Δ-DiT: Accelerating Diffusion Transformers without training via Denoising Property Alignment

Anonymous authors
Paper under double-blind review

Figure 1: Accelerating PIXART-α and DiT-XL by $1.6\times$ speedup with 20 DPMSolver++ steps.

Abstract

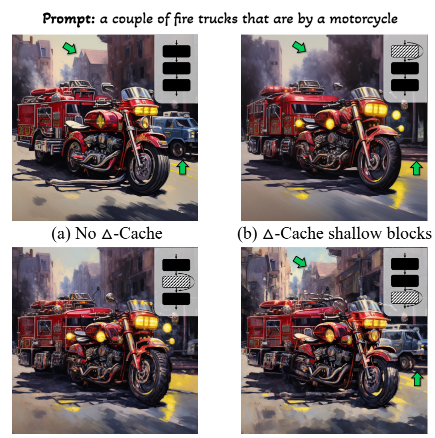
Diffusion models are now commonly used for producing high-quality and diverse images, but the iterative denoising process is time-intensive, limiting their usage in real-time applications. As a result, various acceleration techniques have been developed, though these primarily target UNet-based architectures and are not directly applicable to Transformer-based diffusion models (DiT). To address the specific challenges of the DiT architecture, we first analyze the relationship between the depth of DiT blocks and the quality of image generation. While skipping blocks can lead to large degradations in generation quality, we propose the Δ-Cache method, which captures and stores the incremental changes of different blocks, thereby mitigating the performance gap and maintaining closer alignment with the original results. Our analysis indicates that the shallow DiT blocks primarily define the global structure of images such as compositions and outlines, while the deep blocks mainly refine details, and the role of middle blocks lies between the two. Based on this, we introduce a denoising property alignment method that selectively bypasses computations of different blocks at various timesteps while preserving performance. Comprehensive experiments on PIXART-α and DiT-XL demonstrate that Δ-DiT achieves a $1.6\times$ speedup in 20-step generation and enhances performance in most cases. In the 4-step consistent model generation scenario, and with a more demanding $1.12\times$ acceleration, our approach significantly outperforms existing methods.

1 INTRODUCTION

In recent years, the field of generative models has experienced rapid advancements. Among these, diffusion models (Ho et al., 2020; Rombach et al., 2022; Song et al., 2021b) have emerged as pivotal, attracting widespread attention for their ability to generate high-quality and diverse images (Dhariwal & Nichol, 2021). This has also spurred the development of many meaningful applications, such as image editing (Kawar et al., 2023; Zhang et al., 2023a), 3D generation (Tang et al., 2023b;a; Mo et al., 2023), and video generation (Wu et al., 2023a; Khachatryan et al., 2023; Blattmann et al., 2023; Luo et al., 2023b). Although diffusion models have strong generation capabilities, their iterative denoising nature results in poor real-time performance. Subsequently, numerous inference acceleration frameworks have been proposed, which include general model compression methods for denosing network (Fang et al., 2023; Zhang et al., 2024a; Shang et al., 2023; So et al., 2023; Kim et al., 2023; Salimans & Ho, 2022; Luo et al., 2023a; Zhao et al., 2023; Sauer et al., 2023), fast sampling solver (Song et al., 2021a; Lu et al., 2022a;b; Zhang & Chen, 2023; Karras et al., 2022), and cache-based acceleration methods (Ma et al., 2023; Li et al., 2023b). However, almost all of these acceleration techniques are designed for the UNet-based (Ronneberger et al., 2015) architecture.

Recently, Diffusion Transformers (DiT) (Peebles & Xie, 2023) have emerged as dominant foundational models, exemplified by PIXART-α (Chen et al., 2023), SD3.0 (Esser et al., 2024), and Sora (Brooks et al., 2024). Despite this success, the acceleration of DiT inference is under-explored. Existing methods, such as early stopping (Moon et al., 2023), require retraining and are unsuitable for small-step generation. DiT's isotropic architecture, with no long skip connections found in UNet, makes it difficult to apply UNet-based acceleration techniques. For instance, cache-based methods (Ma et al., 2023; Li et al., 2023b) may result in information loss, as DiT lacks the long shortcuts that facilitate feature reuse in UNet. Moreover, skipping computations for branches can introduce significant degradations. To address this, we propose **Δ-Cache**, a caching method tailored for transformer architectures that caches Δ change between different blocks instead of the original feature maps, preventing large degradation and making caching more effective for DiT.

In our caching framework, we first investigate the degradations introduced by caching at different blocks within the transformer. We observe that the shallow transformer blocks in DiT primarily define the global structure of images such as compositions, and outlines, while the deep blocks focus on refining image details, as illustrated in Figure 2. While previous studies (Wang & Vastola, 2023; Liu et al., 2023; Hertz et al., 2023) have pointed out a property of diffusion models: creating contours in the earlier timesteps (early denoising stage) and generating details in the later timesteps (later stage). Building on this property and our findings, this paper proposes a denoising property alignment inference acceleration method, Δ-DiT. Specifically, the method applies Δ-Cache to the deeper blocks during the early denoising stage to soften the details and preserve the contours, while applying Δ-Cache to the shallow blocks during later sampling to maintain the details, thus aligning with the property of the diffusion models (Wang & Vastola, 2023; Liu et al., 2023; Hertz et al., 2023). We evaluated our approach across multiple datasets, including MS-COCO2017 (Lin et al., 2014) and PartiPrompts (Yu et al., 2022), using various DiT architectures such as PIXART-α (Chen et al., 2023), DiT-XL (Peebles & Xie, 2023), and PIXART-α-LCM (Chen et al., 2023; Luo et al., 2023a). Extensive quantitative results confirm the effectiveness of our method. In the 20-step generation, we achieved a 1.6x speedup, with FID improving from 39.002 to 35.882. In the more challenging 4-step generation scenarios, our method also significantly outperformed existing baselines in terms of FID score from 44.198 to 40.118. The contributions of our paper are three-fold:



(a) No Δ-Cache (b) Δ-Cache shallow blocks (c) Δ-Cache middle blocks (d) Δ-Cache deep blocks

Figure 2: Images generated by Δ-Cache for various blocks within the DiT.

- We adapt the caching method to transformers using Δ-Cache, which stores the incremental changes in feature maps. Furthermore, we identify a correlation between different DiT blocks and the final generation results: shallow blocks focus on generating image outlines, while deep blocks emphasize image details.

- To align with the denoising property of generating outlines first and details later, we propose a training-free acceleration framework, termed Δ-DiT. Specifically, we accelerates inference by caching deep blocks during the early stages of denoising and shallower blocks in the later stages.
- We show empirically that Δ-DiT achieves a 1.6× speedup in 20-step generation while improving image quality. On more challenging image generation scenarios (eg. 4-step), Δ-DiT also outperforms existing approaches in generation quality by a significant margin.

2 Related Work

Efficient Diffusion Model. To improve the real-time performance of diffusion models, various lightweight and acceleration techniques have emerged. Currently, methods for accelerating diffusion models for image generation can be broadly categorized into three perspectives: a lightweight denoising model, and reduced denoising timestep, and the intersection of the model and timestep dimension. Similar to traditional model compression, many efforts focus on pruning (Fang et al., 2023; Zhang et al., 2024a), quantization (Shang et al., 2023; So et al., 2023; He et al., 2023; Li et al., 2023c), and distillation (Kim et al., 2023; Salimans & Ho, 2022; Luo et al., 2023a; Zhao et al., 2023; Sauer et al., 2023) to obtain a smaller yet comparable denoising network. Besides, reduced denoising timestep is a unique dimension for diffusion models. Most methods currently focus on exploring efficient ODE solvers (Song et al., 2021a; Lu et al., 2022a;b; Zhang & Chen, 2023; Karras et al., 2022), aiming to obtain high-quality images with fewer sampling steps. LCM (Song et al., 2023; Luo et al., 2023a) proposes consistency loss and knowledge distillation to achieve the goal of fewer steps. Lastly, there's a focus on jointly optimizing denoising modes and timesteps. For instance, OMS-DPM (Liu et al., 2023) and Autodiffusion (Li et al., 2023a) simultaneously optimize skips and allocate noise estimation networks of specific sizes for each timestep. However, most of the aforementioned work is implemented and validated on the UNet architecture. One previous work (Moon et al., 2023) proposes an early stopping strategy for DiT, which cannot be easily transferred to fewer timestep settings. Therefore, there is currently a lack of novel acceleration methods specifically designed for the DiT architecture.

Cache Mechanism. The cache mechanism is a key concept in computer systems, designed to temporarily store data for reuse, improving processing efficiency. In Large Language Models, the KV cache (Zhang et al., 2023b; Ge et al., 2023) is widely used, caching key and value matrices from attention blocks to accelerate inference. Cache-based techniques have also been applied to diffusion models, with DeepCache (Ma et al., 2023) accelerating UNet by caching feature maps from up-sampling blocks, and Faster Diffusion (Li et al., 2023b) optimizing computation by caching outputs from UNet encoders. On a finer granularity, studies such as (Zhang et al., 2024b; Wimbauer et al., 2024; So et al., 2024) focus on caching feature maps within specific blocks to save computations. These methods target feature maps at various locations and in differing quantities as caching objectives, and most of them are targeted at the UNet architecture. However, this paper introduces a feature map offsets caching method, specifically tailored to the isotropic architecture of DiT.

3 Preliminary

The concept of diffusion originates from a branch of non-equilibrium thermodynamics (De Groot & Mazur, 2013) in physics. In recent years, researchers have applied this concept to image generation (Ho et al., 2020; Rombach et al., 2022; Song et al., 2021b; Dhariwal & Nichol, 2021; Song & Ermon, 2019), transforming the process into two stages: noise diffusion and denoising.

Noising Process. This is also the training phase of the diffusion model. Given an original image $\boldsymbol{x_0}$ and a random time step $t \in [1, T]$ (where T is the total steps), the image after t steps of diffusion is $\sqrt{\bar{\alpha}_t}\boldsymbol{x_0} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, where $\bar{\alpha}_t$ is constant related to t. The noise estimation network is then used to estimate the noise in the diffused result, making the estimated noise $\boldsymbol{\epsilon_\theta}$ as close as possible to the actual noise $\boldsymbol{\epsilon}$ added during diffusion. The learning objective is defined as follows (Ho et al., 2020):

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{t,\boldsymbol{x_0}\sim q(\boldsymbol{x}),\boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon_\theta}(\sqrt{\bar{\alpha}_t}\boldsymbol{x_0} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right], \tag{1}$$

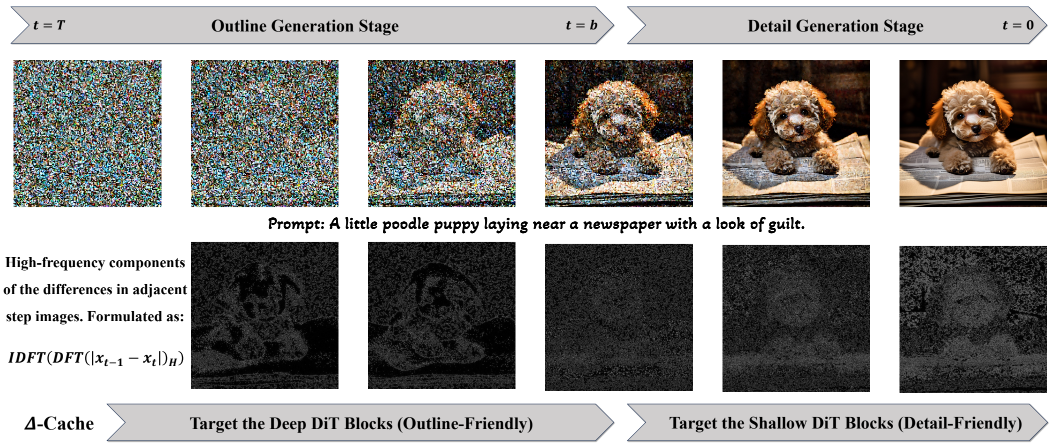


Figure 3: **Overview of the Δ-DiT**: The denoising property emphasizes generating outlines early in denoising and details later. Our previously proposed Δ-Cache method caches deep blocks for outline-friendly generation and shallow blocks for detail-friendly results. In the Δ-DiT, the properties of Denoising and Δ-Cache are aligned in stages, that is, Δ-Cache is applied to the deep blocks in the DiT during the early outline generation stage of the diffusion model, and on shallow blocks during the detail generation stage. The stage is bounded by a hyperparameter b.

where $q(\boldsymbol{x})$ is the dataset distribution, and $\mathcal{N}$ is the Gaussian distribution. In most current works, the noise estimation networks are mostly based on UNet architecture. However, in isotropic architectures like DiT, $\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t)$ can be further transformed into $\boldsymbol{f}^t_{N_b}(\boldsymbol{f}^t_{N_b-1}(\cdots(\boldsymbol{f}^t_1(\boldsymbol{x}_t))) = \boldsymbol{f}^t_{N_b} \circ \boldsymbol{f}^t_{N_b-1} \circ \cdots \circ \boldsymbol{f}^t_1(\boldsymbol{x}_t) = \boldsymbol{F}^t_{1:N_b}(\boldsymbol{x}_t)$, where $\boldsymbol{f}^t_n$ represents the mapping of the n-th DiT block at timestep t, and $\boldsymbol{F}^t_{1:N_b}$ represents the mapping of the first to the N_b-th DiT blocks. N_b denotes the number of blocks.

Denoising Process. During this process, Gaussian noise is iteratively denoised into a generated image, and our goal is to accelerate this denoising process without requiring additional training. Initially, a random Gaussian noise $\boldsymbol{x}_T$ is given. It is then fed into the denoising network $\boldsymbol{\epsilon_\theta}$ to obtain the estimated noise $\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_T)$. With sampling solvers, the noisy image is denoised to produce the denoised sample $\boldsymbol{x}_{T-1}$ for each timestep. After iterating this process T times, the final generated image is obtained. Using the DDPM (Ho et al., 2020) solver as an example, the iterative denoising process can be defined as follows:

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{z}, \tag{2}$$

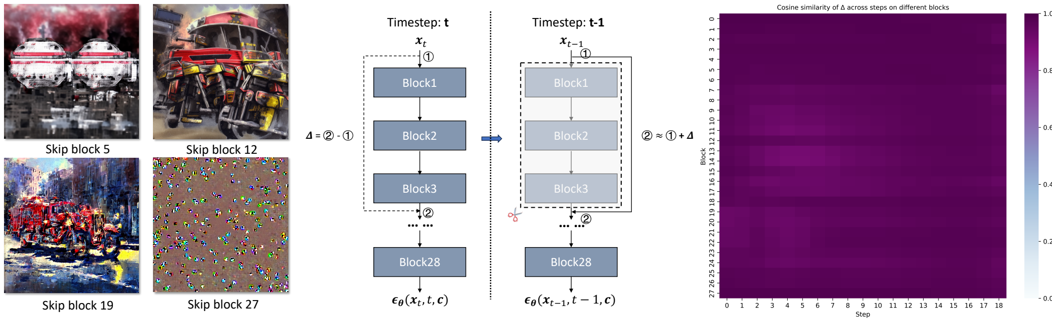
where α_t, β_t and σ_t is constant related to t, and $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. For other solvers (Song et al., 2021a; Lu et al., 2022a;b; Zhang & Chen, 2023; Karras et al., 2022), the sampling formula differs slightly from Eq. 2, but they are all functions of $\boldsymbol{x}_t$ and $\boldsymbol{\epsilon_\theta}$. In many scenarios, the noise estimation network $\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t, t, \boldsymbol{c})$ has another input $\boldsymbol{c}$. It is conditional control information, which can be either a class embedding or a text embedding.

4 Methodology

In this section, we present our denoising property alignment method for training-free acceleration of DiT. First, we introduce Δ-Cache, a novel caching method specifically designed for DiT. Then, leveraging this framework, we explore the specific effects of different parts of blocks on generation. Finally, by integrating the previous findings with the properties of the denoising process, we propose Δ-DiT to accelerate DiT generation. The overall framework is shown in Figure 3.

4.1 Effect of DiT Blocks on Generation

In accelerating DiT, methods like skipping blocks (Raposo et al., 2024) offer a straightforward way to reduce computational overhead. However, skipping blocks during inference without additional training introduces significant degradations from the original results as shown in Figure 4a. Each DiT



(a) Skip different DiT block (b) Δ-Cache method (c) Cosine similarity heatmap of Δ

Figure 4: (a) **Visualization of images generated by skipping different blocks**. (b) **Illustration of Δ-Cache**. The difference between the feature maps at both ends of the block is used as Δ. Then, Δ is employed in the next step to compensate for the skipped computation of the block. (c) **Cosine similarity heatmap of** Δ. Similarity of Δ of blocks with different steps and different depths.

block plays a critical role in estimating noise at each timestep, and directly omitting critical blocks can cause the final image to deteriorate into noise. Thus, it is necessary to compensate for the large discrepancies introduced by skipping blocks to maintain image quality.

Δ-Cache. Inspired by recent cache-based methods (Ma et al., 2023; Li et al., 2023b), caching and reusing previous feature maps offers a potential solution. However, this approach cannot be directly applied to transformer architectures due to the absence of long skip-connection. Therefore, we propose Δ-Cache to cache incremental changes between blocks (e.g., in Figure 4b, the differences between the features at points ① and ② is cached), which based on Δ exhibits a high degree of similarity. As shown in Figure 4c, the cosine similarity of Δ between blocks at the same position across adjacent timesteps remains above 0.9. This high similarity allows the Δ-Cache method to incur minimal information loss. During denoising, some timesteps skip the computation of blocks and apply the cached Δ as compensation to minimize the degradation in the inference. Based on the mathematical framework described in Section 3, Δ-Cache process can be defined as follows:

$$
\begin{aligned}
\boldsymbol{x}_{t-1} &= \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{z} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}F^t_{1:N_b}(\boldsymbol{x}_t)\right) + \sigma_t \boldsymbol{z} \\
&= \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}F^t_{I+N_c:N_b}(F^t_{1:I+N_c}(\boldsymbol{x}_t))\right) + \sigma_t \boldsymbol{z} \\
&= \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}F^t_{I+N_c:N_b}(F^t_{1:I}(\boldsymbol{x}_t) + \underline{F^t_{1:I+N_c}(\boldsymbol{x}_t) - F^t_{1:I}(\boldsymbol{x}_t)})\right) + \sigma_t \boldsymbol{z} \\
&\approx \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}F^t_{I+N_c:N_b}(F^t_{1:I}(\boldsymbol{x}_t) + \underline{F^{t+1}_{1:I+N_c}(\boldsymbol{x}_{t+1}) - F^{t+1}_{1:I}(\boldsymbol{x}_{t+1})})\right) + \sigma_t \boldsymbol{z}
\end{aligned} \tag{3}
$$

Here, the underlined part is Δ, I indicates the starting block position of the Δ-Cache, N_b denotes the number of blocks and N_c refers to the number of cached blocks.

Qualitative Analysis. Within the caching framework, we aim to explore the impact of different blocks on the final generated image. Given that the effect of Δ-Cache on individual blocks is minimal (as shown in Figure 4c), we analyze the impact at a coarser granularity to make the results more discernible, rather than performing a block-by-block analysis. For a 28-block transformer like DiT-XL and PIXART-α, we divide the network into three main sections: (1) **Shallow Blocks (1-21)**: the first 21 blocks, (2) **Middle Blocks (4-24)**: the middle 21 blocks, and (3) **Deep Blocks (8-28)**: the last 21 blocks. This segmentation allows us to better assess the influence of different regions within the model. As shown in Figure 2, we can conclude that:

1) **For Shallow Blocks**. Applying Δ-Cache to the shallow blocks results in inaccurate outline generation. As shown in Figure 2a (green arrows), the blue car's outline on the right is clear. However, in Figure 2b, the outline is absent, despite the image is better generated in detail.
2) **For Deep Blocks**. In contrast, applying Δ-Cache to the deep blocks preserves the global outline but reduces detail accuracy. As shown in Figure 2d, the blue car's outline is retained, but some noise appears in the finer details.

3) **For Middle Blocks**. Δ-Cache applied to the middle part provides a compromise.

From this qualitative analysis, we can conclude that the shallow blocks of DiT are more related to outline generation, the deep blocks are more connected to detail generation, and the middle blocks represent a balance between the two.

Quantitative Analysis. Despite the qualitative analysis indicating a correlation between DiT blocks and the generated output, we further validate this observation through statistical analysis. To quantify the ability to generate details and outlines, we employ the Fourier transform as an effective method (Broughton & Bryan, 2018). In Fourier analysis, high-frequency components correspond to rapid intensity changes, typically associated with details like textures or edges, while low-frequency components represent the global structure or outline. A higher proportion of high-frequency components suggests better detail generation, while strong outline generation reflects more low-frequency components. Specifically, we can calculate the relative log magnitude from the Fourier transform as follows:

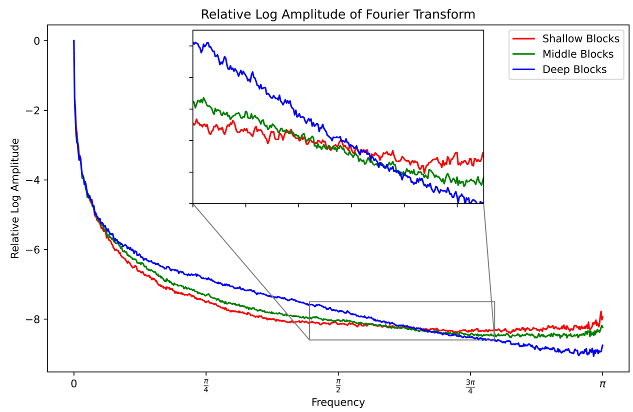


Figure 5: Fourier relative log magnitude of images generated by applying Δ-Cache to various blocks of the DiT.

$$\mathcal{F}(\boldsymbol{x}) = \mathcal{F}(u,v) = \sum_{x=1}^{H}\sum_{y=1}^{W} \boldsymbol{x}(x,y) \cdot e^{-2\pi i\left(\frac{u}{H}x+\frac{v}{W}y\right)} \tag{4}$$

$$\text{Relative Log Magnitude}(u,v) = log\left(\frac{\sqrt{\text{Re}(\mathcal{F}(u,v))^2 + \text{Im}(\mathcal{F}(u,v))^2}}{\max\limits_{u,v}\sqrt{\text{Re}(\mathcal{F}(u,v))^2 + \text{Im}(\mathcal{F}(u,v))^2}}\right) \tag{5}$$

Where $\mathcal{F}$ represents the Fourier transform, $\boldsymbol{x}(x,y)$ denotes the pixel values of the image, and H and W stand for the image's height and width, respectively. The coordinates (u,v) correspond to the frequency domain. Re$(\cdot)$ and Im$(\cdot)$ represent the real parts and imaginary parts of a complex number, respectively.

Using Fourier analysis, we can further compare how different parts of the blocks influence the generated image. To draw statistical conclusions, we perform this analysis on a subset of the MS-COCO2017 dataset Zhao et al. (2024). We compute the relative log magnitude of the Fourier transform for images generated under three different settings, converting it into a radial relative magnitude-frequency space, and calculated the expectation across the dataset. The results are presented in Figure 5 , leading to the following conclusions:

1) **For Shallow Blocks**. When Δ-Cache is applied to the shallow blocks, where the shallow blocks are lossy, the generated images show a high proportion of high-frequency components, indicating strong detail generation, that is, friendly to detail generation.
2) **For Deep Blocks**. Applying Δ-Cache to deep blocks exhibits a high proportion of low-frequency components, indicating strong outline generation, that is, friendly to outline generation.
3) **For Middle Blocks**. Middle blocks with Δ-Cache applied exhibit a balance between these two extremes.

These findings align with the qualitative analysis presented earlier.

4.2 Accelerating DiT via Denoising Property Alignment

Denoising property. Previous research (Wang & Vastola, 2023; Liu et al., 2023; Hertz et al., 2023) has demonstrated that the denoising process in diffusion models follows a generation pattern. During the early stages, these models primarily generate image outlines, while the focus shifts to details in the later stages. To further illustrate this, the bottom part of Figure 3 shows the difference between images generated at adjacent timesteps. After applying a Fourier transform to this difference and extracting the high-frequency components—representing areas with significant changes. We can observe that in the early denoising stages (first two images in Figure 3), these regions primarily capture the outline of the dog, while in later stages (last two images), the focus shifts to finer details like the dog's fur.

Denoising property alignment framework. Therefore, in terms of outline and detail generation, we can align the Δ-Cache with the denoising property. As shown in Figure 3, image denoising at various timesteps (from $t = T$ to $t = 0$) is presented. Since denoising properties emphasize outline generation in the early stages, and the deeper blocks in the Δ-Cache method are more suited for generating outlines, Δ-Cache is applied to the deep blocks during the outline generation stage. Conversely, as diffusion models focus more on detail generation in the later stages, and the shallow blocks of the Δ-Cache method are more appropriate for detail generation, Δ-Cache is applied to the shallow blocks during the detail generation stage.

To this end, we propose a training-free framework termed Δ-DiT, which can generate images with better quality. In our framework, we introduce two hyperparameters. One is denoted as b, representing the boundary between the outline generation stage and the detail generation stage. When $t \leq b$, Δ-Cache is applied to the deep blocks; when $t > b$, Δ-Cache is applied to the shallow blocks. The number of blocks requiring Δ-Cache is determined based on the actual computational requirements. Assuming the computation cost of one block is M_b and the expected total computation cost is M_g, as previously mentioned, the cache interval is N, and the number of DiT blocks is N_b. First, we roughly determine the value of N as:

$$N = \lceil \frac{T \times N_b \times M_b}{M_g} \rceil, \tag{6}$$

In some current low-step scenarios, the value of N is set to 2. After determining N, the actual number of blocks to cache at the timestep is:

$$N_c = [(\underbrace{\frac{M_g - (T \bmod N) \times N_b \times M_b}{\lfloor T/N \rfloor \times M_b}}_{\text{the computation in each } N \text{ step}} - \underbrace{N_b \times M_b}_{\text{the first full DiT}})/\underbrace{(M_b \times (N-1))}_{\text{the remaining cached steps}}]. \tag{7}$$

Once these hyperparameters are determined, the inference process becomes fixed and remains unchanged regardless of the input, enabling acceleration without the need for further training.

5 Experiment

5.1 Experimental Settings

Models, Evaluation Data and Solvers. We conduct experiments on three diffusion transformer-based architectures: DiT-XL (Peebles & Xie, 2023), PIXART-α(Chen et al., 2023), and PIXART-α-LCM(Chen et al., 2023; Luo et al., 2023a). For DiT-XL, we generat 50k images using 1000 ImageNet classes (Russakovsky et al., 2015) for evaluation. For the PIXART-α models, we evaluate image quality using 1.632k prompts from PartiPrompt (Yu et al., 2022) and 5k prompts from the MS-COCO2017 validation dataset (Lin et al., 2014). In our main experiment, we use the 20-step DPMSolver++ (Lu et al., 2022b), the default setting for PIXART-α. For consistency model generation, we apply the 4-step LCMSolver (Song et al., 2023). To demonstrate the effectiveness of our method, we compare with several fast-generation techniques, including the feature map caching method from Faster Diffusion (Li et al., 2023b), TGATE (Zhang et al., 2024b).

Evaluation Metrics. We use a range of metrics to evaluate both generation efficiency and image quality. For generation efficiency, we measure the theoretical computational complexity using MACs and the practical time to generate an image using latency. Lower MACs and latency indicate higher efficiency, while the speedup reflects the acceleration rate. To assess generation quality, we employ widely used metrics such as FID (Heusel et al., 2017), IS (Salimans et al., 2016), and CLIP-Score (Hessel et al., 2021).

Table 1: The MS-COCO2017 and PartiPrompts generation results for PIXART-α are evaluated. Gate is the hyperparameter defined in TGATE (Zhang et al., 2024b). T represents the number of timesteps, and I indicates the starting block index for caching. Latency, measured in milliseconds, is tested on an Nvidia A100 GPU.

Method	MACs ↓	Speedup ↑	Latency ↓	MS-COCO2017 FID ↓	MS-COCO2017 IS ↑	MS-COCO2017 CLIP ↑	PartiPrompts CLIP ↑
PIXART-α ($T = 20$) (Chen et al., 2023)	85.651T	1.00×	2290.668	39.002	31.385	**30.417**	<u>30.097</u>
PIXART-α ($T = 13$) (Chen et al., 2023)	55.673T	1.54×	1565.175	39.989	30.822	<u>30.399</u>	29.993
Faster Diffusion ($I = 14$) (Li et al., 2023b)	64.238T	1.33×	1777.144	41.560	31.233	30.300	29.958
Faster Diffusion ($I = 21$) (Li et al., 2023b)	53.532T	1.60×	1517.698	42.763	30.316	30.227	29.922
TGATE (Gate=10) (Zhang et al., 2024b)	61.075T	1.40×	1718.308	37.413	31.079	29.782	29.347
TGATE (Gate=8) (Zhang et al., 2024b)	56.170T	1.52×	1603.250	37.539	30.124	29.021	28.654
Δ-Cache (Shallow Blocks)	53.532T	1.60×	1522.346	41.702	30.276	30.288	29.964
Δ-Cache (Middle Blocks)	53.532T	1.60×	1522.528	<u>35.907</u>	**33.063**	30.183	30.078
Δ-Cache (Deep Blocks)	53.532T	1.60×	1522.669	**34.819**	<u>32.736</u>	29.898	<u>30.099</u>
Ours ($b = 12$)	53.532T	1.60×	1534.551	<u>35.882</u>	<u>32.222</u>	<u>30.404</u>	**30.123**

Table 3: The MS-COCO 2017 and PartiPrompts results for the PIXART-α-LCM model are evaluated, using the default number of generation steps, $T = 4$.

Method	MACs ↓	Speedup ↑	Latency ↓	MS-COCO2017 FID ↓	MS-COCO2017 IS ↑	MS-COCO2017 CLIP ↑	PartiPrompts CLIP ↑
PIXART-α-LCM (Chen et al., 2023)	8.565T	1.00×	415.255	40.433	30.447	29.989	29.669
Faster Diffusion ($I = 4$) (Li et al., 2023b)	7.953T	1.08×	401.137	468.772	1.146	-1.738	1.067
Faster Diffusion ($I = 6$) (Li et al., 2023b)	7.647T	1.12×	391.081	468.471	1.146	-1.746	1.057
TGATE (Gate=2) (Zhang et al., 2024b)	7.936T	1.08×	400.256	42.038	29.683	29.908	29.549
TGATE (Gate=1) (Zhang et al., 2024b)	7.623T	1.12×	398.124	44.198	27.865	29.074	28.684
Ours ($b = 2$, $N_c = 4$)	7.953T	1.08×	400.132	39.967	29.667	29.751	29.449
Ours ($b = 2$, $N_c = 6$)	7.647T	1.12×	393.469	40.118	29.177	29.332	29.226

5.2 Comparison with Acceleration Methods

We comprehensively compare with efficient generation methods for PIXART-α on both the generation efficiency and image quality in Table 1. The proposed method exceeds the baseline PIXART-$\alpha(T = 20)$ on all metrics except for a small gap in the MS-COCO2017 CLIP-Score, with a 1.60× speedup. With similar inference cost, we surpass PIXART-$\alpha(T = 13)$ on all metrics by a large margin (e.g., FID: 39.989 → 35.882). Moreover, our proposed method also outperforms Faster Diffusion and TGATE in all metrics on both datasets with similar or even higher generation efficiency. Finally, to further illustrate the superior generative performance of our method, refer to the visualizations generated by different methods in Figure 8.

Table 2: Results on the DiT-XL (cfg=4.0). Because the TGATE can only handle cross-attention, it cannot be used for DiT-XL.

Method	ImageNet-50k MACs ↓	ImageNet-50k Latency ↓	ImageNet-50k FID ↓	ImageNet-50k IS ↑
DiT-XL ($T = 20$) (Peebles & Xie, 2023)	4.579T	578.201	15.893	<u>440.797</u>
DiT-XL ($T = 13$) (Peebles & Xie, 2023)	2.976T	382.607	15.982	436.730
Faster Diffusion ($I = 14$) (Li et al., 2023b)	3.434T	458.409	15.084	417.903
Faster Diffusion ($I = 21$) (Li et al., 2023b)	2.862T	383.812	15.145	416.609
Δ-Cache (Shallow Blocks)	2.862T	367.148	15.112	420.198
Δ-Cache (Middle Blocks)	2.862T	368.984	<u>14.270</u>	**442.921**
Δ-Cache (Deep Blocks)	2.862T	367.042	<u>13.391</u>	439.700
Ours ($b = 12$)	2.862T	370.290	**13.289**	<u>442.028</u>

In Table 2, we further validate our proposed method on the DiT-XL architecture Peebles & Xie (2023). The method achieves a 1.6× speedup over the baseline DiT-XL ($T = 20$) while improving the FID from 15.893 to 13.289. Additionally, it surpasses Faster Diffusion ($I = 21$) in terms of IS, improving from 416.609 to 442.028 by a significant margin, while maintaining comparable inference speed. Although Δ-Cache does not lead in all metrics, its strong performance in both tables demonstrates its overall effectiveness, offering a favorable balance between quality and efficiency.

Table 4: Performance under different advanced solvers which are measured on MS-COCO2017.

Solver	PIXART-α			+ Δ-DiT		
	FID $\downarrow$	**IS** $\uparrow$	**CLIP** $\uparrow$	**FID** $\downarrow$	**IS** $\uparrow$	**CLIP** $\uparrow$
EulerD (Karras et al., 2022)	39.688	31.413	30.359	35.735	32.290	30.239
DEIS (Zhang & Chen, 2023)	37.675	32.362	30.420	35.302	32.721	30.377
DPMSolver++ (Lu et al., 2022a)	39.002	31.385	30.417	35.882	32.222	30.404

5.3 Comparison Under LCM Settings

Latent Consistency Model (LCM) (Song et al., 2023; Luo et al., 2023a) introduces a method to accelerate the denoising process using consistency loss, reducing timesteps from over 30 to just 4, making it highly difficult to accelerate further. We test our approach in this challenging scenario to assess its generalizability, as shown in Table 3. Methods like Faster Diffusion (Li et al., 2023b), which lack supervision from previous step images, perform poorly in small-step settings, with significantly high FID scores (FID=468.471). While existing approaches like TGATE (Zhang et al., 2024b) achieve reasonable results, they suffer notable performance degradation (FID: 40.433 $\rightarrow$ 44.198) at an acceleration ratio of approximately 1.12. In contrast, our method maintains superior performance even with better FID scores.

Table 5: Results of opposite denoising property alignment on PIXART-α and DiT-XL.

Method	FID $\downarrow$	IS $\uparrow$	CLIP $\uparrow$
PIXART-α-Ours	35.882	32.222	30.404
Opposite	41.374	30.980	30.259
DiT-XL-Ours	13.289	442.028	/
Opposite	15.255	426.949	/

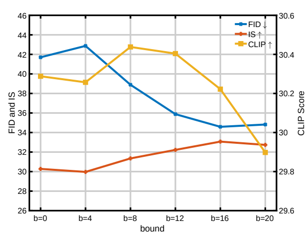


Figure 6: The choice of bound value b.

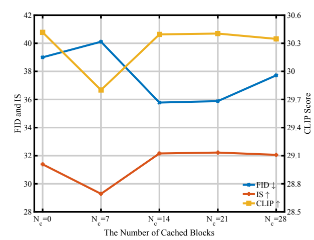


Figure 7: The choice of total cached blocks N_c.

5.4 Ablation Study

Compatibility with fast sampling solvers. Our experiments use the default solver, DPMSolver++(Lu et al., 2022b), but we also demonstrate compatibility with more advanced solvers. As shown in Table 4, the performance improvements are consistent across different solvers. Notably, for all three solvers—EulerD (Karras et al., 2022), DEIS (Zhang & Chen, 2023), and DPMSolver++ (Lu et al., 2022b)—we observe significant gains, particularly in FID scores. EulerD shows a substantial improvement (FID: 39.688 $\rightarrow$ 35.735), as does DEIS (FID: 37.675 $\rightarrow$ 35.882).

Effect of opposite denoising property alignment. The Δ-DiT framework uses Δ-Cache for deep blocks during early sampling and for shallow blocks during later stages. In this experiment, we reverse the cache order, applying Δ-Cache to shallow blocks in the early stages and deep blocks in the later stages. As shown in Table 5, for PIXART-α, although the CLIP score shows a minor difference, FID and IS significantly deteriorate (FID: 35.882 $\rightarrow$ 41.374; IS also drops). While CLIP score reflects semantic alignment with text, FID and IS better capture the image's finer details, highlighting the effectiveness of the original caching strategy in enhancing image quality.

Illustration of the increasing bound b. Figure 6 illustrates the effect of the bound value on generation outcomes. As b increases from 0 to 20, FID and IS improve and reach their optimal values around $b = 16$, while the CLIP score peaks at $b = 8$. Given that a decreasing CLIP score can significantly impact image-text alignment, we empirically determine that setting $b = 12$ offers the best trade-off between FID, IS, and CLIP score, balancing both image quality and semantic alignment.

Illustration of the increasing number of cached blocks N_c. Figure 7 depicts the effect of the number of cached blocks (N_c) on generation performance. As N_c increases from 0 to 28, FID reaches its best value around $N_c = 14$, while IS and CLIP score peak around $N_c = 21$. To balance performance and acceleration, we select $N_c = 21$, which results in over 37% MACs reduction, as shown in Table 1.

Figure 8: **Comparison of images generated by various methods.** High-resolution images are generated based on different strategies using prompts randomly selected from six distinct scenes in the MS-COCO2017 dataset.

6 Conclusion and Limitation

This paper considers the unique structure of DiT and proposes a training-free cache mechanism, Δ-Cache, specifically designed for DiT. Furthermore, we qualitatively and quantitatively explore the relationship between shallow blocks in DiT and outline generation, as well as deep blocks and detail generation. Based on these findings and the denoising properties of diffusion, we propose the denoising property alignment acceleration method, Δ-DiT, which applies Δ-Cache to different part blocks of DiT at various denoising stages. Extensive experiments confirm the effectiveness of our approach. We believe that more refined search or learning strategies will yield even greater benefits.

REFERENCES

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. `https://openai.com/research/video-generation-models-as-world-simulators`, 2024.

S Allen Broughton and Kurt Bryan. *Discrete Fourier analysis and wavelets: applications to signal and image processing*. John Wiley & Sons, 2018.

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, et al. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. *arXiv preprint arXiv:2310.00426*, 2023.

Sybren Ruurds De Groot and Peter Mazur. *Non-equilibrium thermodynamics*. Courier Corporation, 2013.

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

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. *Advances in neural information processing systems*, 2023.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.

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*, 2023.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross-attention control. In *International Conference on Learning Representations*, 2023.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *EMNLP (1)*, pp. 7514–7528. Association for Computational Linguistics, 2021.

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.

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

Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking fid: Towards a better evaluation metric for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9307–9315, 2024.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

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*, pp. 6007–6017, 2023.

Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15954–15964, 2023.

Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. On architectural compression of text-to-image diffusion models. *arXiv preprint arXiv:2305.15798*, 2023.

Lijiang Li, Huixia Li, Xiawu Zheng, Jie Wu, Xuefeng Xiao, Rui Wang, Min Zheng, Xin Pan, Fei Chao, and Rongrong Ji. Autodiffusion: Training-free optimization of time steps and architectures for automated diffusion model acceleration. In *ICCV*, pp. 7082–7091. IEEE, 2023a.

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*, 2023b.

Yanjing Li, Sheng Xu, Xianbin Cao, Xiao Sun, and Baochang Zhang. Q-DM: an efficient low-bit quantized diffusion model. *Advances in neural information processing systems*, 2023c.

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*, pp. 740–755. Springer, 2014.

Enshu Liu, Xuefei Ning, Zinan Lin, Huazhong Yang, and Yu Wang. OMS-DPM: optimizing the model schedule for diffusion probabilistic models. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pp. 21915–21936, 2023.

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, 2022a.

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*, 2022b.

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*, 2023a.

Zhengxiong Luo, Dayou Chen, Yingya Zhang, Yan Huang, Liang Wang, Yujun Shen, Deli Zhao, Jingren Zhou, and Tieniu Tan. Videofusion: Decomposed diffusion models for high-quality video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10209–10218, 2023b.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. *arXiv preprint arXiv:2312.00858*, 2023.

Shentong Mo, Enze Xie, Ruihang Chu, Lanqing Hong, Matthias Nießner, and Zhenguo Li. Dit-3d: Exploring plain diffusion transformers for 3d shape generation. *Advances in neural information processing systems*, 2023.

Taehong Moon, Moonseok Choi, EungGu Yun, Jongmin Yoon, Gayoung Lee, and Juho Lee. Early exiting for accelerated inference in diffusion models. In *ICML 2023 Workshop on Structured Probabilistic & Inference Generative Modeling*, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

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*, pp. 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI (3)*, volume 9351 of *Lecture Notes in Computer Science*, pp. 234–241. Springer, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

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.

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

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*, pp. 1972–1981, 2023.

Junhyuk So, Jungwon Lee, Daehyun Ahn, Hyungjun Kim, and Eunhyeok Park. Temporal dynamic quantization for diffusion models. *Advances in neural information processing systems*, 2023.

Junhyuk So, Jungwon Lee, and Eunhyeok Park. Frdiff : Feature reuse for universal training-free acceleration of diffusion models, 2024. URL `https://arxiv.org/abs/2312.03517`.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, pp. 11895–11907, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023a.

Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22819–22829, 2023b.

Binxu Wang and John J Vastola. Diffusion models generate images like painters: an analytical theory of outline first, details later. *arXiv preprint arXiv:2303.02490*, 2023.

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*, pp. 6211–6220, 2024.

Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7623–7633, 2023a.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023b.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36, 2024.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Trans. Mach. Learn. Res.*, 2022.

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*, 2024a.

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*, pp. 3836–3847, 2023a.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *International Conference on Learning Representations*, 2023.

Wentian Zhang, Haozhe Liu, Jinheng Xie, Francesco Faccio, Mike Zheng Shou, and Jürgen Schmidhuber. Cross-attention makes inference cumbersome in text-to-image diffusion models. *arXiv preprint arXiv:2404.02747*, 2024b.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H2O: heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 2023b.

Lin Zhao, Tianchen Zhao, Zinan Lin, Xuefei Ning, Guohao Dai, Huazhong Yang, and Yu Wang. Flasheval: Towards fast and accurate evaluation of text-to-image diffusion generative models. *arXiv preprint arXiv:2403.16379*, 2024.

Yang Zhao, Yanwu Xu, Zhisheng Xiao, and Tingbo Hou. Mobilediffusion: Subsecond text-to-image generation on mobile devices. *arXiv preprint arXiv:2311.16567*, 2023.

A Additional Metric Evaluation

In the main text, we evaluate the performance of various methods using FID, CLIP, and IS metrics. Here, we have added more benchmarks for a more comprehensive evaluation. Specifically, we included CMMD (an upgraded version of the FID metric) (Jayasumana et al., 2024), as well as IR (ImageReward (Xu et al., 2024)) and HPSv2 (Wu et al., 2023b), two widely accepted human preference metrics. Refer to Table 6 for the specific experimental results. Our method obtains comprehensive optimal results on all those generation metrics.

Table 6: The MS-COCO2017 generation results for PIXART-α are evaluated. Gate is the hyper-parameter defined in TGATE (Zhang et al., 2024b). T represents the number of timesteps, and I indicates the starting block index for caching. Latency, measured in milliseconds, is tested on an Nvidia A100 GPU. Underscore and bold indicate the top 3 results.

Method	**MACs** ↓	**Speedup** ↑	**MS-COCO2017**					
			FID ↓	**IS** ↑	**CLIP** ↑	**CMMD** ↓	**IR** ↓	**HPSv2** ↑
PIXART-α ($T=20$) (Chen et al., 2023)	85.651T	1.00×	39.002	31.385	**30.417**	1.104	3.961	**29.466**
PIXART-α ($T=13$) (Chen et al., 2023)	55.673T	1.54×	39.989	30.822	30.399	1.113	4.265	29.338
Faster Diffusion ($I=21$) (Li et al., 2023b)	53.532T	1.60×	42.763	30.316	30.227	1.119	5.048	29.009
TGATE (Gate=8) (Zhang et al., 2024b)	56.170T	1.52×	37.539	30.124	29.021	1.086	6.081	28.552
Δ-Cache (Shallow Blocks)	53.532T	1.60×	41.702	30.276	30.288	1.162	4.908	29.028
Δ-Cache (Middle Blocks)	53.532T	1.60×	35.907	**33.063**	30.183	1.091	4.160	29.229
Δ-Cache (Deep Blocks)	53.532T	1.60×	**34.819**	32.736	29.898	**1.075**	3.848	29.109
Ours ($b=12$)	53.532T	1.60×	35.882	32.222	30.404	1.077	**3.729**	29.390

B Results on Stable Diffusion 3.0

In the main text, we mainly conducted experiments on the traditional classical DiT architecture (Peebles & Xie, 2023). Recently, some new DiT architectures have emerged, such as the MMDiT of SD3 (Esser et al., 2024). Therefore, we also evaluated the performance on these new DiT architectures, and the results are shown in Table 7. Even with the unique dual-branch architecture of SD3's DiT, our method remains applicable and achieves overall optimal performance in generation metrics, surpassing all baseline methods with comparable MACs.

Table 7: Results on the Stable Diffusion 3.0.

Method	**MS-COCO2017**				
	MACs ↓	**Speedup** ↑	**FID** ↓	**IS** ↑	**CLIP** ↑
SD3 ($T=28$) (Esser et al., 2024)	168.256T	1.00×	32.288	**35.326**	**32.314**
SD3 ($T=18$) (Esser et al., 2024)	108.164T	1.55×	31.875	33.890	32.156
Faster Diffusion ($I=21$) (Li et al., 2023b)	105.160T	1.60×	30.823	33.349	32.172
Δ-Cache (Shallow Blocks)	105.160T	1.60×	30.410	33.583	32.124
Δ-Cache (Middle Blocks)	105.160T	1.60×	30.595	33.902	32.065
Δ-Cache (Deep Blocks)	105.160T	1.60×	30.617	33.725	32.156
Ours	105.160T	1.60×	**30.270**	33.939	32.200

C Generation Performance Under Different Speedup

In the Table 1, we present the experimental results for a fixed speedup (1.6×). To demonstrate the performance of various methods under different speedups, we plotted the Pareto curve of CLIP-Score (more widely recognized in T2I tasks) versus computational cost, as shown in Figure 9. The red line represents the performance of our proposed method, which is positioned at the top-left of the performance curves of other methods, indicating overall Pareto-optimal results. And more results on LCM are shown in Table 8. Under the same speedup ratio (1.12×), our method achieves better generation results compared to existing methods. At a higher speedup ratio (1.4×), the proposed method still maintains an advantage in generation metrics, outperforming TGATE at a 1.12× speedup.

It is worth noting that Faster Diffusion fails to generate properly at a 1.12× speedup, and TGATE's maximum speedup is only 1.17×. Our speedup ratio is groundbreaking, especially for challenging tasks like LCM.

Table 8: The MS-COCO 2017 results for the PIXART-α-LCM model are evaluated, using the default number of generation steps, $T = 4$.

Method	MACs ↓	Speedup ↑	Latency ↓	MS-COCO2017 FID ↓	IS ↑	CLIP ↑
PIXART-α-LCM (Chen et al., 2023)	8.565T	1.00×	415.255	40.433	30.447	29.989
Faster Diffusion ($I = 4$) (Li et al., 2023b)	7.953T	1.08×	401.137	468.772	1.146	-1.738
Faster Diffusion ($I = 6$) (Li et al., 2023b)	7.647T	1.12×	391.081	468.471	1.146	-1.746
TGATE (Gate=2) (Zhang et al., 2024b)	7.936T	1.08×	400.256	42.038	29.683	29.908
TGATE (Gate=1) (Zhang et al., 2024b)	7.623T	1.12×	398.124	44.198	27.865	29.074
Ours ($b = 2$, $N_c = 4$)	7.953T	1.08×	400.132	39.967	29.667	29.751
Ours ($b = 2$, $N_c = 6$)	7.647T	1.12×	393.469	40.118	29.177	29.332
Ours ($b = 2$, $N_c = 11$)	6.883T	1.24×	350.539	42.653	29.810	29.689
Ours ($b = 2$, $N_c = 16$)	6.118T	1.40×	306.334	44.043	29.303	29.268

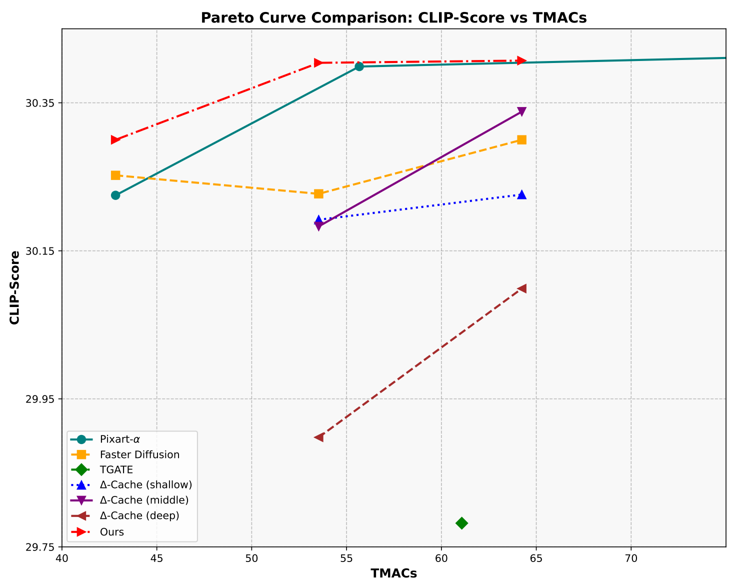


Figure 9: **Pareto curves of various methods' performance**: Evaluation results of the Pixart-α (Chen et al., 2023) on MS-COCO2017.

D ANALYSIS OF Δ-CACHE AND Δ-DIT

Δ-Cache is the foundational module of Δ-DiT, and Δ-DiT utilizes alignment techniques built on top of Δ-Cache. In this section, we visually demonstrate the advantages of Δ-DIT over Δ-Cache, which lacks alignment techniques. Figure 10 visualizes the generation results of different strategies. Similar to Section 4.1, strategies (b) and (c) have poor contour generation ability (an extra horse is generated), while (d) suffers significant detail loss (with many noise points in the image). On the other hand, strategy (e), which applies our alignment technique, maintains good overall contours and preserves details without introducing much noise.

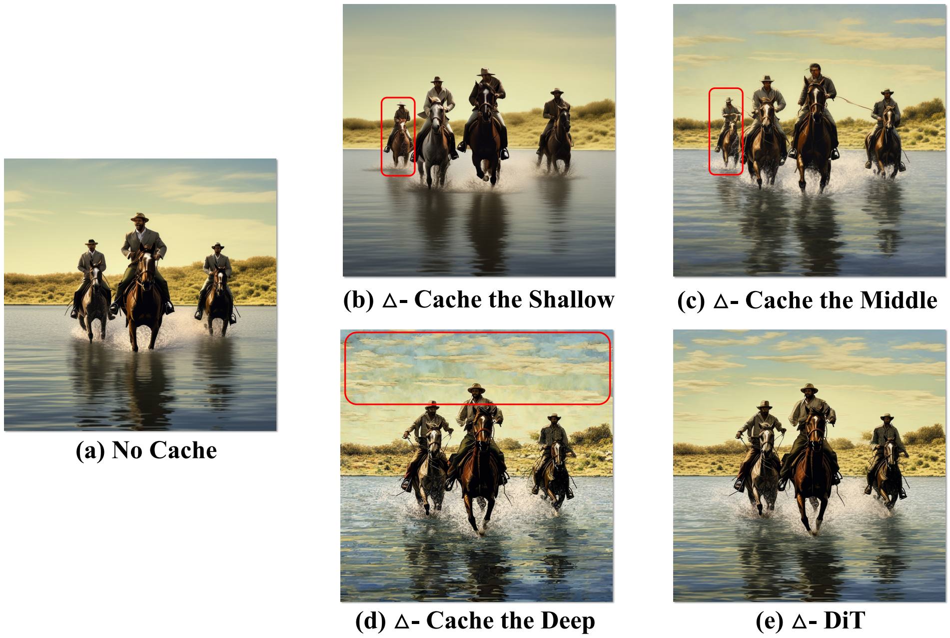


Figure 10: **Comparison of the images generated by our proposed methods.** The red line indicates areas with anomalies in the image. Compared to (a), both (b) and (c) show worse contour generation, with (b) and (c) introducing an extra horse in the contours. (d) exhibits poorer detail generation; while the contour is similar to (a), the image contains more noise. In contrast, (e) leverages alignment techniques, resulting in improved performance in both contour and detail generation.

E Orthogonality With Other Acceleration Methods

In this section, we demonstrate the orthogonality between Δ-DiT and latent consistency models. In fact, our method can also be orthogonal to classical quantization methods. For the Pixart-α, we performed INT8 quantization, and the resulting orthogonal outcomes are shown in Table 9. With the integration of quantization techniques, our speedup can reach $2\times$, while still maintaining good generation metrics.

Table 9: Orthogonal experimental results of our method and model quantization.

Method	MS-COCO2017				
	Latency ↓	**Speedup ↑**	**FID ↓**	**IS ↑**	**CLIP ↑**
Pixart-α (Chen et al., 2023)	2290.668	1.00×	39.002	31.385	30.417
Pixart-α + Quantization	1609.016	1.42×	39.044	31.482	30.418
Ours	1534.551	1.60×	35.882	32.222	30.403
Ours + Quantization	1114.004	2.06×	35.855	32.305	30.394

F Exploration of Hyperparameter Optimization Methods

In Δ-DiT, there are two hyperparameters: the boundary b for detail generation and contour generation, and the number of cache blocks N_c. There are optimization techniques that can be applied to these hyperparameters. For example, by using FlashEval's fast evaluation algorithm to search for optimal values of b and N_c. First, obtain 50 prompts that align well with CLIP metrics using the algorithm. Next, evaluate the CLIP score for different combinations of b and N_c using these prompts. Finally, identify the top 10 hyperparameter combinations, which provide the best text-image matching. Thanks to the speed of FlashEval's evaluation, this process takes about 6 GPU hours to run on a single A100 for the Pixart-α. The quantitative results are shown in Table 10. It can be observed that the CLIP score of the hyperparameters obtained through the search algorithm is better than that of the ones set by experience, further validating the effectiveness of the hyperparameter optimization algorithm.

Table 10: Results of searching for N_c and b after applying CLIP Score metric evaluation based on FlashEval (Zhao et al., 2024) method.

Method	**MS-COCO2017**				
	(N_c, b)	**TMACs** ↓	**CLIP** ↑	**FID** ↓	**IS** ↑
Pixart-α (Chen et al., 2023)	0, None	85.651	30.417	39.002	31.385
Ours	21, 12	53.532	30.403	35.882	32.222
Ours + CLIP Search (Zhao et al., 2024)	12, 10	67.297	30.472	37.547	31.409
Ours + CLIP Search (Zhao et al., 2024)	20, 8	55.061	30.445	38.670	31.330

G Potential Within the UNet Architecture

Although Δ-DiT is a method specifically designed for the DiT architecture, the Δ-Cache concept we proposed is still applicable to the U-Net architecture. Specifically, the Δ-Cache method can be applied to any position with the same resolution in U-Net, as shown in Figure 11. While the widely adopted DeepCache (Ma et al., 2023) uses feature maps as the cache target, our Δ-Cache targets the difference in feature maps as the cache. Experiments on the SD1.5 Rombach et al. (2022) show that our method also achieves competitive results. For detailed quantitative data, refer to Table 11. Our method outperforms the DeepCache method across all three generation metrics under the same MACs.

Table 11: Applicability of our proposed Δ-Cache on the U-Net architecture.

Method	**MS-COCO2017**				
	MACs ↓	**Speedup** ↑	**FID** ↓	**IS** ↑	**CLIP** ↑
Stable Diffusion v1.5 (Rombach et al., 2022)	13.553T	1.00×	25.133	**33.406**	29.953
DeepCache (Ma et al., 2023)	7.923T	1.71×	23.313	32.620	30.146
Ours (Δ-Cache)	7.923T	1.71×	**23.117**	33.014	**30.148**

H More Fine-Grained Block Analysis

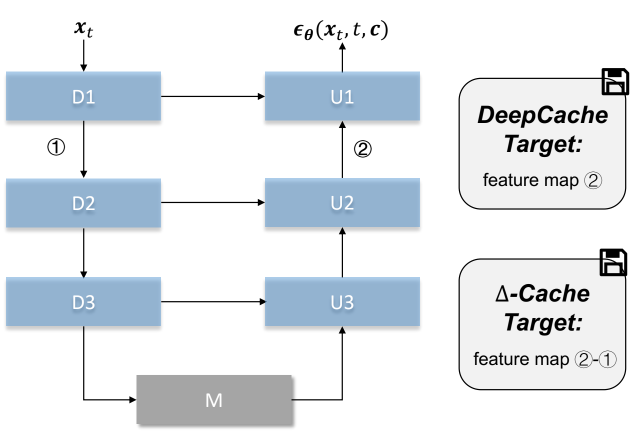


Figure 11: Comparison of Δ-Cache and DeepCache in U-Net.

In the main manuscript, we explored the effect of blocks in three sections: shallow blocks (1-21), middle blocks (4-24), and deep blocks (8-28). Here, we present qualitative and quantitative results in a more fine-grained manner. Specifically, we applied Δ-Cache to blocks 1-7, 7-14, 14-21, and 21-28, and the resulting qualitative and quantitative outcomes are shown in Figure 12. Qualitatively, we observed that applying Δ-Cache to blocks closer to the shallow significantly impacts the contours compared to not using caching. For example, as shown in Figure 12b, a blue car is directly lost. In contrast, applying Δ-Cache to later blocks has a more pronounced effect on the details. Quantitatively, when Δ-Cache is applied to

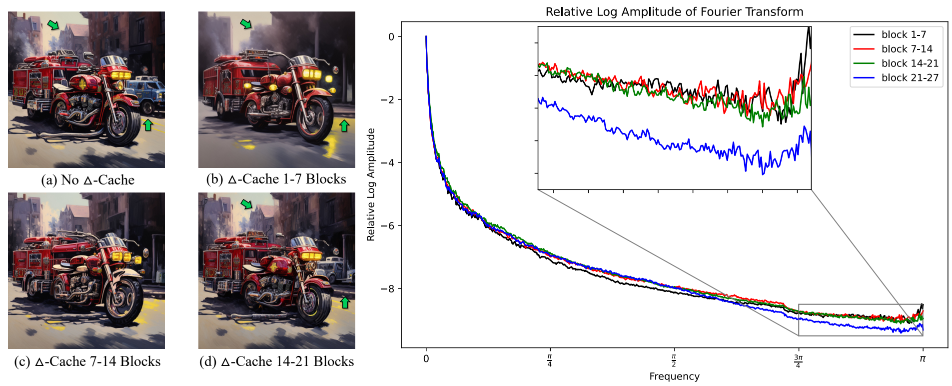


Figure 12: Qualitative and quantitative evaluation results of more fine-grained blocks division.

blocks 1-7, the loss of high-frequency information is minimal, while the loss of blocks 21-27 is large, which also means that the loss of detail is large. This conclusion aligns with the findings presented in the main manuscript.

I Experiments in More Step Scenarios

Our goal is to generate images with an extremely small number of steps, so we set the generation process to 20 steps. Although the FID is slightly higher compared to results with more steps, it aligns with those reported in some literature under the same conditions. References (Zhang et al., 2024b) and (So et al., 2024) provide baseline results for 20-step PIXART-α and DiT-XL, respectively, with FID similar to ours. Finally, given that DiT-XL experiments are typically configured with 250 steps (cfg=1.5) (Peebles & Xie, 2023), we also conducted validation under this setting, as shown in Table 12. The experimental results are consistent with the findings in the paper, demonstrating that our method nearly outperforms the existing baseline approaches in terms of performance. Note that in our experiments, we used the pytorch-fid package to evaluate FID.

Table 12: Results on the DiT-XL (cfg=1.5). * indicates the results we replicated under the official code.

Method	ImageNet-50k			
	MACs ↓	**Latency ↓**	**FID ↓**	**IS ↑**
DiT-XL ($T = 250$) (Peebles & Xie, 2023)	57.24T	-	2.27	278.24
*DiT-XL ($T = 250$) (Peebles & Xie, 2023)	57.24T	7064.60	2.29	277.67
*DiT-XL ($T = 157$) (Peebles & Xie, 2023)	35.94T	4445.56	2.37	267.26
Faster Diffusion ($I = 14$) (Li et al., 2023b)	42.93T	5338.08	2.63	261.40
Faster Diffusion ($I = 21$) (Li et al., 2023b)	35.77T	4671.74	2.58	262.20
Δ-Cache (Shallow Blocks)	35.77T	4652.80	2.52	264.28
Δ-Cache (Middle Blocks)	35.77T	4641.43	2.37	270.84
Δ-Cache (Deep Blocks)	35.77T	4680.23	2.35	269.87
Ours	35.77T	4642.53	2.31	271.03