iCATS: Fast Video Generation via Interaction-Aware Sparse Attention and Timestep-Adaptive Sparsity

Chengfeng Han[1,2*] **Ang Wang**[3] **Baole Ai**[3] **Xianlu Bian**[1,2*] **Jie Yao**[2]
Zilong Huang[2] **Dandan Ding**[1†]

[1]School of Information Science and Technology, Hangzhou Normal University
[2]T-Head Semiconductor Co., Ltd. [3]Alibaba Group

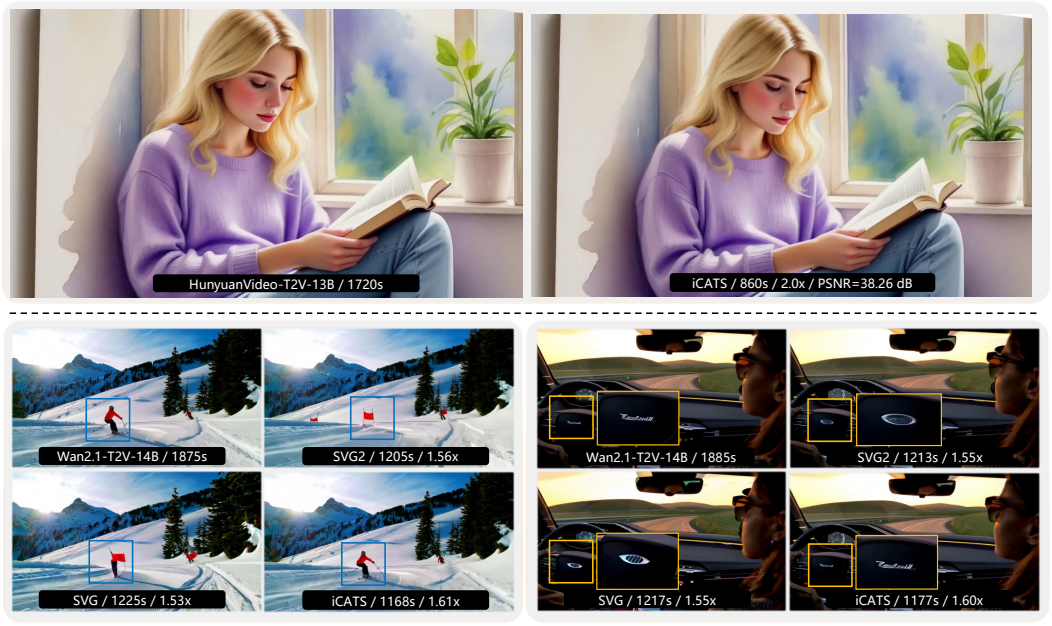


Figure 1: Visualizations on HunyuanVideo-T2V-13B and Wan2.1-T2V-14B show that iCATS better preserves visual details while achieving higher speedup than other training-free methods, producing results closer to native videos. On average, it achieves 2.03× acceleration with 31.017 dB PSNR on HunyuanVideo-T2V-13B, and 1.55× acceleration with 29.301 dB PSNR on Wan2.1-T2V-14B.

Abstract

Training-free sparse attention offers a practical acceleration solution to Diffusion Transformers (DiTs) via reducing computations without fine-tuning. It typically involves estimating the importance of query-key regions and deriving sparse masks to compute only the important candidates, which inevitably introduces approximation errors that may degrade generation quality. To better balance the efficiency-quality trade-off, we propose iCATS, integrating improved importance estimation and sparse mask construction with an efficient hardware execution strategy. Specifically, for importance estimation, unlike previous works that perform independent clustering over query and key tokens based on feature similarity to estimate attention scores, iCATS demonstrates that clustering based on query-key dot-product interactions is more accurate and further reformulates this objective as a simple

*Intern at T-Head Semiconductor Co., Ltd.
†Corresponding author. DandanDing@hznu.edu.cn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

quadratic form for low-cost computation. For sparse mask construction, instead of using a fixed top-p rule, we observe that tolerance to sparse approximation errors varies across denoising timesteps and therefore introduce an SNR-guided sparsity schedule to adjust sparsity dynamically, leading to higher accuracy. Finally, for hardware execution, we devise a tail-merging strategy to reduce padding overhead caused by irregular cluster sizes, improving GPU kernel utilization. Extensive experiments show that iCATS achieves 2.03$\times$ acceleration with 31.017 dB PSNR on HunyuanVideo-T2V-13B and 1.55$\times$ acceleration with 29.301 dB PSNR on Wan2.1-T2V-14B, delivering a state-of-the-art efficiency-quality trade-off.

1 Introduction

Diffusion Transformers (DiTs) [1] have driven remarkable progress in video generation [2–12], emerging as the dominant backbone architecture. However, as the demand for generating longer and higher-resolution videos continues to grow, the length of token sequences increases substantially, causing the quadratic computational cost of attention mechanisms to rise sharply [13]. This quadratic scaling has become a critical bottleneck during inference for DiT-based video generation models, motivating efficient attention acceleration while preserving generated video quality.

Sparse attention [14–23] is an effective way to accelerate DiT inference by computing only a subset of attention interactions. Existing works fall into two categories: training-based and training-free methods. Training-based methods [24–27] fine-tune models with predefined sparse patterns, achieving high sparsity (e.g., $\sim$90% [25]) but requiring retraining when sparsity changes. In contrast, training-free methods [28–33] introduce sparsity without modifying model parameters. With lower sparsity (e.g., $\sim$70% [17]), they provide greater flexibility and plug-and-play usability across various pretrained DiT models, which is more practical for real applications. Therefore, this work focuses on training-free sparse attention for DiT acceleration.

The core idea of training-free sparse attention is retaining the most important attention computations only during inference. This is typically achieved by first **estimating the importance** of query-key regions in the attention map and then **deriving sparse masks** through rules such as top-p selection. A common strategy is to partition query and key tokens independently and use group mean features to form a smaller query-key matrix for this importance estimation [29, 21, 17]. Based on this matrix, sparse masks are constructed by selecting the most important blocks or clusters. Among these methods, SVG2 [17] groups tokens by feature similarity via k-means clustering and employs cluster centroids to estimate attention scores for cluster selection, achieving state-of-the-art performance.

While this clustering strategy is efficient, its optimization objective is not fully aligned with minimizing the importance estimation error (see Section 3.1 for details). A more principled approach is to cluster tokens based on their query-key dot-product interactions. However, explicitly clustering tokens by their full query-key dot-product interactions is prohibitively expensive for long video sequences: constructing the full query-key dot-product matrix already incurs quadratic complexity in sequence length, not to mention that each interaction vector itself also has sequence-length dimensionality. To tackle this issue, we reformulate the clustering objective derived from query-key interactions as a quadratic form in the original query/key space, avoiding explicit construction of the full matrix. This yields a more efficient clustering scheme for more accurate importance estimation.

Upon the estimated importance, sparse masks are determined by pre-defined rules. The fixed top-p rule [34, 35] is widely used by selecting the highest-scoring query-key regions until the cumulative importance reaches p. While simple and effective, this rule yields a similar sparsity across the denoising process, overlooking the fact that tolerance to sparse approximation errors varies across timesteps. Since signal-to-noise ratio (SNR) characterizes the denoising state [36–38] by reflecting the relative strength of noise and signal at different timesteps, it provides a natural basis for adaptive sparsity control. Based on this, we propose an SNR-guided sparsity schedule, in which the top-p value is computed offline from a smooth function of the logarithmic SNR, effectively driving sparse masks to match various denoising timesteps.

When deploying these optimization methods on hardware, **execution efficiency** is critical. In existing cluster-based sparse attention, mapping clusters to block-based GPU kernels often produces underfilled tail blocks due to irregular cluster sizes [39–42], increasing computational overhead and reducing hardware utilization. This problem is particularly severe on the query side, where the same

padding overhead may be incurred repeatedly across multiple selected key clusters. To alleviate this, we introduce a tail-merging strategy that reduces padding overhead and improves utilization during sparse attention execution.

Building on these designs, we propose iCATS (Fast Video Generation via Interaction-Aware Sparse Attention and Timestep-Adaptive Sparsity), a training-free framework that performs optimization throughout the sparse attention pipeline. The main contributions of this work are as follows:

- We propose a training-free method, called iCATS, for DiT acceleration. iCATS clusters tokens according to query-key interactions via a quadratic-form objective defined in the original query/key space, avoiding complicated full query-key dot-product computation while improving the accuracy of importance estimation.
- For sparse mask construction, iCATS introduces an SNR-guided sparsity schedule that dynamically adjusts the top-p value across denoising timesteps, thereby allocating the computation budget to the selected informative query-key interactions.
- Moreover, when deployed on hardware, iCATS develops a tail-merging strategy that reduces padding overhead from irregular query cluster sizes, improving GPU kernel utilization during sparse attention execution.
- Extensive experiments on representative DiT-based video generation models show that iCATS achieves state-of-the-art performance among training-free sparse attention methods, delivering an advanced trade-off between generation quality and inference efficiency.

2 Related Work

2.1 Training-based Sparse Attention

Training-based sparse attention methods accelerate DiT-based video generation by learning model-adaptive sparse computation patterns. DSV [43] trains separate low-rank predictors to identify critical key-value pairs and achieves acceleration using specialized kernels and sparsity-aware parallelism. However, the decoupled predictor design limits end-to-end optimization. VSA [20] addresses this issue by introducing an end-to-end trainable sparse attention design: it first performs coarse cube-level attention on pooled representations to locate important regions, and then computes token-level attention within the selected cube blocks, enabling joint optimization of selection and computation. VMoBA [44] likewise adopts block-level selection, but with a different motivation: it exploits the spatio-temporal locality of videos through recurrent 1D-2D-3D block partitioning, global block selection, and threshold-based selection.

The above methods all directly discard unselected tokens, whereas SLA [24] revisits this design and instead partitions blocks into three computational levels: sparse attention for critical blocks, linear attention for marginal blocks, and direct skipping for negligible ones. Building on this idea, SLA2 [25] replaces heuristic splitting with a learnable router and further introduces ratio-based branch fusion and quantization-aware low-bit sparse attention.

Despite their different designs, these methods all rely on additional model-specific training or fine-tuning, which limits their flexibility when the target model changes.

2.2 Training-free Sparse Attention

Training-free sparse attention improves inference efficiency without modifying model parameters, which is more flexible for deployment. One line of work uses predefined sparse patterns derived from spatio-temporal priors. SVG [31] profiles per-head spatial and temporal patterns online and applies the resulting masks through FlexAttention [45]. RainFusion [15] and Sparse-vDiT [18] extend this line with richer predefined patterns, while STA [14] replaces full attention with sliding tile attention over local 3D neighborhoods. Although these methods are fast, their fixed or hand-designed sparse patterns limit adaptivity to diverse video content.

Another line of work dynamically constructs sparse masks based on token features. SpargeAttn [46] identifies important blocks through a two-stage online filtering scheme. XAttention [47] estimates block importance using strided antidiagonal sums followed by cumulative thresholding. SVG2 [17] further improves dynamic sparse attention by first applying k-means clustering to group tokens with

similar features, then using cluster centroids to estimate attention importance for group selection, and finally reordering the selected groups into contiguous layouts for more efficient sparse attention execution. Compared with predefined-pattern methods, these dynamic approaches are more content-adaptive. However, their token grouping operates independently in the query and key feature spaces; while simple, it sacrifices accuracy.

Following importance estimation, most existing methods [31, 17] apply a fixed sparsity level across denoising timesteps to construct sparse masks. This produces similar sparsity patterns across steps, even though different denoising timesteps exhibit distinct characteristics during the generation process. These observations motivate a more accurate and efficient sparse attention framework that optimizes the entire pipeline.

3 Motivation

3.1 Independent Clustering Mismatches Q-K Interaction Objectives

As aforementioned, existing sparse attention methods [17] typically adopt a clustering-based approach for importance estimation: clustering centroids are used to construct a smaller query-key matrix that replaces the original full matrix. Therefore, the clustering objective is to minimize the error between the original and the smaller query-key matrices.

To further explain, we fix the query matrix Q and analyze key clustering as an illustrative example. Let $G = \{B_t\}_{t=1}^{P}$ denote a partition of key tokens, where each key token $k_j \in \mathbb{R}^{1\times d}$ is a row vector and $\mu_t \in \mathbb{R}^{1\times d}$ is the centroid of cluster B_t. If each key token is replaced by its cluster centroid when forming the smaller query-key matrix, the optimization objective is then to minimize the resulting error in the query-key dot-product interactions, i.e., minimizing $E(G) = \sum_{t=1}^{P}\sum_{j\in B_t} \|Q(k_j^\top - \mu_t^\top)\|_2^2$. By contrast, previous key-space clustering methods like k-means assume that $Q^\top Q \propto I$, i.e., the query matrix treats all directions in the key space isotropically, and simplify $E(G)$ to $\hat{E}(G) = \sum_{t=1}^{P}\sum_{j\in B_t} \|k_j - \mu_t\|_2^2$. Apparently, this $\hat{E}(G)$ optimization is not aligned with the true objective $E(G)$ of minimizing the query-key interaction error, thereby reducing estimation accuracy.

Instead, this paper proposes to directly optimize $E(G)$. However, this is computationally expensive for long video sequences, as it requires constructing the full query-key dot-product matrix and representing each token with a sequence-length interaction vector, which increases inference time. Therefore, we develop an efficient method, termed **Interaction-aware Clustering**, which minimizes $E(G)$ using a low-complexity scheme without explicitly constructing the full matrix.

3.2 Fixed Sparsity Overlooks Denoising Dynamics

In flow-based video generation [38], samples are synthesized through an iterative denoising process. Let a latent representation $x_0 \in \mathbb{R}^{T\times H\times W\times C}$ be mapped from a clean video X, where T, H, W, and C denote the numbers of frames, height, width, and channels. At denoising timestep $t \in \{0, \ldots, N\}$, the latent is $x_t = a_t x_0 + b_t \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$, N is the total number of solver steps, and a_t and b_t are determined by the timestep schedule. As t increases, x_t transitions from signal-dominated to noise-dominated, while sampling follows the reverse denoising process back to x_0.

Existing methods typically use a fixed top-p rule to construct sparse masks across denoising timesteps. Although the exact number of selected blocks or clusters may vary slightly due to score estimation, the overall sparsity remains broadly similar throughout the denoising trajectory (see our analysis in appendix A). While simple, this implicitly assumes that approximation errors between dense and sparse computations contribute uniformly across timesteps. However, denoising is a recursive process in which the state at each timestep depends on the previous update. Consequently, errors introduced at one timestep accumulate and propagate along the denoising trajectory. As a result, errors at different timesteps have varying impacts on the final output.

Therefore, under a fixed computational budget, the key challenge is not only how much attention computation can be reduced but also how to allocate this budget appropriately across timesteps to best preserve quality. This motivates our **SNR-guided Sparsity** that adjusts sparse mask construction according to the characteristics of each denoising timestep. As the SNR directly reflects denoising progress and signal dominance, it serves as an ideal indicator for such adaptation.

3.3 Varying Cluster Sizes Incur Tail Overhead

Unlike block-based sparse attention, which naturally matches fixed-size block kernels (Figure 2a), cluster-based sparse attention selects computation at the cluster level (Figure 2b). This easily produces irregular cluster sizes that do not align with the fixed execution granularity of sparse kernels.

More specifically, in cluster-based sparse attention with block-based kernels, tokens from selected key clusters are gathered into a contiguous sequence for each query cluster, and attention is computed block by block. The last block is often partially filled, yet the kernel still executes full-block computation, introducing overhead on padded positions. This inefficiency is especially severe on the query side, where padding in the last query block repeats across all key/value blocks, causing overhead to scale with the selected key length. Therefore, reducing tail execution overhead is crucial for improving hardware efficiency, motivating our **Tail-merging Strategy**.

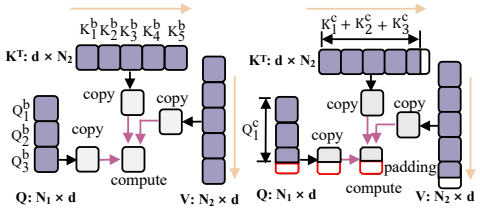


Figure 2: Visualization of (a) block-based and (b) cluster-based sparse attention.

4 Method

This section details our proposed iCATS. As shown in Figure 3, iCATS consists of three components: interaction-aware clustering for accurate importance estimation (Section 4.1), SNR-guided sparsity for sparse mask construction (Section 4.2), and tail merging for sparse attention execution (Section 4.3).

4.1 Interaction-Aware Clustering

Recalling Section 3.1, iCATS performs interaction-aware clustering by grouping tokens using their query-key dot-product interactions. We represent all token features as row vectors. Let $Q = [q_1^\top, q_2^\top, \ldots, q_n^\top]^\top \in \mathbb{R}^{n\times d}$ $(d \ll n)$ denote the query matrix, where $q_i \in \mathbb{R}^{1\times d}$ is the i-th query token. Here, n is the sequence length and d is the token dimension, e.g., $n = 75,600$ and $d = 128$ in Wan2.1-T2V-14B. Let B_t denote the t-th key cluster and $\mu_t \in \mathbb{R}^{1\times d}$ its centroid, where $t \in \{1, \ldots, P\}$ with P being the total number of key clusters. We define the key clustering objective as:

$$\begin{aligned}\mathcal{L} &= \sum_{t=1}^{P}\sum_{j\in B_t} \left\|Qk_j^\top - Q\mu_t^\top\right\|_2^2 = \sum_{t=1}^{P}\sum_{j\in B_t}\sum_{i=1}^{n}\left(q_i k_j^\top - q_i\mu_t^\top\right)^2 \\ &= \sum_{t=1}^{P}\sum_{j\in B_t}(k_j-\mu_t)\left(\sum_{i=1}^{n} q_i^\top q_i\right)(k_j^\top - \mu_t^\top) = \sum_{t=1}^{P}\sum_{j\in B_t}(k_j-\mu_t)\,Q^\top Q\,(k_j^\top-\mu_t^\top).\end{aligned} \tag{1}$$

As seen, clustering using query-key interactions requires computing, for each key, its dot products with all queries to form an n-dimensional interaction vector, followed by clustering in this space. Equation (1) finally reformulates this objective as a quadratic form in the original key feature space of dimension d, where $d \ll n$, avoiding explicit construction of the full query-key dot-product matrix. Query clustering is defined symmetrically, yielding a corresponding quadratic form induced by $K^\top K$. Although $Q^\top Q$ and $K^\top K$ are only of size $d \times d$, computing them still requires aggregating statistics over all tokens. To reduce this overhead, we approximate them using cluster centroids and sizes. Specifically, for query clusters $\{A_r\}_{r=1}^{O}$, where O is the number of query clusters, with centroid μ_r and size $|A_r|$, and key clusters $\{B_t\}_{t=1}^{P}$ defined similarly with size $|B_t|$, we define $M_Q = \sum_{r=1}^{O}|A_r|\,\mu_r^\top\mu_r$ and $M_K = \sum_{t=1}^{P}|B_t|\,\mu_t^\top\mu_t$. Approximating $Q^\top Q$ with M_Q and $K^\top K$ with M_K, and noting that M_Q and M_K are symmetric matrices, the quadratic-form distances for clustering become:

$$\begin{aligned}(k_j-\mu_t)M_Q(k_j^\top-\mu_t^\top) &= k_j M_Q k_j^\top + \mu_t M_Q \mu_t^\top - 2\,k_j M_Q \mu_t^\top, \\ (q_i-\mu_r)M_K(q_i^\top-\mu_r^\top) &= q_i M_K q_i^\top + \mu_r M_K \mu_r^\top - 2\,q_i M_K \mu_r^\top.\end{aligned} \tag{2}$$

Built on Flash-Kmeans [48], we precompute the token-side terms (e.g., $k_j M_Q$, $k_j M_Q k_j^\top$, $q_i M_K$, and $q_i M_K q_i^\top$) only once before the clustering iterations begin, as token features remain fixed. The centroid-side terms are recomputed from the current centroids at each iteration.

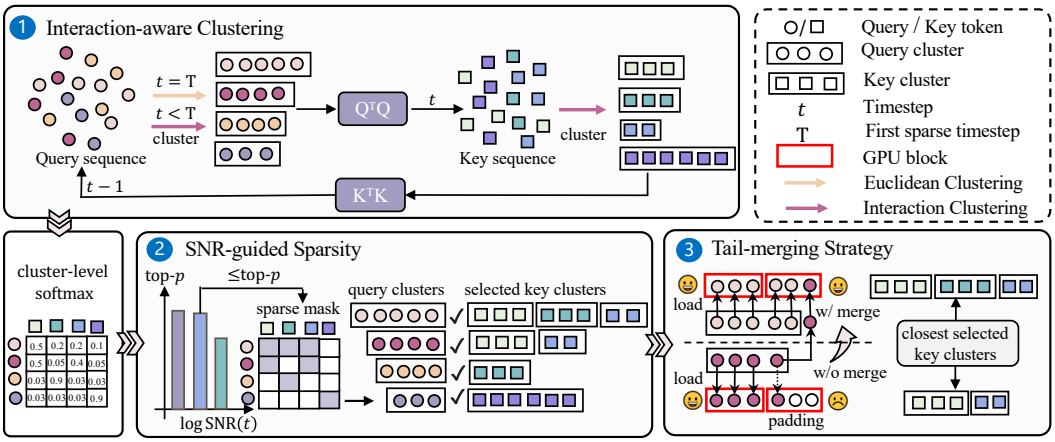


Figure 3: Overview of iCATS pipeline. For each attention layer, query and key tokens are first grouped by ① **Interaction-aware Clustering**. Then, the corresponding query-key cluster centroids are used to approximate softmax scores at the cluster level. Based on these approximated scores, ② **SNR-guided Sparsity** determines the top-p value at each denoising timestep and selects the corresponding attention computation region. Finally, ③ **Tail-merging Strategy** is applied before sparse attention kernel computation by reassigning tail tokens of non-block-aligned query clusters, thereby reducing padding overhead caused by irregular cluster sizes. In this figure, interaction clustering denotes interaction-aware query-key clustering, while Euclidean clustering is used to initialize query clusters solely at the first sparse timestep.

Based on this design, we perform interaction-aware clustering alternately. At the first sparse timestep, query tokens are clustered in the original feature space, and the resulting query centroids and cluster sizes are used to construct M_Q. Key tokens are then clustered using the quadratic form induced by M_Q, and the resulting key centroids and cluster sizes are used to construct M_K. At later timesteps, we use the previous M_K to guide query clustering, leveraging the similarity of quadratic structures between adjacent timesteps. The updated query clusters define the current M_Q, which in turn induces the quadratic form for key clustering. The resulting key clusters form a new M_K for the next timestep.

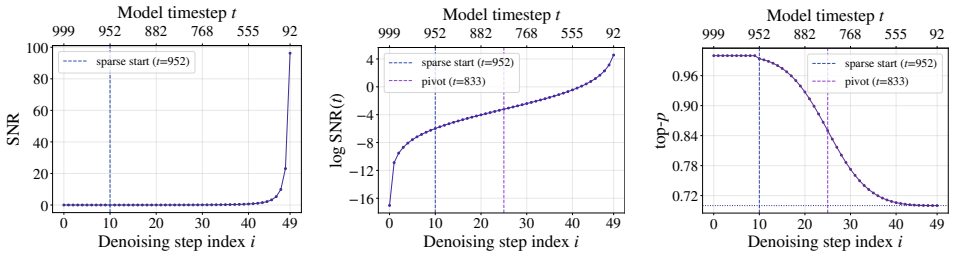
4.2 SNR-Guided Sparsity

Motivated by the observation in Section 3.2, iCATS devises the SNR-guided sparsity schedule, which dynamically adjusts the sparsity of sparse masks across timesteps. Specifically, we employ the timestep-wise SNR to characterize the denoising state: a low SNR typically corresponds to noise-dominated latents at earlier denoising stages, where approximation errors may affect more subsequent updates, while a high SNR indicates the emergence of cleaner signal components and more stable semantic structures. This makes SNR a practical indicator for adapting top-p across timesteps. Formally, the timestep-wise SNR is defined as $\mathrm{SNR}(t) = (a_t/b_t)^2$.

Figure 4a shows that the raw SNR spans a wide range across timesteps, making early- and late-stage differences hard to distinguish. We therefore operate in the logarithmic domain and define $\ell_t = \log \mathrm{SNR}(t)$ (see Figure 4b). This converts ratio changes in SNR into additive differences, making variations across timesteps easier to model and analyze.

As analyzed in Figure 4b, the $\log\mathrm{SNR}(t)$ curve exhibits a clear stage-wise pattern: the early and late stages change more sharply, while the middle stage is comparatively smooth. Based on this observation, we use an inverted $\mathrm{sigmoid}$ function to generate timestep-adaptive top-p values for sparse mask construction. The inverted $\mathrm{sigmoid}$ provides a simple schedule shape that aligns well with the stage-wise pattern of the $\log\mathrm{SNR}(t)$ curve: its two saturated ends help maintain relatively stable computation in the early and late stages, while its transition region allows top-p to decrease gradually in the smoother middle stage. Overall, we keep dense attention before the first sparse timestep t_{sparse}, and apply sparse attention for $t \le t_{\text{sparse}}$ with

$$\text{top-}p_t = p_{\min} + (1 - p_{\min}) \left(1 - \frac{1}{1 + \exp(-k x_t)}\right), \qquad t \le t_{\text{sparse}}, \tag{3}$$



(a) SNR (b) $\log\mathrm{SNR}(t)$ (c) Top-p from $\log\mathrm{SNR}(t)$

Figure 4: Visualization of proposed SNR-guided sparsity schedule on Wan2.1-T2V-14B. (a) Raw SNR over denoising timesteps. (b) $\log\mathrm{SNR}(t)$, where the pivot marks the center of the flattest region. (c) The resulting top-p schedule generated by an inverted sigmoid.

where $p_{\min}$ denotes the minimum top-p value, and $x_t = \ell_t - \ell_{t_{\text{pivot}}}$ is defined by the $\log\mathrm{SNR}(t)$ difference between timestep t and a pivot timestep t_{pivot}. The pivot is chosen automatically as the timestep immediately following the minimum difference between consecutive $\log\mathrm{SNR}(t)$ values, which typically lies near the smoothest denoising stage.

We choose $\tau \in (0, 1)$ to be very close to 1 in practice (e.g., 0.99 or 0.999), which controls how close the inverted $\mathrm{sigmoid}$ is to its upper bound at the first sparse timestep and thus determines the sigmoid slope as $k = \frac{\log \frac{\tau}{1-\tau}}{\left|\ell_{t_{\text{sparse}}} - \ell_{t_{\text{pivot}}}\right|}$. Finally, rather than introducing an additional parameter, we set the minimum retained ratio of key clusters by linearly scaling top-p as $\text{min-}p_t = 0.1 \cdot \frac{\text{top-}p_t}{\text{top-}p_{t_{\text{pivot}}}}$.

As illustrated in Figure 4c, the final top-p schedule resulting from $\log\mathrm{SNR}(t)$ keeps the sparse masking budget more stable in the early and late stages while reducing top-p more smoothly in the middle stage, which better aligns with the stage-wise characteristics of denoising. Since the SNR sequence is fully determined by the predefined noise schedule, the top-p values can be computed offline before video generation, largely saving runtime overhead during inference. To further illustrate this behavior, we visualize the computational density of iCATS across timesteps in Appendix A (Figure 1), where density denotes the proportion of retained computations.

4.3 Tail-Merging Strategy

Motivated by the tail execution issue analyzed in Section 3.3, iCATS develops a tail-merging strategy before the kernel computation of sparse attention to reduce the extra overhead caused by irregular cluster sizes. The core idea is to restructure query clusters in a hardware-friendly manner while preserving the original sparse attention pattern as much as possible, thereby reducing both scheduling and padding overhead. To this end, we first exploit merging opportunities that do not alter the original sparse pattern, and then further reduce the remaining tail waste.

Specifically, we merge query clusters within the same attention head that attend to the same set of key clusters. Because these clusters share the same attended key-cluster set, this operation preserves the original sparse attention pattern without introducing additional error, while reducing the number of query clusters and subsequent scheduling overhead.

We then handle query clusters whose sizes are not divisible by the block execution size, since their final execution blocks are only partially occupied while still fully computed, leading to tail overhead. Our goal is to reassign only tokens in these tail blocks to improve execution-block utilization while minimizing perturbation to the sparse attention pattern. To this end, we need to find suitable target clusters for these tail tokens. Specifically, we measure the compatibility between query clusters based on the similarity of their key-cluster masks. Each mask is represented as a packed binary vector, from which pairwise Hamming distances between clusters are computed. A smaller Hamming distance indicates that the key-cluster masks of two query clusters are more similar; when the merged mask is updated by union, this usually leads to fewer newly introduced key clusters, making them better candidates for merging. Under size and alignment constraints, we iteratively identify compatible cluster pairs and reassign the tail-block tokens of the source cluster to the target cluster. After merging, empty clusters are removed, and the remaining clusters are compactly reindexed. In implementation, we perform two rounds of tail merging. Detailed pseudocode is presented in Appendix E.1.

5 Experiments and Analysis

5.1 Setup

Experimental setup. We evaluate iCATS on three mainstream video generation DiT models, including HunyuanVideo-T2V-13B, Wan2.1-T2V-14B, and Wan2.1-I2V-14B, for 5-second 720p generation. Following prior works [31, 17], we use the Penguin Benchmark with VBench [49, 50] prompt optimization for text-to-video evaluation. For image-to-video evaluation, we use VBench prompt-image pairs and resize the input images to 16:9 to match the 720p setting. Ablations are conducted on Wan2.1-T2V-14B using 138 randomly sampled text prompts. More setting details can be found in Appendix B.

Evaluation metrics. We measure the similarity between videos generated by the accelerated DiT models and the original DiT models using PSNR, SSIM [51], and LPIPS [52] and evaluate video quality with VBench [49]. The main paper reports the average VBench score, while the detailed breakdown over subject consistency, background consistency, motion smoothness, aesthetic quality, and imaging quality is reported in Appendix C. To assess practical efficiency, we report the speedup over the original dense attention. We also report sparsity, defined as the theoretical computation reduction ratio of sparse attention methods relative to dense attention.

Comparison setup. We compare iCATS against two state-of-the-art sparse attention methods, SVG [31] and SVG2 [17], using their official open-source implementations and settings for fair comparisons. Moreover, we evaluate two iCATS variants, Base and Turbo, whose hyperparameters are chosen to match the speed or sparsity of prior methods. Unless otherwise stated, experiments are conducted on eight NVIDIA H100 GPUs.

5.2 Quality & Efficiency Evaluation

Table 1 shows that iCATS-Base and iCATS-Turbo consistently achieve a better quality-efficiency trade-off than SVG and SVG2 on both HunyuanVideo-T2V-13B and Wan2.1-T2V-14B. At comparable speedup, iCATS-Base achieves better PSNR, SSIM, LPIPS, and VBench than SVG2, indicating stronger quality preservation. Meanwhile, iCATS-Turbo provides higher acceleration at similar sparsity while maintaining better reconstruction quality and competitive VBench scores. The same trend also holds on Wan2.1-I2V-14B, where iCATS-Base improves all reconstruction metrics and VBench over SVG2 at comparable speed, while iCATS-Turbo achieves the fastest inference among sparse methods and still maintains better reconstruction quality with matched VBench. Detailed VBench results can be found in Appendix C. Visualization comparisons are shown in Figure 1 and Appendix H. Results of cross-model and cross-task hyperparameter transfer without retuning are provided in Appendix F.

Table 1: Comparison with existing works on various benchmark models

Method	PSNR↑	SSIM↑	LPIPS↓	VBench↑	Time (s)↓	Sparsity
HunyuanVideo-T2V-13B (Dense)	–	–	–	0.844	1730	–
SVG	20.362	0.725	0.235	0.843	874 (1.98×)	75.8%
SVG2	26.247	0.869	0.097	0.842	853 (2.03×)	71.5%
iCATS-Base	**31.017**	**0.926**	**0.050**	**0.844**	854 (2.03×)	66.1%
iCATS-Turbo	30.466	0.922	0.054	**0.844**	827 (2.09×)	70.3%
Wan2.1-T2V-14B (Dense)	–	–	–	0.839	1889	–
SVG	21.497	0.741	0.197	0.836	1220 (1.55×)	65.7%
SVG2	25.606	0.853	0.098	0.837	1226 (1.54×)	68.6%
iCATS-Base	**29.301**	**0.908**	**0.058**	**0.838**	1222 (1.55×)	61.7%
iCATS-Turbo	28.172	0.895	0.068	0.837	1176 (1.61×)	69.0%
Wan2.1-I2V-14B (Dense)	–	–	–	0.847	1519	–
SVG	26.533	0.847	0.101	0.844	1088 (1.40×)	65.7%
SVG2	29.082	0.890	0.067	0.845	1081 (1.41×)	69.9%
iCATS-Base	**30.606**	**0.911**	**0.056**	**0.846**	1084 (1.41×)	63.2%
iCATS-Turbo	30.097	0.905	0.060	**0.846**	1057 (1.44×)	68.9%

To understand the quality gains from interaction-aware clustering, we compare both clustering schemes with exact dense attention using two metrics. (i) Token-level top-p selection overlap: for each query token q, we compute its exact dense attention weights (full softmax over all keys) and apply top-p ($p = 0.9$) to obtain the keys selected for q; the method computes keys in the key clusters selected for the query cluster containing q; overlap is the fraction of dense-selected keys covered by the method, averaged over all query tokens. (ii) Attention output MSE between dense and masked attention outputs (attention weights multiplied by V), with masked weights zeroed and the remaining weights renormalized. The diagnostics are conducted on Wan2.1-T2V-14B with 20 randomly sampled prompts, covering 5 timesteps ($t = 952, 882, 768, 555, 92$), 5 layers (1, 10, 20, 30, 39), and 4 heads (0, 10, 20, 30) per layer. Both schemes share the same top-p (0.9), min-p (0.1), cluster configuration (400 query / 1000 key clusters), clustering iterations (50 at the first sparse timestep, then 2), warmup, and kernel implementation; only the clustering scheme differs. As shown in Table 2, interaction-aware clustering achieves higher selection overlap (+1.7 points) and a 36% lower output MSE, with consistent gains at every sampled timestep, layer, and head (Appendix D). These results indicate that interaction-aware clustering better matches dense attention in both region selection and attention outputs. On Wan2.1-T2V-14B over 20 randomly sampled prompts, replacing M_Q/M_K with exact $Q^\top Q/K^\top K$ under the same SNR-guided sparsity schedule yields the same reported token-level overlap (0.912 for both), while increasing denoising time from 1219 s to 1257 s.

Table 2: Attention fidelity of different clustering schemes on Wan2.1-T2V-14B.

Method	**Top-p overlap↑**	**Output MSE ($\times 10^{-4}$)↓**
Independent Euclidean clustering	0.896	8.99
Interaction-aware clustering (ours)	**0.913**	**5.75**

To quantify the effect of tail merging, we measure the reduction in execution blocks before and after merging at each sparse attention timestep, where execution blocks are defined following FlashInfer [53]. Figure 5 shows the average reduction ratio of iCATS, with the shaded region indicating the min-max range across layers. The reduction is largest at early timesteps and gradually decreases during denoising. The merging procedure is efficiently implemented in Triton [54]. On Wan2.1-T2V-14B running on an NVIDIA H100 GPU, each invocation of the merging function takes $\sim$3 ms, while merging itself takes $\sim$10 s per generation, which is negligible relative to the total inference time of 1222 s. More details are provided in Appendix E.2. A detailed breakdown of the denoising runtime is provided in Appendix G.

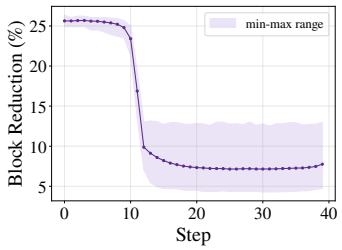


Figure 5: Reduction ratio of execution blocks on Wan2.1-T2V-14B.

5.3 Cross-Hardware Validation

To evaluate the hardware generalization of iCATS, we further deploy iCATS-Turbo on Wan2.1-T2V-14B on the T-Head Zhenwu M890P[3], an AI accelerator developed by Alibaba T-Head, with sparse kernel operators adapted to this platform. As shown in Table 3, iCATS achieves performance comparable to that on H100 GPUs, demonstrating strong adaptability and generalization across hardware platforms.

Table 3: Comparison of different methods on Wan2.1-T2V-14B using the Zhenwu M890P.

Method	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**Time (s)↓**	**Speedup↑**	**Sparsity**
Wan2.1-T2V-14B (Dense)	–	–	–	2828	–	–
SVG	21.847	0.774	0.186	2208	1.28×	65.7%
SVG2	26.321	0.870	0.088	2214	1.28×	68.9%
iCATS-Turbo	**28.849**	**0.903**	**0.061**	2073	1.36×	68.9%

[3]Technical documentation: `https://developer.t-head.cn/docs_center/doc_detail/index.html?projectId=48`

5.4 Human Evaluation

We blindly compare iCATS-Base with SVG2 on Wan2.1-T2V-14B using 20 VBench-optimized Penguin Benchmark prompts covering complex scenes, highly dynamic human actions, and simple static scenes. Eleven video-domain professionals with normal vision view anonymized, randomly ordered A/B pairs side by side on 2K screens, selecting the perceptually better video or "Similar Quality," yielding 220 votes. Table 4 shows 54.1% preference for iCATS-Base, 18.6% for SVG2, and 27.3% similar quality, supporting iCATS-Base's perceptual advantage on these samples beyond automatic metrics.

Table 4: Human preference on Wan2.1-T2V-14B.

Preference	Votes	Percentage
SVG2	41	18.6%
iCATS-Base	**119**	**54.1%**
Similar Quality	60	27.3%

5.5 Ablation Study

We conduct ablation studies on the turbo variant on Wan2.1-T2V-14B to evaluate the contribution of interaction-aware clustering (IC), SNR-guided sparsity (SG), and tail-merging strategy (TM). Here, "Anchor" denotes the starting configuration built upon SVG2 settings.

As reported in Table 5, both IC and SG improve quality over Anchor, and their combination brings further gains, indicating that they are complementary. In particular, SG controls how the sparsity budget is allocated across timesteps, whereas IC improves how this budget is used for sparse region selection. Adding TM yields the best overall trade-off, improving speedup from 1.56× to 1.61× while achieving the best quality metrics. Although TM slightly reduces sparsity, it improves speed by reducing padding overhead. Overall, all three components contribute positively to final performance.

Table 5: Ablation study on Wan2.1-T2V-14B

Method	PSNR↑	SSIM↑	LPIPS↓	Time↓	Sparsity
Wan2.1-T2V-14B	–	–	–	1890s	–
Anchor	25.894	0.868	0.101	1227s (1.54×)	68.7%
w/ IC	26.435	0.878	0.093	1224s (1.54×)	69.9%
w/ SG	27.309	0.890	0.085	1219s (1.55×)	69.5%
w/ TM	26.085	0.872	0.098	1223s (1.55×)	67.9%
w/ IC & SG	28.283	0.903	0.074	1213s (1.56×)	70.0%
w/ IC & SG & TM	**28.372**	**0.905**	**0.072**	1171s (1.61×)	69.0%

6 Conclusion & Limitation

We propose iCATS, a training-free sparse attention framework accelerating DiT-based video generation through improved importance estimation, sparse masking, and hardware execution. It replaces independent clustering with query–key interaction clustering and reformulates the resulting optimization for low-cost computation. SNR-guided scheduling adapts sparsity across denoising timesteps, while hardware-friendly tail merging reduces execution overhead. Together, these designs achieve state-of-the-art performance across DiT models.

Currently, the clustering procedure in iCATS is not optimized for multi-GPU parallel execution and is left for future work. The token-level sparse masking kernel also remains to be further optimized.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 62571174), Alibaba Group through the Alibaba Innovative Research Program, and the Advanced Computing Center of Hangzhou Normal University. We thank the anonymous area chair and reviewers for their feedback on earlier drafts, and the outstanding prior works on video generation and sparse attention.

References

[1] Peebles, W., S. Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

[2] Team Wan, A. Wang, B. Ai, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[3] Kong, W., Q. Tian, Z. Zhang, et al. HunyuanVideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

[4] Yang, Z., J. Teng, W. Zheng, et al. CogVideoX: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

[5] Gao, Y., H. Guo, T. Hoang, et al. Seedance 1.0: Exploring the boundaries of video generation models. *arXiv preprint arXiv:2506.09113*, 2025.

[6] Team Seedance, D. Chen, L. Chen, et al. Seedance 2.0: Advancing video generation for world complexity, 2026.

[7] Kling Team, J. Chen, Y. Ci, et al. Kling-Omni technical report, 2025.

[8] Wu, B., C. Zou, C. Li, et al. HunyuanVideo 1.5 technical report, 2025.

[9] Liu, Y., K. Zhang, Y. Li, et al. Sora: A review on background, technology, limitations, and opportunities of large vision models, 2024.

[10] Feng, W., C. Yang, H. Qin, et al. QuantSparse: Comprehensively compressing video diffusion transformer with model quantization and attention sparsification. In *The Fourteenth International Conference on Learning Representations*. 2026.

[11] Li, S., R. Lu, Q. Chen, et al. DSA: Efficient inference for video generation models via distributed sparse attention. In *The Fourteenth International Conference on Learning Representations*. 2026.

[12] Zhou, X., Q. Mang, S. Yang, et al. SVG-EAR: Parameter-free linear compensation for sparse video generation via error-aware routing, 2026.

[13] Vaswani, A., N. Shazeer, N. Parmar, et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, vol. 30. 2017.

[14] Zhang, P., Y. Chen, R. Su, et al. Fast video generation with sliding tile attention. In *Forty-second International Conference on Machine Learning*. 2025.

[15] Chen, A., B. Dong, J. Li, et al. Rainfusion: Adaptive video generation acceleration via multi-dimensional visual redundancy. *arXiv preprint arXiv:2505.21036*, 2025.

[16] Chen, A., Y. Liu, J. Huang, et al. RainFusion2.0: Temporal-spatial awareness and hardware-efficient block-wise sparse attention. *arXiv preprint arXiv:2512.24086*, 2025.

[17] Yang, S., H. Xi, Y. Zhao, et al. Sparse VideoGen2: Accelerate video generation with sparse attention via semantic-aware permutation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 2025.

[18] Chen, P., X. Zeng, M. Zhao, et al. Sparse-vDiT: Unleashing the power of sparse attention to accelerate video diffusion transformers. *arXiv preprint arXiv:2506.03065*, 2025.

[19] Li*, X., M. Li*, T. Cai, et al. Radial Attention: $\mathcal{O}(n \log n)$ sparse attention with energy decay for long video generation. *arXiv preprint arXiv:2506.19852*, 2025.

[20] Zhang, P., Y. Chen, H. Huang, et al. Faster video diffusion with trainable sparse attention. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, N. Chen, eds., *Advances in Neural Information Processing Systems*, vol. 38, pages 152509–152534. Curran Associates, Inc., 2025.

[21] Liu, X., Z. Li, J. Zhang, et al. Rectified SpaAttn: Revisiting attention sparsity for efficient video generation. *arXiv preprint arXiv:2511.19835*, 2025.

[22] Agarwal, K., Z. Chen, C. Luo, et al. MonarchRT: Efficient attention for real-time video generation, 2026.

[23] Luo, J., J. Chen, J. Wang, et al. Training-free sparse attention for fast video generation via offline layer-wise sparsity profiling and online bidirectional co-clustering, 2026.

[24] Zhang, J., H. Wang, K. Jiang, et al. SLA: Beyond sparsity in diffusion transformers via fine-tunable sparse–linear attention. In *The Fourteenth International Conference on Learning Representations*. 2026.

[25] —. SLA2: Sparse-linear attention with learnable routing and qat. *arXiv preprint arXiv:2602.12675*, 2026.

[26] Chen, J., Y. Zhao, J. Yu, et al. SANA-Video: Efficient video generation with block linear diffusion transformer. In *The Fourteenth International Conference on Learning Representations*. 2026.

[27] Fang, T., H. Zhang, R. Xie, et al. SALAD: Achieve high-sparsity attention via efficient linear attention tuning for video diffusion transformer, 2026.

[28] Shmilovich, D., T. Wu, A. Dahan, et al. LiteAttention: A temporal sparse attention for diffusion transformers, 2025.

[29] Zhang, Y., J. Xing, B. Xia, et al. Training-free efficient video generation via dynamic token carving. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, N. Chen, eds., *Advances in Neural Information Processing Systems*, vol. 38, pages 81913–81946. Curran Associates, Inc., 2025.

[30] Li, X., Y. Gu, X. Lin, et al. PSA: Pyramid sparse attention for efficient video understanding and generation, 2025.

[31] Xi, H., S. Yang, Y. Zhao, et al. Sparse Video-Gen: Accelerating video diffusion transformers with spatial-temporal sparsity. In *Forty-second International Conference on Machine Learning*. 2025.

[32] Liang, C., H. Chen, L. Hou, et al. VMonarch: Efficient video diffusion transformers with structured attention, 2026.

[33] Xia, Y., S. Ling, F. Fu, et al. Training-free and adaptive sparse attention for efficient long video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15982–15993. 2025.

[34] Zhu, K., T. Tang, Q. Xu, et al. Tactic: Adaptive sparse attention with clustering and distribution fitting for long-context LLMs. In *The Fourteenth International Conference on Learning Representations*. 2026.

[35] Lin, C., J. Tang, S. Yang, et al. Twilight: Adaptive attention sparsity with hierarchical top-p pruning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, N. Chen, eds., *Advances in Neural Information Processing Systems*, vol. 38, pages 119406–119432. Curran Associates, Inc., 2025.

[36] Ho, J., A. Jain, P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[37] Song, Y., J. Sohl-Dickstein, D. P. Kingma, et al. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*. 2021.

[38] Lipman, Y., R. T. Chen, H. Ben-Hamu, et al. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[39] Dao, T., D. Fu, S. Ermon, et al. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

[40] Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[41] Shah, J., G. Bikshandi, Y. Zhang, et al. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Advances in Neural Information Processing Systems*, 37:68658–68685, 2024.

[42] Zadouri, T., M. Hoehnerbach, J. Shah, et al. Flashattention-4: Algorithm and kernel pipelining co-design for asymmetric hardware scaling, 2026.

[43] Tan, X., Y. Chen, Y. Jiang, et al. Dsv: Exploiting dynamic sparsity to accelerate large-scale video dit training. *arXiv preprint arXiv:2502.07590*, 2025.

[44] Wu, J., L. Hou, H. Yang, et al. VMoBA: Mixture-of-block attention for video diffusion models. In *The Fourteenth International Conference on Learning Representations*. 2026.

[45] Dong, J., B. Feng, D. Guessous, et al. Flex Attention: A programming model for generating optimized attention kernels, 2024.

[46] Zhang, J., C. Xiang, H. Huang, et al. SpargeAttention: Accurate and training-free sparse attention accelerating any model inference. In *Forty-second International Conference on Machine Learning*. 2025.

[47] Xu, R., G. Xiao, H. Huang, et al. XAttention: Block sparse attention with antidiagonal scoring. In *Forty-second International Conference on Machine Learning*. 2025.

[48] Yang, S., H. Xi, Y. Zhao, et al. Flash-kmeans: Fast and memory-efficient exact k-means, 2026.

[49] Huang, Z., Y. He, J. Yu, et al. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024.

[50] Zheng, D., Z. Huang, H. Liu, et al. VBench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025.

[51] Wang, Z., A. C. Bovik, H. R. Sheikh, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[52] Zhang, R., P. Isola, A. A. Efros, et al. 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.

[53] Ye, Z., L. Chen, R. Lai, et al. Flashinfer: Efficient and customizable attention engine for LLM inference serving. In *Eighth Conference on Machine Learning and Systems*. 2025.

[54] Tillet, P., H.-T. Kung, D. Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19. 2019.

Appendix

A SNR-Guided Sparsity

Figure 1 illustrates the efficacy of our proposed SNR-guided sparsity in iCATS. Specifically, we present the density changes across timesteps on Wan2.1-T2V-14B. As shown, iCATS allocates computation dynamically along the denoising trajectory: it keeps full computation during warmup, gradually reduces density in the middle stage, and stabilizes in the late stage. Compared with the fixed-density baseline SVG2 (the light blue dotted line in Figure 1), iCATS preserves more computation early on while using much lower density later, with iCATS-Turbo being more aggressive.

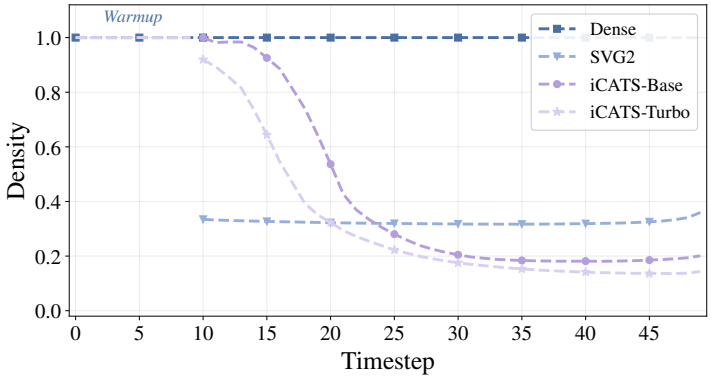


Figure 1: Density of the original full attention, SVG2, and our iCATS across timesteps on Wan2.1-T2V-14B.

B Detailed Experimental Settings

This appendix provides the detailed settings for all compared methods.

Compared methods. We compare iCATS with two state-of-the-art sparse attention baselines, SVG and SVG2. On HunyuanVideo-T2V-13B, Wan2.1-T2V-14B and Wan2.1-I2V-14B, all baseline methods follow their official open-source implementations and default settings, including warmup timesteps, warmup layers, and the number of clustering iterations.

iCATS variants. We evaluate two variants of iCATS, Base and Turbo, designed to match the speed or sparsity of prior methods under different models. For fair comparison, the warmup timesteps, warmup layers, and clustering iterations of iCATS follow the baseline configuration of each model, as summarized in Table 1.

Table 1: Warmup and clustering configurations for each model.

Model	Warmup timesteps	Warmup layers	Cluster iters (first/later)
HunyuanVideo-T2V-13B	5/50	1/60	50/2
Wan2.1-T2V-14B	10/50	1/40	50/2
Wan2.1-I2V-14B	14/40	1/40	50/2

For **HunyuanVideo-T2V-13B**, both iCATS variants use 500 query clusters and 1000 key clusters. The Base variant uses $p_{\min} = 0.7$ and $\tau = 0.98$, while the Turbo variant uses $p_{\min} = 0.65$ and $\tau = 0.97$.

For **Wan2.1-T2V-14B**, both iCATS variants use 400 query clusters and 1000 key clusters. The Base variant uses $p_{\min} = 0.7$ and $\tau = 0.999$, while the Turbo variant uses $p_{\min} = 0.65$ and $\tau = 0.97$.

For **Wan2.1-I2V-14B**, both iCATS variants use 400 query clusters and 1000 key clusters. The Base variant uses $p_{\min} = 0.8$ and $\tau = 0.999$, while the Turbo variant uses $p_{\min} = 0.75$ and $\tau = 0.98$.

These settings are chosen to provide comparable speed or sparsity to prior sparse attention methods.

C Additional VBench Results

Table 2 reports detailed VBench results for all compared methods across all models. Specifically, we report subject consistency, background consistency, motion smoothness, aesthetic quality, imaging quality, and the averaged VBench score. These fine-grained results complement the average scores reported in the main paper and provide a clearer view of quality preservation under sparse attention.

Overall, iCATS remains highly competitive with dense attention and strong sparse baselines across all models. On HunyuanVideo-T2V-13B, both iCATS-Base and iCATS-Turbo achieve an average VBench score of 0.844, exactly matching dense attention. More specifically, both variants match dense attention in subject consistency, background consistency, motion smoothness, and imaging quality. Compared with existing sparse baselines, iCATS further achieves the strongest subject consistency while remaining tied for the best background consistency and motion smoothness. These results indicate that the proposed sparse attention design preserves identity consistency, scene coherence, temporal stability, and visual fidelity without measurable degradation on this model.

On Wan2.1-T2V-14B, iCATS-Base achieves an average VBench score of 0.838, outperforming SVG (0.836) and slightly surpassing SVG2 (0.837), while iCATS-Turbo matches SVG2 at 0.837. In addition, iCATS remains competitive in subject consistency and background consistency, preserves motion smoothness close to other sparse methods, and achieves the strongest imaging quality among the sparse variants. These results suggest that the proposed sparse attention design preserves both structural coherence and visual fidelity well, while still enabling substantial efficiency gains.

On Wan2.1-I2V-14B, iCATS likewise remains highly competitive with dense attention. Both iCATS-Base and iCATS-Turbo achieve an average VBench score of 0.846, exceeding SVG (0.844) and SVG2 (0.845) while staying very close to dense attention (0.847). Looking at the detailed dimensions, iCATS-Base matches dense attention in imaging quality, iCATS-Turbo matches dense attention in background consistency, and both variants preserve the same motion smoothness as dense attention. Meanwhile, subject consistency and aesthetic quality also remain competitive with the strongest sparse baselines, showing that iCATS generalizes well beyond text-to-video generation and can maintain strong appearance consistency, scene coherence, motion quality, and perceptual fidelity in the image-to-video setting.

Table 2: Detailed VBench comparison with existing methods on benchmark models

Method	**Sub.Cons.↑**	**Back.Cons.↑**	**Mot.Smooth↑**	**Aes.Qual.↑**	**Img.Qual.↑**	**VBench (Avg.)↑**
HunyuanVideo-T2V-13B (Dense)	0.940	0.959	0.994	0.628	0.698	0.844
SVG	0.938	**0.959**	**0.994**	**0.632**	0.693	0.843
SVG2	0.939	**0.959**	0.993	0.627	0.695	0.842
iCATS-Base	**0.940**	**0.959**	**0.994**	0.627	**0.698**	**0.844**
iCATS-Turbo	**0.940**	**0.959**	**0.994**	0.627	**0.698**	**0.844**
Wan2.1-T2V-14B (Dense)	0.934	0.955	0.987	0.630	0.689	0.839
SVG	0.929	0.953	**0.987**	0.629	0.686	0.836
SVG2	0.931	**0.954**	0.986	**0.630**	0.684	0.837
iCATS-Base	**0.932**	**0.954**	0.986	0.629	**0.687**	**0.838**
iCATS-Turbo	0.931	0.953	0.986	0.629	**0.687**	0.837
Wan2.1-I2V-14B (Dense)	0.958	0.958	0.986	0.618	0.716	0.847
SVG	0.953	0.954	**0.986**	0.612	0.715	0.844
SVG2	**0.956**	0.955	**0.986**	0.614	0.715	0.845
iCATS-Base	**0.956**	0.957	**0.986**	0.615	**0.716**	**0.846**
iCATS-Turbo	0.955	**0.958**	**0.986**	**0.616**	0.715	**0.846**

D Attention Fidelity Breakdown

Table 3 breaks down the attention fidelity diagnostics in Table 2 of the main paper by timestep, layer, and head. Interaction-aware clustering consistently outperforms independent Euclidean clustering on both metrics across all sampled points.

Table 3: Breakdown of attention fidelity on Wan2.1-T2V-14B by timestep, layer, and head. Each entry is reported as "interaction-aware (ours) / Euclidean"; MSE values are in units of 10^{-4}.

Timestep t	952	882	768	555	92
Top-p overlap↑	0.902 / 0.891	0.914 / 0.896	0.915 / 0.896	0.916 / 0.897	0.917 / 0.901
Output MSE↓	8.91 / 11.75	6.33 / 9.52	5.41 / 8.27	4.71 / 7.48	3.40 / 7.91
Layer	1	10	20	30	39
Top-p overlap↑	0.947 / 0.941	0.879 / 0.863	0.881 / 0.863	0.904 / 0.880	0.952 / 0.935
Output MSE↓	1.36 / 4.19	18.42 / 20.22	4.76 / 5.59	2.68 / 10.74	1.55 / 4.19
Head	0	10	20	30	–
Top-p overlap↑	0.933 / 0.919	0.920 / 0.910	0.903 / 0.892	0.894 / 0.866	–
Output MSE↓	5.25 / 8.54	4.81 / 5.04	7.04 / 7.28	5.91 / 15.08	–

E Tail-Merging Strategy

E.1 Algorithm Overview

Algorithm 1 Proposed Tail-merging Strategy

Input:
$\mathbf{L} \in \mathbb{Z}^{BH\times N}$: query-cluster labels; $\mathbf{s} \in \mathbb{Z}^{BH\times C}$: cluster sizes; C: number of query-clusters
$\mathbf{P} \in \{0,1\}^{BH\times C\times W}$: key-cluster mask per cluster; W: number of key-clusters; b: block stride
Output:
$\mathbf{L}' \in \mathbb{Z}^{BH\times N}$: updated labels; $\mathbf{s}' \in \mathbb{Z}^{BH\times C'}$: merged sizes ($C' \le C$); $\mathbf{M}' \in \{0,1\}^{BH\times C'\times W}$: merged masks

$\mathbf{P}_{\text{pack}} \leftarrow \text{PackToInt64}(\mathbf{P})$; $(\mathbf{P}^*, \pi) \leftarrow \text{Unique}([\text{head_id}, \mathbf{P}_{\text{pack}}])$ ▷ merge clusters with identical key-cluster masks within each head
if duplicate masks exist **then**
  $\mathbf{s}^* \leftarrow \text{IndexAdd}(\pi, \mathbf{s})$; $\mathbf{L}^* \leftarrow \text{Remap}(\mathbf{L}, \pi)$ ▷ accumulate sizes of merged clusters; rewrite token labels via π
else
  $\mathbf{L}^*, \mathbf{s}^*, \mathbf{P}^* \leftarrow \mathbf{L}, \mathbf{s}, \mathbf{P}_{\text{pack}}$
end if
$\boldsymbol{\tau} \leftarrow \mathbf{s}^* \bmod b$; $\mathbf{T} \leftarrow \text{ComputeIsTail}(\mathbf{L}^*, \mathbf{s}^*, b)$ ▷ $\boldsymbol{\tau}_i$: tail length of cluster i; $\mathbf{T} \in \{0,1\}^{BH\times N}$ flags tail tokens
if $\exists\, \boldsymbol{\tau}_i > 0$ **then**
  $\mathbf{D} \leftarrow \text{HammingDist}(\mathbf{P}^*, \mathbf{P}^*)$; $\mathbf{D}_{ii} \leftarrow \infty$
  $\mathbf{D}_{ij} \leftarrow \infty$ if $\boldsymbol{\tau}_i = 0 \vee \boldsymbol{\tau}_j = 0 \vee \boldsymbol{\tau}_i + \boldsymbol{\tau}_j > b$ ▷ forbid self-merge, clusters without tails, and tail overflow
  for $r = 1, 2$ **do**
    $j^*(i) \leftarrow \arg\min_j \mathbf{D}_{ij}$; $\mathcal{E} \leftarrow \{(i,j) : j^*(i) = j,\, j^*(j) = i,\, i < j\}$ ▷ nearest neighbor by mask similarity; mutually selected pairs only
    if $\mathcal{E} = \emptyset$ **then break**
    end if
    for $(i,j) \in \mathcal{E}$ **do**
      $\mathbf{P}^*_i \leftarrow \mathbf{P}^*_i \cup \mathbf{P}^*_j$; $\mathbf{s}^*_i \mathrel{+}= \boldsymbol{\tau}_j$; $\mathbf{s}^*_j \mathrel{-}= \boldsymbol{\tau}_j$ ▷ merge masks via bitwise OR; transfer $\boldsymbol{\tau}_j$ tail tokens from j to i
      $\mathbf{L}^*_n \leftarrow i \;\; \forall n :\; \mathbf{L}^*_n{=}j \wedge \mathbf{T}_n{=}1$; $\mathbf{D}_{j,\cdot} = \mathbf{D}_{\cdot,j} \leftarrow \infty$ ▷ reassign tail tokens of j to i; invalidate j in this round
    end for
    if $r < 2$ **then**
      $\boldsymbol{\tau} \leftarrow \mathbf{s}^* \bmod b$; $\mathbf{T} \leftarrow \text{ComputeIsTail}(\mathbf{L}^*, \mathbf{s}^*, b)$; recompute $\mathbf{D}$ and reapply constraints to $\mathbf{D}$
    end if
  end for
end if
$\mathbf{L}' \leftarrow \text{Remap}(\mathbf{L}^*, \mathbf{s}^* > 0)$; $\mathbf{s}' \leftarrow \mathbf{s}^*[\mathbf{s}^* > 0]$; $\mathbf{M}' \leftarrow \text{Unpack}(\mathbf{P}^*[\mathbf{s}^* > 0])$ ▷ remove empty clusters and compact indices
return $\mathbf{L}',\ \mathbf{s}',\ \mathbf{M}'$

Algorithm 1 summarizes the complete tail merging procedure in iCATS. Given per-token cluster assignments, cluster sizes, and binary key-cluster masks, we first pack each mask into a compact integer and, within each head, merge clusters with identical masks exactly by summing their sizes and remapping token labels. We then selectively reassign tail tokens from non-block-aligned clusters: a pair (i, j) with $i < j$ is merged only if the two clusters are mutually closest in Hamming distance and their tails fit into one block, moving the tail tokens of j into i and merging their masks by union, which improves block utilization while largely preserving the original sparsity structure. After two rounds, empty clusters are removed and the rest are compactly reindexed within each head for subsequent sparse attention computation.

E.2 Implementation Details

In line with the previous cluster-based method [17], in our implementation, we employ FlashInfer [53] to execute sparse attention. To evaluate the effectiveness of the tail-merging strategy, we measure the relative reduction in the number of executed blocks under the iCATS-Base setting, with SNR-guided sparsity configured by $\tau = 0.999$ and $p_{\min} = 0.7$. FlashInfer organizes sparse attention computation into execution blocks, e.g., with a query-side block size of 128 and a key/value-side block size of 96. As shown in Figure 2, the average reduction across layers is about 25% at early denoising timesteps where the sparsity level remains low. As denoising progresses and the sparsity increases according to the SNR-guided sparsity schedule (see Figure 1), the key/value sequence length also decreases, and thus this average reduction gradually decreases to around 7%. The min-max range in Figure 2 further shows the variation in block reduction ratio across sparse attention layers at each denoising timestep.

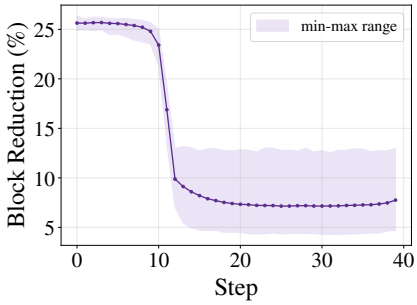


Figure 2: Relative reduction in the number of executed FlashInfer blocks on Wan2.1-T2V-14B under the iCATS-Base setting.

F Hyperparameter Sensitivity and Transferability

In the main experiments, model-specific hyperparameters are chosen to match the runtime or sparsity of the corresponding baselines. To examine whether these settings can be reused across models and tasks, we directly transfer the query/key cluster counts and SNR-guided sparsity hyperparameters of HunyuanVideo-Base to Wan2.1-T2V-14B and Wan2.1-I2V-14B without retuning. Warmup timesteps, warmup layers, and clustering iterations follow each target model's baseline settings, while tail merging uses two rounds throughout. We evaluate the transferred configuration using 100 randomly sampled prompts.

Table 4 shows that the transferred configuration achieves better PSNR, SSIM, and LPIPS than SVG2 on both tasks at comparable speedup. On Wan2.1-T2V-14B, it achieves 28.656 dB PSNR, compared with 25.411 dB for SVG2, while differing from the Wan-tuned iCATS-Base configuration by only 0.404 dB. On Wan2.1-I2V-14B, it improves PSNR from 29.068 to 29.810 dB with slightly higher speedup. These results support the reuse of a shared clustering and sparsity-schedule configuration across the evaluated models and tasks.

We further examine hyperparameter sensitivity on Wan2.1-T2V-14B using the same prompt set. Starting from the Base setting ($\tau = 0.999$, $p_{\min} = 0.7$), we independently decrease τ to 0.99, 0.98, and 0.97, or decrease $p_{\min}$ to 0.6, while keeping all other settings fixed. Unlike the configuration-transfer experiment, these runs retain the Wan-specific query/key cluster counts to isolate the effects of the two sparsity-schedule hyperparameters.

Table 4: Hyperparameter transfer from HunyuanVideo-Base to Wan2.1 without retuning.

Method	PSNR↑	SSIM↑	LPIPS↓	Speedup↑	Sparsity
Wan2.1-T2V-14B					
SVG2	25.411	0.845	0.101	1.54×	68.7%
iCATS-Base (Wan-tuned)	29.060	0.904	0.060	1.54×	60.5%
iCATS (transferred)	28.656	0.899	0.065	1.55×	66.2%
Wan2.1-I2V-14B					
SVG2	29.068	0.890	0.067	1.42×	70.1%
iCATS (transferred)	29.810	0.901	0.063	1.44×	71.5%

As shown in Table 5, decreasing either parameter increases sparsity and speedup, with a gradual reduction in reconstruction quality. Across the tested τ values, sparsity increases from 60.5% to 66.4%, while speedup increases from 1.54× to 1.58×. All tested settings retain better PSNR, SSIM, and LPIPS than SVG2, indicating that the quality advantage is maintained across the evaluated parameter range rather than depending on a single setting.

Table 5: Hyperparameter sensitivity on Wan2.1-T2V-14B. Each variant changes only the indicated parameter from iCATS-Base.

Setting	PSNR↑	SSIM↑	LPIPS↓	Speedup↑	Sparsity
SVG2	25.411	0.845	0.101	1.54×	68.7%
iCATS-Base	29.060	0.904	0.060	1.54×	60.5%
$\tau = 0.99$	28.716	0.900	0.063	1.55×	63.4%
$\tau = 0.98$	28.537	0.897	0.065	1.56×	65.2%
$\tau = 0.97$	28.392	0.895	0.067	1.58×	66.4%
$p_{\min} = 0.6$	28.576	0.896	0.066	1.57×	63.8%

G Runtime Breakdown

We profile iCATS-Turbo on Wan2.1-T2V-14B over 10 randomly sampled prompts using NVIDIA H100 GPUs and CUDA events. We use 50 denoising steps, 40 layers, 10 dense warmup steps, one dense warmup layer, and classifier-free guidance, with approximately 70% average sparsity over sparse timesteps. Timings cover denoising only, excluding model loading, text encoding, and VAE decoding.

Table 6: Runtime breakdown of iCATS-Turbo.

Component	Time (s)
Dense attention	286.5
Sparse attention kernel	263.3
Clustering	78.5
Token permutation	11.2
Tail merging	10.6
Sparse mask construction	3.8
Others (proj., QK-norm, RoPE, FFN, cross-attn., norm, etc.)	507.1
Total	1161.0

Clustering accounts for 6.8% of the total denoising time. Following prior methods, we use 50 initialization iterations and two refinement iterations at later timesteps, warm-started from previous centroids. Other DiT operations include projections, QK normalization, RoPE, FFNs, cross-attention, and normalization.

H Additional Visualizations

We provide additional visualization comparisons between iCATS and dense attention on Wan2.1-T2V-14B. These examples demonstrate that iCATS preserves high visual fidelity and maintains generation quality that is visually close to the original dense attention across diverse scenes and motion patterns.

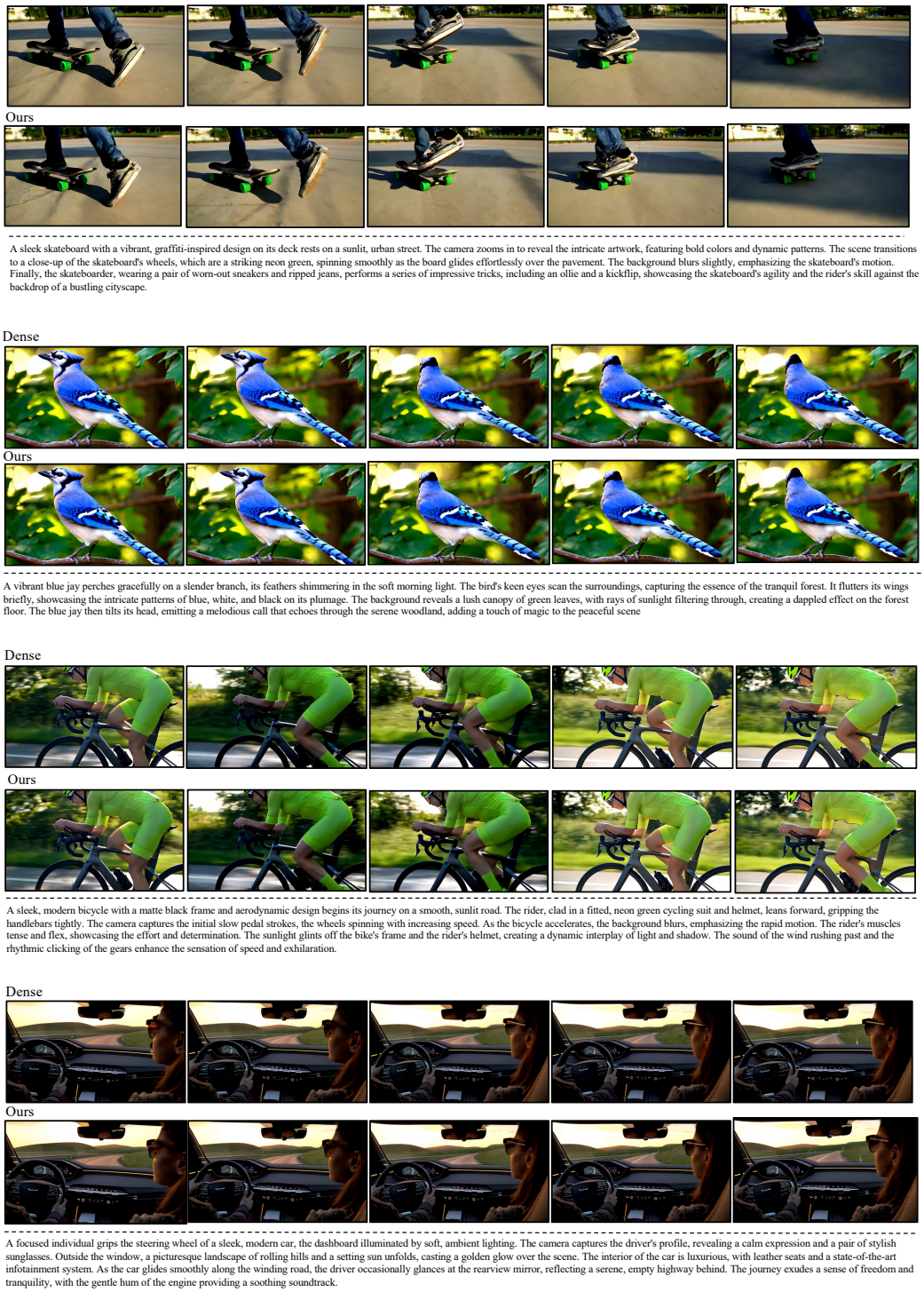
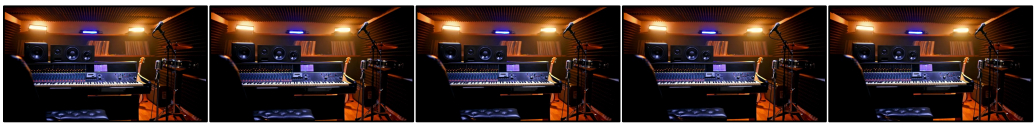
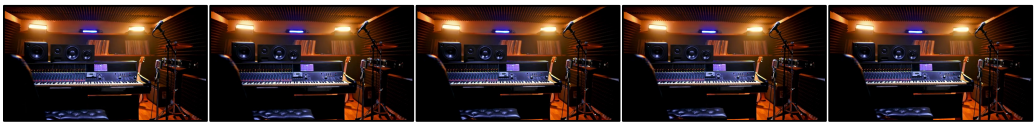


Figure 3: Visualization comparison between iCATS and dense attention on Wan2.1-T2V-14B.

Dense



Ours



A dimly lit music studio, filled with an array of high-end equipment, sets the scene. The room is adorned with soundproofing foam panels, creating an intimate and professional atmosphere. A sleek black grand piano sits in one corner, its polished surface reflecting the soft glow of ambient lighting. Nearby, a vintage microphone on a stand awaits the next vocal performance. The mixing console, with its myriad of buttons and sliders, is the heart of the studio, surrounded by monitors displaying intricate waveforms. Shelves lined with vinyl records and musical instruments, including guitars and a drum set, add to the creative vibe. The air is thick with the promise of musical magic, as the studio stands ready to capture the next hit.

Dense



Ours



A vibrant orange ceramic bowl sits on a rustic wooden table, bathed in the soft glow of morning sunlight streaming through a nearby window. The bowl's glossy surface reflects the light, highlighting its smooth curves and rich color. Inside, a collection of fresh, colorful fruits—red apples, green grapes, and yellow bananas—create a striking contrast against the bowl's vivid hue. The scene is serene and inviting, with the background featuring a blurred view of a cozy kitchen, complete with potted plants and vintage decor, enhancing the warm, homely atmosphere.

Figure 3: *Figure 3 continued.*

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately summarize the main contributions and scope of the paper. The corresponding methodological and experimental evidence is provided in Sections 4 and 5, with limitations discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes. Section 6 explicitly discusses two limitations of the current method: the lack of a highly efficient token-level sparse masking kernel and the additional overhead caused by the current clustering procedure in multi-GPU distributed settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide the mathematical derivation for interaction-aware clustering in Section 4.1.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides the model settings, evaluation protocols, compared methods, implementation details, and hardware setup needed to reproduce the main experimental results and conclusions.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The datasets used in this work are publicly available. However, the code is not released at submission time, so the paper does not provide open access to the full codebase and reproduction scripts. We plan to release the code upon publication.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes. Section 5 provides the experimental settings and implementation details necessary to understand the reported results, including the evaluated models, datasets, evaluation metrics, and key hyperparameters.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We test over multiple random seeds and the results are highly deterministic and reproducible. So we don't include error bars in the figures.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes. Section 5 provides the hardware platform and runtime information needed to understand and reproduce the reported experiments, including the type of GPU used and the execution time of the main evaluations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes. The authors have reviewed the NeurIPS Code of Ethics and confirm that the research complies with it.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This paper proposes an acceleration method for video generation models and therefore does not have direct societal impact. However, societal impacts associated with video generation models may still be amplified by improved efficiency.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a new high-risk model or dataset. It studies an inference acceleration method for existing DiT-based video generation models, so safeguards for releasing data or models are not directly applicable.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: This paper uses publicly available datasets and models in Section 5, and their use complies with the corresponding licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets, models, or code assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: Section 5.4 describes the participant instructions, viewing setup, and voting procedure for the blind human evaluation.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: N/A.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not used as an important, original, or non-standard component of the core methodology in this research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.