DDiT: Dynamic Patch Scheduling for Efficient Diffusion Transformers

Anonymous CVPR submission

Paper ID *****

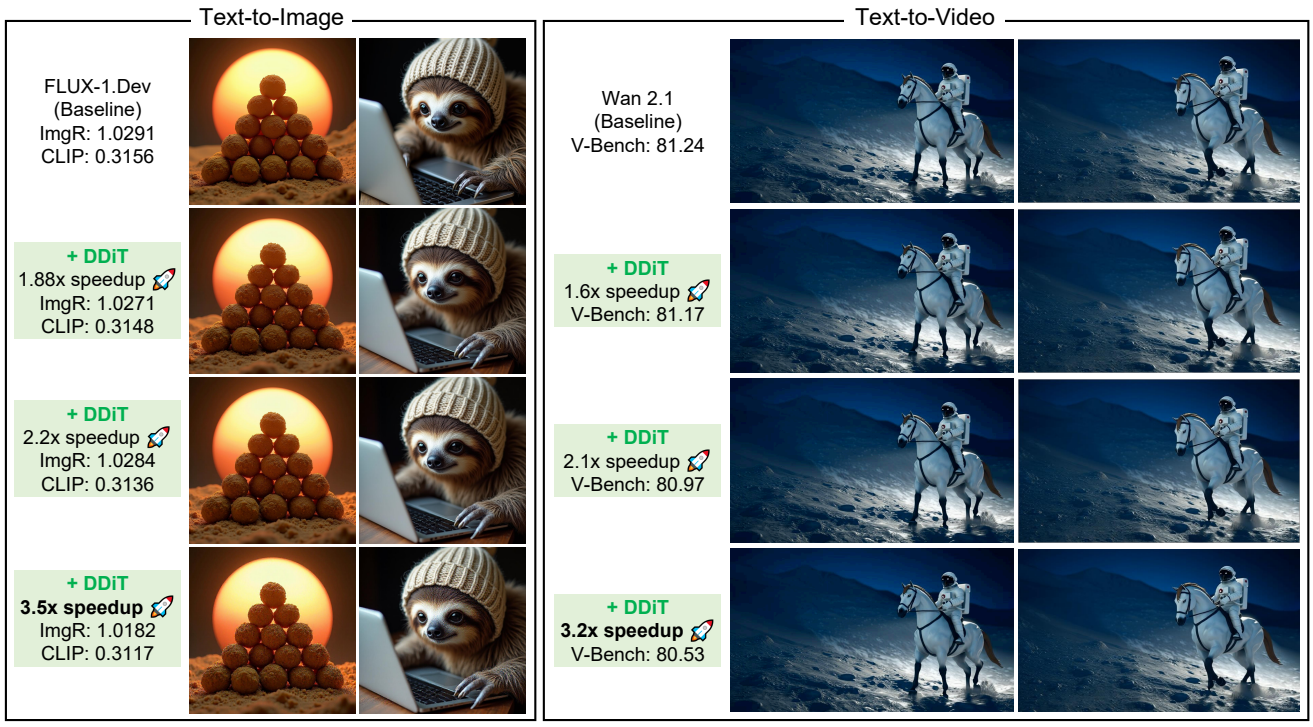

Figure 1. **DDiT dynamically selects the optimal patch size** at each denoising step at inference yielding significant computational gains at no loss of perceptual quality. Results are shown for FLUX-1.Dev [10] for text-to-image and Wan-2.1 [27] for text-to-video generation. The top panel denotes the baseline (original model), while the remaining panels illustrate outputs from DDiT at different acceleration rates. ImageReward [30], CLIP [20], and VBench [8] scores are reported (higher is better).

Abstract

Diffusion Transformers (DiTs) have achieved state-of-the-art performance in image and video generation, but their success comes at the cost of heavy computation. This inefficiency is largely due to the fixed tokenization process, which uses constant-sized patches throughout the entire denoising phase, regardless of the content's complexity. We propose dynamic tokenization*, an efficient* ***test-time strategy*** *that varies patch sizes based on content complexity and the denoising timestep. Our key insight is that early timesteps only require coarser patches to model global structure, while later iterations demand finer (smaller-sized) patches to refine local details. During inference, our method dynamically reallocates patch sizes across denoising steps for image and video generation and substantially reduces cost while preserving perceptual generation quality. Extensive experiments demonstrate the effectiveness of our approach: it achieves up to* $3.52\times$ *and* $3.2\times$ *speedup on FLUX-1.Dev and Wan* 2.1*, respectively, without compromising the generation quality and prompt adherence.*

1. Introduction

Diffusion transformers (DiTs) [3, 19] have emerged as a dominant framework for content generation, producing high-quality and photorealistic results in both image and video synthesis. However, this impressive performance comes with substantial computational cost, significantly limiting the usage of these models in practice. Such high

computational demands of generative models have catapulted the development of more efficient generation methods. Existing research has broadly focused on acceleration techniques such as feature caching [15, 32], feature pruning [1, 4], vector quantization [23, 24], and model distillation [12, 22].

Although these approaches show promise, they suffer from two key limitations. First, many methods [5, 7, 15, 29] typically employ a **hard, static reduction strategy**, such as removing a fixed amount of weights, operations, or tokens. Such static approach can lead to significant quality degradation, as computations critical to a specific output might be permanently discarded [2, 29]. Second, most existing methods [15, 16, 29] apply a rigid, one-size-fits-all strategy, that is agnostic to the input. This is problematic, as different prompts require varying levels of computational detail [17, 28]. A simple prompt like "a blue sky" should not require the same amount of computational resources compared to a prompt "a scene crowded with many zebras." The rigidity in all existing solutions prevents us from dynamically allocating resources where they are needed most.

In this work, we address the rigid, one-size-fits-all computation of existing methods. Our approach is based on a key observation: the visual content generated by a diffusion model evolves at varied levels of detail. Some denoising timesteps establish coarse scene structure, while others refine fine-grained visual details. Recent studies [9, 11, 25, 26] show that features generated at different timesteps of the denoising process encode different information, thus selecting the right timestep of diffusion features is important for successful downstream tasks such as classification [11] and visual reasoning [9]. Furthermore, [18, 28] note that this information can also be used for image editing, catering to different levels of detail in an image [17, 18, 28], and for generating more prompt-aligned images by injecting different levels of prompt information at different timesteps [21].

This leads us to a critical question: **should every denoising step process the latent at the same granularity?** Or, could some steps operate on a coarser latent, thereby yielding computational benefits, while others use a finer latent to preserve detail? Thus, unlike prior works [1, 4] which approach efficiency by discarding weights or operations, we *dynamically allocate* it. Specifically, at every timestep, we adjust the patch size of the latent and adaptively use larger patches (coarser granularity) when less detail is required and smaller patches (finer granularity) when high fidelity is needed.

This, however, raises a new question: **how do we determine the optimal patch size at any given timestep and for any given prompt?** For this, we measure the *rate of change of the latent manifold* over time. We hypothesize that this rate correlates with the level of detail being generated. If the underlying latent evolves slowly within a short timestep window, we posit that coarse-grained details are being generated. Consequently, we divide the latent into coarser patches and process them, saving computational resources. Conversely, if the underlying latent evolves rapidly, we infer that fine-grained details are being generated and fall back to using finer-grained latent patches.

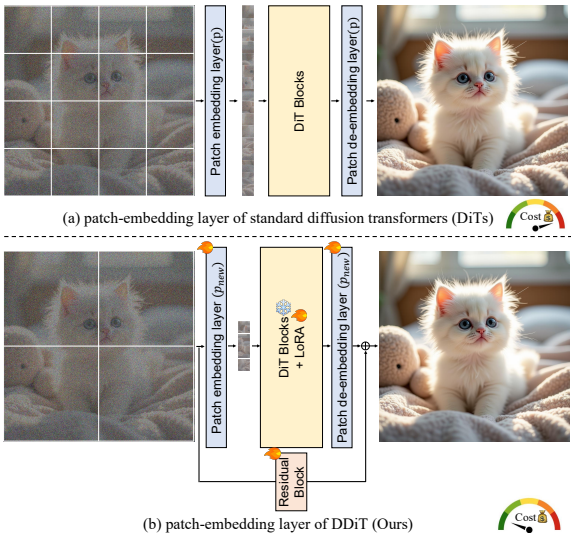

Figure 2. **Revised patch-embedding layer to support patches of varied resolutions.** We modify the standard patch-embedding layer, designed for a fixed patch size p, to additionally support patch sizes p_{new}.

Thus, this dynamic strategy tailors the computation load to each timestep and each prompt, allocating more resources when needed and conserving them where possible. Ultimately, our approach gives us explicit control over the computational budget while generating the highest possible quality content given the computational budget (Fig. 1). In summary, our contributions are:

- We introduce a simple, intuitive, and low-cost strategy to **dynamically vary latent's granularity** in diffusion models, that requires minimal architectural changes.
- We propose a test-time **Dynamic Patch Scheduler** that automatically determines the optimal patch size at each timestep, adapting the computational load based on generation complexity and the input prompt.
- We demonstrate through extensive experiments that our approach generalizes across both image and video diffusion transformers and achieves significant speedups – up to 3.52× on FLUX-1.Dev [10] and 3.2× speedups on Wan 2.1 [27], while maintaining high perceptual quality, photo-realism, and prompt alignment.

2. Approach

2.1. Dynamic Patching and Tokenization

We aim to modify a pre-trained DiT to seamlessly operate under different patch sizes with minimal architectural modifications (Fig. 2). To this end, we adapt the patch embedding layer, originally operating on patch size p, to also

handle new patch sizes p_{new} and allow input latents of varying spatial resolutions. We define p_{new} a positive integer multiple of p, *i.e.*, $\{p, 2p, 4p, ...\}$.

Modifications to the patch embedding layer. To generalize DiT to p_{new}, we introduce patch-specific embedding layers for each patch size we wish to support. Let C denotes the number of latent channels and d represents the embedding dimension. Let $\mathbf{w}^{\text{emb}}_{p_{\text{new}}} \in \mathbb{R}^{p_{\text{new}} \times p_{\text{new}} \times C \times d}$ and $\mathbf{b}^{\text{emb}}_{p_{\text{new}}} \in \mathbb{R}^{d}$ denote the weight matrix and bias vector of the patch embedding layer corresponding to p_{new}. Each patch of size p_{new} is linearly projected into an embedding of dimension d using this newly added embedding layer. This results in a total of $N_{p_{\text{new}}} = \frac{HW}{p^2_{\text{new}}}$ patches. Since $N_{p_{\text{new}}}$ is smaller than N by a factor of $(p_{\text{new}}/p)^2$, DiT now processes fewer patches and yields significant computational gains.

To minimize the training cost, we retain the base model originally trained on the latent patch size p and introduce a Low-Rank Adaptation (LoRA) branch [6] into each transformer block in DiT. This LoRA branch serves as an *adaptive pathway* and enables the model to process patches of different sizes. Additionally, as shown in Fig. 2, we add a residual connection from *before* the patch embedding layer to *after* the patch de-embedding block. This helps strike a balance between the base latent manifold and the new manifold being learnt by LoRA for p_{new}.

Finally, to distill the knowledge from the frozen base model to the LoRA-augmented model, we fine-tune the LoRA branch with a distillation loss. Let ϵ_{θ_L} and ϵ_{θ_T} denote the predicted noise from the LoRA-fine-tuned and frozen base models respectively. The distillation loss is:

$$\mathcal{L} = ||\epsilon_{\theta_L}(\mathbf{z}_t^{p_{\text{new}}}, t) - \epsilon_{\theta_T}(\mathbf{z}_t^{p}, t)||_2^2. \tag{1}$$

These minor architectural tweaks allow us to dynamically support larger patch sizes while maintaining the base model's perceptual output quality.

2.2. Dynamic Patch Scheduling

Now that we enabled processing of multiple size input patches, how do we learn *when* to adapt to a larger patch-size (*i.e.*, coarser token) and when to switch back to a smaller patch-size (*i.e.*, fine-grained token)? To this end, we introduce a dynamic patch scheduling mechanism to determine the appropriate patch size at each diffusion timestep. Since different timesteps correspond to different levels of generative detail [17, 18, 21, 25, 26, 28], selecting the proper patch size at each stage is crucial for maintaining both efficiency and quality. We design a **training-free dynamic scheduler** that adaptively selects the patch size based on the *rate of evolution of the latent representations* within a window of timesteps.

Latent evolution estimation. We employ finite-difference approximations of increasing order to quantify how latent representations evolve during the denoising process. Let $\mathbf{z}_t$ denote the latent at timestep t. The first-order finite difference captures the displacement of latent features between consecutive timesteps:

$$\Delta \mathbf{z}_t = \mathbf{z}_t - \mathbf{z}_{t+1}. \tag{2}$$

Similarly, the second-order difference describes the *rate* of change of this displacement, representing the local velocity of the denoising trajectory, defined by

$$\Delta^{(2)} \mathbf{z}_{t-1} = \Delta \mathbf{z}_{t-1} - \Delta \mathbf{z}_t. \tag{3}$$

Finally, the third-order finite difference quantifies the variation in this velocity. This can be interpreted as a measure of *acceleration* of a latent's evolution during denoising within a short temporal window.

$$\Delta^{(3)} \mathbf{z}_{t-1} = \Delta^{(2)} \mathbf{z}_{t-1} - \Delta^{(2)} \mathbf{z}_t = 2\left(\frac{\Delta \mathbf{z}_{t-1} + \Delta \mathbf{z}_{t+1}}{2} - \Delta \mathbf{z}_t\right), \tag{4}$$

We hypothesize that if the acceleration is slow at a given timestep, there is a relatively minor difference in the underlying latent manifold in the local temporal window. On the other hand, **a high acceleration value suggests a larger difference in the structure of the underlying manifold.** We use this measure as a proxy to identify transition points where the generative process intrinsically shifts between generating coarse to fine structures or vice versa.

Empirically, we find that the third-order difference captures this variation more effectively and remains more stable, while the first- and second-order differences fail to do so, likely because they capture relatively short-term temporal changes. This observation is consistent with [31], which shows that the difference between neighboring noise predictions is explicitly related to the third-order finite difference.

Spatial variance estimation. We use latent z_t in the above formulation for simplicity, but in practice, z_t is always divided into patches of size $p_{\text{new}} \times p_{\text{new}}$ (Sec. 2.1). Our final task now is to select the right patch size at each latent manifold. This requires quantifying and aggregating the acceleration at which the latent patches evolve. Thus, we divide

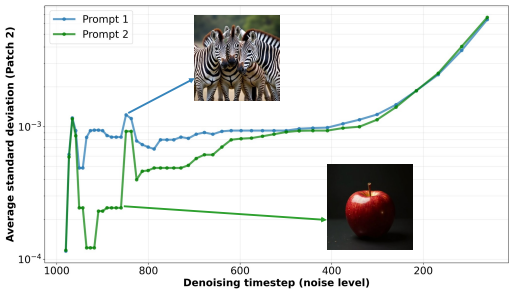

Figure 3. **Visualization of $\boldsymbol{\sigma}^{2p,(\rho)}_{t-1}$ for two prompts (log scale).** *Prompt 1: "Several zebras are standing together behind a fence." Prompt 2: "A simple red apple on a black background."* Prompts requiring different levels of spatial granularity exhibit distinct $\boldsymbol{\sigma}^{2p,(\rho)}_{t-1}$ patterns across timesteps. For the fine-grained zebra pattern, $\boldsymbol{\sigma}^{2p,(\rho)}_{t-1}$ remains higher, indicating higher detail sensitivity, whereas for the simpler apple scene, $\boldsymbol{\sigma}^{2p,(\rho)}_{t-1}$ is lower, thus we can seamlessly use larger patch sizes during generation.

Table 1. **Quantitative comparison of text-to-image generation performance with state-of-the-art methods** on COCO, DrawBench, and PartiPrompts. If not specified, all results are reported using 50 inference steps by default. Each color (**Yellow** , **Blue**) indicates methods operating at similar inference speeds. As highlighted in **Blue** , our method achieves the best overall image quality, evidenced by the lowest FID scores, strong prompt alignment (CLIP and ImageReward), and high perceptual similarity (SSIM and LPIPS). **Bold**: best. Underline: second-best.

Model	Speed↓ (secs/image)	COCO		DrawBench				PartiPrompts	
		FID↓	CLIP↑	CLIP↑	ImgR↑	SSIM↑	LPIPS↓	CLIP↑	ImgR↑
FLUX-1.Dev (50 steps)	12.0	33.07	0.314	0.3156	1.0291	–	–	0.3197	1.192
FLUX-1.Dev (28 steps)	6.72	33.35	0.312	0.3140	1.0107	0.739 ± 0.155	0.175 ± 0.126	0.3118	1.115
FLUX-1.Dev (15 steps)	3.6	34.02	0.311	0.3121	0.9865	0.591 ± 0.155	0.298 ± 0.128	0.3072	0.9613
TeaCache ($\delta = 0.6$)	6.0	34.95	0.303	0.3071	0.9968	$\mathbf{0.654 \pm 0.145}$	$\mathbf{0.241 \pm 0.114}$	0.3018	0.9701
TeaCache ($\delta = 0.8$)	5.33	35.57	0.303	0.3075	0.9780	0.612 ± 0.146	0.279 ± 0.114	0.2991	0.9699
TaylorSeer ($N = 3, O = 2$)	6.0	34.74	0.303	0.3085	0.9721	0.632 ± 0.173	0.271 ± 0.149	0.3142	0.9813
TaylorSeer ($N = 6, O = 1$)	3.5	35.02	0.302	0.3040	0.9535	0.583 ± 0.130	0.284 ± 0.106	0.3004	0.9714
DDiT	5.5	**33.42**	**0.317**	**0.3136**	**1.0284**	0.635 ± 0.183	0.264 ± 0.190	**0.3192**	**1.189**
DDiT + Teacache ($\delta = 0.4$)	3.4	33.60	0.315	0.3117	1.0182	0.592 ± 0.118	0.267 ± 0.120	0.3172	1.062

z_{t-1} into patches of size $p_i \times p_i$, where $p_i \in p_{\text{new}}$. Then, we compute the **standard deviation** $\boldsymbol{\sigma}_{t-1}^{p_i}$ of the acceleration (defined in Eqn. 4) within each patch. We hypothesize that if the per-(latent) pixel standard deviation within a latent patch is high, then the denoising process is focusing on generating finer-grained details. On the other hand, if the standard deviation is low, then the underlying evolving latent is smooth. As shown in Fig. 3, prompts with different levels of granularity exhibit distinct variance profiles across timesteps.

Given $\boldsymbol{\sigma}_{t-1}^{p_i}$, our goal is to determine the appropriate patch size at each timestep. A straightforward way to do this is to aggregate $\boldsymbol{\sigma}_{t-1}^{p_i}$ by taking their mean across patches, which provides a simple measure of the overall latent variation at that timestep. However, this fails to effectively capture the generative dynamics occurring at that timestep. For example, when generating an image containing both a uniform white background and a highly textured region, averaging might smoothen the higher standard deviation values, leading to the scheduler choosing larger patches and thus overlooking fine details in the textured area. To better capture such spatial heterogeneity in the underlying latent manifold, we instead take the ρ-th percentile of the per-patch variances, denoted as $\sigma_{t-1}^{p_i,(\rho)}$. This percentile-based aggregation allows us to capture meaningful information across patches without averaging out important signals, while also avoiding bias toward a few high-variance outliers.

Concretely, we compare $\sigma_{t-1}^{p_i,(\rho)}$ against a predefined variance threshold τ. For each timestep, we select the largest patch size whose corresponding variance is below the threshold (τ). If no such patch satisfies this condition, it defaults to the smallest patch size, which is 1. We formulate this **patch size scheduling** as follows:

$$p_t = \begin{cases} \max(p_i), & \text{if } \sigma_{t-1}^{p_i,(\rho)} < \tau, \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

Controlling τ gives us explicit control over the speed: if users prefer faster generation, a higher τ can be selected; otherwise, a smaller τ can be used for higher quality with less speed gain. We select τ and ρ empirically and balance generation stability and visual quality.

3. Experiments

3.1. Text-to-Image Generation

We evaluate our method on text-to-image generation using FLUX-1.dev [10] and compare against state-of-the-art acceleration methods [13, 14]. Our approach achieves comparable FID and CLIP scores to the base model while providing a $2.18\times$ speedup, consistently outperforming prior methods under similar computational budgets. Furthermore, combining our method with TeaCache [13] yields up to a $3.52\times$ speedup, demonstrating that dynamic patch scheduling improves efficiency while preserving perceptual quality.

3.2. Text-to-Video Generation

We further evaluate the effectiveness of our method in the text-to-video (T2V) generation setting and compare it with the base model [27]. As shown in Fig. 1, our approach significantly reduces inference time while maintaining competitive video quality, as reflected by the VBench score [8].

4. Conclusion

We present an intuitive and highly-computationally efficient method, *DDiT*, to adapt diffusion transformers to patches of different sizes during denoising while maintaining visual quality. *DDiT* demonstrates a critical insight: not all timesteps require the underlying latent space to be equally fine-grained. Building on this insight, we dynamically select the optimal patch size at every timestep and achieve significant computational gains, with no loss in perceptual visual quality. Our approach requires just adding a simple plug-and-play LoRA adapter to make the patch-embedding (and de-embedding) blocks amenable to varied input patch sizes. This minimal architectural tweak allows any DiT-based model to benefit from fast inference.

References

[1] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *CVPR*, 2023. 2

[2] Juncan Deng, Shuaiting Li, Zeyu Wang, Hong Gu, Kedong Xu, and Kejie Huang. Vq4dit: Efficient post-training vector quantization for diffusion transformers. In *AAAI*, 2025. 2

[3] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICLR*, 2024. 1

[4] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *NeurIPS*, 2023. 2

[5] Gongfan Fang, Kunjun Li, Xinyin Ma, and Xinchao Wang. Tinyfusion: Diffusion transformers learned shallow. In *CVPR*, 2025. 2

[6] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022. 3

[7] Taihang Hu, Linxuan Li, Joost van de Weijer, Hongcheng Gao, Fahad Shahbaz Khan, Jian Yang, Ming-Ming Cheng, Kai Wang, and Yaxing Wang. Token merging for training-free semantic binding in text-to-image synthesis. In *NeurIPS*, 2024. 2

[8] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *CVPR*, 2024. 1, 4

[9] Dahye Kim, Xavier Thomas, and Deepti Ghadiyaram. Revelio: Interpreting and leveraging semantic information in diffusion models. In *ICCV*, 2025. 2

[10] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024. 1, 2, 4

[11] Alexander C Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier. In *ICCV*, 2023. 2

[12] 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. In *NeurIPS*, 2023. 2

[13] Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *CVPR*, 2025. 4

[14] Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. *arXiv preprint arXiv:2503.06923*, 2025. 4

[15] Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. In *NeurIPS*, 2024. 2

[16] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *CVPR*, 2024. 2

[17] Shweta Mahajan, Tanzila Rahman, Kwang Moo Yi, and Leonid Sigal. Prompting hard or hardly prompting: Prompt inversion for text-to-image diffusion models. In *CVPR*, 2024. 2, 3

[18] Or Patashnik, Daniel Garibi, Idan Azuri, Hadar Averbuch-Elor, and Daniel Cohen-Or. Localizing object-level shape variations with text-to-image diffusion models. In *ICCV*, 2023. 2, 3

[19] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 1

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1

[21] Ketan Suhaas Saichandran, Xavier Thomas, Prakhar Kaushik, and Deepti Ghadiyaram. Progressive prompt detailing for improved alignment in text-to-image generative models. *arXiv preprint arXiv:2503.17794*, 2025. 2, 3

[22] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 2

[23] Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *CVPR*, 2023. 2

[24] Junhyuk So, Jungwon Lee, Daehyun Ahn, Hyungjun Kim, and Eunhyeok Park. Temporal dynamic quantization for diffusion models. In *NeurIPS*, 2023. 2

[25] Nick Stracke, Stefan Andreas Baumann, Kolja Bauer, Frank Fundel, and Björn Ommer. Cleandift: Diffusion features without noise. In *CVPR*, 2025. 2, 3

[26] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. In *NeurIPS*, 2023. 2, 3

[27] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 1, 2, 4

[28] Luozhou Wang, Shuai Yang, Shu Liu, and Ying-cong Chen. Not all steps are created equal: Selective diffusion distillation for image manipulation. In *ICCV*, 2023. 2, 3

[29] Zhendong Wang, Yifan Jiang, Huangjie Zheng, Peihao Wang, Pengcheng He, Zhangyang Wang, Weizhu Chen, Mingyuan Zhou, et al. Patch diffusion: Faster and more data-efficient training of diffusion models. In *NeurIPS*, 2023. 2

[30] 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. In *NeurIPS*, 2023. 1

[31] Hancheng Ye, Jiakang Yuan, Renqiu Xia, Xiangchao Yan, Tao Chen, Junchi Yan, Botian Shi, and Bo Zhang. Training-free adaptive diffusion with bounded difference approximation strategy. In *NeurIPS*, 2024. 3

[32] Hui Zhang, Tingwei Gao, Jie Shao, and Zuxuan Wu. Blockdance: Reuse structurally similar spatio-temporal features to accelerate diffusion transformers. In *CVPR*, 2025. 2