

Research Article

Efficiency Meets Fidelity: A Novel Quantization Framework for Stable Diffusion

Shuaiting Li^{1,2}, Juncan Deng^{1,2}, Zeyu Wang¹, Hong Gu², Kedong Xu², Haibin Shen¹, Kejie Huang¹

1. Zhejiang University, China; 2. vivo Mobile Communication Co., Ltd, China

Text-to-image generation of Stable Diffusion models has achieved notable success due to its remarkable generation ability. However, the repetitive denoising process is computationally intensive during inference, which renders Diffusion models less suitable for real-world applications that require low latency and scalability. Recent studies have employed post-training quantization (PTQ) and quantization-aware training (QAT) methods to compress Diffusion models. Nevertheless, prior research has often neglected to examine the consistency between results generated by quantized models and those from floating-point models. This consistency is crucial in fields such as content creation, design, and edge deployment, as it can significantly enhance both efficiency and system stability for practitioners. To ensure that quantized models generate high-quality and consistent images, we propose an efficient quantization framework for Stable Diffusion models. Our approach features a Serial-to-Parallel calibration pipeline that addresses the consistency of both the calibration and inference processes, as well as ensuring training stability. Based on this pipeline, we further introduce a mix-precision quantization strategy, multi-timestep activation quantization, and time information precalculation techniques to ensure high-fidelity generation in comparison to floating-point models.

Through extensive experiments with Stable Diffusion v1-4, v2-1, and XL 1.0, we have demonstrated that our method outperforms the current state-of-the-art techniques when tested on prompts from the COCO validation dataset and the Stable-Diffusion-Prompts dataset. Under W4A8 quantization settings, our approach enhances both distribution similarity and visual similarity by 45%~60%.

Shuaiting Li and Juncan Deng equally contributed to this work.

1. Introduction

Diffusion models have yielded remarkable achievements and demonstrated exceptional performance across various generative tasks,^{[1][2][3][4][5][6]} particularly in the realm of text-to-image generation^{[1][3][4]}. Nonetheless, these models often entail significant computational expenses, primarily due to two factors. Firstly, within a Diffusion model, a UNet^{[7][8]} carries out a time-consuming iterative sampling process to progressively denoise a random latent variable. Secondly, the pursuit of superior image quality and higher resolutions has resulted in larger model sizes, necessitating extensive time and memory resources. These challenges render Diffusion models (e.g., Stable Diffusion^[3] and Stable Diffusion XL^[9]) computationally demanding and difficult to deploy in real-world applications requiring low latency and scalability.

Recently, many researchers have investigated quantization strategies for compressing Diffusion models^{[10][11][12][13][14][15]}, predominantly utilizing Post-Training Quantization (PTQ)^{[16][17]}. PTQ does not require retraining or fine-tuning the network and therefore is more attractive than Quantization-Aware Training (QAT)^[18] for large models. However, PTQ methods experience substantial performance degradation at 4 bits and below. Furthermore, the quantization of large text-to-image models, such as Stable Diffusion XL 1.0, can still require 1 day.

Meanwhile, another issue is that most existing research primarily concentrates on optimizing quantized models for high-quality image generation, paying little attention to the consistency of results produced by quantized and floating-point models. In the context of content creation and design, ensuring consistency in expected results is of paramount importance. It is imperative that quantized models exhibit a high degree of similarity to the style and content of images generated by floating-point models. Otherwise, users will encounter significant challenges in predicting and controlling the final results, necessitating extensive debugging and modification of cues, which will inevitably impact their productivity and creative expression. Moreover, alterations in the style of images generated by a quantized model will affect the performance of the downstream tasks^[19]^[20] and the overall reliability of the system.

To address the aforementioned issues, we propose a novel Stable Diffusion quantization framework that is specifically designed to achieve high fidelity and efficiency. We analyze the strengths and

weaknesses of the existing pipeline for the joint optimization of the UNet. By leveraging their advantages, we propose our Serial-to-Parallel pipeline, which ensures consistency in the generated outputs while improving training stability. To further enhance fidelity, several techniques are introduced, including the preservation of temporal information, the utilization of multiple time-step activation quantizers, and a Hessian-based mixed-precision strategy.

The quality of the generated results is evaluated in terms of both distributional and visual similarity. In comparison to previous PTQ methods, our framework demonstrates superior generation consistency in shorter training times across multiple Stable Diffusion models.

2. Related Work

2.1. Diffusion Model Acceleration

While Stable Diffusion models can generate high-quality samples, their slow generation speeds pose a significant challenge for large-scale applications. To tackle this problem, significant efforts have focused on improving the efficiency of the sampling process, which can be categorized into two methods.

The first method involves designing advanced samplers for pre-trained models, such as analytical trajectory estimation^{[21][22]}, implicit sampler^{[5][23][24][25]}, stochastic differential equations^{[6][26][27]} and ordinary differential equations^{[28][29][30]}. Although these methods can reduce the number of sampling iterations required, the significant parameter count and computational demands of Stable Diffusion models limit their application on edge devices.

The second method involves retraining the model, such as diffusion scheme optimization^{[31][32][33][34]}, knowledge distillation^{[35][36]}, sample trajectory optimization^{[25][37]}, and noise scale adjustment^{[38][39]}. Though these techniques effectively speed up the sampling process, re-training a Diffusion model is computationally intensive, especially for resource-constrained devices.

2.2. Diffusion Model quantization

Quantization is a widely used technique that aids in reducing memory usage and speeding up computation. It is generally categorized into two types: QAT^{[40][41][42][43][44]} and PTQ^{[16][17][45][46][47]}. EfficientDM^[15] is representative of QAT work, it proposes a data-free distillation framework and

applies a quantization-aware variant of the low-rank adapter. While QAT is time-consuming and computationally heavy, recent studies focus on PTQ for Diffusion models, which does not require fine-tuning and only necessitates a small amount of unlabeled data for calibration. PTQ4DM^[12] and Q-diffusion^[11] focus on sampling the noise of the floating-point model across different timesteps, Q-diffusion further propose to split the activation of shortcut layers. PTQD^[10] disentangle the quantization noise into its correlated and residual uncorrelated parts and correct them individually. PCR^[14] progressively calibrates the activation quantizer considering the accumulated quantization error across timesteps and selectively relaxing the bit-width for several of those timesteps. However, these PTQ methods seldom consider the consistency of the generated output, and many of them are not designed for new, large pre-trained text-to-image models, such as Stable Diffusion.

3. Preliminaries

3.1. Diffusion models

Diffusion models^{[2][5]} gradually add Gaussian noise with a variance schedule $\beta_1, \dots, \beta_T \in (0, 1)$ to real image $x_0 \sim q(x)$ for T times as sampling process, resulting in a sequence of noisy samples x_1, \dots, x_T . In DDPMs^[2], the sampling process is a Markov chain, which can be formulated as:

$$q(x_{1:T} | x_0) = \prod_{t=1}^T q(x_t | x_{t-1}), q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, \beta_t \mathbf{I}) \quad (1)$$

where $\beta_t = 1 - \alpha_t$. Conversely, the denoising process removes noise from a sample from Gaussian noise $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to gradually generate high-fidelity images. However, due to the unavailability of the true reverse conditional distribution $q(x_{t-1} | x_t)$, Diffusion models approximate it via variational inference by learning a Gaussian distribution $p_\theta(x_{t-1} | x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$, the μ_θ can be derived by reparameterization trick as follows:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) \quad (2)$$

where $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$ and $\epsilon_\theta(\cdot)$ is a trainable model to predict noise. The variance $\Sigma_\theta(x_t, t)$ can be either learned^[39] or fixed to a constant schedule^[2] σ_t . When it uses a constant schedule, x_{t-1} can be expressed as:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t \mathbf{z} \quad (3)$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

The formulas outlined in our research are based on the DDPM framework but can be easily adjusted for other accelerated sampling techniques such as DDIM^[5], PNDM^[29], and Euler^[48].

3.2. Model Quantization

Quantization^[49] is a key technique in model compression. This method compresses neural networks by reducing the number of bits used for model weights and activations. The quantization process can be formulated as:

$$w_q = \text{clip} \left(\text{round} \left(\frac{w}{s} \right) + z, q_{\min}, q_{\max} \right) \quad (4)$$

where s is the scaling factor, z is the zero-point, and q_{\min} and q_{\max} are the minimum and maximum quantization values, respectively. Reversely, the dequantization process is formulated as:

$$\hat{w} = (w_q - z) \times s \quad (5)$$

We utilize uniform quantization in all our study experiments.

4. Method

As illustrated in Fig. 1, we present a novel quantization framework for large pre-trained text-to-image diffusion models, including Stable Diffusion v1-4 and Stable Diffusion XL. We begin by introducing a Serial-to-Parallel training pipeline that not only addresses the consistency between the training and inference processes but also guarantees stability during training. Subsequently, several techniques are integrated into the pipeline. Multi-timestep activation quantizer is set to separately optimize the parameters associated with each timestep. Additionally, the time feature is precalculated and the accurate projection information is saved for training and inference. Furthermore, we implement a mixed-precision quantization strategy that assigns higher bit-width to sensitive layers and lower bit-width to insensitive layers.

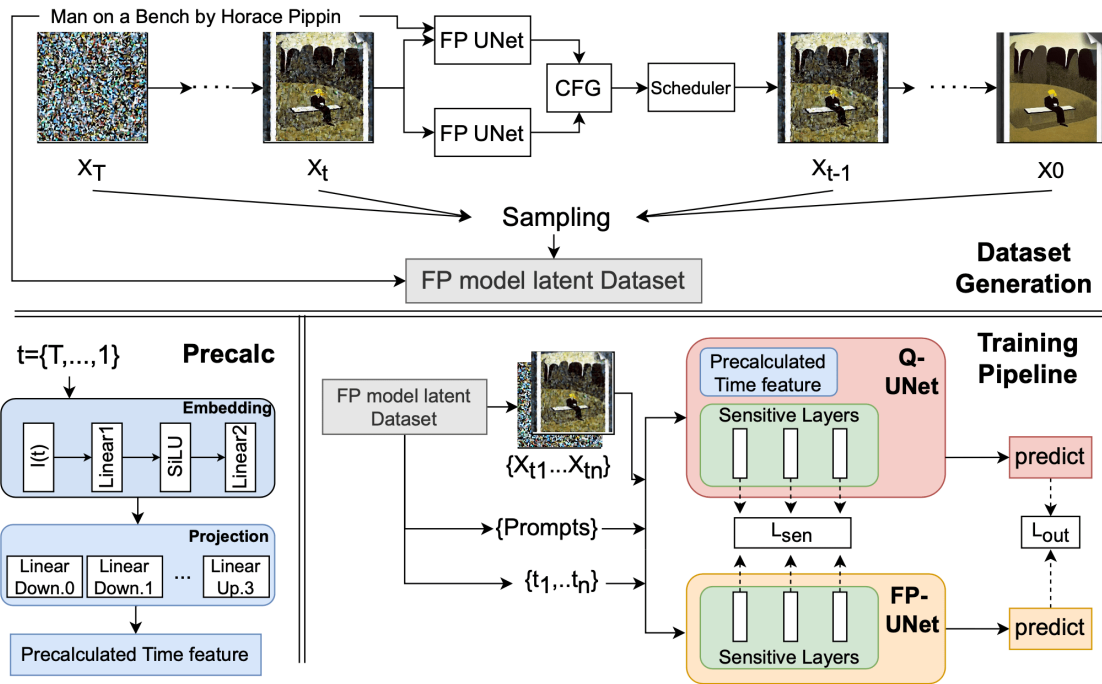


Figure 1. Overview of our proposed quantization framework. (a) Dataset Generation: During the inference of the floating-point model, latent generated from various timesteps for each prompt are randomly sampled. (b) Time information precalculation: The feature map of time projection layers is precalculated for training and inference. Subsequently, the time embedding and projection layers are removed from UNet. (c) Serial-to-Parallel training pipeline: A Hessian-based sensitive ranking is assigned to each layer as well as different bit-widths. At each iteration, latent from various timesteps along with the corresponding prompts are selected from the dataset. The Loss function is calculated between the output and the sensitive layers.

4.1. Serial-to-Parallel Training Pipeline

Previous works on jointly optimizing quantized models and distillation-based compression can be roughly divided into two categories: (a) 'Serial' (e.g. [15]) and (b) 'Parallel' (e.g. [50][51]), as illustrated in Fig. 2. The serial pipeline operates in a data-free manner, requiring only a few prompts to generate the latent of the floating-point model. This latent is then used as input for the quantized model, which is updated in a chronological sequence. In contrast, the parallel pipeline is more closely aligned with the original Stable Diffusion training process. This approach relies on an image-text pair dataset, where the image is processed through a Variational Autoencoder (VAE) to derive the initial latent. In

each iteration, multiple timesteps are randomly sampled, and the latent is then augmented with varying levels of Gaussian noise, as determined by the scheduler.

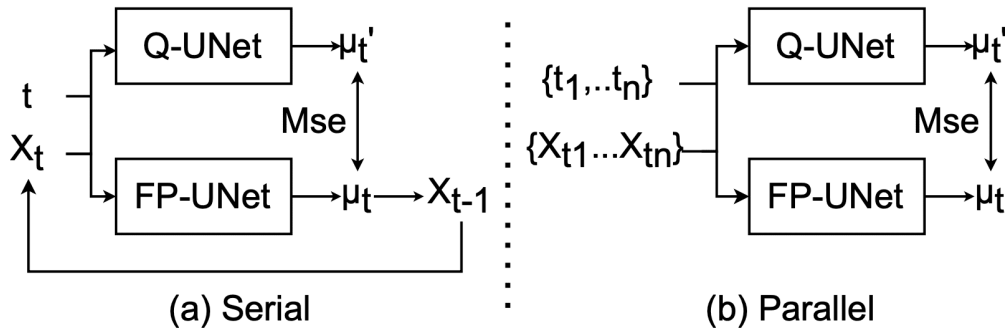


Figure 2. Comparison of 'Serial' and 'Parallel' training pipeline.

Both frameworks have their own advantages and disadvantages. As illustrated in Fig. 3, the theoretical latent generated by adding Gaussian noise differs markedly from the actual latent range when reasoning with the floating-point model.

$$\mu_{fp}(x_t, t) \neq x_{t-1} \leftarrow \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, \beta_t \mathbf{I}) \quad (6)$$

Consequently, it is more beneficial to use the latent from the floating-point model as input during the distillation process. This approach enhances the consistency of the quantized model's outputs in comparison to those of the floating-point model.

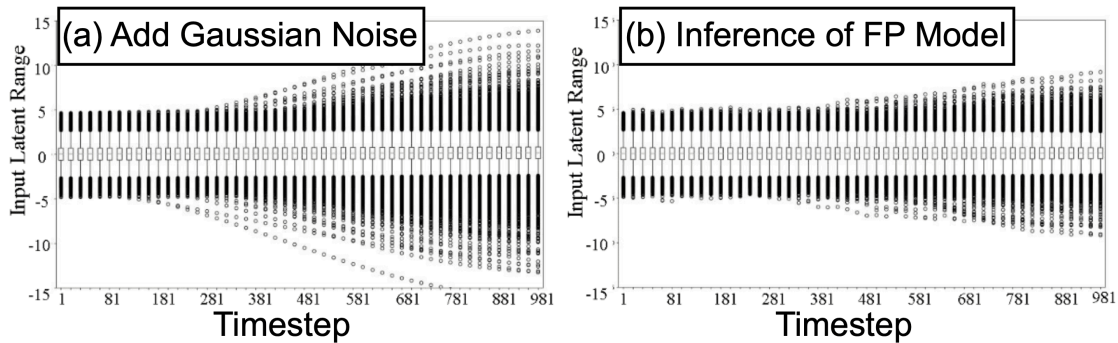


Figure 3. Difference in input latent range at each timestep with the same initial latent. (a) Gradually adding Gaussian noise based on Eq. (1). (b) Step-by-step denoising during inference of floating-point Stable Diffusion v1-4.

Also, in Stable Diffusion models, where all timesteps share the same weight, it is more appropriate to average the gradients across multiple timesteps rather than relying solely on their sequential order. As demonstrated in Fig. 4, we have documented the changes in gradients for both serial and parallel pipelines. It can be observed that the gradients remain relatively stable during parallel training, whereas they exhibit periodic oscillations during serial training. Previous research^{[52][53]} have indicated that Adam optimizer may sometimes underperform in the presence of periodic oscillating gradients.

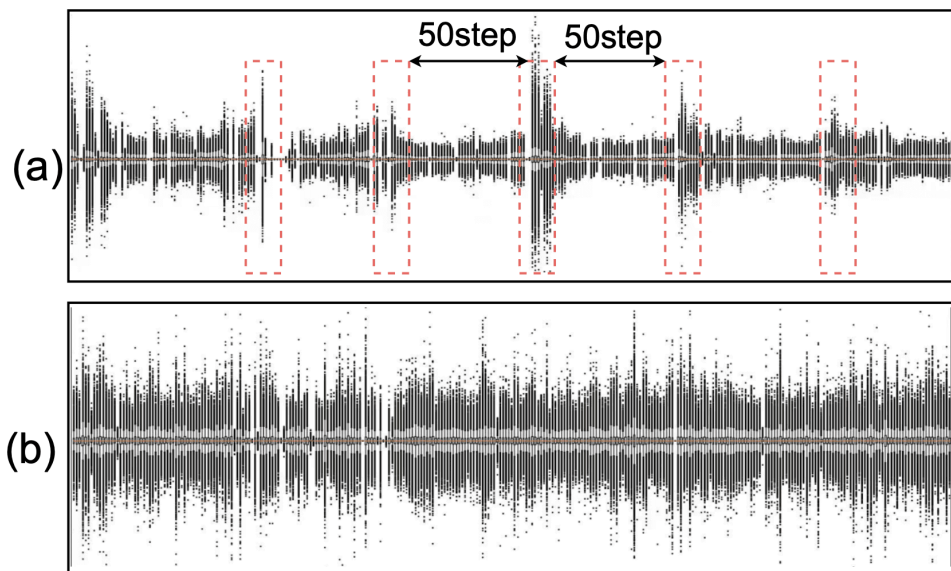


Figure 4. Box plot illustrating the gradient variations of the 'down_blocks.0.attentions.0.proj_in' layer in the quantized Stable Diffusion v1-4 model during training. (a) represents the serial pipeline, and (b) represents the parallel pipeline.

It can thus be concluded that the latent of the serial pipeline is more appropriate while the training procedure of the parallel pipeline is more reasonable. Building on this analysis, we introduce our method, termed 'Serial-to-Parallel', which harnesses the strengths of both serial and parallel pipelines. The advantages of different pipelines are summarized in Table 1. Initially, the inference is conducted with the floating-point model, whereby the latent is randomly sampled from various timesteps for each prompt. During the training process, at each iteration, the latent is sampled from different timesteps along with their corresponding prompts from the latent dataset. This strategy

renders our framework data-free, relying solely on prompts, while simultaneously enhancing generation consistency and ensuring training stability.

Pipeline	Data-free	Consistency	Stability
Serial	✓	✓	X
Parallel	X	X	✓
Ours	✓	✓	✓

Table 1. Comparison of our pipeline and previous pipeline.

4.2. Components For Higher Fidelity

Moreover, a variety of techniques are employed to guarantee the fidelity of the generated results.

Accurate activation quantization

Previous studies on Diffusion models^{[12][11][54][55][56]} have shown that the activation distribution at different timesteps varies greatly, posing a challenge for activation quantization. We adopt different activation quantization parameter sets for different timesteps, which can be expressed as:

$$\mathbf{s}_l = \{s_l^0, s_l^1, \dots, s_l^{T-1}\}, \quad \mathbf{z}_l = \{z_l^0, z_l^1, \dots, z_l^{T-1}\} \quad (7)$$

where s_l^t and z_l^t are the scaling factor and zero-point of activation quantization parameter for the l -th layer at timestep t . The memory consumption of these parameters is negligible and does not influence the inference speed. With regard to the inputs of different time steps within the same batch, our pipeline is capable of efficiently optimizing the activation quantization parameters for these time steps simultaneously.

Low memory time information precalculation

In a Stable Diffusion model, the time-step t is firstly encoded by time-embedding layers, then passed through time-projection layers in each Bottleneck block. The Time information e_p inserted into the UNet is calculated as follows:

$$e_t = emb(t), \quad ep_{i,i} = proj_i(e_t) \quad (8)$$

We observe that the quantization of both the time embedding and time embedding projection layers of the model has a significant impact on the quality of the generated images. When the inference configuration is determined, the output of the time embedding module e_t is only related to timesteps, and ep is dependent only on e_t . Consequently, ep is finite and invariant. Therefore, we remove time-embedding and time-projection from the model and save the ep , which is directly input into the Resnet blocks of the model. The memory usage and computational cost of ep are much smaller than the parameters of the time embedding module and the time embedding projection layers.

Mixed-Precision Quantization Strategy

Recent studies^{[55][50]} observe that compressing different blocks in Diffusion models can lead to different image generation quality. However, quantizing a certain block, fine-tuning, and then evaluating the model, are computationally intensive and time-consuming. Moreover, the sensitivity of each block is coarse-grained, which may lead to suboptimal compression. To address these issues, we propose a mixed-precision quantization strategy to identify sensitive layers and assign different bit-width to different layers based on their sensitivity.

Our objective is to evaluate the sensitivity S_i of each layer. We consider i -th layer (e.g. linear layer) of the model with weight $W_i \in \mathbb{R}^{C_{in} \times C_{out}}$, given dataset $\mathcal{D} = \{x_m, y_m\}_{m=1}^n$ comprising n samples. Specifically, to estimate S_i , the deviation in the loss function caused by W_i from original value to zero can be formulated as :

$$S_i = \left| \frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_i}^\top W_i + \frac{1}{2} W_i^\top H W_i + \mathcal{O}(\|W_i\|^3) \right| \quad (9)$$

where H is the Hessian matrix and \mathcal{L} denotes the loss function. However, the formula cannot be directly computed, since the computation of H on the model is impractical. By employing the Fisher information matrix approximation^[57], the computation of loss deviation can be rewritten as:

$$S_i \approx \left| \frac{\partial \mathcal{L}(\mathcal{D})}{\partial W_i}^\top W_i - \frac{1}{2} \left(\frac{1}{n} \sum_{m=1}^n \frac{\partial \mathcal{L}(\mathcal{D}_n)}{\partial W_i} W_i \right)^2 \right| \quad (10)$$

where the redundant term can be neglected. Finally, the mix-precision quantization is then conducted based on the sensitivity ranking. We selected the top 5% of layers with the highest sensitivity as sensitive layers, while the bottom 5% of layers are designated as insensitive layers. For A8 quantization, sensitive layers are set to A16, while insensitive layers are set to A4. For W4

quantization, sensitive layers are set to W8. The detailed average bit-width will be displayed in the experiment section.

4.3. Objective Function

We optimize quantized UNet ϵ_q to mimic the output of the floating-point UNet ϵ_{fp} . Given the latent $\{x_{t_1} \dots x_{t_n}\}$ at timestep $\{t_1 \dots t_n\}$, text embedding $\{p_1 \dots p_n\}$ from frozen text encoder, and precalculated projection e_p , the output loss is defined as the mean squared error between the quantized and floating-point UNet outputs:

$$\mathcal{L}_{out} = \mathbb{E} \left[\|\epsilon_{fp}(x_{t_1 \dots t_n}, p_{1 \dots n}, e_p) - \epsilon_q(x_{t_1 \dots t_n}, p_{1 \dots n}, e_p)\|_2^2 \right] \quad (11)$$

where ϵ_{fp} and ϵ_q indicate the floating-point UNet and the quantized UNet, respectively.

The loss function of the feature maps by the sensitive layers is added to ensure they receive more attention:

$$\mathcal{L}_{sen} = \mathbb{E} \left[\|f_{fp}^s(x_{t_1 \dots t_n}, p_{1 \dots n}, e_p) - f_q^s(x_{t_1 \dots t_n}, p_{1 \dots n}, e_p)\|_2^2 \right] \quad (12)$$

where f_{fp}^s and f_q^s indicate the floating-point and quantized feature maps of the sensitive layer, respectively.

The final loss function is: $\mathcal{L} = \mathcal{L}_{out} + \mathcal{L}_{sen}$

5. Experiments

5.1. Experimental Setup

Datasets. In this paper, we conduct experiments using two distinct datasets: COCO^[58] and Stable-Diffusion-Prompts. We utilize prompts from the COCO training dataset to construct the latent dataset. In terms of evaluation, the process is twofold. Following^[14], firstly 5,000 prompts are selected from the COCO validation dataset, which has been extensively employed in previous studies. Secondly, an additional 5,000 prompts from the Stable-Diffusion-Prompts dataset are used to assess the generalization capabilities of our quantized model in different prompt scenarios.

Metrics. We evaluate the generative results of the quantized model from the perspectives of distributional similarity and visual similarity. For distributional similarity, we refer to the FID-to-FP^[14], which is the Fréchet Inception Distance between images generated by the quantized model and

floating-point model. For visual similarity, we consider the commonly used metrics of SSIM^[59], LPIPS^[60], and PSNR^[61]. Additionally, we use CLIP score^[62] to evaluate the matching degree between images and prompts. The evaluation code is adopted from^[63] and^[64].

Baselines and implementation. We compare our proposed approach against advanced techniques: Q-diffusion^[11], PTQ4DM^[12], and PCR^[14]. Results are obtained from^[14] or reproduced. We employ the Stable Diffusion v1-4 (resolution of 512x512), Stable Diffusion v2-1 (resolution of 512x512), and the Stable Diffusion XL 1.0 (resolution of 768x768), both sourced from Hugging Face. We compare exclusively with PCR^[14] and align the quantization and generation settings with it. Except for special declaration, the standard setup involves a 50-step PNDM sampling process for the Stable Diffusion model and a 50-step Euler sampling process for the Stable Diffusion XL model, with both configurations using a Classifier-Free Guidance (CFG) scale of 7.5. All experiments are conducted using a single NVIDIA A100.

5.2. Dataset Generation Analysis

First of all, we discussed the trade-off of some crucial hyperparameters in the latent dataset generation.

More prompts or more timesteps?

In the dataset generation process, we can randomly sample varying amounts of latent for each prompt. For comparison, two datasets have been constructed. The first dataset comprises 4000 prompts, with 50 latent per prompt. The second consists of 20000 prompts, each prompt with just 1 latent. As demonstrated in Table 2, despite the first dataset having 10× more latent, it exhibits a similar FID-to-FP. We can infer that a dataset with more prompts is more resistant to overfitting and has a smaller size but with a longer generation time. Given the comparable outcomes of the two strategies, users can select their sampling strategies based on their time requirements or storage requirements.

Methods	Prompts	Size	Time	FID-to-FP
50steps/prompt	4000	6G	1.8h	10.12
1steps/prompt	20000	0.6G	4h	10.14

Table 2. Comparison of different sampling strategies for dataset generation.

Training hyperparameters

Since we only perform limited training iterations, the amount of prompts is crucial. A smaller dataset requires less storage but can result in overfitting. In our experiments, the training iteration is fixed to 10000 and the batch size is fixed to 12 (resp. 4) for Stable Diffusion v1-4 and v2-1 (resp. Stable Diffusion XL). Experiments are conducted on datasets of varying lengths for both sampling strategies. As illustrated in Fig. 5, the challenge of overfitting is evident with smaller datasets. To avoid severe overfitting, the default settings for Stable Diffusion v1-4 (resp. Stable Diffusion XL) are set at 6000 prompts (resp. 2000 prompts) with 50-step sampling and 40000 prompts (resp. 10000 prompts) with 1-step sampling. A summary of the training efficiency comparison is provided in Table 3. It is evident that our approach significantly reduces training time compared to PTQ methods, especially for larger models like Stable Diffusion XL.

Methods		Time Cost			
		SD1.4		SDXL	
PCR		≈13h		≈25h	
Ours	Gen Data (Once for all)	(a)	(b)	(a)	(b)
		3h	≈7.5h	2.5h	≈5.5h
	Train	≈4.5h		≈7.5h	

Table 3. Efficiency comparison on Stable Diffusion v1-4 and Stable Diffusion XL. (a) denotes sampling 50 steps per prompt, and (b) denotes sampling 1 step per prompt

5.3. Main Results

For the following experiments, 1-step per prompt sampling strategy is selected. The quantization results of Stable Diffusion v1-4 on the COCO and Stable-Diffusion-Prompts validation datasets are presented in Table 4. For better comparison, we have additionally listed the results without mixed precision. On the COCO dataset, our approach demonstrates a 40% reduction in FID-to-FP on W4A8 compared to PCR, highlighting the effectiveness of our proposed method. Furthermore, our approach exhibits significant improvements on the Stable-Diffusion-Prompts dataset, which illustrates the generalizability of our approach across diverse prompt styles. Notably, our method still achieves smaller FID-to-FP compared to PCR with smaller latent dataset, as shown in Fig. 5.

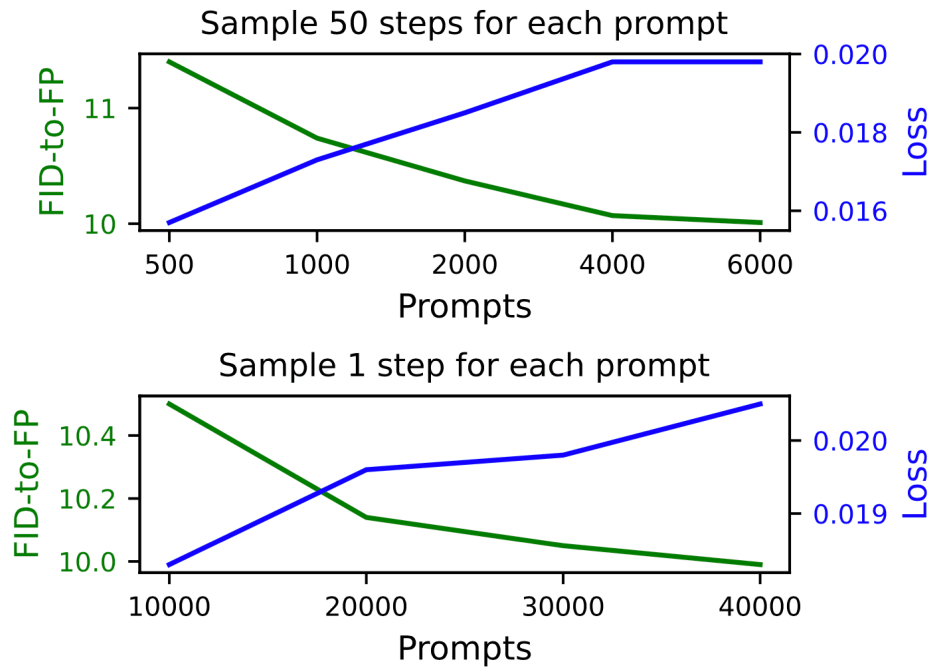


Figure 5. Comparison of Loss and FID-to-FP Curves under Different Prompt Quantities.

Prompts	Method	W/A	FID-to-FP↓	ΔCLIPscore↑
	FP32	32/32	0.0	0.00
SD v1.4				
COCO Prompts	Q-diff	8/8	18.6	-0.31
	PTQ4DM	8/8	14.6	-0.13
	PCR	8/8.4*	8.3/8.7†	+0.01/-0.03†
	Ours	8/8.35*	8.1	+0.06
	Q-diff	4/8	20.4	-0.31
	PTQ4DM	4/8	17.7	-0.21
	PCR	4/8.4*	14.2/16.3†	+0.02/-0.23†
	Ours	4/8	10.6	-0.04
	Ours	4.02*/8.35*	10.0	-0.06
	SD Prompts	Q-diff	8/8	16.2
PTQ4DM		8/8	13.2	-1.02
PCR		8/8.4*	9.5/9.7†	-0.05/-0.25†
Ours		8/8.35*	9.2	-0.15
Q-diff		4/8	17.4	-1.48
PTQ4DM		4/8	17.2	-1.40
PCR		4/8.4*	17.9/19.2†	-0.74/-1.33†
Ours		4/8	12.8	-0.24
Ours		4.02*/8.35*	12.0	-0.10
SD v2.1				
COCO Prompts	PCR	4/8.4*	35.9†	-0.81†
	Ours	4/8	14.0	+0.03
	Ours	4.03*/8.34*	12.4	+0.07
SD Prompts	PCR	4/8.4*	46.3†	-1.30†
	Ours	4/8	18.2	-0.40
	Ours	4.03*/8.34*	15.3	-0.30

Table 4. 512×512 generation results on COCO and Stable-Diffusion-Prompts for Stable Diffusion v1-4 and Stable Diffusion v2-1. ↓ means lower is better. ↑ means higher is better. † denotes reproduced results on our machine. * denotes mix-precision.

A visual comparison is provided in Fig. 6. Previous methods, when quantized to 4-bit, result in noticeable style changes in the generated images compared to those produced by the floating-point model. Such changes include but are not limited to, alterations in scene layout and facial features, loss of color and object, and the blending of multiple objects. In contrast, the images generated by our method are consistently of high fidelity.

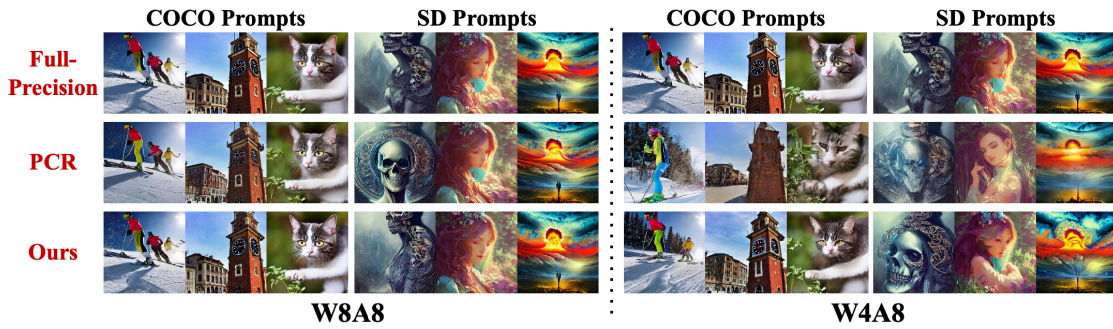


Figure 6. Comparison of Loss and FID-to-FP Curves under Different Prompt Quantities.

In the case of Stable Diffusion v2-1, due to the absence of results from PCR, we utilized its settings from Stable Diffusion v1-4 to replicate outcomes. As illustrated in Table 4, our approach exhibits a substantial superiority over PCR.

To further validate our method, we conduct experiments using Stable Diffusion XL to generate images at a resolution of 768x768. The results, presented in Table 5, demonstrate superior performance on both COCO and SD prompts. Moreover, as illustrated in Fig. 7, our method consistently produces high-quality images that closely resemble those generated by floating-point models. In comparison with PCR^[14], Our method achieves a significant reduction in FID by up to 45%.



Figure 7. Stable Diffusion XL 768x768 image generation using COCO prompts and Stable-Diffusion-Prompts.

Prompts	Method	W/A	FID-to-FP↓	ΔCLIPscore↑
	FP32	32/32	0.0	0.00
COCO	Q-diff	8/8	38.1	-10.47
	PTQ4DM	8/8	38.6	-10.46
	PCR	8/8.4*	12.0	-2.39
	Ours	8/8.1*	7.6	-0.02
	Prompts	Q-diff	4/8	44.0
	PTQ4DM	8/8	46.4	-10.52
	PCR	4/8.4*	18.2	-2.59
	Ours	4/8	10.92	+0.06
	Ours	4.2*/8.1*	10.6	+0.03
SD	Q-diff	8/8	22.8	-9.73
	PTQ4DM	4/8	22.8	-9.65
	PCR	8/8.4*	11.7	-4.00
	Ours	8/8.1*	6.8	-0.07
	Prompts	Q-diff	4/8	24.9
	PTQ4DM	4/8	28.2	-9.61
	PCR	4/8.4*	18.2	-4.03
	Ours	4/8	10.67	-0.02
	Ours	4.2*/8.1*	10.2	-0.01

Table 5. 768×768 generation results on COCO and Stable-

Diffusion-Prompts validation datasets for Stable Diffusion XL.

↑ means higher is better. * Denotes mix-precision. ↑ means higher is better.

In addition to the distribution similarity, the visual similarity results are summarized in Table 6. Our method achieves significantly better results in LPIPS, SSIM, and PSNR metrics, further demonstrating that our approach can generate images that are highly consistent with those produced by the floating-point model.

Prompts	Methods	LPIPS↓	SSIM↑	PSNR↑
SD v1.4				
COCO Prompts	PCR	0.53†	0.47†	13.6†
	Ours	0.32	0.58	15.5
SD Prompts	PCR	0.52†	0.50†	14.9†
	Ours	0.38	0.59	16.9
SD v2.1				
COCO Prompts	PCR	0.55†	0.47†	13.5†
	Ours	0.38	0.57	15.3
SD Prompts	PCR	0.54†	0.49†	14.6†
	Ours	0.42	0.56	16.1
SD XL				
COCO Prompts	PCR	0.68†	0.44†	12.6†
	Ours	0.43	0.61	16.0
SD Prompts	PCR	0.65†	0.46†	12.7†
	Ours	0.39	0.67	17.5

Table 6. Visual comparison with PCR^[14] under W4A8 quantization setting. ↓ means lower is better, ↑ means higher is better. † denotes reproduced results on our machine.

5.4. Ablation Study

An ablation study is conducted to analyze the impact of different components. For clarity, we refer to time-feature precalculation, multiple time-step activation, and mix-precision as 'Components'. The serial-to-parallel pipeline modification is denoted as 'pipeline'. The term 'Base' refers to the original serial pipeline as illustrated in Table 1.

The quantization results for Stable Diffusion v1-4, tested on COCO prompts using the W4A8 quantization setting, are detailed in Table 7. Each of the proposed components significantly enhances the fidelity of the generated images. Notably, the Serial-to-Parallel pipeline exhibits the most

pronounced effect, underscoring the essential role of multiple timesteps in achieving stable training. Our method incorporates all these components effectively.

Method	FID↓	sFID↓	LPIPS↓	SSIM↑
FP32	0.00	0.00	0.00	1.00
Base	12.64	69.74	0.48	0.50
+ Components	11.48	68.81	0.45	0.52
+ Pipeline(Ours)	9.99	65.47	0.32	0.58

Table 7. Ablation results on the COCO validation prompts for Stable Diffusion v1-4 under W4A8 settings.

6. Conclusion

This research explores the application of quantization to Stable Diffusion models. In this paper, we propose an efficient quantization framework for Stable Diffusion models aiming for high generation consistency. We introduce a Serial-to-Parallel pipeline which not only considers the consistency of the training process and the inference process but also ensures the training stability. With the aid of multi-timestep activation quantization, time information precalculation, and mix-precision quantization strategy, high-fidelity generation is guaranteed. Extensive experiments demonstrate that our method generates high-fidelity figures within a shorter time and outperforms state-of-the-art techniques.

Supplementary Material

7. Module Sensitivity Details

We demonstrate the module sensitivity for Stable-diffusion-v1.4 and Stable-diffusion-XL-v1.0. The results are shown in Fig. 8 and Fig. 9.

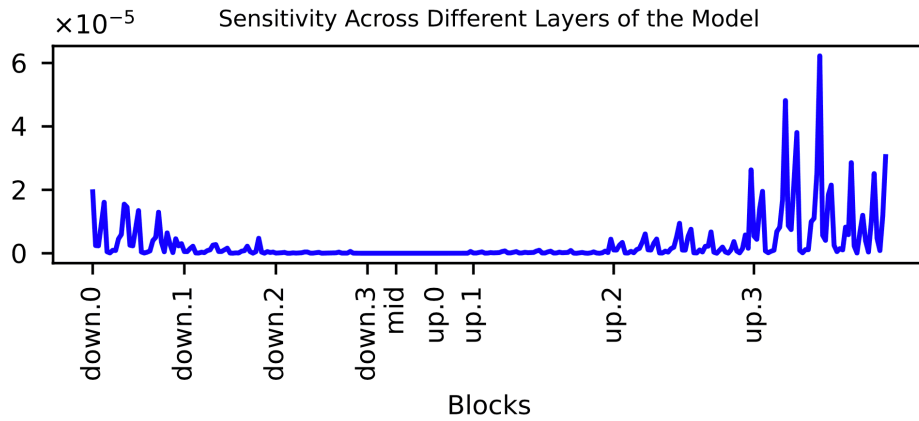


Figure 8. Layer sensitivity for Stable diffusion v1.4

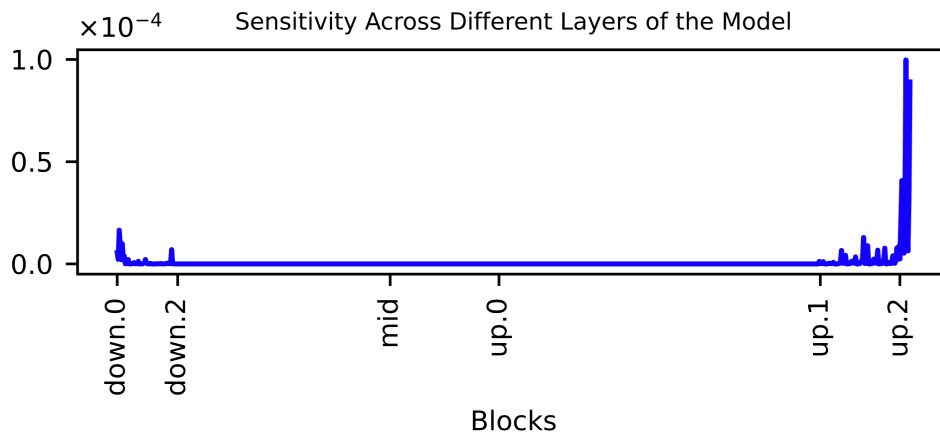


Figure 9. Layer sensitivity for Stable diffusion XL

Moreover, we also list the 5% most sensitive layers and 5% insensitive layers in both models. As mentioned in the paper, For A8 quantization, sensitive layers are set to A16, while insensitive layers are set to A4. For W4 quantization, sensitive layers are set to W8.

7.1. Stable Diffusion v1-4

Sensitive layers

- up_blocks.3.attentions.2.proj_in

- up_blocks.3.attentions.1.proj_in
- up_blocks.3.attentions.1.transformer_blocks.0.attn1.to_out.0
- up_blocks.3.resnets.2.conv_shortcut
- up_blocks.3.attentions.2.proj_out
- up_blocks.3.attentions.0.proj_in
- up_blocks.3.attentions.1.proj_out
- up_blocks.3.resnets.1.conv_shortcut
- up_blocks.3.attentions.1.transformer_blocks.0.attn1.to_v
- up_blocks.3.attentions.2.transformer_blocks.0.attn1.to_out.0
- down_blocks.0.attentions.0.proj_in
- up_blocks.3.attentions.0.transformer_blocks.0.attn1.to_out.0
- up_blocks.3.attentions.2.transformer_blocks.0.attn1.to_v

Insensitive layers

- down_blocks.3.resnets.0.conv2
- mid_block.resnets.1.conv2
- mid_block.resnets.0.conv2
- mid_block.resnets.1.conv1
- down_blocks.3.resnets.1.conv2
- down_blocks.3.resnets.0.conv1
- mid_block.attentions.0.transformer_blocks.0.attn2.to_k
- mid_block.resnets.0.conv1
- down_blocks.3.resnets.1.conv1
- up_blocks.0.resnets.0.conv1
- mid_block.attentions.0.transformer_blocks.0.attn2.to_q
- mid_block.attentions.0.transformer_blocks.0.attn1.to_q
- mid_block.attentions.0.transformer_blocks.0.attn1.to_k

7.2. Stable Diffusion XL

Sensitive layers

- up_blocks.2.resnets.1.conv_shortcut
- up_blocks.1.attentions.1.proj_in
- up_blocks.1.attentions.0.proj_in
- up_blocks.2.resnets.2.conv_shortcut
- up_blocks.1.attentions.0.transformer_blocks.0.attn1.to_out.0
- up_blocks.1.attentions.2.proj_in
- up_blocks.1.attentions.0.transformer_blocks.1.attn1.to_out.0
- up_blocks.1.attentions.0.transformer_blocks.1.attn1.to_v
- up_blocks.1.attentions.0.transformer_blocks.0.attn1.to_v
- up_blocks.1.attentions.1.transformer_blocks.0.attn1.to_out.0
- up_blocks.2.resnets.0.conv_shortcut
- up_blocks.1.resnets.1.conv_shortcut
- up_blocks.1.attentions.1.transformer_blocks.1.attn1.to_v
- up_blocks.1.attentions.1.transformer_blocks.0.attn1.to_v
- down_blocks.1.resnets.0.conv_shortcut
- up_blocks.1.resnets.2.conv_shortcut
- up_blocks.1.attentions.0.proj_out
- up_blocks.1.attentions.1.transformer_blocks.1.attn1.to_out.0
- up_blocks.1.attentions.2.transformer_blocks.1.attn1.to_v
- up_blocks.1.attentions.2.transformer_blocks.0.attn1.to_out.0
- up_blocks.1.attentions.1.proj_out
- up_blocks.1.attentions.1.transformer_blocks.0.ff.net.2
- up_blocks.1.attentions.2.transformer_blocks.1.attn1.to_out.0
- up_blocks.2.resnets.1.conv2
- up_blocks.1.attentions.2.proj_out
- up_blocks.2.resnets.0.conv2
- up_blocks.1.attentions.2.transformer_blocks.0.attn1.to_v
- up_blocks.1.attentions.1.transformer_blocks.1.attn1.to_q

- down_blocks.0.resnets.0.conv2
- up_blocks.1.attentions.0.transformer_blocks.0.ff.net.2
- up_blocks.1.attentions.1.transformer_blocks.1.attn1.to_k
- up_blocks.1.resnets.0.conv_shortcut
- up_blocks.2.resnets.1.conv1
- down_blocks.1.attentions.0.proj_in
- up_blocks.1.attentions.1.transformer_blocks.0.ff.net.0.proj
- up_blocks.1.attentions.1.transformer_blocks.1.ff.net.0.proj
- up_blocks.1.attentions.1.transformer_blocks.1.ff.net.2
- up_blocks.2.resnets.2.conv2
- down_blocks.0.resnets.1.conv2

Insensitive layers

- up_blocks.0.attentions.0.transformer_blocks.7.attn2.to_q
- up_blocks.0.attentions.2.transformer_blocks.5.attn2.to_q
- mid_block.attentions.0.transformer_blocks.4.attn2.to_k
- mid_block.attentions.0.transformer_blocks.4.attn2.to_q
- down_blocks.2.attentions.0.transformer_blocks.5.attn2.to_k
- up_blocks.0.attentions.2.transformer_blocks.7.attn2.to_q
- mid_block.attentions.0.transformer_blocks.5.attn2.to_k
- down_blocks.2.attentions.1.transformer_blocks.9.attn2.to_k
- mid_block.attentions.0.transformer_blocks.5.attn2.to_q
- down_blocks.2.attentions.1.transformer_blocks.9.attn2.to_q
- up_blocks.0.attentions.2.transformer_blocks.5.attn2.to_k
- up_blocks.0.attentions.0.transformer_blocks.7.attn2.to_k
- down_blocks.2.attentions.0.transformer_blocks.9.attn2.to_q
- down_blocks.2.attentions.0.transformer_blocks.9.attn2.to_k
- up_blocks.0.attentions.0.transformer_blocks.8.attn2.to_q
- up_blocks.0.attentions.1.transformer_blocks.9.attn2.to_q
- up_blocks.0.attentions.0.transformer_blocks.8.attn2.to_k
- down_blocks.2.attentions.0.transformer_blocks.7.attn2.to_q

- down_blocks.2.attentions.0.transformer_blocks.7.attn2.to_k
- up_blocks.0.attentions.1.transformer_blocks.9.attn2.to_k
- down_blocks.2.attentions.0.transformer_blocks.8.attn2.to_k
- up_blocks.0.attentions.2.transformer_blocks.7.attn2.to_k
- up_blocks.0.attentions.2.transformer_blocks.8.attn2.to_q
- mid_block.attentions.0.transformer_blocks.6.attn2.to_q
- mid_block.attentions.0.transformer_blocks.6.attn2.to_k
- up_blocks.0.attentions.2.transformer_blocks.8.attn2.to_k
- up_blocks.0.attentions.1.transformer_blocks.8.attn2.to_q
- down_blocks.2.attentions.0.transformer_blocks.8.attn2.to_q
- down_blocks.2.attentions.0.transformer_blocks.6.attn2.to_k
- up_blocks.0.attentions.1.transformer_blocks.8.attn2.to_k
- mid_block.attentions.0.transformer_blocks.7.attn2.to_q
- mid_block.attentions.0.transformer_blocks.7.attn2.to_k
- down_blocks.2.attentions.0.transformer_blocks.6.attn2.to_q
- up_blocks.0.attentions.2.transformer_blocks.6.attn2.to_q
- up_blocks.0.attentions.2.transformer_blocks.6.attn2.to_k
- up_blocks.0.attentions.0.transformer_blocks.9.attn2.to_q
- up_blocks.0.attentions.0.transformer_blocks.9.attn2.to_k
- up_blocks.0.attentions.2.transformer_blocks.9.attn2.to_k
- up_blocks.0.attentions.2.transformer_blocks.9.attn2.to_q

8. Pipeline Details

The latent dataset creation process is described by Algorithm 1.

Algorithm 1 Dataset Generation

Require: Pretrained floating-point model F **Require:** Small prompt dataset D **Require:** Sample latent N_l per prompt**Require:** Inference steps T

```
1: for each prompt  $p$  in  $D$  do
2:   Initialize random noise  $X_T$ 
3:   Randomly select sample steps  $\{t_1, \dots, t_{N_p}\}$ 
4:   for  $t = T, T - 1, \dots, 1$  do
5:     if  $t \in \{t_1, \dots, t_{N_p}\}$  then
6:       Add  $X_t$  to latent dataset
7:     end if
8:     if  $t == t_{N_p}$  then
9:       break
10:    end if
11:    Predict noise  $\hat{\theta} = F(X_t, t, p)$ 
12:    Update  $X_{t-1} \leftarrow \text{scheduler}(X_t, t, \hat{\theta})$ 
13:  end for
14: end for
```

9. Comparison On Other FID Metrics

We supplement the spatial Fréchet Inception Distance (sFID) results which better capture the spatial relationships. Moreover, we provide the FID-to-FP scores based on the CLIP feature extractor (using clip_vit_b_32 model). Results are shown in Table 8.

Methods	FID-to-FP \downarrow	sFID-to-FP \downarrow	FID-to-FP(clip) \downarrow
PCR	16.3 \dagger /14.2	72.7 \dagger	2.57 \dagger
Ours	10.0	65.4	0.85

Table 8. Comparison on 50 steps PNDM, W4A8, SD v1-4, coco prompt. \dagger denotes reproduced results on our machine.

10. Experiments With Fewer Sampling Steps

We validate our approach with fewer sampling steps. For Stable Diffusion v1-4, we consider PNDM scheduler with 25 steps and UNIPCM scheduler with 10 steps. While for Stable Diffusion XL v1.0, we employ Euler scheduler with 30 steps. Results summarized in Table 9, Table 10, and Table 11 demonstrate that our method still generates high-consistency images under fewer steps.

Method	FID-to-FP↓	LPIPS↓	SSIM↑	PSNR↑
PCR	20.45†	0.53†	0.45†	13.7†
Ours	14.45	0.39	0.54	15.3

Table 9. Comparison on 25 steps PNDM, W4A8, SD v1-4, coco prompt. † denotes reproduced results on our machine.

Method	FID-to-FP↓	FID-to-FP(clip)↓	CLIP score↑
PCR	8.98	3.30	26.41
Ours	8.22	0.62	26.44

Table 10. Comparison on 10 steps UNIPCM, W4A8, SD v1-4, coco prompt.

Method	FID-to-FP↓	LPIPS↓	SSIM↑	Δ CLIPscore↑
Ours	7.33	0.27	0.73	+0.01

Table 11. Results on 30 steps Euler, W4A8, SD XL, coco prompt.

11. More Visualized Results

We provide more visualized results in Fig. 10 and Fig. 11.

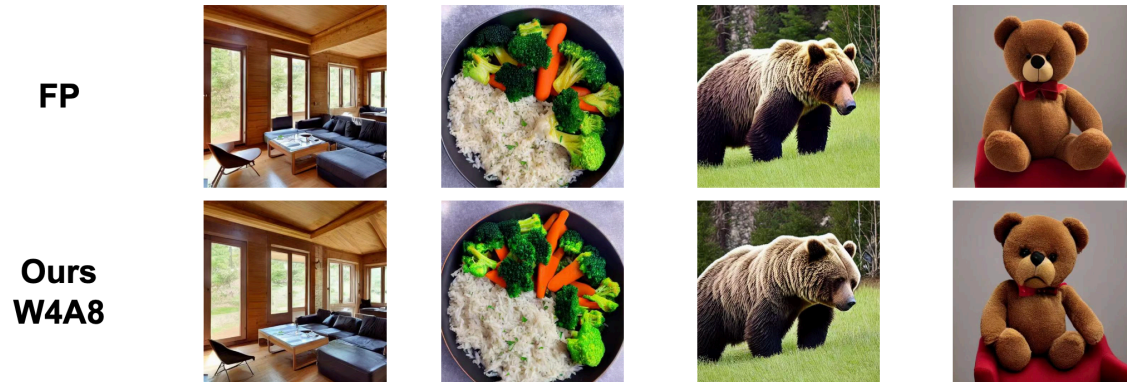


Figure 10. Stable Diffusion v1-4 512x512 generation using DDIM scheduler, 50 steps.



Figure 11. Comparison between the floating-point model and quantized models, using PNDM scheduler, 25 steps.

Notes

Shuaiting Li and Juncan Deng: Work done during an internship at vivo Mobile Communication.

References

1. ^{a, b}Chen H, Zhang Y, Wu S, Wang X, Duan X, Zhou Y, Zhu W. DisenBooth: Identity-Preserving Disentangled Tuning for Subject-Driven Text-to-Image Generation. 2024. Available from: <https://arxiv.org/abs/2305.03374>.
2. ^{a, b, c, d}Ho J, Jain A, Abbeel P (2020). "Denoising diffusion probabilistic models". *Advances in neural information processing systems*. 33: 6840–6851.
3. ^{a, b, c}Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B (2022). "High-resolution image synthesis with latent diffusion models". *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pages 10684–10695.
4. ^{a, b}Ruiz N, Li Y, Jampani V, Pritch Y, Rubinstein M, Aberman K (2023). "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 22500–22510.
5. ^{a, b, c, d}Song J, Meng C, Ermon S (2022). "Denoising Diffusion Implicit Models". *arXiv*. Available from: <https://arxiv.org/abs/2010.02502>.
6. ^{a, b}Song Y, Sohl-Dickstein J, Kingma DP, Kumar A, Ermon S, Poole B (2021). "Score-Based Generative Modeling through Stochastic Differential Equations". *arXiv*. [arXiv:2011.13456 \[cs.LG\]](https://arxiv.org/abs/2011.13456).
7. ^aDhariwal P, Nichol A (2021). "Diffusion models beat gans on image synthesis". *Advances in neural information processing systems*. 34: 8780–8794.
8. ^aRonneberger O, Fischer P, Brox T. 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*. Springer; 2015. p. 234–241.
9. ^aPodell D, English Z, Lacey K, Blattmann A, Dockhorn T, Müller J, Penna J, Rombach R (2023). "SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis". *arXiv*. Available from: <https://arxiv.org/abs/2307.01952>.
10. ^{a, b}He Y, Liu L, Liu J, Wu W, Zhou H, Zhuang B. "Ptq4: Accurate post-training quantization for diffusion models". *Advances in Neural Information Processing Systems*. 36, 2024.
11. ^{a, b, c, d}Li X, Liu Y, Lian L, Yang H, Dong Z, Kang D, Zhang S, Keutzer K. "Q-diffusion: Quantizing diffusion models". *In: Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023:17535–17545.

12. ^{a, b, c, d}Shang Y, Yuan Z, Xie B, Wu B, Yan Y (2023). "Post-training quantization on diffusion models". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023: 1972–1981.
13. [^]Wang C, Wang Z, Xu X, Tang Y, Zhou J, Lu J (2024). "Towards Accurate Post-training Quantization for Diffusion Models". *arXiv*. Available from: <https://arxiv.org/abs/2305.18723>.
14. ^{a, b, c, d, e, f, g, h, i}Tang S, Wang X, Chen H, Guan C, Wu Z, Tang Y, Zhu W (2024). "Post-training Quantization for Text-to-Image Diffusion Models with Progressive Calibration and Activation Relaxing". *arXiv*. [arXiv:2311.06322 \[cs.CV\]](https://arxiv.org/abs/2311.06322).
15. ^{a, b, c}He Y, Liu J, Wu W, Zhou H, Zhuang B (2023). "Efficientdm: Efficient quantization-aware fine-tuning of low-bit diffusion models". *arXiv preprint arXiv:2310.03270*. Available from: <https://arxiv.org/abs/2310.03270>.
16. ^{a, b}Li Y, Gong R, Tan X, Yang Y, Hu P, Zhang Q, Yu F, Wang W, Gu S (2021). "BRECQ: Pushing the Limit of Post-Training Quantization by Block Reconstruction". *arXiv*. [arXiv:2102.05426 \[cs.LG\]](https://arxiv.org/abs/2102.05426).
17. ^{a, b}Nagel M, Amjad RA, Van Baalen M, Louizos C, Blankevoort T. "Up or down? adaptive rounding for post-training quantization." In: *International Conference on Machine Learning*. PMLR; 2020. p. 7197–7206.
18. [^]Nagel M, Fournarakis M, Bondarenko Y, Blankevoort T. Overcoming oscillations in quantization-aware training. In: *International Conference on Machine Learning*. PMLR; 2022. p. 16318–16330.
19. [^]Zhang L, Rao A, Agrawala M (2023). "Adding conditional control to text-to-image diffusion models". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 3836–3847.
20. [^]Wu R, Yang T, Sun L, Zhang Z, Li S, Zhang L. "Seesr: Towards semantics-aware real-world image super-resolution." In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2024. p. 25456–25467.
21. [^]Bao F, Li C, Zhu J, Zhang B (2022). "Analytic-DPM: an Analytic Estimate of the Optimal Reverse Variance in Diffusion Probabilistic Models". *arXiv*. Available from: <https://arxiv.org/abs/2201.06503>.
22. [^]Bao F, Li C, Sun J, Zhu J, Zhang B (2022). "Estimating the Optimal Covariance with Imperfect Mean in Diffusion Probabilistic Models". *arXiv*. [arXiv:2206.07309 \[cs.LG\]](https://arxiv.org/abs/2206.07309).
23. [^]Kong Z, Ping W (2021). "On Fast Sampling of Diffusion Probabilistic Models". *arXiv*. Available from: <https://arxiv.org/abs/2106.00132>.
24. [^]Zhang Q, Tao M, Chen Y (2023). "gDDIM: Generalized denoising diffusion implicit models". *arXiv*. Available from: <https://arxiv.org/abs/2206.05564>.

25. ^{a, b}Watson D, Ho J, Norouzi M, Chan W (2021). "Learning to Efficiently Sample from Diffusion Probabilistic Models". arXiv. [arXiv:2106.03802 \[cs.LG\]](https://arxiv.org/abs/2106.03802).
26. ^ΔJolicœur-Martineau A, Li K, Piché-Taillefer R, Kachman T, Mitliagkas I (2021). "Gotta Go Fast When Generating Data with Score-Based Models". arXiv. Available from: <https://arxiv.org/abs/2105.14080>.
27. ^ΔKim B, Ye JC (2022). "Denoising MCMC for Accelerating Diffusion-Based Generative Models". arXiv. [arXiv:2209.14593 \[cs.LG\]](https://arxiv.org/abs/2209.14593).
28. ^ΔLu C, Zhou Y, Bao F, Chen J, Li C, Zhu J (2022). "Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps". *Advances in Neural Information Processing Systems*. 35: 5775–5787.
29. ^{a, b}Liu L, Ren Y, Lin Z, Zhao Z (2022). "Pseudo Numerical Methods for Diffusion Models on Manifolds". arXiv. [arXiv:2202.09778 \[cs.CV\]](https://arxiv.org/abs/2202.09778).
30. ^ΔZhang Q, Chen Y (2023). "Fast Sampling of Diffusion Models with Exponential Integrator". arXiv. Available from: <https://arxiv.org/abs/2204.13902>.
31. ^ΔChung H, Sim B, Ye JC (2022). "Come-closer-diffuse-faster: Accelerating conditional diffusion models for inverse problems through stochastic contraction". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pages 12413–12422.
32. ^ΔFranzese G, Rossi S, Yang L, Finamore A, Rossi D, Filippone M, Michiardi P (2023). "How much is enough? a study on diffusion times in score-based generative models". *Entropy*. 25 (4): 633.
33. ^ΔZheng H, He P, Chen W, Zhou M (2023). "Truncated Diffusion Probabilistic Models and Diffusion-based Adversarial Auto-Encoders". arXiv. [arXiv:2202.09671 \[stat.ML\]](https://arxiv.org/abs/2202.09671).
34. ^ΔLyu Z, Xu X, Yang C, Lin D, Dai B (2022). "Accelerating diffusion models via early stop of the diffusion process". arXiv. Available from: <https://arxiv.org/abs/2205.12524>.
35. ^ΔLuhman E, Luhman T (2021). "Knowledge Distillation in Iterative Generative Models for Improved Sampling Speed". arXiv. Available from: <https://arxiv.org/abs/2101.02388>.
36. ^ΔSalimans T, Ho J (2022). "Progressive Distillation for Fast Sampling of Diffusion Models". arXiv. [arXiv:2202.00512](https://arxiv.org/abs/2202.00512).
37. ^ΔLam MWY, Wang J, Su D, Yu D (2022). "BDDM: Bilateral Denoising Diffusion Models for Fast and High-Quality Speech Synthesis". arXiv. [arXiv:2203.13508 \[eess.AS\]](https://arxiv.org/abs/2203.13508).
38. ^ΔKingma D, Salimans T, Poole B, Ho J (2021). "Variational diffusion models". *Advances in neural information processing systems*. 34: 21696–21707.

39. ^a ^bNichol AQ, Dhariwal P. "Improved denoising diffusion probabilistic models." In: International conference on machine learning. PMLR; 2021. p. 8162–8171.
40. [△]Gong R, Liu X, Jiang S, Li T, Hu P, Lin J, Yu F, Yan J (2019). "Differentiable soft quantization: Bridging full-precision and low-bit neural networks". *Proceedings of the IEEE/CVF international conference on computer vision*. 2019: 4852--4861.
41. [△]Louizos C, Reisser M, Blankevoort T, Gavves E, Welling M (2018). "Relaxed Quantization for Discretized Neural Networks". *arXiv*. [arXiv:1810.01875 \[cs.LG\]](https://arxiv.org/abs/1810.01875).
42. [△]Jacob B, Kligys S, Chen B, Zhu M, Tang M, Howard A, Adam H, Kalenichenko D. Quantization and training of neural networks for efficient integer-arithmatic-only inference. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018. p. 2704–2713.
43. [△]Zhuang B, Shen C, Tan M, Liu L, Reid I. Towards effective low-bitwidth convolutional neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018. p. 7920–7928.
44. [△]Zhang L, He Y, Lou Z, Ye X, Wang Y, Zhou H (2023). "Root quantization: a self-adaptive supplement STE". *Applied Intelligence*. 53 (6): 6266–6275.
45. [△]Hubara I, Nahshan Y, Hanani Y, Banner R, Soudry D (2020). "Improving Post Training Neural Quantization: Layer-wise Calibration and Integer Programming". *arXiv*. [arXiv:2006.10518 \[cs.LG\]](https://arxiv.org/abs/2006.10518).
46. [△]Wei X, Gong R, Li Y, Liu X, Yu F (2023). "QDrop: Randomly Dropping Quantization for Extremely Low-bit Post-Training Quantization". *arXiv*. [arXiv:2203.05740 \[cs.CV\]](https://arxiv.org/abs/2203.05740).
47. [△]Yang Lin, Zhang T, Sun P, Li Z, Zhou S (2023). "FQ-ViT: Post-Training Quantization for Fully Quantized Vision Transformer". *arXiv*. Available from: <https://arxiv.org/abs/2111.13824>.
48. [△]Karras T, Aittala M, Aila T, Laine S (2022). "Elucidating the design space of diffusion-based generative models". *Advances in Neural Information Processing Systems*. 35: 26565–26577.
49. [△]Nagel M, Fournarakis M, Amjad RA, Bondarenko Y, van Baalen M, Blankevoort T (2021). "A White Paper on Neural Network Quantization". *arXiv*. [arXiv:2106.08295 \[cs.LG\]](https://arxiv.org/abs/2106.08295).
50. ^a ^bKim BK, Song HK, Castells T, Choi S (2023). "Bk-sdm: Architecturally compressed stable diffusion for efficient text-to-image generation". In: *Workshop on Efficient Systems for Foundation Models@ ICML2023*.
51. [△]Luo S, Tan Y, Huang L, Li J, Zhao H (2023). "Latent Consistency Models: Synthesizing High-Resolution Images with Few-Step Inference". *arXiv*. Available from: <https://arxiv.org/abs/2310.04378>.

52. [△]Reddi SJ, Kale S, Kumar S (2019). "On the convergence of adam and beyond". arXiv preprint arXiv:1904.09237. Available from: <https://arxiv.org/abs/1904.09237>.
53. [△]Wilson AC, Roelofs R, Stern M, Srebro N, Recht B (2017). "The marginal value of adaptive gradient methods in machine learning". *Advances in neural information processing systems*. 30.
54. [△]So J, Lee J, Ahn D, Kim H, Park E (2024). "Temporal dynamic quantization for diffusion models". *Advances in Neural Information Processing Systems*. 36.
55. [△]^bYang Y, Dai X, Wang J, Zhang P, Zhang H (2023). "Efficient Quantization Strategies for Latent Diffusion Models". arXiv. Available from: <https://arxiv.org/abs/2312.05431>.
56. [△]Wang H, Shang Y, Yuan Z, Wu J, Yan Y (2024). "QuEST: Low-bit Diffusion Model Quantization via Efficient Selective Finetuning". arXiv. [arXiv:2402.03666 \[cs.CV\]](https://arxiv.org/abs/2402.03666).
57. [△]Barshan E, Brunet M-E, Dziugaite GK. Relatif: Identifying explanatory training samples via relative influence. In: *International Conference on Artificial Intelligence and Statistics*. PMLR; 2020. p. 1899–1909.
58. [△]Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL. Microsoft coco: Comm on objects in context. In: *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*. Springer; 2014. p. 740–755.
59. [△]Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004). "Image quality assessment: From error visibility to structural similarity". *IEEE Transactions on Image Processing*. 13 (4): 600–612.
60. [△]Zhang R, Isola P, Efros AA, Shechtman E, Wang O (2018). "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 586–595.
61. [△]Gonzalez RC, Woods RE. *Digital Image Processing*. Prentice Hall; 2002.
62. [△]Hessel J, Holtzman A, Forbes M, Le Bras R, Choi Y (2021). "Clipscore: A reference-free evaluation metric for image captioning". arXiv preprint arXiv:2104.08718.
63. [△]Teng J, Zheng W, Ding M, Hong W, Wangni J, Yang Z, Tang J (2023). "Relay Diffusion: Unifying diffusion process across resolutions for image synthesis". arXiv. [arXiv:2309.03350 \[cs.CV\]](https://arxiv.org/abs/2309.03350).
64. [△]Chen C, Mo J. IQA-PyTorch: PyTorch Toolbox for Image Quality Assessment [Online]. Available: <https://github.com/chaofengc/IQA-PyTorch>, 2022.

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.