, Anime, Children Sketch, and Yarn Art) to our INT4 quantized models.

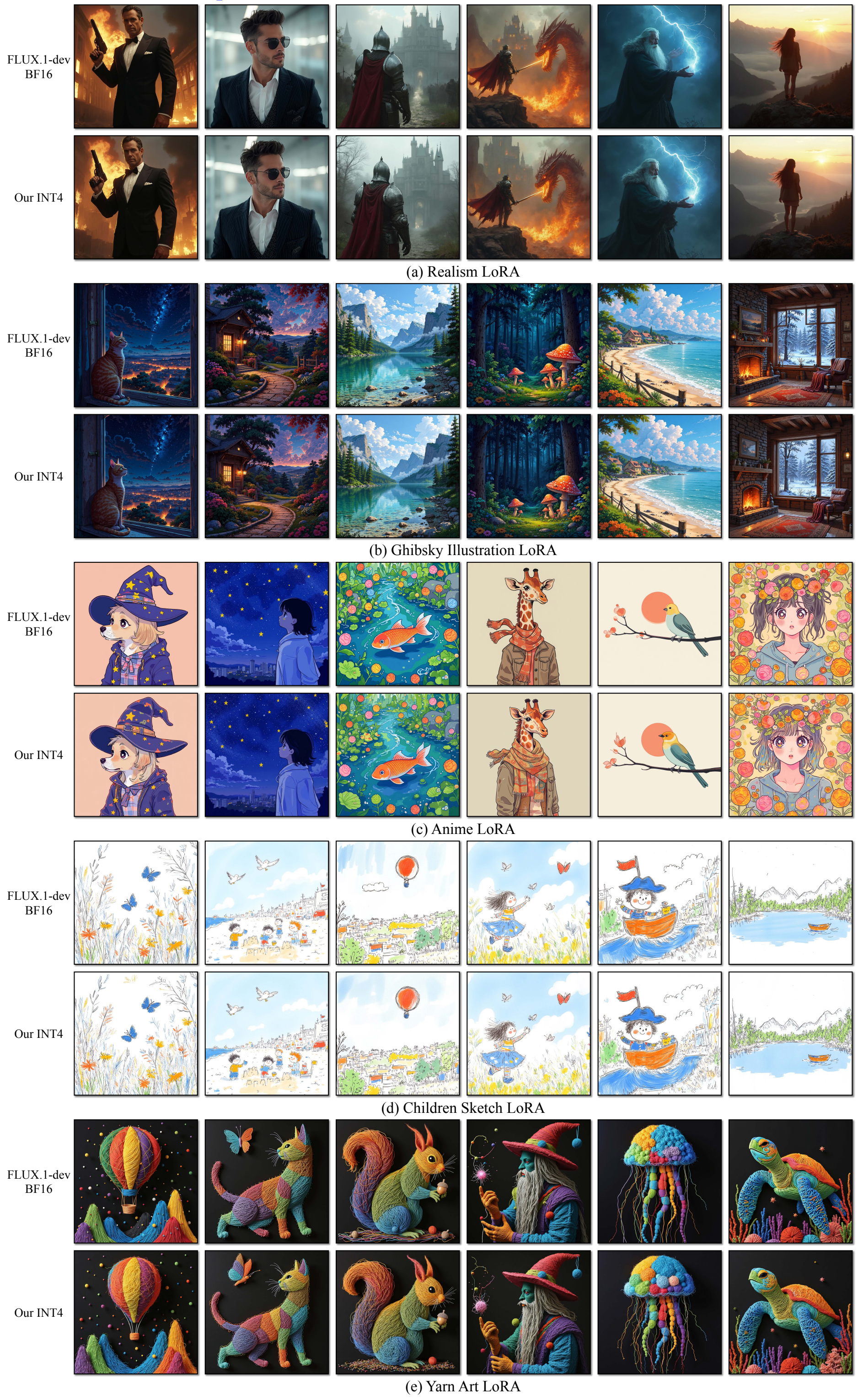


Figure 17: Additional LoRA results on FLUX.1-dev. When applying LoRAs, our INT4 model matches the image quality of the original BF16 model. See App. F for the detailed used text prompts.

D.3 Latency Results

In Tab. 4, we compare FLUX latency on a laptop-level 4090 GPU across different precisions. Compared to INT8, 4-bit quantization delivers a 1.3× speedup. However, without optimization, SVDQuant incurs an 18% overhead due to the low-rank branch. By eliminating redundant memory access, LoRunner achieves latency comparable to naive INT4.

Table 4: Single-step latency comparisons of FLUX on a desktop-level 4090 GPU.

Method	BF16	INT8	Naïve INT4	SVDQuant	SVDQuant +LoRunner
Latency (ms)	657	282	212	250	218

D.4 Comparisons with LLM Baselines

In Tab. 5, we adapted LLM quantization methods (SmoothQuant (Xiao et al., 2023), QLLM (Liu et al., 2024a), QuaRot (Ashkboos et al., 2024), and AffineQuant (Ma et al., 2024d)) for diffusion models and show the W4A4 results on MJHQ. SVDQuant outperforms all baselines by a wide margin across all metrics on PixArt-Σ. On FLUX.1-schnell, SVDQuant achieves the best FID and Image Reward scores and ranks second in LPIPS and PSNR, only behind NF4.

Table 5: 4-bit quantitative quality comparisons on PixArt-Σ and FLUX.1-schnell.

Model	Method	FID (↓)	Image Rewad (↑)	LPIPS (↓)	PSNR (↑)
PixArt-Σ	Naive	206	-1.23	0.762	9.08
	SmoothQuant	48.6	0.617	0.607	12.9
	QLLM	35.8	0.763	0.581	13.1
	AffineQuant	29.6	0.816	0.540	14.5
	QuaRot	28.2	0.847	0.459	15.3
	SVDQuant (Ours)	**20.1**	**0.898**	**0.394**	**16.2**
FLUX.1-schnell	Naive	18.1	0.962	0.345	16.3
	SmoothQuant	18.4	0.943	0.323	16.7
	QLLM	18.3	0.959	0.295	17.3
	AffineQuant	22.8	0.937	0.292	16.9
	QuaRot	19.3	0.951	0.287	17.4
	NF4 (W4A16)	18.9	0.943	**0.257**	**18.2**
	SVDQuant (Ours)	**18.1**	**0.965**	0.292	17.5

D.5 Tradeoff between Quality and Bitwidth

We evaluate LPIPS across different bitwidths for various quantization methods on PixArt-Σ and FLUX.1-schnell using the MJHQ dataset in Fig. 18, with weights and activations sharing the same bitwidth. Following the convention (Xiao et al., 2023; Lin et al., 2024a;b; Li et al., 2023a; Zhao et al., 2024d; Dettmers et al., 2022), for bitwidths above 4, we apply per-channel quantization; for 4 or below, we use per-group quantization (group size 64). SVDQuant consistently outperforms naive quantization and SmoothQuant. Notably, on PixArt–Σ and FLUX.1-schnell, our 4-bit results match 7-bit and 6-bit naive quantization, respectively.

Our SVDQuant can still generate images in the 3-bit settings on both PixArt-Σ and FLUX.1-schnell, performing much better than SmoothQuant. Below this precision (e.g., W2A4 or W4A2), SVDQuant cannot produce images either, since 2-bit symmetric quantization is essentially a ternary quantization. Prior work (Ma et al., 2024a; Wang et al., 2023a) has shown that ternary neural networks require quantization-aware training even for weight-only quantization to adapt the weights and activations to the low-bit distribution.

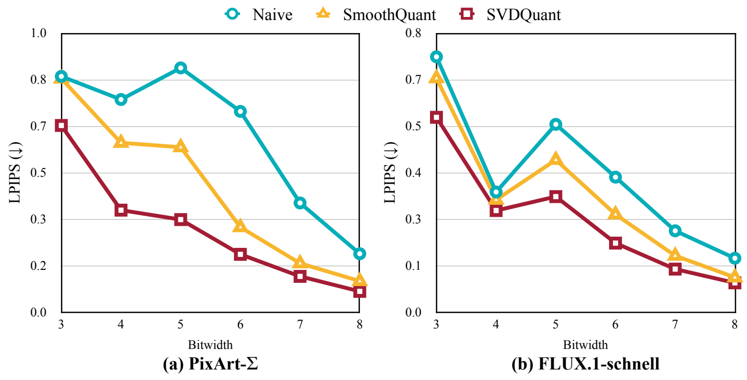


Figure 18: LPIPS of different quantization methods on PixArt-Σ and FLUX.1-schnell across different bitwidths.

D.6 Results on Stale Diffusion 1.4

In Tab. 6, we present additional results comparing Q-Diffusion (Li et al., 2023a) and QuEST (Wang et al., 2024b) on Stable Diffusion 1.4 (Rombach et al., 2022) using the MJHQ dataset with W4A4 precision. Specifically, QuEST is a quantization-aware training method that requires additional training. However, SVDQuant can still outperform it by a large margin regarding LPIPS and PSNR.

Table 6: 4-bit quantitative quality comparisons on Stable Diffusion 1.4.

Method	FID (↓)	LPIPS (↓)	PSNR (↑)
Q-Diffusion	368	0.862	8.00
QuEST	**25.1**	0.771	8.48
SVDQuant (Ours)	38.3	**0.393**	**15.9**

E Limitations

In this work, we do not report the speedups for our FP4 models. This is because we have no access to Blackwell GPUs, which natively support the precision and microscaling for group quantization. On Blackwell hardware, we anticipate greater speedups compared to our INT4 results on 4090 GPUs.

F Text Prompts

Below we provide the text prompts we use in Fig. 9 (from left to right).

```
a man in armor with a beard and a sword
GHIBSKY style, a fisherman casting a line into a peaceful village lake
↪  surrounded by quaint cottages
girl, neck tuft, white hair, sheep horns, blue eyes, nm22 style
sketched style, A squirrel wearing glasses and reading a tiny book under
↪  an oak tree
a panda playing in the snow, yarn art style
```

The text prompts we use in Fig. 17 are (in the rasterizing order):

```
A male secret agent in a tuxedo, holding a gun, standing in front of a
↪  burning building
A handsome man in a suit, 25 years old, cool, futuristic
A knight in shining armor, standing in front of a castle under siege
A knight fighting a fire-breathing dragon in front of a medieval castle,
↪  flames and smoke
A male wizard with a long white beard casting a lightning spell in the
↪  middle of a storm
A young woman with long flowing hair, standing on a mountain peak at dawn,
↪  overlooking a misty valley
```

```
GHIBSKY style, a cat on a windowsill gazing out at a starry night sky and
↪  distant city lights
GHIBSKY style, a quiet garden at twilight, with blooming flowers and the
↪  soft glow of lanterns lighting up the path
GHIBSKY style, a serene mountain lake with crystal-clear water,
↪  surrounded by towering pine trees and rocky cliffs
GHIBSKY style, an enchanted forest at night, with glowing mushrooms and
↪  fireflies lighting up the underbrush
GHIBSKY style, a peaceful beach town with colorful houses lining the
↪  shore and a calm ocean stretching out into the horizon
GHIBSKY style, a cozy living room with a view of a snow-covered forest,
↪  the fireplace crackling and a blanket draped over a comfy chair

a dog wearing a wizard hat, nm22 anime style
a girl looking at the stars, nm22 anime style
a fish swimming in a pond, nm22 style
a giraffe with a long scarf, nm22 style
a bird sitting on a branch, nm22 minimalist style
a girl wearing a flower crown, nm22 style

sketched style, A garden full of colorful butterflies and blooming
↪  flowers with a gentle breeze blowing
sketched style, A beach scene with kids building sandcastles and seagulls
↪  flying overhead
sketched style, A hot air balloon drifting peacefully over a patchwork of
↪  fields and forests below
sketched style, A sunny meadow with a girl in a flowy dress chasing
↪  butterflies
sketched style, A little boy dressed as a pirate, steering a toy ship on
↪  a small stream
sketched style, A small boat floating on a peaceful lake, surrounded by
↪  trees and mountains


a hot air balloon flying over mountains, yarn art style
a cat chasing a butterfly, yarn art style
a squirrel collecting acorns, yarn art style
a wizard casting a spell, yarn art style
a jellyfish floating in the ocean, yarn art style
a sea turtle swimming through a coral reef, yarn art style
```