CacheQuant: Comprehensively Accelerated Diffusion Models

Xuewen Liu[1,2], Zhikai Li[1,2]*, Qingyi Gu[1]*
[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
{liuxuewen2023, lizhikai2020, qingyi.gu}@ia.ac.cn

Abstract

Diffusion models have gradually gained prominence in the field of image synthesis, showcasing remarkable generative capabilities. Nevertheless, the slow inference and complex networks, resulting from redundancy at both temporal and structural levels, hinder their low-latency applications in real-world scenarios. Current acceleration methods for diffusion models focus separately on temporal and structural levels. However, independent optimization at each level to further push the acceleration limits results in significant performance degradation. On the other hand, integrating optimizations at both levels can compound the acceleration effects. Unfortunately, we find that the optimizations at these two levels are not entirely orthogonal. Performing separate optimizations and then simply integrating them results in unsatisfactory performance. To tackle this issue, we propose CacheQuant, a novel training-free paradigm that comprehensively accelerates diffusion models by jointly optimizing model caching and quantization techniques. Specifically, we employ a dynamic programming approach to determine the optimal cache schedule, in which the properties of caching and quantization are carefully considered to minimize errors. Additionally, we propose decoupled error correction to further mitigate the coupled and accumulated errors step by step. Experimental results show that CacheQuant achieves a 5.18× speedup and 4× compression for Stable Diffusion on MS-COCO, with only a 0.02 loss in CLIP score. Our code are open-sourced.

1. Introduction

Recently, diffusion models [7, 19, 70] with different frameworks, such as UNet [58] and DiT [54], have come to dominate the field of image synthesis, exhibiting remarkable generative capabilities, as evidenced by the open platform K-sort Arena [32]. Numerous compelling applications have been implemented with diffusion models, including but not limited to image editing [2, 20, 46], image enhancing [10, 24, 61], image-to-image translation [5, 59, 71], text-to-image generation [51, 56, 60, 80] and text-to-3D generation [34, 43, 55]. Despite their appeal, the slow inference and complex networks, resulting from thousands of denoising iterations and billions of model parameters, pose significant challenges to deploy these models in real-world applications. For instance, even on high-performance A6000 GPU, a single inference of Stable Diffusion [57] requires over a minute and consumes 16GB of memory.

To address the above challenges, the research community accelerates diffusion models primarily at two levels: the temporal level and the structural level. For the former,

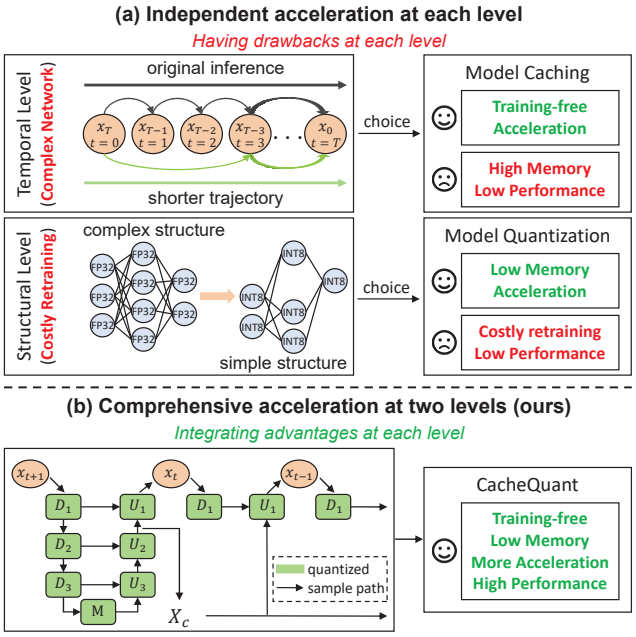

Figure 1. An overview of motivations. (a) The principles and properties of the traditional acceleration methods at each level. (b) Our approach integrates the advantages of model caching and quantization while eliminating their drawbacks, achieving comprehensive acceleration at two levels.

*Corresponding author: {lizhikai2020, qingyi.gu}@ia.ac.cn.

existing methods [26, 41, 47, 62, 69] tackle the slow inference by shortening the denoising trajectory. In contrast, other methods [3, 9, 27, 38, 39] focus on simplifying the network structure to address the complex networks for the latter. Although these methods have achieved significant results, each has its own drawbacks. As shown in Figure 1, temporal-level methods fail to reduce or even exacerbate the complexity of the networks, while structural-level methods require costly retraining processes. Moreover, independent optimization at each level to push the acceleration limits, such as employing a shorter denoising path [45] or further reducing model parameters [72], results in significant performance degradation. Therefore, we seek to develop a comprehensive acceleration solution for diffusion models across both temporal and structural levels, aiming to integrating the advantages of each while eliminating their respective drawbacks. This allows us to push the acceleration boundaries further without compromising performance.

We start by analyzing the properties of methods at each level. At the temporal level, model caching [4, 66, 74] utilize caching mechanisms to eliminate redundant computations at per step without any retraining, which preserve temporal continuity and maintain performance within equivalent computational budgets compared to other methods [8, 37, 52, 69, 81]. At the structural level, quantization-based methods [15, 67] are more efficient in terms of training overhead and hardware friendly compared to other compression-based methods [13, 28, 36, 64, 76]. Thus, we choice model caching and quantization to comprehensively accelerate diffusion models. Moreover, these two techniques exhibit a synergistic relationship: quantization reduces the memory usage increased by caching, while caching alleviates the quantization difficulties caused by temporal redundancy.

Based on the above analysis, theoretically, integrating optimized model caching with quantization methods can yield more substantial acceleration while maintaining controlled performance degradation. However, in practice, we find that the optimizations at these two methods are not entirely orthogonal. Independently optimizing and then simply combining them results in unsatisfactory performance. The underlying issue is that both caching and quantization introduce errors into the original models. These errors couple and accumulate iteratively, further exacerbating their impact on model performance and hindering the effective integration of optimization methods. More specifically, if model quantization is applied directly to caching methods, the quantization error causes significant deviation in the denoising path of the cache. Conversely, if model caching is directly added to quantization methods, the caching error leads to substantial accumulation of quantization errors. In both cases, model performance degrades severely.

To this end, we introduce CacheQuant that solves the above issues by jointly optimizing model caching and quantization techniques. Specifically, we propose Dynamic Programming Schedule (DPS) that models the design of the cache schedule as a dynamic programming problem, aiming to minimize the errors introduced by both caching and quantization. Through optimization, the computational complexity of DPS is significantly reduced, requiring only 8 minutes for LDM on ImageNet. To further mitigate the coupled and accumulated errors, we propose Decoupled Error Correction (DEC), which performs channel-wise correction separately for caching and quantization errors at each time step in a training-free manner. Since the correction for quantization errors can be absorbed into weight quantization, EDC introduces only one additional matrix multiplication and addition during network inference. To the best of our knowledge, this is the first work to investigate diffusion model acceleration at both the temporal and structural levels. We also evaluate the acceleration capabilities of CacheQuant by deploying it on various hardware platforms (GPU, CPU, ARM).

In summary, we make the following contributions:

- We introduce CacheQuant, a novel training-free paradigm that comprehensively accelerates diffusion models with different frameworks at both temporal and structural levels. Our method further pushes accelerated limits and maintains performance.
- CacheQuant minimizes the errors from caching and quantization through DPS and further mitigates these errors via DEC. It achieves a complementary advantage of model caching and quantization techniques by jointly optimizing them.
- We conduct experiments on diffusion models with UNet and DiT frameworks. Extensive experiments demonstrate that our approach outperforms traditional acceleration methods (solver, caching, distillation, pruning, quantization) in both speedup and performance.

2. Related Work

Diffusion models have gradually surpassed GANs [1, 12] and VAEs [18, 22], emerging as the dominant approach in image generation. However, slow inference and complex networks hinder their low-latency applications in real-world scenarios. Current research focuses on two main levels to accelerate diffusion models.

Temporal-level Acceleration methods focus on shortening the sampling trajectory. Some approaches adjust variance schedule [52] or modify denoising equations [69, 81] to remove certain steps entirely. Studies further dive into the fast solver of SDE [8, 37] or ODE [41, 42] to create efficient sampling steps. Others [25, 68, 82] conduct parallel sampling to speed up inference. In contrast, cache-based methods [4, 45, 74] reduce the inference path at each step by caching the output of block.

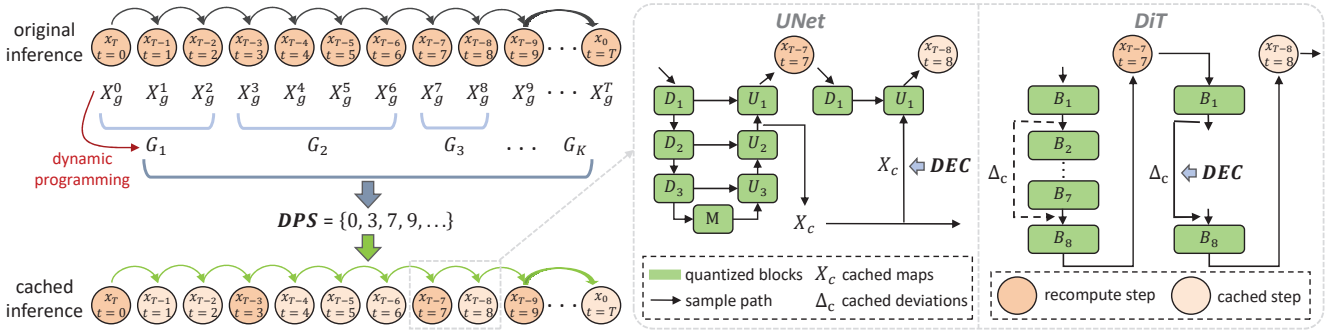

Figure 2. An overview of CacheQuant. **DPS** selects the optimal cache schedule and **DEC** mitigates the coupled and accumulated errors.

Structural-level Acceleration methods concentrate on simplifying the network architecture. Previous studies redesign lightweight network [28] or incorporate frequency priors into model design [76]. OMS-DPM [36] creates a diffusion model zoo to select different models at various steps. Some methods [44, 47, 62] simplify model architecture with distillation technology. On the other hand, pruning-based methods [3, 9, 79] reduce the number of model parameters, while quantization-based approaches [27, 29, 30, 38, 39] achieve model compression by utilizing lower bit-width representations.

3. Preliminary

In the following, we present the two key techniques employed in our work.

Model Caching accelerates inference by storing intermediate network outputs. For diffusion models with temporal networks, this technique leverages the inherent similarity of feature maps between adjacent denoising steps to eliminate temporal computational redundancies. For example, we cache the output activation X_g^t of block at step t as X_c^t. When inferring at step $t+1$, X_c^t is reused in place of the ground truth X_g^{t+1}, thereby eliminating the computations of X_g^{t+1}. Existing methods implement caching at various network layers. Deepcache [45] and Faster Diffusion [26] cache the output feature maps of upsampling blocks and UNet encoder, respectively. Block Caching [74] further adaptively caches all blocks. Δ-DiT [4] selectively caches blocks based on their impact at different denoising stages. Besides, this mechanism can be extended to cover more steps, with the cached features X_c^t calculated once and reused in the consecutive $N-1$ steps:

$$X_c^t \rightarrow X_g^{t+1} \Rightarrow X_c^t \rightarrow \{X_g^{t+1}, X_g^{t+2}, ..., X_g^{t+N}\} \quad (1)$$

Determining the cache schedule, i.e., where to recompute cached features, directly impacts model performance. For instance, in a diffusion model with T steps, when the cache frequency N is fixed, a uniform cache schedule is represented by $\{0, N, 2N, ..., (T/N-1)N\}$, and the corresponding cached features are $\{X_c^0, X_c^N, X_c^{2N}, ..., X_c^{(T/N-1)N}\}$. To reduce the errors introduced by caching, previous methods have developed various cache schedules. [4, 45, 74] determine the schedule by conducting experiments and tuning hyperparameters, while [26] directly specifies the schedule manually. In this work, as shown in Figure 2, for the UNet framework, we cache the outputs of a single upsampling block as the cached features X_c, similar to the approach used in DeepCache [45]. For the DiT framework, we cache the deviations Δ_c between two blocks as the cached features, similar to Δ-DiT [4]. We model the selection of schedule as a dynamic programming problem and our goal is to minimize the errors introduced by caching and quantization, thereby achieving the optimal schedule.

Model Quantization represents model parameters and activations with low-precision integer values, compressing model size and accelerating inference. Given a floating-point vector $\mathbf{x}$, it can be uniformly quantized as follows:

$$\hat{\mathbf{x}} = clip\left(\lfloor \mathbf{x}/s \rceil + z, 0, 2^b - 1\right) \quad (2)$$

where $\hat{\mathbf{x}}$ is the quantized value, scale factor s and zero point z are quantization parameters, $\lfloor \cdot \rceil$ denotes rounding function, and the bit-width b determines the range of clipping function $clip(\cdot)$. Depending on whether fine-tuning of the model is necessitated, this technique can be categorized into two approaches: post-training quantization (PTQ) and quantization-aware training (QAT). Initial PTQ methods [31, 48] calibrate the quantization parameters in a training-free manner using a small calibration set. Subsequently, reconstruction-based methods [33, 49, 73] employ backpropagation to optimize quantization parameters. On the other hand, QAT methods [11, 50] entail fine-tuning the model weights on the original dataset. While this approach preserves performance, it requires significant time cost and computational resources. Notably, all existing quantization methods for diffusion models are either reconstruction-based [27, 39] or fine-tuning-based [14, 38]. In stark con-

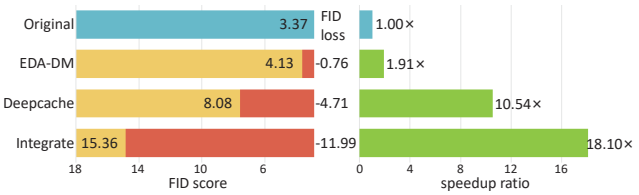

Figure 3. Performance and acceleration of different optimization strategies. EDA-DM and Deepcache are optimization methods for model quantization and caching, respectively.

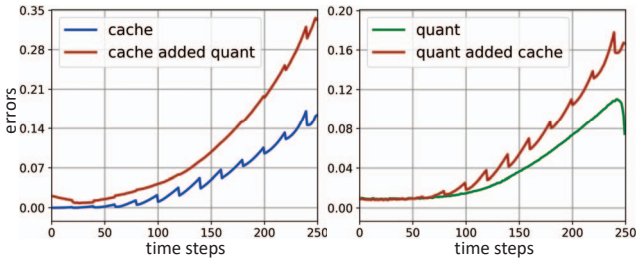

Figure 4. Output errors of network at each time step.

trast, we propose a PTQ correction strategy to mitigate errors, preserving the advantages of being training-free.

4. CacheQuant

In this section, we introduce **CacheQuant**, a novel training-free paradigm that jointly optimizes caching and quantization techniques to comprehensively accelerate diffusion models. We start by analyzing the challenges of comprehensive acceleration in Sec 4.1, followed by our proposed methods to address these challenges in Sec 4.2 and Sec 4.3. The overview of CacheQuant is shown in Figure 2.

4.1. Challenges of Comprehensive Acceleration

Leveraging model caching and quantization enables comprehensive acceleration of diffusion models. Unfortunately, we find that, although independently optimizing and then simply integrating these two methods yields more noticeable acceleration, the model performance remains far from satisfactory. To analyze the above issues, we conduct experiments for LDM on ImageNet. As shown in Figure 3, when the original model is independently optimized through model quantization and caching, the FID score drops 0.76 and 4.71, respectively. However, simply integrating the two optimizations results in an FID loss of 11.99. The underlying issue is that both caching and quantization inherently introduce errors into the original models. These errors couple and accumulate iteratively, further exacerbating their impact on model performance and hindering the effective combination of optimization methods. As shown in Figure 4, if model quantization is directly added to caching methods, the quantization error causes significant deviation in the denoising path of the cache. Conversely, if model caching is applied directly to quantization methods, the caching error leads to substantial accumulation of quantization errors. This suggests that the optimizations of these two methods are not entirely orthogonal, highlighting the need for joint optimization.

4.2. Dynamic Programming Schedule

We illustrate our method with the UNet framework as an example. To minimize errors, we analyze the feature maps $X = \{X_g^0, X_g^1, ..., X_g^{T-1}\}$ at all steps to guide the selection of the cache schedule, reframing the problem as one of grouping ordered samples.

For a diffusion model with T steps and a cache frequency of N, all feature maps are divided into $K = T/N$ groups, forming a grouping set $G = \{G_1, G_2, ..., G_K\}$. Time steps within the same group share the same cached features. To achieve optimal grouping, we propose **D**ynamic **P**rogramming **S**chedule (**DPS**):

First, we consider two constraints: 1) Each feature map belongs to exactly one group, ensuring that no step is duplicated or omitted; 2) The order of the feature maps within each group must remain unchanged to preserve the temporal consistency of the denoising process. Specified as:

$$\begin{aligned} G_1 &= \{X_g^0, X_g^1, ..., X_g^{s_1-1}\}, \\ G_2 &= \{X_g^{s_1}, X_g^{s_1+1}, ..., X_g^{s_2-1}\}, \\ &... \\ G_K &= \{X_g^{s_{K-1}}, X_g^{s_{K-1}+1}, ..., X_g^{T-1}\} \end{aligned} \tag{3}$$

where the time step of the first element $X_g^{s_*}$ of each group denotes the dividing point, which forms the cache schedule.

Second, we define the intra-group error as $D_k(i, j)$, which represents the error introduced by caching and quantization when steps i to j are assigned to the k-th group. Notably, since X_g^i is cached and replaces $\{X_g^{i+1}, ..., X_g^j\}$, the error is calculated by sequentially comparing X_g^i with $\{X_g^{i+1}, ..., X_g^j\}$ and summing the resulting differences. Additionally, quantization error arises from the absolute numerical differences between feature maps, and is therefore measured using the $L1$ norm. Consequently, the mathematical formulation of $D_k(i, j)$ is as follows:

$$D_k(i, j) = \sum_{t=i+1}^{j} \|X_g^i - X_g^t\|_1 \tag{4}$$

Third, we denote the partitioning of T steps into K groups as $b(T, K)$. The grouping loss function is defined as $L[b(T, K)] = \sum_{k=1}^{K} D_k(i, j)$. The solution for K-th optimal group can be expressed as:

$$L[b(T, K)] = L[b(s-1, K-1)] + D(s, T) \tag{5}$$

$$M(T, K) = \min_{K \le s \le T} L[b(T, K)] \tag{6}$$

Algorithm 1 : Dynamic Programming Schedule

Input: all steps T, cache frequency N
Output: optimal schedule DPS

$K = T/N$ ▷ init number of groups
$M = S = 0_{T\times K}$ ▷ init loss M and dividing point S
for $t = 1$ to T **do**
  $M(t,1) = D(1,t)$ ▷ calculate boundary conditions
  $S(t,1) = t$
end for
for $k = 1$ to K **do**
  for $t = k$ to T **do**
    clear L = [ ]
    for $s = k$ to t **do**
      limit $\frac{1}{2}N \leq t - s \leq 2N$ ▷ optimization limits
      $L[b(t,k)] = M(s-1, k-1) + D(s,t)$
      append $L[b(t,k)]$ to L
    end for
    $M(t,k) = \min(L)$ ▷ store mininum loss
    $S(t,k) = argmin_s(L)$ ▷ store dividing point
  end for
end for
t = T
for $k = K$ to 1 **do**
  $s_k = S(t,k)$ ▷ dividing point for k-th group
  append s_k to DPS ▷ store for optimal schedule
  $t = s_k - 1$ ▷ number of remaining steps
end for

where s denotes the dividing point, $K \leq s \leq T$ ensures that each feature map belongs to exactly one group, $M(T,K)$ minimizes the grouping loss to obtain the K-th optimal group $G_K = \{X_g^s, X_g^{s+1}, ..., X_g^{T-1}\}$. Briefly, the above formula can be reformulated as:

$$M(T,K) = \min_{K\leq s\leq T}\{M(s-1,K-1) + D(s,T)\} \quad (7)$$

As can be seen, the K-th optimal group is based on the assignment of the $s-1$ feature maps to $K-1$ optimal groups. Thus, all optimal groups can be iteratively solved based on the boundary conditions $M(t,1)$. The workflow of DPS is shown in Algorithm 1.

However, due to the nested loops, the computation of DPS is complex, resulting in slow convergence. We consider practical grouping factors, optimizing the group length to no more than $2N$ and no less than $\frac{1}{2}N$. This significantly reduces the computational complexity of DPS. For instance, the solution time for LDM with 250 steps on ImageNet is reduced from 4 hours to 8 minutes. Finally, DPS efficiently obtains the optimal schedule that minimizes both caching and quantization errors.

4.3. Decoupled Error Correction

To further mitigate the coupled and accumulated errors while maintaining acceleration efficiency, we explore a training-free solution. We begin by analyzing the outputs of block receiving cached features under different conditions:

$$O_g = X_g W_g \ , \ O_c = X_c W_g \ , \ O_{cq} = X_{cq} W_q \quad (8)$$

where $O \in \mathbb{R}^{B\times C^o}$, $X \in \mathbb{R}^{B\times C^i}$, and $W \in \mathbb{R}^{C^i\times C^o}$ denote the output, activation, and weight, respectively. The B, C^i and C^o denote the batch size, in-channel dimension, and out-channel dimension, respectively. The subscripts g, c, and q represent the different conditions: ground truth, cached, and quantized, respectively. We observe a strong correlation in channel-wise granularity between O_g and O_{cq}, as shown in Figure 5(a). Therefore, we can calculate correction parameters along the out-channel dimension for O_{cq}, aiming to reduce their error relative to O_g. And we correct at each step to alleviate accumulated errors. The corrected formula is as follows:

$$O_g = a \cdot O_{cq} + b \quad (9)$$

where $a \in \mathbb{R}^{C^o}$ and $b \in \mathbb{R}^{C^o}$ are correction parameters. We solve them using the least squares method. For instance, the correction parameters for k-th channel are as follow:

$$\begin{aligned} a_k &= \frac{\mathrm{Cov}(O_{cq(:,k)}, O_{g(:,k)})}{\mathrm{Var}(O_{cq(:,k)})} \\ b_k &= \bar{O}_{g(:,k)} - a_k * \bar{O}_{cq(:,k)} \end{aligned} \quad (10)$$

Here, $\bar{O}_{g(:,k)}$ and $\bar{O}_{cq(:,k)}$ represent the mean of the k-th out-channel. When adjusting O_{cq} using the correction parameters, although the mean error is eliminated, the variance of the error remains large, resulting in ineffective correction, as shown in Figure 5(b)(1) and (3). The underlying issue is that directly correcting O_{cq} cannot efficiently eliminate caching errors, as these errors fundamentally arise from the difference between X_g and X_c.

To address this, we propose **D**ecoupled **E**rror **C**orrection (**DEC**) that decouples error E_o introduced by caching and quantization into cache error E_c and quantization error E_q:

$$\begin{aligned} E_o &= X_g W_g - X_{cq} W_q = O_g - O_{cq} \\ E_c &= X_g W_g - X_c W_g = O_g - O_c \\ E_q &= X_c W_g - X_{cq} W_q = O_c - O_{cq} \end{aligned} \quad (11)$$

Similar to Eq. 9, we correct X_c to reduce E_c and correct O_{cq} to reduce E_q:

$$\begin{aligned} X_g &= a_1 \cdot X_c + b_1 \\ O_c &= a_2 \cdot O_{cq} + b_2 \end{aligned} \quad (12)$$

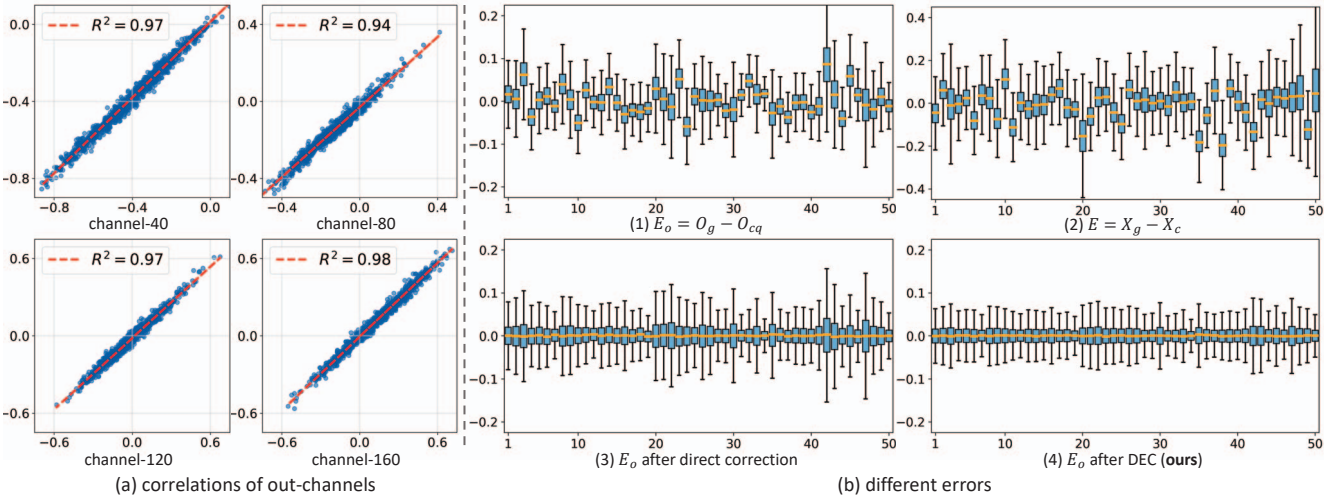

Figure 5. (a) Correlations between the different out-channels of O_g and O_{cq}. (b) Box plots visualize the mean and variance of different errors. Data comes from steps $t = 192$ and $t = 210$ for LDM on ImageNet, which are assigned to the same group by the DPS.

where the correction parameters $(a_1, b_1) \in \mathbb{R}^{C^i}$, $(a_2, b_2) \in \mathbb{R}^{C^o}$ are solved like Eq. 10. Experimental results demonstrate that DEC not only eliminates the mean error but also efficiently reduces error variance (shown in Figure 5(b)(4)), significantly improving performance. For instance, compared to direct correction, DEC enhances the FID score for LDM on ImageNet by 0.91.

We also provide a theoretical proof that DEC outperforms direct correction. Through equivalent transformations, the two correction methods express O_{cq} as:

$$O_{cq} = X_{cq}W_q = \frac{X_g W_g}{a} - \frac{b}{a} \tag{13}$$

$$O_{cq} = X_{cq}W_q = \frac{X_g}{a_1} \cdot \frac{W_g}{a_2} - \frac{b_1}{a_1} \cdot \frac{W_g}{a_2} - \frac{b_2}{a_2} \tag{14}$$

As can be seen, compared to direct correction on the out-channels, DEC adjusts the mean and variance across both in-channels and out-channels. The two expressions are equivalent when assuming $a_1 = \mathbf{1}$ and $b_1 = \mathbf{0}$, which implies that the mean error between X_g and X_c is zero and the variance is negligible. However, as shown in Figure 5(b)(2), this assumption clearly does not hold, making DEC the more reasonable approach. Additionally, by incorporating (a_2, b_2) into weight quantization, DEC introduces only one additional matrix multiplication and addition during network inference.

5. Experiment

5.1. Experimental Setup

Models, Datasets, and Metrics. To demonstrate the effectiveness of our method, we conduct evaluations on DDPM, LDM, and Stable Diffusion [57, 69] with UNet framework and DiT-XL/2 [53] with DiT framework. We present experimental results on six commonly used datasets: CIFAR-10, LSUN-Bedroom, LSUN-Church, ImageNet, MS-COCO, and PartiPrompt [6, 23, 35, 77, 78]. Following previous works [4, 38, 45, 75], we utilize 5k validation set from MS-COCO and 1.63k captions from PartiPrompt as prompts for Stable Diffusion, and generate 10k images for DiT-XL/2. For other tasks, we generate 50k images to assess the generation quality. The evaluation metrics include FID, IS, and CLIP Score (on ViT-g/14) [16, 17, 63]. Besides, we employ Bops ($Bops = MACs \times b_w \times b_x$), Speed Up (on GPU), and Model Size (MB) to visualize acceleration and compression performance.

Caching and Quantization Settings. Our method uses Deepcache [45], Δ-DiT [4], and EDA-DM [39] as the baseline. We select the last 3/1/1-th blocks as cached blocks for DDPM, LDM, and Stable Diffusion models, respectively, and maintain Middle Blocks ($I = 7$ and $N_c = 14$ in [4]) as cached object for DiT-XL/2. For model quantization, we utilize the temporal quantizer from [38] to quantize all layers, with channel-wise quantization for weights and layer-wise quantization for activations, as this is the common practice. Additionally, CacheQuant seamlessly integrates with quantization reconstruction to enhance performance.

5.2. Comparison with Temporal-level Methods

The mainstream temporal-level acceleration methods for diffusion models include model caching and fast solvers. We first compare CacheQuant with cache-based methods (Deepcache [45], Δ-DiT [4]), as reported in Table 2 and 3. Our method achieves comparable or even superior performance to cache-based methods, while delivering a 4× model compression and significant speedup improvement. Furthermore, CacheQuant demonstrates robustness to cache

Table 1. Unconditional generation quality on CIFAR-10, LSUN-Church, and LSUN-Bedroom using DDPM, LDM-8, and LDM-4, respectively. The notion 'WxAy' is employed to represent the bit-widths of weights 'W' and activations 'A'.

Dataset	Method	Bops ↓	Speed ↑	Size ↓	Retrain	FID ↓
CIFAR 32 × 32 T = 100	DDPM	6.21T	1.00×	143.0	✗	4.19
	Deepcache-N=3	3.62T	1.61×	1.00×	✗	4.70
	Ours-N=3 (W8A8)	0.23T	3.57×	3.98×	✗	4.61
	Deepcache-N=5	3.08T	1.85×	1.00×	✗	5.73
	Ours-N=5 (W8A8)	0.19T	4.11×	3.98×	✗	5.28
	Deepcache-N=10	2.69T	2.07×	1.00×	✗	9.74
	Ours-N=10 (W8A8)	0.17T	4.62×	3.98×	✗	8.19
LSUN-Church 256 × 256 T = 100 eta=0.0	LDM-8	19.10T	1.00×	1514.5	✗	3.99
	Deepcache-N=2	10.07T	1.86×	1.00×	✗	4.43
	Ours-N=2 (W8A8)	0.63T	3.10×	3.99×	✗	3.52
	Deepcache-N=3	7.18T	2.54×	1.00×	✗	5.10
	Ours-N=3 (W8A8)	0.45T	4.14×	3.99×	✗	3.66
	Deepcache-N=5	4.65T	3.67×	1.00×	✗	6.74
	Ours-N=5 (W8A8)	0.29T	5.98×	3.99×	✗	3.71
LSUN-Bedroom 256 × 256 T = 100 eta=0.0	LDM-4	98.36T	1.00×	1317.4	✗	10.49
	Deepcache-N=2	52.23T	1.79×	1.00×	✗	11.21
	Ours-N=2 (W8A8)	3.26T	3.05×	3.99×	✗	8.85
	Deepcache-N=3	37.49T	2.68×	1.00×	✗	11.86
	Ours-N=3 (W8A8)	2.34T	4.72×	3.99×	✗	9.27
	Deepcache-N=5	24.59T	4.08×	1.00×	✗	14.28
	Ours-N=5 (W8A8)	1.54T	7.06×	3.99×	✗	10.29

frequency, as evidenced by its consistent outperformance in Table 1. At smaller cache frequency, our method even achieves lower FID score than the full-precision models. This is a common occurrence observed in prior works [27, 38, 39], suggesting that the generated image quality is comparable to that produced by the full-precision models. We demonstrate the superiority of CacheQuant over fast solvers by comparing it with the PLMS solver [37]. As shown in Table 4, using Stable Diffusion with 50 PLMS steps as a baseline, reducing the PLMS steps to 20 severely degrades performance. In contrast, our method maintains performance while achieving a 4× model compression and more than a 5× speedup.

5.3. Comparison with Structural-level Methods

The structural-level acceleration methods primarily include model quantization, pruning, and distillation. We compare CacheQuant with quantization-based methods (EDA-DM [39]) in Table 2. At 8-bit precision, CacheQuant with N=5 cache frequency outperforms EDA-DM (FID 4.03 vs 4.13). Importantly, CacheQuant avoids costly retraining and achieves significant acceleration improvements (Speed 7.87× vs 1.91×). As the bit width decreases, EDA-DM with the 4-bit precision achieves an 8× compression and a 3.35× speedup. However, the FID score significantly drops to 44.12. In stark contrast, CacheQuant combined with reconstruction maintains an FID score of 12.65, achieving 8× compression and 18.06× speedup. We conduct a comparison with pruning-based method in Table 2. As can be seen, CacheQuant surpasses Diff-Pruning [9] in terms of efficiency, performance, acceleration, and compression. We also compare CacheQuant with distillation-based methods, including Small SD [40] and BK-SDM [21], which are developed by retraining on LAION [65] dataset, using Stable Diffusion as the baseline. As reported in Table 4, our method achieves superior performance and faster acceleration compared to these approaches.

Table 2. Class-conditional generation quality on ImageNet using LDM-4 with UNet framework, employing 250 DDIM steps.

	ImageNet 256 × 256					
Method	Bops ↓	Speed ↑	Size ↓	Retrain	FID ↓	IS ↑
LDM-4	102.22T	1.00×	1824.6	✗	3.37	204.56
EDA-DM (W8A8)	6.39T	1.91×	457.1	✓	4.13	186.78
EDA-DM (W4A8)	3.19T	1.91×	229.2	✓	4.79	176.43
EDA-DM (W4A4)	1.61T	3.35×	229.2	✓	44.12	62.04
Diff-Pruning	53.98T	1.51×	757.7	✓	9.27	214.42
Deepcache-N=5	24.06T	4.12×	1824.6	✗	3.79	199.58
ours-N=5 (W8A8)	1.50T	7.87×	457.1	✗	4.03	193.90
ours-N=5 (W4A8)	0.75T	7.87×	229.2	✓	6.26	168.46
Deepcache-N=10	14.31T	6.96×	1824.6	✗	4.60	188.81
ours-N=10 (W8A8)	0.89T	12.20×	457.1	✗	4.68	184.38
ours-N=10 (W4A8)	0.45T	12.20×	229.2	✓	6.90	158.27
Deepcache-N=15	11.17T	9.19×	1824.6	✗	5.91	175.50
ours-N=15 (W8A8)	0.70T	16.55×	457.1	✗	5.51	174.81
ours-N=15 (W4A8)	0.35T	16.55×	229.2	✓	9.40	139.64
Deepcache-N=20	9.62T	10.54×	1824.6	✗	8.08	159.27
ours-N=20 (W8A8)	0.60T	18.06×	457.1	✗	7.21	160.68
ours-N=20 (W4A8)	0.30T	18.06×	229.2	✓	12.65	124.13

Table 3. Class-conditional generation quality on ImageNet using DiT-XL/2 with DiT framework, employing 50 DDIM steps.

	ImageNet 256 × 256					
Method	Bops ↓	Speed ↑	Size ↓	Retrain	FID ↓	IS ↑
DiT-XL/2	117.18T	1.00×	2575.42	✗	6.02	246.24
Δ-DiT-N=2	87.88T	1.31×	2575.42	✗	9.06	205.95
ours-N=2 (W8A8)	5.49T	2.72×	645.72	✗	7.86	213.08
Δ-DiT-N=3	75.53T	1.51×	2575.42	✗	13.75	171.68
ours-N=3 (W8A8)	4.72T	3.08×	645.72	✗	12.42	173.17

5.4. Analysis

Ablation Study. To assess the efficacy of each proposed component, we conduct a comprehensive ablation study on ImageNet, as presented in Table 5. We add 8-bit quantization to DeepCache with N=20 cache frequency as a baseline, resulting in an increase of the FID score to 15.36. With DPS introduced to select the optimal cache schedule, the FID score significantly improves to 8.47. This demonstrates that DPS effectively minimizes errors caused by caching and quantization. By further incorporating EDC that corrects decoupled errors in a training-free manner, the FID score is improved to 7.21. Moreover, our method, combined with reconstruction approach, further enhances performance, notably increasing the IS score to 180.42.

Table 4. Text-conditional generation quality on PartiPrompt and MS-COCO using Stable Diffusion with UNet framework.

Method	Prec.(W/A)	Bops ↓	Size ↓	Retrain	MS-COCO Speed ↑	MS-COCO FID ↓	MS-COCO IS ↑	MS-COCO CLIP Score ↑	PartiPrompts Speed ↑	PartiPrompts CLIP Score ↑
PLMS - 50 steps	32/32	346.96T	4112.5	✗	1.00×	25.59	41.02	26.89	1.00×	27.23
PLMS - 20 steps	32/32	138.78T	4112.5	✗	2.43×	24.70	39.72	26.74	2.46×	27.04
Deepcache - N=10	32/32	133.58T	4112.5	✗	3.52×	23.45	39.21	26.71	3.56×	26.86
Small SD	16/16	57.28T	1158.6	✓	2.93×	29.43	32.55	26.02	2.80×	25.99
BK-SDM - Base	16/16	57.30T	1160.2	✓	2.79×	28.47	36.79	26.21	2.66×	26.53
BK-SDM - Small	16/16	55.75T	966.6	✓	2.88×	30.10	35.57	25.22	2.80×	25.76
BK-SDM - Tiny	16/16	52.50T	648.5	✓	2.92×	31.82	32.82	25.10	2.87×	25.51
Ours - N=5	8/8	8.44T	1029.7	✗	**5.18**×	23.74	**39.81**	**26.87**	**5.20**×	27.13
Ours - N=5	4/8	**4.27T**	**515.9**	✓	**5.18**×	**23.23**	39.41	26.77	**5.20**×	**27.15**

Table 5. The effect of different components proposed in the paper.

Method	Prec.(W/A)	Retrain	FID ↓	IS ↑
Deepcache-N=20	32/32	✗	8.08	159.27
baseline	8/8	✗	15.36	121.78
+DPS	8/8	✗	8.47	154.07
+DPS+EDC	8/8	✗	7.21	160.68
+DPS+EDC+Recon	8/8	✓	6.34	180.42

Acceleration vs. Performance Tradeoff. We investigate the tradeoff between acceleration and performance for various approaches, as presented in Figure 6. As speedup ratio increases, traditional acceleration methods, such as cache (Deepcache), quantization (EDA-DM), and solvers (PLMS), suffer from significant performance degradation. In sharp contrast, our method comprehensively accelerates diffusion models at two levels, further pushing acceleration limits while maintaining performance.

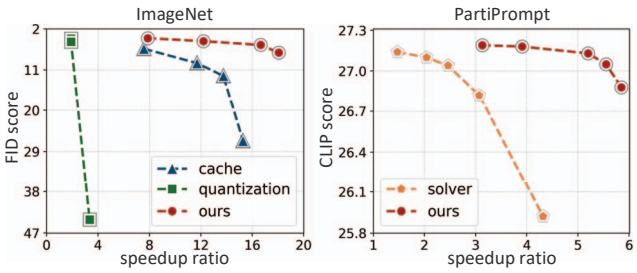

Figure 6. An overview of the acceleration-vs-performance tradeoff across various approaches. Data from LDM-4 on ImageNet and Stable Diffusion on PartiPrompt.

Study on Efficiency. As shown in Figure 7, our method significantly outperforms traditional approaches in efficiency. For instance, compression-based methods require over 10 hours of GPU runtime, while distillation-based methods demand more than 10 days to complete.

Deployment of accelerated models. To evaluate the real-world speedup, we deploy our accelerated diffusion models on various hardware platforms. As shown in Figure 8, the acceleration on GPU is significantly more pronounced compared to CPU and ARM. Our method achieves a 5× GPU speedup of Stable Diffusion on MS-COCO, significantly facilitating its applications in real-world scenarios.

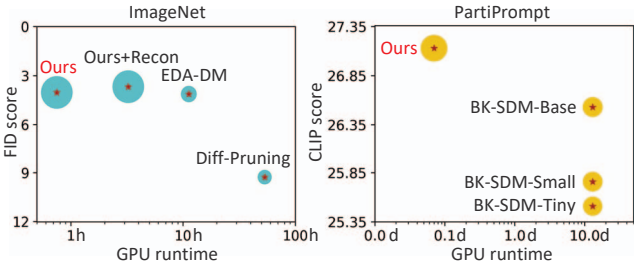

Figure 7. Comparison of the efficiency across various approaches. Data from LDM-4 on ImageNet and Stable Diffusion on PartiPrompt. The circle size denotes speedup ratio.

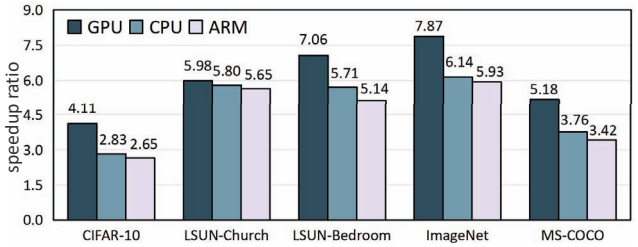

Figure 8. Speedup ratio of diffusion models with 8-bit precision and N=5 cache frequency.

6. Conclusion

In this paper, we introduce CacheQuant, a novel training-free paradigm that comprehensively accelerates diffusion models at both temporal and structural levels. To address the non-orthogonality of optimization, we propose DPS that selects the optimal cache schedule to minimize errors caused by caching and quantization. Additionally, we employ DEC to further mitigate the coupled and accumulated errors without any retraining. Empirical evaluations on several datasets and different model frameworks demonstrate that CacheQuant outperforms traditional acceleration methods. Importantly, the proposed paradigm pushes the boundaries of diffusion model acceleration while maintaining performance, thereby offering a new perspective in the field.

7. Acknowledge

This work is supported in part by the National Science and Technology Major Project of China under Grant 2022ZD0119402; in part by the National Natural Science Foundation of China under Grant 62276255.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[2] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18208–18218, 2022.

[3] Thibault Castells, Hyoung-Kyu Song, Bo-Kyeong Kim, and Shinkook Choi. Ld-pruner: Efficient pruning of latent diffusion models using task-agnostic insights. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 821–830, 2024.

[4] Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. Delta-dit: A training-free acceleration method tailored for diffusion transformers. *arXiv preprint arXiv:2406.01125*, 2024.

[5] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. *arXiv preprint arXiv:2108.02938*, 2021.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[8] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. Genie: Higher-order denoising diffusion solvers. *Advances in Neural Information Processing Systems*, 35:30150–30166, 2022.

[9] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

[10] Sicheng Gao, Xuhui Liu, Bohan Zeng, Sheng Xu, Yanjing Li, Xiaoyan Luo, Jianzhuang Liu, Xiantong Zhen, and Baochang Zhang. Implicit diffusion models for continuous super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10021–10030, 2023.

[11] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4852–4861, 2019.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[13] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Joshua M Susskind. Boot: Data-free distillation of denoising diffusion models with bootstrapping. In *ICML 2023 Workshop on Structured Probabilistic Inference $\{\backslash\&\}$ Generative Modeling*, 2023.

[14] Yefei He, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Efficientdm: Efficient quantization-aware fine-tuning of low-bit diffusion models. *arXiv preprint arXiv:2310.03270*, 2023.

[15] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptqd: Accurate post-training quantization for diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

[16] 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.

[17] 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. *Advances in neural information processing systems*, 30, 2017.

[18] Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3, 2017.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[20] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6007–6017, 2023.

[21] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. Bk-sdm: Architecturally compressed stable diffusion for efficient text-to-image generation. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*, 2023.

[22] Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[24] Haoying Li, Yifan Yang, Meng Chang, Shiqi Chen, Huajun Feng, Zhihai Xu, Qi Li, and Yueting Chen. Srdiff: Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, 479:47–59, 2022.

[25] Muyang Li, Tianle Cai, Jiaxin Cao, Qinsheng Zhang, Han Cai, Junjie Bai, Yangqing Jia, Kai Li, and Song Han. Distrifusion: Distributed parallel inference for high-resolution diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7183–7193, 2024.

[26] Senmao Li, Taihang Hu, Fahad Shahbaz Khan, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, and Jian

Yang. Faster diffusion: Rethinking the role of unet encoder in diffusion models. *arXiv preprint arXiv:2312.09608*, 2023.

[27] Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17535–17545, 2023.

[28] 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. *Advances in Neural Information Processing Systems*, 36, 2024.

[29] Zhikai Li and Qingyi Gu. I-vit: Integer-only quantization for efficient vision transformer inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17065–17075, 2023.

[30] Zhikai Li, Liping Ma, Mengjuan Chen, Junrui Xiao, and Qingyi Gu. Patch similarity aware data-free quantization for vision transformers. In *European Conference on Computer Vision*, pages 154–170, 2022.

[31] Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. Repq-vit: Scale reparameterization for post-training quantization of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17227–17236, 2023.

[32] Zhikai Li, Xuewen Liu, Dongrong Fu, Jianquan Li, Qingyi Gu, Kurt Keutzer, and Zhen Dong. K-sort arena: Efficient and reliable benchmarking for generative models via k-wise human preferences. *arXiv preprint arXiv:2408.14468*, 2024.

[33] Zhikai Li, Xuewen Liu, Jing Zhang, and Qingyi Gu. Repquant: Towards accurate post-training quantization of large transformer models via scale reparameterization. *arXiv preprint arXiv:2402.05628*, 2024.

[34] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023.

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[36] Enshu Liu, Xuefei Ning, Zinan Lin, Huazhong Yang, and Yu Wang. Oms-dpm: Optimizing the model schedule for diffusion probabilistic models. In *International Conference on Machine Learning*, pages 21915–21936. PMLR, 2023.

[37] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

[38] Xuewen Liu, Zhikai Li, and Qingyi Gu. Dilatequant: Accurate and efficient diffusion quantization via weight dilation. *arXiv preprint arXiv:2409.14307*, 2024.

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

[40] Chengqiang Lu, Jianwei Zhang, Yunfei Chu, Zhengyu Chen, Jingren Zhou, Fei Wu, Haiqing Chen, and Hongxia Yang. Knowledge distillation of transformer-based language models revisited. *ArXiv*, abs/2206.14366, 2022.

[41] 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. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

[42] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

[43] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2837–2845, 2021.

[44] 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.

[45] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15762–15772, 2024.

[46] 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. *arXiv preprint arXiv:2108.01073*, 2021.

[47] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023.

[48] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019.

[49] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020.

[50] Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In *International Conference on Machine Learning*, pages 16318–16330. PMLR, 2022.

[51] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[52] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[53] William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

[54] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[55] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[56] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1 (2):3, 2022.

[57] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[58] 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*, pages 234–241. Springer, 2015.

[59] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022.

[60] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[61] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE transactions on pattern analysis and machine intelligence*, 45(4):4713–4726, 2022.

[62] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[63] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

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

[65] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

[66] Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. Fora: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425*, 2024.

[67] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1972–1981, 2023.

[68] Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

[69] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[70] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[71] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1921–1930, 2023.

[72] 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*, 2024.

[73] Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. *arXiv preprint arXiv:2203.05740*, 2022.

[74] Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, et al. Cache me if you can: Accelerating diffusion models through block caching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6211–6220, 2024.

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

[76] Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made slim. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 22552–22562, 2023.

[77] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[78] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

[79] Dingkun Zhang, Sijia Li, Chen Chen, Qingsong Xie, and Haonan Lu. Laptop-diff: Layer pruning and normalized distillation for compressing diffusion models. *arXiv preprint arXiv:2404.11098*, 2024.

[80] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[81] Qinsheng Zhang, Molei Tao, and Yongxin Chen. gddim: Generalized denoising diffusion implicit models. *arXiv preprint arXiv:2206.05564*, 2022.

[82] Hongkai Zheng, Weili Nie, Arash Vahdat, Kamyar Azizzadenesheli, and Anima Anandkumar. Fast sampling of diffusion models via operator learning. In *International conference on machine learning*, pages 42390–42402. PMLR, 2023.