CoReDiT: Spatial Coherence-Guided Token Pruning and Reconstruction for Efficient Diffusion Transformers

Anonymous CVPR submission

Paper ID 9

Abstract

Diffusion Transformers (DiTs) deliver remarkable image and video generation quality but incur high computational cost, limiting scalability and on-device deployment. We introduce CoReDiT, a structured token pruning framework for DiTs across vision tasks. CoReDiT uses a linear-time spatial coherence *score to estimate local redundancy in the latent token lattice and skips high-coherence (redundant) tokens in self-attention. To maintain a dense representation and avoid visual discontinuities, we reconstruct skipped attention outputs via coherence-guided aggregation of spatially neighboring retained tokens. We further introduce a progressive, block-adaptive pruning schedule that increases pruning gradually and allocates larger budgets to blocks and denoising steps with higher redundancy. Across state-of-the-art diffusion backbones including PixArt-α and MagicDrive-V2, CoReDiT achieves up to* ***55%*** *self-attention FLOPs reduction and inference speedups of* ***1.33***$\times$ *on cloud GPUs and* ***1.72***$\times$ *on mobile NPUs, while maintaining high visual quality. Notably, CoReDiT also increases on-device memory headroom, enabling higher-resolution generation.*

1. Introduction

Diffusion models learn to synthesize data through an iterative denoising process, and have achieved state-of-the-art results across various generative applications, such as conditional image generation, image editing (inpainting and super-resolution) and video synthesis. Building on this success, Diffusion Transformers (DiTs) [21] replace convolutional U-Nets with attention-based backbones that effectively model long-range dependencies among tokenized patches; this architecture scale effectively with model size and facilitate the incorporation of conditioning from various modalities (e.g., class labels, text, and depth maps).

However, these benefits come with demanding computational cost. Self-attention scales quadratically with the number of tokens, making high-resolution images and multi-frame videos particularly expensive and often impractical on compute- and memory-constrained mobile platforms. Empirically, a large fraction of tokens correspond to visually redundant or low-saliency regions (e.g., uniform backgrounds and smooth textures), suggesting substantial room for compute reduction. Prior work reduces inference cost by pruning or merging tokens using attention-based saliency [12, 27], feature similarity [2], or learned routing/predictors [23, 34]. However, several challenges limit their applicability to diffusion inference: (1) localizing low-saliency tokens efficiently and effectively, (2) preserving visual semantics for the tokens that do not participate in attention, and (3) determining a pruning schedule that adapts across transformer blocks and denoising timesteps.

Contributions: To address these challenges, we propose CoReDiT, a token pruning framework for accelerating *pre-trained* DiTs that targets self-attention while preserving output fidelity. CoReDiT consists of three components: (1) *Spatial coherence-based selector* (Sec. 3.1): We observe that redundant tokens in low-saliency regions are highly similar to their spatial neighbors. To exploit this, we partition the token lattice into small, non-overlapping grids and estimate a *spatial coherence* score for each token, based on its feature similarity to tokens in the same grid. This score can be implemented with linear-time, hardware-friendly reductions (e.g., grid means and dot products), adding minimal overhead while reliably selecting redundant tokens (with high coherence) to bypass attention. (2) *Coherence-based reconstruction* (Sec. 3.2): Skipping tokens can break local spatial continuity, leading to artifacts in the generated outputs. To preserve a dense token lattice for subsequent blocks, we reconstruct skipped tokens from retained neighbors using similarity-weighted aggregation within a local neighborhood. This content-aware interpolation (unlike zeroing or naive forwarding) preserves visual semantics and mitigates artifacts while maintaining compatibility with the original DiT architecture. (3) *Progressive, block-adaptive pruning* (Sec. 3.3): Pruning tolerance varies across blocks and timesteps: different blocks carry varying semantic im-

portance, and late diffusion steps are typically less tolerant than early ones [34]. Hence, we fine-tune with a progressive schedule that *automatically allocates* the pruning budget to the most redundant blocks. At regular intervals during fine-tuning, we estimate a timestep-weighted redundancy score for each transformer block and increment the pruning ratio of the block with the highest redundancy; gradually increasing the pruning ratio enables fast recovery after each update and ensures stable adaptation.

Combining these components results in an efficient pruning pipeline that reduces attention computation while preserving high-fidelity generation. Across text-to-image and video generation, CoReDiT enables higher-resolution synthesis and stable conditional alignment, achieving up to 55% reduction in self-attention FLOPs and wall-clock speedups of 1.33× on cloud GPUs and 1.72× on mobile devices.

2. Related Work

Diffusion Models. Diffusion models have demonstrated impressive generative performance across image, video, and 3D domains. Early works commonly adopt U-Net backbones [11, 25], and latent diffusion [24] improves scalability by denoising in a compressed latent space. Recently, Diffusion Transformers (DiTs) [21] replace U-Net architectures with stacks of transformer blocks, offering better scalability to high resolution and flexible conditioning, but incurring high inference cost from quadratic attention and multi-step denoising.

Efficient Diffusion Transformers. Prior studies have improved the efficiency of DiTs along several complementary directions: (1) *Token reduction within a timestep*. Training-free token reduction often merges or downsamples tokens to reduce the effective sequence length, including token merging methods (e.g., ToMe [1]) and diffusion-specific variants (e.g., ToMA [18]) that aim to reduce merge overhead. Other training-free strategies reduce attention cost by downsampling only key/value tokens (e.g., ToDo [28]). Learning-based methods learn per-layer/timestep keep ratios or token routers for content generation, including DiffCR [34] and dynamic token-density schemes such as FlexDiT [5]. A key challenge in DiTs (vs. ViTs) is that naively discarding tokens can introduce spatial discontinuities (texture breaks, boundary artifacts), motivating approaches that either preserve structure via merge/unmerge or explicitly reconstruct skipped content. (2) *Cross-timestep caching and feature reuse*. Caching-based methods exploit temporal redundancy across denoising steps by reusing intermediate features with selective refresh [19, 31]. Recent DiT-specific works further cache at finer granularity (e.g., token-wise feature caching in ToCa [39]). These approaches primarily target *cross-step redundancy* and are conceptually orthogonal to within-step token reduction. *(3) Dynamic computation and conditional routing.* Dynamic DiT architectures adjust compute across timesteps and/or spatial regions, e.g., DyDiT [38] uses timestep-wise dynamic width and spatial token skipping, while DiffCR [34] fine-tunes token routers with differentiable compression ratios. Compared to token reduction, these methods introduce explicit routing modules and typically optimize a dynamic compute graph. *(4) Sparse/structured attention kernels.* Another line accelerates DiTs by replacing full attention with sparse attention patterns and hardware-efficient kernels, including training-free profiling-based sparse attention (e.g., Sparse VideoGen [32]) and trainable sparse attention (e.g., VSA [35]). These methods change the attention operator itself rather than reducing the token set, and can be complementary to token pruning/merging. (5) *Weight-pruning and architecture editing*. Beyond tokens, structured and unstructured pruning remove channels, heads, or even full DiT blocks, with brief recovery fine-tuning or lightweight calibration to retain quality [8]. Other work (e.g., grafting [4]) edits the architectures of pretrained DiTs to explore more efficient backbones under small compute budgets; these approaches are naturally complementary to token sparsity methods.

Positioning of CoReDiT. CoReDiT is a *post-training* method that targets *within-timestep* redundancy in DiT latent grids: it performs *structured token skipping* guided by local spatial coherence, and *reconstructs* skipped tokens to preserve a full lattice for subsequent layers. This differs from caching-based methods (e.g., FasterCache/ToCa) that exploit *cross-timestep reuse*, from dynamic routing approaches (e.g., FlexDiT/DiffCR) that introduce learnable routing policies, and from token-merging methods (e.g., ToMe) that explicitly merge/unmerge tokens. Compared to purely training-free schemes, CoReDiT uses lightweight fine-tuning to learn block/timestep-specific pruning schedules to improve hardware-relevant speed/quality trade-offs.

3. Proposed Approach: CoReDiT

Fig. 1 summarizes the workflow of CoReDiT: In each transformer block, we partition the input tokens (patch embeddings) X into a retained set X_R and a skipped set X_S according to a spatial coherence score that quantifies local redundancy of each token; only X_R participates in multi-head self-attention, producing $Y_R = \mathrm{MHSA}(X_R)$ (Sec. 3.1). For skipped tokens X_S, we synthesize their attention outputs Y_S using nearby retained outputs Y_R with coherence-guided weights (Sec. 3.2). Importantly, *attention among retained tokens remains global*; locality is used only for efficient selection and reconstruction, not to impose a local-attention pattern. To determine which block to prune more aggressively, we learn a progressive, block-adaptive pruning schedule during fine-tuning of a pretrained DiT, based

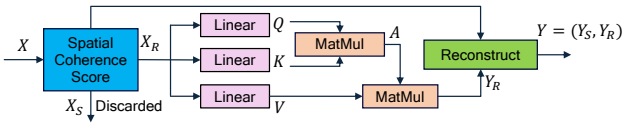


Figure 1. Workflow of CoReDiT. In each transformer block, input tokens X are partitioned into retained tokens X_R and skipped tokens X_S using the proposed spatial coherence score that captures *local redundancy* (Sec. 3.1). Only the retained tokens X_R participate in self-attention, reducing attention compute by skipping X_S. We then reconstruct the missing attention outputs Y_S from nearby retained outputs Y_R via coherence-guided aggregation (Sec. 3.2), preserving a dense token lattice for subsequent blocks. Finally, we learn an adaptive pruning-ratio schedule via coherence-guided progressive pruning during lightweight fine-tuning (Sec. 3.3).

on the estimated redundancy statistics (Sec. 3.3).

3.1. Token Selection

Motivation. A common token pruning criterion is attention score-based importance [12, 27]: Let $A = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})$ denote the attention matrix, where the column-sum in A measures the total attention a token receives. As Fig. 2a shows, pruning high-attention tokens severely damages visual semantics (salient regions), whereas pruning low-attention tokens tends to be less harmful (background regions). However, computing global attention score A requires quadratic time and memory in token count, which substantially offsets pruning gains. Our key observation is that low-saliency tokens typically exhibit *strong local redundancy*: their embeddings are highly similar to spatially nearby tokens. Motivated by this, we design an efficient selection score based on local similarity that avoids forming the full $N \times N$ attention matrix.

Spatial Coherence (SC). Given N tokens $X = \{x_i\}_{i=1}^N$ reshaped into a 2D spatial lattice $H \times W$ (given $N = HW$), we partition tokens into non-overlapping spatial grids of size $w \times w$ to estimate local (rather than global) similarity (Fig. 2b). For a token x_i, define its spatial coherence (SC) as the mean cosine similarity to other tokens in the same grid:

$$\text{SC}(x_i) := \frac{1}{|\mathcal{G}(i)|} \sum_{j \in \mathcal{G}(i)} \text{sim}(x_i, x_j) \tag{1}$$

where $\mathcal{G}(i)$ indexes tokens in the same grid as token x_i. Specifically, we adopt cosine similarity $\text{sim}(x_i, x_j) = \frac{x_i^T x_j}{||x_i||_2 ||x_j||_2}$ due to its computational efficiency and strong empirical performance in prior work [2]. Averaging within each grid normalizes SC values across blocks with different w, enabling comparisons in our block-adaptive schedule (Sec. 3.3).

Efficient computation and overhead. Naively computing within-grid pairwise similarities costs $O(w^2 N)$ dot products. Instead, let $\hat{x}_i = \frac{x_i}{||x_i||_2}$ denote the normalized token embedding, and $\hat{g}_i = \frac{1}{|\mathcal{G}(i)|} \sum_{j \in \mathcal{G}(i)} \hat{x}_j$ denote the mean normalized embeddings within the grid of token x_i. Then we can compute the spatial coherence score efficiently:

$$\text{SC}(x_i) = \frac{1}{|\mathcal{G}(i)|} \sum_{j \in \mathcal{G}(i)} \hat{x}_i{}^T \hat{x}_j = \frac{\hat{x}_i{}^T}{|\mathcal{G}(i)|} \sum_{j \in \mathcal{G}(i)} \hat{x}_j = \hat{x}_i{}^T \hat{g}_i. \tag{2}$$

Essentially, the spatial coherence is an inner product between a token's normalized embedding and the mean normalized embedding of its grid. Thus, SC can be computed with one reduction per grid plus one dot product per token, i.e., $O(N)$ time, which is linear in token count, regardless of grid size w.

Selection rule. High SC indicates a token is well-explained by nearby patches (redundant), so we skip the top-K tokens by SC:

$$X_S = \text{Top}_K \left(\{\text{SC}(x_i)\}_{i=1}^N \right), \quad X_R = X \setminus X_S \tag{3}$$

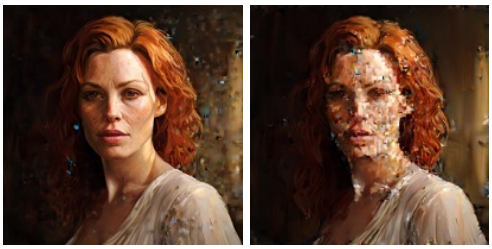
where $K = r(l, s) \cdot N$ is determined by the pruning ratio $r(l, s)$ for block l and timestep s. Empirically, as shown in Fig. 2b, high-SC tokens (brighter colors) concentrate in backgrounds, while boundaries/textures tend to have lower SC. Note that our proposed spatial coherence is not only used for token selection, but also utilized to update skipped tokens and guide our progressive pruning (Sec. 3.3).

3.2. Token Reconstruction

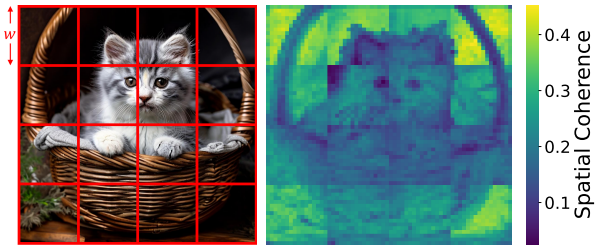
Motivation. Unlike ViTs for discriminative tasks (e.g., classification and object detection) where dropping tokens can be tolerated [15], generative DiTs require spatially coherent representations at every block and timestep. Naively skipping tokens in MHSA can introduce discontinuities (inconsistent textures, distorted boundaries), as in Fig. 3a. We therefore reconstruct the missing attention outputs for skipped tokens using nearby retained outputs, preserving a dense lattice for subsequent blocks. Because our selector preferentially skips *redundant* (high-SC) tokens, these tokens are particularly well-suited to reconstruction from neighbors.

Coherence-based reconstruction. Let X_R be tokens retained for attention and $Y_R = \text{MHSA}(X_R)$. For a skipped token $x_i \in X_S$, a natural reconstruction is similarity-weighted aggregation over nearby retained tokens:

$$y_i = \frac{\sum_{j \in \mathcal{S}(i)} \text{sim}(x_i, x_j)\, y_j}{\sum_{k \in \mathcal{S}(i)} \text{sim}(x_i, x_k)} \tag{4}$$



(a) Token pruning with attention scores-based importance: removing tokens with the lowest scores (left) vs. the highest scores (right).



(b) Example generated from PixArt-α-1024. Left: image with grid partition overlay. Right: computed SC scores over 64×64 tokens.

Figure 2. Token selection motivation and visualization.

(a) Simply skipping tokens (b) Visual artifacts on border (c) m-stride token retention

Figure 3. Issues and proposed improvements. (a) Naively skipping tokens causes inconsistent textures, which can be mitigated by token reconstruction. (b) Using a fixed grid size during partitioning causes visual artifacts along grid borders; alternating grid sizes across consecutive blocks mitigates these artifacts. (c) m-stride token retention ensures at least one token is retained in each sub-grid to serve as a reconstruction anchor.

where $\mathcal{S}(i)$ denotes retained tokens in a local neighborhood of x_i. Intuitively, the transformation result of a token should be more alike to that of a token with a higher similarity. However, computing $\text{sim}(x_i, x_j)$ per pair adds non-trivial overhead.

Approximate sim(x_i, x_j) with SC(x_j). Skipped tokens are selected to have *high* coherence within their grid (Sec. 3.1), meaning $\hat{x_i}$ is close to the grid mean $\hat{g_i}$. Formally, define $\epsilon_i = \|\hat{x_i} - \hat{g_i}\|_2$; high-SC implies small ϵ_i. Then for any $j \in \mathcal{G}(i)$:

$$\text{sim}(x_i, x_j) = \hat{x_i}^T \hat{x_j} = \hat{g_i}^T \hat{x_j} + (\hat{x_i} - \hat{g_i})^T \hat{x_j} = \text{SC}(x_j) + O(\epsilon_i) \tag{5}$$

since $\|\hat{x_j}\|_2 = 1$ bounds the residual term by ϵ_i. Thus, for the tokens we prune (high-SC, small ϵ_i), using SC(x_j) yields a close approximation to similarity-based weights while avoiding pairwise similarity computation. We therefore approximate Eq. (4) as:

$$y_i \approx \frac{\sum_{j \in \mathcal{S}(i)} \text{SC}(x_j)\, y_j}{\sum_{k \in \mathcal{S}(i)} \text{SC}(x_k)}. \tag{6}$$

Sub-grids and complexity. If $\mathcal{S}(i)$ spans an entire grid, Eq. (6) may over-smooth and reduce spatial variation. We address this by subdividing each $w \times w$ grid into smaller *sub-grids* of size $w_s \times w_s$ and defining $\mathcal{S}(i)$ as retained tokens in the *same sub-grid* as x_i. This preserves locality at boundaries while keeping computation linear. Reconstruction uses only local weighted sums over at most w_s^2 tokens per skipped token; with small w_s (e.g., 3), the overhead is modest and scales linearly with token count, i.e., $O(N)$. Notably, this locality is *only* for reconstruction; MHSA among X_R is still global, so CoReDiT does not impose a local attention pattern.

Micro designs. (1) Alternating grid size: Fixed grid boundaries can introduce border artifacts due to limited inter-grid mixing in reconstruction (Fig. 3b). We mitigate this by alternating grid sizes across blocks (e.g., for 64×64 tokens, alternate grid size 16×16 and 9×9), so boundaries shift and information can propagate across grid borders over consecutive blocks. This strategy enables cross-border information flow over consecutive blocks, eliminating visible discontinuities. (2) m-stride token retention: In highly redundant regions, an entire sub-grid could be skipped, leaving no anchors for reconstruction. We therefore enforce retaining at least one token per m positions ($m \leq w_s$) by protecting a deterministic stride pattern. For block index l, we retain tokens at spatial coordinates $[i, j]$ satisfying $(i + j - l) \bmod m = 0$ by subtracting a large offset from their SC so they are never selected into X_S.

3.3. Pruning Ratio Schedule

Motivation. Token redundancy varies across both denoising timesteps and transformer blocks. Fig. 4 depicts the evolution of spatial coherence across denoising steps and blocks. As shown, early steps display broadly elevated coherence (coarse structure), mid steps accentuate background redundancy, and late steps concentrate low-coherence pockets around edges and details [5, 34]; besides, within a timestep, different blocks exhibit different degrees of locality [4]. Motivated by this, we propose adapting the

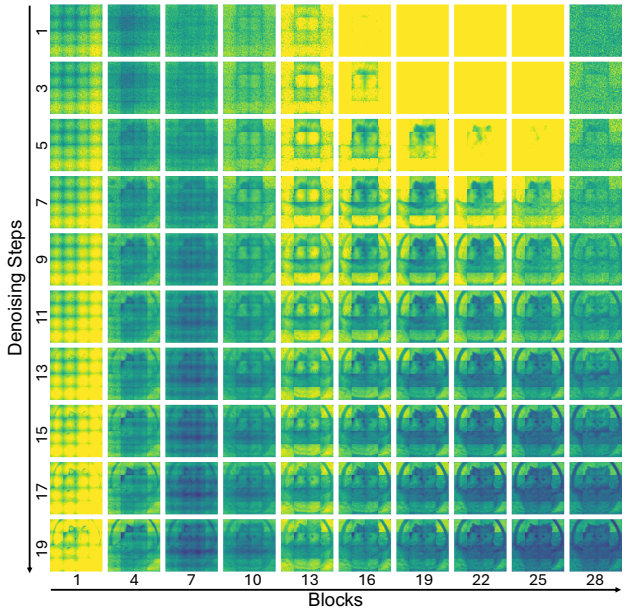


Figure 4. SC score evolution across transformer blocks and denoising timesteps (brighter = higher SC) for generating the image in Fig. 2b. (a) Early timestep in the first block: representations are noise-dominated and exhibit high local redundancy. (b) Early timestep in the last block: weaker spatial structure and reduced locality. (c) Late timestep in the first block: refined spatial details with clearer object-level coherence. (d) Late timestep in the last block: partial refinement with moderate spatial structure. These patterns indicate that redundancy varies with both timestep and block, motivating adaptive pruning ratios across depth and denoising phase.

pruning ratio across blocks based on the statistics observed from real data. Specifically, we adopt progressive pruning during fine-tuning: gradually increasing pruning budgets while prioritizing blocks/timesteps with higher measured redundancy. This progressive design also conforms to the empirical finding in [33]: Applying a large pruning ratio abruptly to a pretrained model can be hard to recover, while progressive pruning leads to better results at comparable pruning levels.

Coherence-based progressive pruning. Let $K(l,s) = r(l,s) \cdot N$ denote the number of tokens pruned in block l at denoising step s. During fine-tuning, we increase $K(l,s)$ in increments of Δk every T training iterations until reaching a target budget. To decide which block to increase next, we sort the SC scores in block l in descending order, $\{\mathrm{SC}_i^l\}_{i=1}^N$. We estimate the redundancy of the *next* Δk candidate tokens as:

$$\Delta R_l = \sum_{i=K(l,s)+1}^{K(l,s)+\Delta k} \mathrm{SC}_i^l, \tag{7}$$

where ΔR_l is computed on the current minibatch. Every T fine-tuning iterations, we assign the next increment Δk to the block with largest ΔR_l, i.e., the block whose next pruned tokens are most redundant.

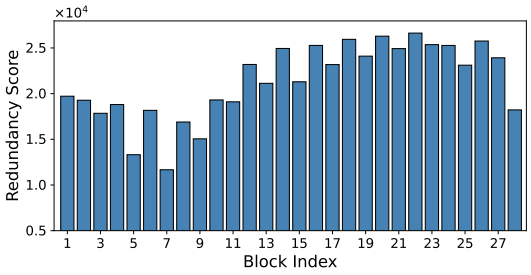
Timestep-wise decay. Pruning schedule should also reflect denoising phase: Early timesteps are noise-dominated and set global structure (more redundancy), while late timesteps refine local details (less redundancy). Accordingly, we adopt more aggressive pruning early and reduce pruning late. In principle, one could learn a distinct pruning ratio for each denoising step. However, changing $r(l,s)$ across steps changes how many tokens are retained and therefore modifies the execution path of MHSA and reconstruction. Implementing a fully step-specific schedule would require instantiating (and often caching) many step-dependent computation graphs and associated buffers (e.g., on mobile NPUs), which increases memory footprint and engineering complexity. We therefore adopt a lightweight two-phase decay that captures the major redundancy shift from early (noise-dominated) to late (detail-refinement) denoising, inspired by prior studies [5, 34], while requiring only *two* graphs per block and keeping memory overhead low: $r(l,s) = \begin{cases} r(l), & 1 \le s \le s_0 \\ c \cdot r(l), & s_0 < s \le S \end{cases}$ where $r(l)$ is the base pruning ratio for block l, S is the total number of denoising steps, s_0 is the phase boundary, and $0 < c < 1$ is the late-phase decay.

Empirical study. We demonstrate progressive pruning on PixArt-α-1024 by increasing the pruned tokens by $\Delta k = 64$ every $T = 15$ fine-tuning steps, with a per-block cap of 60% tokens. Fig. 5a shows that blocks differ substantially in redundancy at initialization. Consequently, the learned schedule (Fig. 5b) allocates higher pruning ratios to consistently redundant blocks while keeping low-redundancy blocks lightly pruned (e.g., blocks 5 and 7). As a result, CoReDiT rapidly reduces attention and end-to-end FLOPs (Fig. 6a) while maintaining generation quality (Fig. 6b).

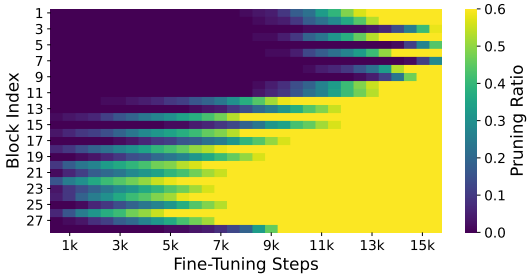
4. Results

4.1. Experimental Setup

Models and training datasets. For text-to-image generation, we apply CoReDiT to the official checkpoints of PixArt-α-1024 [6] and PixArt-Σ-2048 [7]. We perform lightweight fine-tuning with progressive pruning on the 10K synthetic dataset from CLEAR [17], where images are generated by FLUX.1-dev [13]. For video generation, we apply CoReDiT to MagicDrive-V2 [9] and fine-tune the released third-stage checkpoint on nuScenes [3].

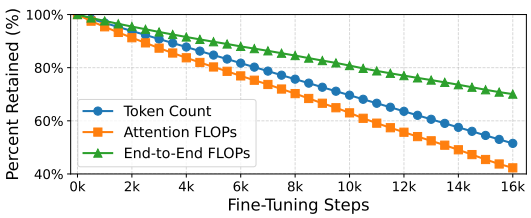


(a) Block-wise redundancy of the next Δk candidate pruned tokens at initialization (computed via Eq. (7)).

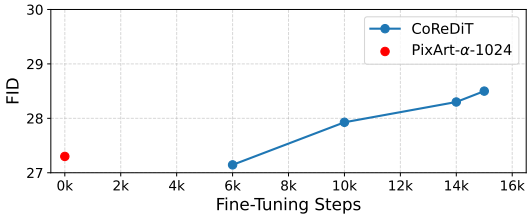


(b) Learned pruning ratios over fine-tuning.

Figure 5. Coherence-guided progressive pruning across transformer blocks for PixArt-α-1024. Higher redundancy blocks receive larger pruning ratios as fine-tuning proceeds.



(a) Efficiency during progressive pruning: retained-token ratio, attention FLOPs, and end-to-end FLOPs.



(b) Quality during progressive pruning (FID). Red dot: pretrained baseline; blue curve: CoReDiT during fine-tuning.

Figure 6. Progressive pruning dynamics for PixArt-α-1024: efficiency gains and quality trend over fine-tuning.

Progressive pruning configuration. For PixArt-α-1024, we fine-tune with batch size of 20 on a single Nvidia H100 GPU; forPixArt-Σ-2048, we use a total batch size of 40 across 8 Nvidia H100 GPUs. Unless otherwise stated, we update the pruning schedule every $T = 15$ fine-tuning steps by pruning additional $\Delta k = 64$ tokens in the selected block, with a per-block pruning ratio cap of 60%; we use a two-phase timestep schedule with decay $c = 0.25$ and $S - s_0 = 5$, i.e., decay only in the final 5 denoising steps, as detailed in Sec. 3.3. Following CLEAR [17], we apply distillation during fine-tuning using both model output loss L_{pred} and the pruned-block loss L_{attn}: $L_{distill} = L_{ori} + \alpha L_{pred} + \beta L_{attn}$, following their hyperparameters $\alpha = \beta = 0.5$. For MagicDrive-V2, we prune only the *spatial* transformer blocks, since they represent the main computational bottleneck, with a significantly higher token count than the temporal blocks. We fine-tune the third-stage checkpoint with SP size of 8 across 8 Nvidia H100 GPUs.

Evaluation metrics. For text-to-image generation, following CLEAR [17], we evaluate on 10k caption-image pairs randomly sampled from MSCOCO 2014 validation [16] using FID [10], CLIP text similarity [22], and Inception Score (IS) [26]. For video generation, we report FVD [29], LPIPS [36], PSNR, and SSIM [30] for video quality; we also evaluate conditional alignment using BEVFormer [14] with mAP and mIoU.

4.2. Main Results: PixArt-α-1024

Quality metrics. Tab. 1 compares PixArt-α-1024 with representative token-reduction and caching baselines [1, 8, 20, 37]. At $r = 40\%$ average pruning, CoReDiT reduces self-attention FLOPs by 48% (24% total FLOPs) while maintaining quality close to the baseline (FID 28.7 vs. 27.3). Notably, CLIP and IS remain competitive, indicating that semantic alignment is preserved under substantial attention compute reduction. With distillation, CoReDiT further narrows the FID gap (27.4) with only minor changes in IS. Fig. 7 shows that CoReDiT preserves global semantics and fine details without introducing noticeable artifacts at 40% pruning.

Efficiency and latency. Tab. 2 reports both self-attention and end-to-end latency on Nvidia H100 GPUs under two attention backends: (i) highly-optimized kernels XFORMERS.MEMORY_EFFICIENT_ATTENTION (Efficient Attn) and (ii) PyTorch native attention implementations (Native Attn). With 45% pruning, CoReDiT reduces self-attention latency by 26% under the efficient attention backend, but end-to-end latency improves by 11% because non-attention components (e.g., FFN/MLP, I/O, and kernel launch overheads) remain unchanged. Under native attention, latency improvements track the reduced quadratic attention cost more closely (37% in self-attention and 25% end-to-end).

Beyond GPU inference, Fig. 8 shows per-block latency on Qualcomm Snapdragon 8 Elite NPUs. At 50% pruning, CoReDiT achieves up to 1.72× per-block speedup at

Model	FLOPs Reduction		Image Quality		
	Self-Attn	End-to-end	FID ↓	CLIP ↑	IS ↑
PixArt-α-1024	-	-	27.3	31.6	37.77
ToMeSD (25% ratio) [1]	-	-7%	174.6	30.2	11.68
DiffPruning [8]	-	-9%	34.6	32.0	-
EcoDiff [37]	-	-9%	32.2	32.0	-
DeepCache (N=2) [20]	-	-25%	31.6	33.1	37.44
CoReDiT ($r = 40\%$)	-48%	-24%	28.7	32.1	37.96
CoReDiT ($r = 40\%$) + distillation	-48%	-24%	27.4	31.9	36.85
CoReDiT ($r = 45\%$)	-55%	-28%	29.3	31.9	36.67
CoReDiT ($r = 45\%$) + distillation	-55%	-28%	28.5	31.9	36.65

Table 1. Results on PixArt-α-1024, where r denotes the average pruning ratio (across blocks and timesteps) in CoReDiT.

Model	FLOPs Reduction		Latency (Efficient Attn)		Latency (Native Attn)	
	Self-Attn	End-to-end	Self-Attn	End-to-end	Self-Attn	End-to-end
PixArt-α-1024	-	-	0.68s	1.68s	2.16s	3.17s
CoReDiT ($r = 45\%$)	-55%	-28%	0.50s (-26%)	1.50s (-11%)	1.36s (-37%)	2.38s (-25%)

Table 2. Comparison of FLOPs and latency on Nvidia H100 GPUs, measured with batch size 64.

1600 resolution. Notably, the baseline PixArt-α encounters out-of-memory (OOM) error at resolutions above 1600, whereas CoReDiT runs up to 1920 resolution and matches the baseline latency at 1600. These results indicate that structured attention skipping can provide not only runtime gains but also meaningful *memory headroom* for higher-resolution deployment.

4.3. High Resolution: PixArt-Σ-2048

We apply CoReDiT to PixArt-Σ-2048 to evaluate high-resolution image generation. As shown in Tab. 3, at an average pruning level of 23%, CoReDiT reduces self-attention FLOPs by 32% (25% total FLOPs), translating into a 22% reduction in self-attention latency and a 15% end-to-end latency reduction (batch size 32, memory-efficient attention). Quality remains high: CLIP remains the same, while FID and IS exhibit only a modest drop (26.0 vs. 28.0 and 37.49 vs. 36.52, respectively). We attribute part of this gap to a resolution-domain mismatch: our fine-tuning dataset is at 1K (upscaled to 2K) while evaluation uses native 2K images. Upscaled 1K data can under-represent 2K high-frequency details and long-range structures; fine-tuning on native 2K images is expected to narrow the remaining quality gap.

4.4. Video Generation: MagicDrive-V2

We extend CoReDiT, which is broadly applicable across vision tasks and agnostic to input modalities, to the state-of-the-art video generation DiT for autonomous driving - MagicDrive-V2 [9]. We prune only spatial transformer blocks and achieve a 39% self-attention FLOPs reduction (8% total FLOPs at the full-model level) as shown in Tab. 4. Despite this reduction, CoReDiT preserves strong per-frame quality (PSNR, LPIPS, and SSIM remain clost to the baseline) and maintains conditional aligment, as measured by BEVFormer mAP and mIoU.

Currently, CoReDiT is applied only to spatial transformer blocks in which each frame is processed independently. Consequently, we observed a drop in FVD, which reflects temporal consistency in the generated video. Future work will focus on integrating temporal consistency objectives, such as cross-frame coherence, to mitigate this limitation. We expect such enhancements will improve FVD performance while maintaining the computational efficiency demonstrated by our current approach.

4.5. Ablation Study

Tab. 5 presents our ablation studies to evaluate key components of CoReDiT on PixArt-α-1024 (all with distillation): (1) *Random Token Selection.* To evaluate the importance of spatial-coherence-based token selection, we conduct experiments of randomly sampling tokens to skip in each block; this yields extremely poor quality (FID 644.5 at 23% pruning), proving that spatial-coherence based selection is effective. (2) *Skipping Tokens without Reconstruction.* To illustrate the contribution of reconstruction, we disable it and simply skip selected tokens. At 30% pruning ratio, FID degrades to 30.4, indicating that reconstruction is necessary to preserve semantic fidelity. (3) *Uniform Pruning then Fine-tuning.* To justify the effectiveness of progressive pruning, we apply a uniform 50% per-block pruning to the pretrained model and then fine-tune; this results in worse FID 32.3 at average 40.6% pruning, highlighting the advantage of progressive, block-adaptive pruning.

(a) Pretrained PixArt-α-1024 model [6]

(b) Proposed CoReDiT with 40% FLOPs Pruning. The semantics are preserved and no additional artifacts.

Figure 7. Qualitative comparison to PixArt-α-1024.

Model (ratio)	FLOPs Reduction		Latency		Image Quality		
	Self-Attn	End-to-end	Self-Attn	End-to-end	FID ↓	CLIP ↑	IS ↑
PixArt-Σ-2048	-	-	4.62s	6.68s	26.0	31.4	37.49
CoReDiT ($r = 23\%$) + distillation	-32%	-25%	3.61s (-22%)	5.67s (-15%)	28.0	31.4	36.52

Table 3. Results on PixArt-Σ-2048, with latency measured on Nvidia H100 GPUs using Efficient Attention.

Model	FLOPs Reduction		Video Quality				Cond. Alignment	
	Self-Attn	End-to-end	PSNR ↑	LPIPS ↓	SSIM ↑	FVD ↓	mAP ↑	mIoU ↑
MagicDrive-V2	-	-	14.28	0.422	0.372	107.8	18.4%	21.5%
CoReDiT ($r = 26\%$)	-39%	-8%	14.25	0.424	0.378	119.8	18.1%	21.2%

Table 4. Video generation results on video quality and conditional alignment via 3D object detection

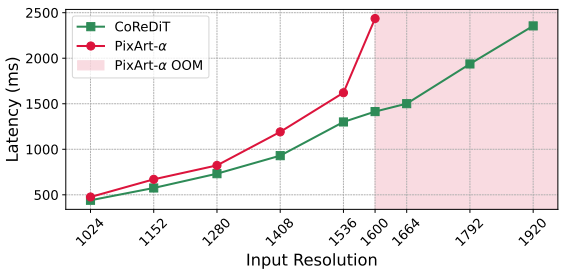


Figure 8. Per-block latency comparison on Qualcomm Snapdragon 8 Elite NPUs. OOM: out-of-memory.

Experiment	Image Quality		
	FID ↓	CLIP ↑	IS ↑
PixArt-α-1024	27.3	31.6	37.77
CoReDiT ($r = 45\%$)	28.5	31.9	36.65
Random selection ($r = 23\%$)	644.5	21.5	1.00
Disable reconstruction ($r = 30\%$)	30.4	31.7	34.20
Uniform ratio ($r = 41\%$)	32.3	31.5	33.68

Table 5. Ablation study on PixArt-α-1024, with distillation in all experiments.

5. Conclusion

We introduce CoReDiT, a token pruning framework for DiTs that exploits spatial coherence as a practical signal of local redundancy. CoReDiT combines (1) an efficient pre-attention selector that skips high-coherence tokens, (2) a coherence-guided reconstruction operator that preserves a dense token lattice and mitigates pruning artifacts, and (3) a progressive, block-adaptive pruning schedule that allocates pruning where redundancy is highest across blocks and denoising steps. Experiments on state-of-the-art image and video generators demonstrate substantial reductions in self-attention compute and measurable end-to-end speedups on both cloud GPUs and mobile NPUs, while maintaining strong perceptual quality and conditional alignment. In future work, we plan to (1) further reduce non-attention overhead by extending efficiency to FFN/MLP components, and (2) improve video temporal consistency by incorporating cross-frame coherence objectives and pruning strategies that explicitly model temporal redundancy.

References

[1] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4599–4603, 2023. 2, 6, 7

[2] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022. 1, 3

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 5

[4] Keshigeyan Chandrasegaran, Michael Poli, Daniel Y Fu, Dongjun Kim, Lea M Hadzic, Manling Li, Agrim Gupta, Stefano Massaroli, Azalia Mirhoseini, Juan Carlos Niebles, et al. Exploring diffusion transformer designs via grafting. *arXiv preprint arXiv:2506.05340*, 2025. 2, 4

[5] Shuning Chang, Pichao Wang, Jiasheng Tang, and Yi Yang. Flexdit: Dynamic token density control for diffusion transformer. *arXiv preprint arXiv:2412.06028*, 2024. 2, 4, 5

[6] 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. 5, 8

[7] Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-σ: Weak-to-strong training of diffusion transformer for 4k text-to-image generation. In *European Conference on Computer Vision*, pages 74–91. Springer, 2024. 5

[8] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *Advances in Neural Information Processing Systems*, 2023. 2, 6, 7

[9] Ruiyuan Gao, Kai Chen, Bo Xiao, Lanqing Hong, Zhenguo Li, and Qiang Xu. Magicdrive-v2: High-resolution long video generation for autonomous driving with adaptive control. *arXiv preprint arXiv:2411.13807*, 2024. 5, 7

[10] 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. 6

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2

[12] Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 784–794, 2022. 1, 3

[13] Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024. 5

[14] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *arXiv preprint arXiv:2203.17270*, 2022. 6

[15] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. *arXiv preprint arXiv:2202.07800*, 2022. 3

[16] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 6

[17] Songhua Liu, Zhenxiong Tan, and Xinchao Wang. Clear: Conv-like linearization revs pre-trained diffusion transformers up. *arXiv preprint arXiv:2412.16112*, 2024. 5, 6

[18] Wenbo Lu, Shaoyi Zheng, Yuxuan Xia, and Shengjie Wang. Toma: Token merge with attention for diffusion models. *arXiv preprint arXiv:2509.10918*, 2025. 2

[19] Zhengyao Lv, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K Wong. Fastercache: Training-free video diffusion model acceleration with high quality. *arXiv preprint arXiv:2410.19355*, 2024. 2

[20] 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. 6, 7

[21] 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. 1, 2

[22] 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 *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 6

[23] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021. 1

[24] 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. 2

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2

[26] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, et al. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016. 6

[27] Manish Kumar Singh, Rajeev Yasarla, Hong Cai, Mingu Lee, and Fatih Porikli. Tosa: Token selective attention for efficient vision transformers. *arXiv preprint arXiv:2406.08816*, 2024. 1, 3

[28] Ethan Smith, Nayan Saxena, and Aninda Saha. Todo: Token downsampling for efficient generation of high-resolution images. *arXiv preprint arXiv:2402.13573*, 2024. 2

[29] Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018. 6

[30] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6

[31] Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, 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. 2

[32] Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025. 2

[33] Huanrui Yang, Hongxu Yin, Maying Shen, Pavlo Molchanov, Hai Li, and Jan Kautz. Global vision transformer pruning with hessian-aware saliency. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18547–18557, 2023. 5

[34] Haoran You, Connelly Barnes, Yuqian Zhou, Yan Kang, Zhenbang Du, Wei Zhou, Lingzhi Zhang, Yotam Nitzan, Xiaoyang Liu, Zhe Lin, et al. Layer-and timestep-adaptive differentiable token compression ratios for efficient diffusion transformers. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18072–18082, 2025. 1, 2, 4, 5

[35] Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. Vsa: Faster video diffusion with trainable sparse attention. *arXiv preprint arXiv:2505.13389*, 2025. 2

[36] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[37] Yang Zhang, Er Jin, Yanfei Dong, Ashkan Khakzar, Philip Torr, Johannes Stegmaier, and Kenji Kawaguchi. Effortless efficiency: Low-cost pruning of diffusion models. *arXiv preprint arXiv:2412.02852*, 2024. 6, 7

[38] Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Yibing Song, Gao Huang, Fan Wang, and Yang You. Dynamic diffusion transformer. *ICLR*, 2025. 2

[39] Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. *arXiv preprint arXiv:2410.05317*, 2024. 2