AdaCluster: Adaptive Query-Key Clustering for Sparse Attention in Video Generation

Haoyue Tan[1,2]*, Shengnan Wang[3]*, Yulin Qiao[4], Juncheng Zhang[1,5], Youhui Bai[1]†,
Ping Gong[1], Zewen Jin[1], Cheng Li[1,2]
[1]University of Science and Technology of China
[2]Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
[3]Independent Researcher
[4]University of Macau
[5]The Chinese University of Hong Kong

Abstract

Video diffusion transformers (DiTs) suffer from prohibitive inference latency due to quadratic attention complexity. Existing sparse attention methods either overlook semantic similarity, or fail to adapt to heterogeneous token distributions across layers, leading to model performance degradation. We propose AdaCluster, a training-free adaptive clustering framework that accelerates the generation of DiTs while preserving accuracy. AdaCluster applies an angle-similarity preserving clustering method to query vectors for higher compression, and designs a euclidean-similarity preserving clustering method for keys, covering cluster number assignment, threshold-wise adaptive clustering, and efficient critical cluster selection. Experiments on CogVideoX-2B, HunyuanVideo, and Wan-2.1 via one A40 GPU demonstrate up to 1.67×-4.31× speedup with negligible quality degradation.
Code: https://github.com/USTC-MLSys-Team/Adacluster

1. Introduction

Diffusion transformer models (DiTs) have emerged as a dominant paradigm for video generation and physical scene simulation, demonstrating strong scalability and high-fidelity synthesis capabilities [3, 6, 26]. However, generation latency remains a major bottleneck due to inherently high computational complexity [4, 41, 43]. The primary source of this cost is the attention module, whose complexity scales quadratically with the input sequence length [38]. In video generation, the sequence length grows with both resolution and the number of frames, resulting in substantially longer input sequences. For instance, in CogVideoX-2B [52], we generated a video with 81 frames at a resolution of 720p on a single NVIDIA A40 GPU. The entire generation takes 1691 seconds, where the input sequence length is 70K tokens and attention accounted for 75% of the total generation time. We also observe similar phenomena in HunyuanVideo [16] and other DiT models.

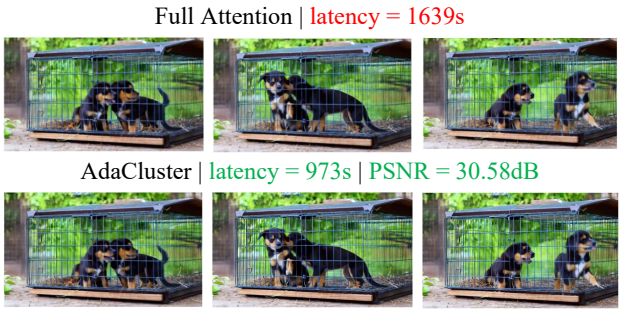

Figure 1. Performance comparison between Full Attention and AdaCluster methods on HunyuanVideo, with videos generated on a single A40. AdaCluster reduces inference latency from 1639s to 973s with a PSNR of 30.58dB, significantly improving inference speed while maintaining visual quality.

To address this, existing works, such as Top-K attention [27, 59], exploit the inherent sparsity of attention to accelerate computation, namely only a few critical tokens dominate output accuracy [40, 62]. One category of methods aggregates consecutive tokens into fixed-size blocks, and select a representative (usually the average of the tokens) from each block for significance estimation [56, 58]. However, consecutive tokens are not necessarily semantically close in the embedding space. Some tokens may be far away from the block representative, and hence leads to de-

*Equal contribution.
†Corresponding author, youhuibai@ustc.edu.cn.

graded attention accuracy. To address this issue, the token clustering methods are adopted in the later methods [51], through which higher intra-group token similarity is ensured, hence achieving better accuracy.

In cluster-based sparse attention methods, queries and keys are typically clustered to achieve significant acceleration. However, existing approaches apply a uniform strategy to both queries and keys, using the same Euclidean distance-based clustering. This overlooks the distinct roles that queries and keys play in the attention mechanism, as well as the significantly different patterns they exhibit.

In this paper, we propose **AdaCluster**, a training-free adaptive clustering framework that introduces a role-aware redesign of the entire "cluster-then-select" pipeline. We design customized clustering strategies for queries and keys separately. The core technical contributions are as follows:

1. **Query Clustering.** For the query tokens, we analyze that the relative magnitude of query–key scores is independent of the query length. Based on this, AdaCluster first normalizes queries and proposes an angle-based clustering method, which can compress the query tokens by a large ratio.
2. **Key Clustering.** For the key tokens, we show that the distribution of the key tokens varies greatly across different layers. Due to this, we propose an layerwise adaptive kmeans clustering method, covering cluster number assignment, threshold-wise adaptive clustering, and efficient critical cluster selection.
3. **Evaluation.** We build custom operators for AdaCluster on Triton [36] and FlashInfer [54], and test it on open-source DiT models including CogVideoX-2B [52], HunyuanVideo [16], and Wan-2.1 [39]. Experimental results on one A40 GPU demonstrate that AdaCluster achieves an end-to-end acceleration of **1.67**×–**4.31**× for resolutions above 720p within our test range, while maintaining high visual fidelity with PSNR of 30.99.

2. Background and Motivation

2.1. DiT Models and Computation Bottleneck

DiT models. Video generation has emerged as a transformative technology, enabling the creation of high-quality, realistic videos for applications in entertainment, advertising, and virtual reality [44]. Recently, Video Diffusion Transformers (DiTs) have demonstrated superior performance in generating temporally consistent and visually compelling videos by leveraging a Transformer-based architecture [5, 22, 42, 48, 60].

Popular DiT models share a similar architecture and inference workflow. Each layer typically includes an attention module to capture spatial-temporal dependencies across frames and a feed-forward network (FFN) for nonlinear feature enhancement [24, 48]. During inference, DiTs generate videos by iteratively denoising random latent noise over dozens of steps, with each step requiring a full forward pass to ensure temporal coherence.

Computation bottleneck of DiT. Despite their capabilities, generating high-quality videos with DiTs remains prohibitively time-consuming and resource-intensive. For instance, generating an 81-frame video at 1280×720 resolution takes more than 27 minutes on an A40 GPU with 48GB HBM. A higher-resolution version (1920×1120) of the same video takes drastically longer (130 minutes).

Our analysis identifies the attention mechanism as the primary performance bottleneck, a finding consistent with prior research [25, 37, 61]. In the two tasks mentioned above, attention computation accounted for 67.1% and 83.8% of the total time, respectively. This bottleneck arises because the computational complexity of attention scales quadratically with the input sequence length [28]. In DiTs, this sequence length is the product of the frame count, height, and width, leading to extremely long sequences [11]. For our examples, the sequence lengths reached 70K and 180K tokens, respectively. We observed a similar phenomenon in CogVideoX-2B [52], another widely used DiT model. In summary, the performance bottleneck caused by excessively long sequences in DiT attention computation severely hinders the development of applications requiring high resolution or long video generation.

2.2. Sparse Attention

To address the bottleneck of full attention, researchers have exploited the inherent sparsity of attention mechanisms [8, 50, 55]. Only a small fraction of key/value tokens contribute significantly to the final output, so selectively retaining these important tokens can substantially reduce computational overhead while preserving model quality [1, 13]. Existing approaches generally fall into two categories: static sparsity and dynamic sparsity.

Static sparsity simplifies the sparsity identification during inference by using predefined and fixed attention patterns. Methods such as MInference [15] and SVG [46] perform offline analysis of attention behavior and assign fixed sparse patterns to each attention head based on empirical statistics or heuristic rules. These sparsity patterns typically capture spatial or temporal dependencies, reflecting the inherent characteristics of attention heads in DiT [23, 49].

However, static patterns lack sufficient generality and fail to capture all important attention regions, potentially omitting critical tokens. To address this, recent methods increasingly resort to **dynamic sparsity** schemes [19, 29, 31, 45, 47, 51, 56, 57]. The most representative family is the Top-K attention approach, which computes the attention score matrix $S = QK^T$ and selects the Top-K key–value pairs via these scores, hence preserving enhanced flexibility and model quality. However, computing the whole query-

key scores for all the queries is very time-consuming. To address this, SpargeAttn [56] detects sparsity patterns at the block level by grouping consecutive tokens. In contrast, Sparse VideoGen2 (SVG2) [51] introduces token clustering for compression, yielding more flexible and efficient sparse attention scheduling. Sparse attention variations with dynamic input patterns are more widely adopted due to their generality and flexibility, which is the focus of this paper.

2.3. Limitations of Dynamic Sparsity Methods

Here, we conduct a study to uncover the limitations of existing dynamic sparsity and motivate our new design. To do so, we generate videos with Hunyuan [16] and Wan-2.1 [39] models, and analyze the distribution of query-key vectors used for computing the attention scores. The main findings are summarized as follows.

Distributions of query-key vectors are ignored. SpargeAttn [56] processes vectors in consecutive blocks without considering inherent properties such as numerical similarity. SVG2 [51] considers similarity but uses a fixed clustering scheme with 100 query clusters and 500 key clusters for all models. However, we observe that the distribution of query–key vectors differs substantially across models and layers. Here, we take the numerical distribution of key vectors as an example. As shown in Figure 2, some layers have highly dispersed token representations, requiring more clusters to preserve critical keys, or even being unsuitable for clustering. In contrast, layers with concentrated token distributions can be effectively compressed with fewer clusters. This diversity makes it difficult for existing static clustering methods to effectively capture critical tokens, potentially causing reduction in attention accuracy and output quality.

Critical tokens may be missed after clustering. After clustering, we need to evaluate the significance of all clusters for each query vector and select tokens from the Top-K critical clusters for attention computation. Existing methods typically estimate cluster significance based on attention scores computed over cluster centers. However, this approach may overlook important tokens, since a cluster center cannot fully represent all critical tokens, especially when those tokens lie near the cluster boundaries rather than close to the center.

Together, these limitations underscore the need for an adaptive query–key clustering mechanism that can dynamically determine the appropriate number of clusters, and efficiently identify critical tokens after clustering.

3. Methodology

This section introduces the overall design of AdaCluster. In the existing clustering based method SVG2 [51], both query and key vectors are treated equally, and they were clustered using the same manner. To maximize efficiency and mean-

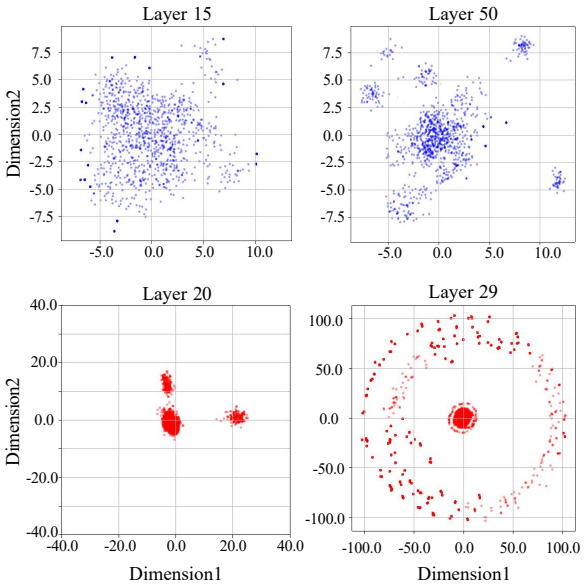

Figure 2. Token distribution visualization across different layers in Hunyuan(blue) and Wan-2.1(red) models. The token distributions are obtained by randomly sampling tokens across all steps of generating, and their 128-dimensional features are projected into a 2-dimensional space using PCA for visualization.

while ensuring accuracy, AdaCluster designs customized clustering methods for the queries and keys, respectively. We analyze that in the attention mechanism, the query and key have different patterns. In query clustering, we only need to ensure a relaxed angle similarity, so we normalize them before clustering, which can increase the compression ratio. Unlike this, the key vectors clustering is relatively complex, since it requires a critical euclidean similarity. To this end, we propose an effective method, covering cluster number assignment, threshold-wise adaptive clustering, and efficient critical cluster selection.

3.1. Query Clustering

We first introduce the method to compress the query vector. Since the query vectors are distributed in the high-dimensional space, and the lengths of the vectors vary greatly, directly clustering on the original space is difficult, only limited compression ratio can be achieved. Note that for any query q, the relative magnitude of the query-key scores s is independent of the query vector length [2]. Hence, we can first normalize the query vectors, and use the normalized query to measure the significance of the key vectors. As shown in Figure 3 , the distribution of the queries is much more compact, so we can cluster the queries by a much high compression ratio. Overall, in our query clustering method, we actually group the queries that

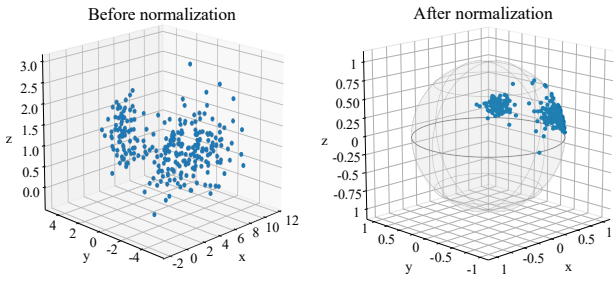

Figure 3. Query vector distributions before (left) and after (right) normalization. We normalize queries onto a unit sphere, making their distribution more compact and clustering more efficient.

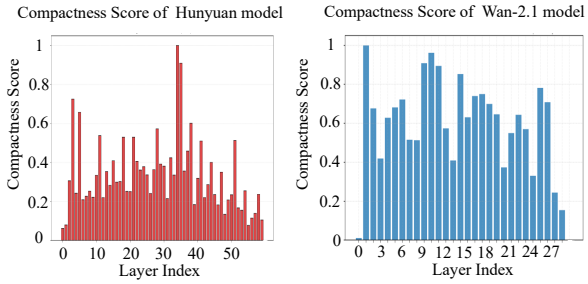

Figure 4. Layer-wise compactness scores of the Wan-2.1 and Hunyuan models. The x-axis represents the layer index, and the y-axis shows the normalized compactness score C_l. Note that the values are computed with respect to 400 cluster centers there.

are close at the angle level, which is much more efficient than the existing method using the Euclidean-level clustering.The corresponding theoretical analysis can be found in Appendix A.2.

3.2. Key Clustering

3.2.1. Observation and Analysis

Though the complexity of attention is greatly reduced after query clustering, when the sequence length is long, identifying significant key tokens is still the most time-consuming step in Top-K attention. To further speed up the denoising process, we also consider to cluster the key vectors. Different from the queries, both the vector direction and the vector lengths of the keys affect the Top-K attention result, so the keys require Euclidean-level clustering. Considering an arbitrary k, assume $c(k)$ is the cluster center of k after token clustering. Note that for any q, we have that

$$q^\top c(k) - q^\top k \leq ||q|| * ||k - c(k)||,$$

so if the tokens in the same cluster are compact enough, namely $||k - c(k)||$ is very small, then we can efficiently approximate the qk score $q^\top k$ by $q^\top c(k)$, for all the keys in this cluster. In this situation, for any query q, we do not need to measure and compare the significance of all the keys. Instead, we only need to measure and compare the significance of clusters.

According to above analysis, the compactness of the key vectors in the same cluster determines whether the critical keys can be accurately select, and hence affects the final model performance. SVG2 adopts uniform number of clusters for all the layers. However, we observe that the range of keys distribution varies greatly across different layers, as show in Figure 2. To measure the intra-cluster compactnessof different layers more explicitly, we randomly sample a request with length N, and cluster the key tokens of all the layers with a uniform number of cluster centers. Then we compute the mean squared reconstruction error (MSE) of each head i in layer l by

$$\mathrm{MSE}_l^i = \frac{1}{N}\sum_{i=1}^{N} \left\| k_l^i - c(k_l^i) \right\|_2^2$$

where k_l^i denotes the i-th key vector of head i from layer l. We define the compactness score of each layer l as $\mathrm{Comp_l} = 1/\mathrm{MSE_l}$, where MSE_l is the average of $\{\mathrm{MSE}_l^i\}$.

Figure 4 measures the compactness scores of different layers for model Wan-2.1 and Hunyuan. It is shown that, the intra-cluster compactness differs significantly across different layers. Hence, we should assign larger cluster counts for the layers with relatively scattered data distribution, and less cluster counts for the layers with relatively concentrated data distribution. The compressity for different prompts within the same layer is similar (Appendix A.1). Therefore, we only need to adjust on a per-layer basis.

3.2.2. Multi-stage K-means Clustering

As analyzed above, different number of clusters are required across layers to ensure the compactness of intra-cluster data distribution. However, it is difficult to determine the suitable number of clusters in advance. In this subsection, we adopt a multi-stage K-means clustering method to finish this task. Specifically, as shown in Figure 5, for each layer, at the initial stage, we cluster the tokens with a moderate number of clusters. After clustering, we select the outlier tokens which is relatively far from their cluster centers (beyond a predefined threshold). Then we re-cluster these outlier tokens with a few number of new clusters. We repeat such a strategy until each token is assigned to a compact cluster. In particular, if the number of clusters exceeds a preset upper limit, we will stop further clustering and treat this layer as a hard-to-compress layer. For such layers, we apply the vanilla full attention. The detailed process is shown in Algorithm 1.

Efficient initialization. DiT involves multiple denoising steps during image/video generation. We observe that for each layer the token distributions change gradually across consecutive denoising steps, i.e., adjacent steps exhibit very similar distributions. Figure 6 visualizes token distributions

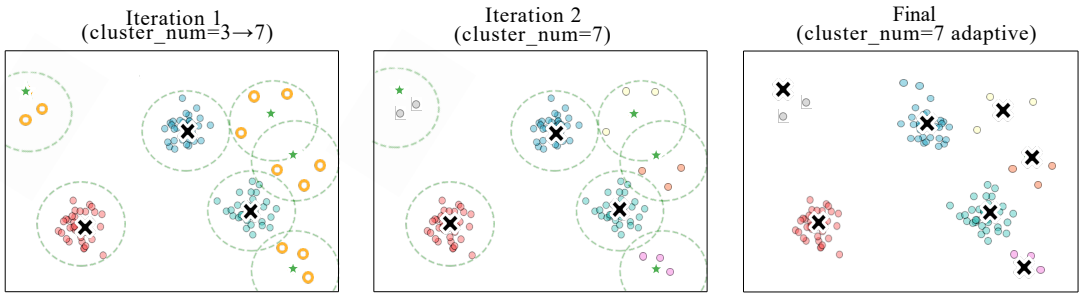

Figure 5. Visualization of Multi-stage K-means clustering. Red crosses represent tokens, black crosses denote cluster centers, and dashed circles indicate cluster boundaries. In each iteration, samples are progressively clustered into finer groups. The number of clusters (*cluster_num*) increases adaptively as unassigned samples are reclustered.

Algorithm 1 MultiStageClustering

Require: Key vectors set $\mathcal{K} = \{\mathbf{k}_1, \ldots, \mathbf{k}_n\}$, distance threshold τ, max cluster count $N_{\max}$
Ensure: Cluster assignments $\mathcal{A}$, Cluster centers $\mathcal{C}$, use full attention flag F

$\mathcal{C} \leftarrow \emptyset$
$\mathcal{U} \leftarrow \mathcal{D}$
$t \leftarrow 0$, $F \leftarrow$ False
while $\mathcal{U} \neq \emptyset$ **do**
  if $|\mathcal{C}| \geq N_{\max}$ **then**
    $F \leftarrow$ True
    break
  end if
  Choose a moderate cluster number m_t
  Index, $\mathcal{C}_t \leftarrow \text{KMeans}(\mathcal{U}, m_t)$
  for each $\mathbf{k}_i \in \mathcal{K}$ **do**
    Compute the distance $d_i \leftarrow \|k_i - c(k_i)\|_2$
    if $d_i < \tau$ **then**
      Remove k_i from $\mathcal{U}$: $\mathcal{U} \leftarrow \mathcal{U} \setminus \{k_i\}$
    end if
  end for
  $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_t$
  $t \leftarrow t + 1$
end while
Obtaining the cluster assignment set $\mathcal{A}$ by assigning each k_i in $\mathcal{K}$ to the closest cluster center in $\mathcal{C}$.
return $\mathcal{A}, \mathcal{C}, F$

across denoising steps (Layer 0 and Layer 24) using PCA. Therefore, the cluster centers computed at step t can approximately represent the token distribution of step $t+1$.

Based on this observation, we can further simplify the whole process of DiT denoising. We only need to apply the above-mentioned multi-stage kmeans clustering at the first denoising step to determine the number of clusters for each layer. In the later denoising steps, the number of clusters is fixed, the specific cluster centers will be updated. In particular, we will use the cluster centers of previous denoising step as the initialization of the current step, which can accelerate the clustering procedure in practice.

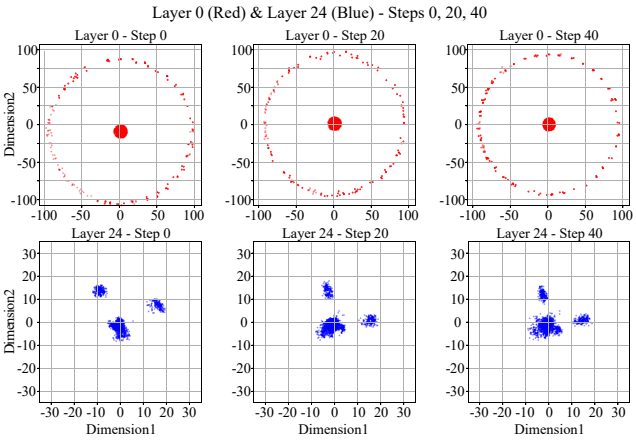

Figure 6. Token distribution consistency of Wan-2.1 across timesteps for Layer 0 (top, red) and Layer 24 (bottom, blue). Each column corresponds to a different denoising step (0, 20, 40).

3.2.3. Efficiently Identify Critical Clusters

To efficiently identify critical tokens after clustering, we take inspiration from the existing method Quest [32], which is able to select the most potentially critical KV cache pages for the current query during LLM decoding. Specifically, given a query q, and a page of key vectors $K := [k_1, k_2, ..., k_n]$, Quest identifies critical attention candidates by estimating the upper bound of attention weights across all key vectors, which is expressed as

$$\text{Quest}(q, K) = \sum_{d=1}^{D} \max(q^d * \max(K^d), q^d * \min(K^d)), \tag{1}$$

where D is the total dimension of q and k, and K^d denote the vector composed of the d-th dimension of K. One can refer to [32] for more details about Quest.

Algorithm 2 TensorQuest

Require:
Query tensor $\mathbf{Q} \in \mathbb{R}^{B\times H\times L_q\times D}$
Key tensor $\mathbf{K} \in \mathbb{R}^{B\times H\times L_k\times D}$
Ensure: Quest score $\mathbf{S}$
$\mathbf{Q}^+ \leftarrow \max(\mathbf{Q}, 0)$
$\mathbf{Q}^- \leftarrow \min(\mathbf{Q}, 0)$
$\mathbf{K}^+ \leftarrow \max(\mathbf{K}, 0)$
$\mathbf{K}^- \leftarrow \min(\mathbf{K}, 0)$
$\mathbf{S} \leftarrow \text{matmul}(\mathbf{Q}^+, (\mathbf{K}^+)^\top) + \text{matmul}(\mathbf{Q}^-, (\mathbf{K}^-)^\top)$
// shape: $[B, H, L_q, L_k]$
return $\mathbf{S}$

Inspired by Quest, we aim to adopt a similar method to measure the significance of a cluster. However, according to (1) we can see that the Quest score is purely computed in the GPU CUDA Core, which is inefficient. Compared with LLM decoding, the attention in diffusion denoising is more computationally intensive, so the existing Quest method cannot be directly applied, due to unacceptable time delay. In this section, we propose TensorQuest, which is equivalent to vanilla Quest method, but enables the main computation to be executed in the powerful tensor core, thereby significantly enhancing efficiency. To achieve this goal, we first extract the positive and negative parts of query and key separately, as follows,

$$\begin{aligned} \mathbf{q}^+ &= \max(\mathbf{q}, 0), \quad \mathbf{q}^- = \min(\mathbf{q}, 0), \\ \mathbf{k}^+ &= \max(\mathbf{k}, 0), \quad \mathbf{k}^- = \min(\mathbf{k}, 0). \end{aligned}$$

Then the Quest score can be computed by

$$\text{Quest}(q, K) = \text{matmul}(q^+, k^+) + \text{matmul}(q^-, k^-). \quad (2)$$

It can be verified that equation (2) is equivalent to (1). Except the light-weight positive and negative parts extraction, the main computation is finished in the CUDA Core. We illustrate the whole process in Algorithm 2.

Algorithm 3 AdaCluster

Require: Query Q, Key K, Value V, distance threshold τ, Top-K parameter $topk$, max cluster count $N_{\max}$
Ensure: AttnOut
$A_q, C_q \leftarrow \text{KMeans}(\text{Normalize}(Q))$
$\mathcal{A}_k, \mathcal{C}_k, F \leftarrow \text{MultiStageClustering}(K, \tau, N_{\max})$
if F is True **then**
  $\text{AttnOut} \leftarrow \text{Attention}(Q, K, V)$
else
  $S \leftarrow \text{TensorQuest}(Q, K)$
  $\text{Index} \leftarrow \text{argtopk}(S, topk)$
  $K^* \leftarrow K[\text{Index}], V^* \leftarrow V[\text{Index}]$
  $\text{AttnOut} \leftarrow \text{Attention}(Q, K^*, V^*)$
end if
return AttnOut

3.3. Overall Process

Finally, we describe the whole procedure of AdaCluster in the Algorithm 3. We cluster normalized queries using standard KMeans, while the keys are processed with our MultiStageClustering to determine whether a given layer should be compressed. For layers identified as compressible, we employ TensorQuest to rapidly select critical tokens for attention computation.

4. Experiment

4.1. Setup

Models. We evaluate the performance of AdaCluster on several Diffusion Transformer (DiT) based video generation models, including HunyuanVideo [16, 33], Wan-2.1-T2V-1.3B [30, 39], and CogVideoX-2B [35, 52]. We generate videos with multiple resolutions (e.g. 480p and 720p) via the Text-to-Video (T2V) method.

Datasets and Metrics. For the generation task, we use the prompt dataset from PenguinVideoBenchmark [34]. To evaluate the performance of the AdaCluster, we measure the generated videos from two dimensions: similarity to original videos and overall video quality, with specific metrics selected as follows. For the similarity assessment between generated videos and reference videos, we choose three metrics: Peak Signal-to-Noise Ratio(PSNR), Learned Perceptual Image Patch Similarity(LPIPS) and Structural Similarity Index Measure(SSIM). For the overall video quality evaluation, we use VBench [14] to measure four aspects of generated videos:imaging quality, background consistency, subject consistency and aesthetic quality.

Baselines. We compare the performance of AdaCluster with baseline models adopting FlashAttention [7]. Additionally, we select SpargeAttn [56] and SVG2 [51] as baselines, which represent the state-of-the-art sparse attention algorithms with dynamic input-driven patterns. We use the default hyperparameter settings reported in their original papers and code repositories.

Platform and configurations. Main experiments are conducted on one NVIDIA A40 GPU with 48GB memory. Experiments on H100 are in Apendix A.6. Our implementation builds upon Transformer Diffuser [9] and integrates custom kernels from FlashInfer [54]. For skipped layers, we use FlashAttention [7] as the full attention, while for layers with clustering, we implement a Triton-based kernel to improve computational efficiency. To balance speed and accuracy, we set k_{max} such that the top 15% of layers use full attention—these layers are relatively small, making the additional latency acceptable. The threshold τ is defined as $1.5\times$ of the average token-to-cluster-center distance computed in the first inference step. We fix the number of clusters for q to 65, while the number of clusters for k is dynamically adjusted according to our algorithm. We report

Table 1. Similarity, quality and efficiency benchmarking results of AdaCluster and baselines.

Model	Config	PSNR↑	SSIM↑	LPIPS↓	ImgeQual↑	BgConsis↑	SubConsis↑	TmpFlick↑	Speedup↑
CogVideoX-2B	720×480	-	-	-	61.15%	96.06%	92.71%	97.31%	1×
	SpargeAttn	28.189	0.517	0.618	55.77%	**95.56%**	**94.35%**	**98.14%**	1.23×
	AdaCluster	**30.989**	**0.767**	**0.231**	**56.76%**	93.95%	89.26%	95.18%	**1.67×**
Wan-2.1-1.3B	832×480	-	-	-	67.53%	96.72%	95.08%	98.04%	1×
	SpargeAttn	28.292	0.437	0.599	64.59%	**97.24%**	**96.29%**	**98.36%**	1.81×
	SVG2	28.230	0.358	0.679	66.43%	96.05%	94.36%	97.96%	1.61×
	AdaCluster	**29.083**	**0.571**	**0.393**	**66.45%**	96.35%	94.58%	97.84%	**1.85×**
HunyuanVideo	1280×720	-	-	-	62.64%	95.84%	93.05%	98.64%	1×
	SpargeAttn	28.155	0.490	0.596	61.76%	95.89%	**92.92%**	98.69%	1.33×
	SVG2	29.319	0.794	0.308	**65.42%**	**96.58%**	92.14%	98.73%	1.57×
	AdaCluster	**30.580**	**0.835**	**0.203**	65.11%	95.58%	92.56%	**98.85%**	**1.68×**

the detailed number of clusters for k under different videos configuration in the Appendix. Hyperparameter sensitivity analysis is in Appendix A.5. Due to memory constraints of A40, we use 30 inference steps for Hunyuan, while setting 50 steps for the other two models. Based on this hardware setup, we ensure that all baselines use the exact same model. For all subsequent timesteps, the cluster centers from the previous step are reused.

4.2. Quality Evaluation

We present all the accuracy evaluation results in Table 1 along two dimensions: similarity to the full-attention model and video quality scores measured by VBench.

In terms of similarity, our results indicate that AdaCluster consistently outperforms the selected baselines across all three models. This demonstrates that AdaCluster effectively leverages diverse sparsity patterns across different model layers, thereby preserving video quality to a greater extent relative to the original full-attention approach.

Regarding video quality evaluation, we observe that AdaCluster and the two baseline methods generally achieve performance comparable to that of the original model. However, two exceptions are noted: first, SpargeAttn occasionally achieves an unexpectedly high score—sometimes even surpassing the original model—which may be attributed to its tendency to generate videos that align more closely with VBench's evaluation standards, thereby making such scores less meaningful in comparative analysis.

Second, accuracy is stable on Wan-2.1 and Hunyuan with little variation across methods. On CogVideoX, however, both SpargeAttn and AdaCluster show a significant drop in image quality; SpargeAttn's sub-consistency is much higher than the original, while ours is much lower. We attribute this to the model itself—CogVideoX exhibits the poorest video quality among the three (see Appendix).

It is worth noting that video quality assessment remains an active area of research. Our objective here is primarily to demonstrate that our approach preserves visual quality without degradation, while simultaneously delivering substantial generation speedup. As shown in Table 1, AdaCluster achieves the highest end-to-end generation speedup while incurring the lowest accuracy loss. Screenshots of the generated video are provided in the Appendix.

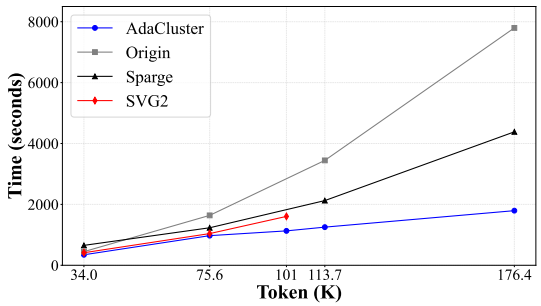

Figure 7. Time cost trend of several methods vs. seqlen.

4.3. Efficiency Evaluation

In this section, we present the end-to-end generation time for various resolutions using HunyuanVideo as the test model. The number of generated frames is fixed at 81, while the resolution is varied, with the corresponding token counts reported in Table 2. As shown in Figure 7, we observe that under different input token lengths, AdaCluster consistently achieves the highest speedup ratio. For instance, with 28.3K tokens, AdaCluster attains a speedup of 1.53×, while SpargeAttn and SVG2 achieve 1.28× and 1.46×, respectively. As the token count increases, the speedup of AdaCluster becomes even more pronounced. At the maximum input length of 176.4K tokens, AdaCluster and SpargeAttn reach their peak speedups of 4.31× and 1.78×, respectively. This trend can be attributed to the inherent sparsity of the attention mechanism—longer sequences tend to exhibit greater sparsity, thereby allowing more room for acceler-

Table 2. Number of tokens corresponding to different resolutions for HunyuanVideo under the 81-frame configuration.

RES	864×480	1280×720	1712×720	1912×720	1912×1120
Tokens	34.0K	75.6K	101.1K	113.7K	176.4K

ation. Additionally, we note that SVG2 is only applicable when the token count is below 101.1K, due to its substantial memory overhead from caching metadata for subsequent processing.

4.4. Ablation Study

All ablation studies are based on HunyuanVideo with videos generated under the configuration of 1280×720 resolution with a frame number of 81 (totally 75.6K tokens). Due to space limitations, we analyze the results from the following three perspectives only. The remaining component ablation studies can be found in Appendix A.4.

Cluster count strategy. To further validate the impact of assigning an appropriate number of clusters to each layer, we compare AdaCluster with another strategy, 'AvgClus', which assigns the same number of clusters to every layer while maintaining the same average number of clusters (specifically, 412 for the selected configuration) to ensure a fair comparison. Shown in Table 3, 'AdaClus' consistently outperforms 'AvgClus' in terms of accuracy, demonstrating that adaptively determining the cluster count for each layer can effectively enhance generation quality.

Table 3. Accuracy of AdaCluster and AverageCluster.

	PSNR↑	SSIM↑	LPIPS↓	ImgeQual↑
AdaClus	30.580	0.835	0.203	65.11%
AvgClus	29.007	0.724	0.378	64.79%

TensorQuest accuracy. We evaluate the accuracy gains from using TensorQuest (Sec 3.2.3) for cluster selection, comparing against a mean-based baseline (w/o Quest). As shown in Table 4, our method achieves higher accuracy, demonstrating that TensorQuest effectively guarantees the selection of important tokens.

Table 4. Accuracy of TensorQuest and w/o Quest.

	PSNR↑	SSIM↑	LPIPS↓	ImgeQual↑
TensorQuest	30.580	0.835	0.203	65.11%
w/o Quest	28.941	0.687	0.410	64.06%

TensorQuest reformulation benefits. Beyond the accuracy ablation study, we further evaluate the performance improvements achieved by the TensorQuest algorithm. As shown in Figure 8, the execution times under varying input token lengths are reported. For the longest input sequence, our proposed design attains a speedup of up to 5×. This performance gain stems from the efficient utilization of GPU tensor cores, which are specialized for high-throughput matrix operations, whereas the original Quest computation relies solely on the less efficient CUDA cores.

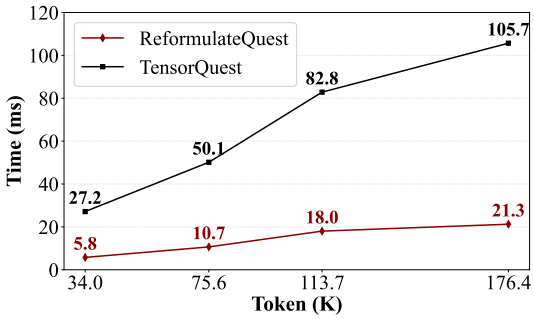

Figure 8. Top-K selection time under different input tokens.

5. Related Work

Diffusion model quantization. Quantization has been widely studied to accelerate diffusion models and reduce memory usage. For example, PTQD [12] and PQD [53] investigate post-training quantization for diffusion pipelines, while Q-DM [21] and Q-Diffusion [17] target low-bit quantization. On the other hand, low-rank quantization methods such as SVDQuant [18] and IntLoRA [10] exploit a low-rank adapter in conjunction with quantization to further improve efficiency. These methods are orthogonal to ours: they lower numerical precision, while we reduce attention workload through token clustering and selection.

Training-based sparse attention. VSA [57] learns dynamic sparsity via two-stage attention: coarse tiling selects critical tokens, followed by fine token-level attention within selected tiles. DSV [31] uses low-rank attention predictors with reduced heads, trained separately without full end-to-end integration. RadialAttention [20] extends generation length in pre-trained video DiT models via efficient LoRA fine-tuning. However, these approaches generally incur substantial re-training or fine-tuning costs.

6. Conclusion

We introduced AdaCluster, a training-free adaptive query–key clustering framework for sparse attention in DiTs. By tailoring clustering strategies to the heterogeneous distributions of query and key vectors, and by integrating efficient critical token selection via TensorQuest, AdaCluster achieves substantial improvement in generation speed while preserving high-fidelity video quality.

Acknowledgements

We thank the anonymous reviewers for their insightful comments. This work was supported by the National Natural Science Foundation of China under Grant No. 62572454. This work is done with the sponsorship of TeleAI. The corresponding author is Youhui Bai.

References

[1] Chenxin An, Jun Zhang, Ming Zhong, Lei Li, Shansan Gong, Yao Luo, Jingjing Xu, and Lingpeng Kong. Why does the effective context length of llms fall short? *arXiv preprint arXiv:2410.18745*, 2024. 2

[2] Ben Anson, Xi Wang, and Laurence Aitchison. Scale-invariant attention. 2025. 3

[3] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22563–22575, 2023. 1

[4] Haoxin Chen, Yong Zhang, Xiaodong Cun, Menghan Xia, Xintao Wang, Chao Weng, and Ying Shan. Videocrafter2: Overcoming data limitations for high-quality video diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7310–7320, 2024. 1

[5] Shoufa Chen, Mengmeng Xu, Jiawei Ren, Yuren Cong, Sen He, Yanping Xie, Animesh Sinha, Ping Luo, Tao Xiang, and Juan-Manuel Perez-Rua. Gentron: Diffusion transformers for image and video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6441–6451, 2024. 2

[6] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(9):10850–10869, 2023. 1

[7] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. 6

[8] Yichuan Deng, Zhao Song, Jing Xiong, and Chiwun Yang. How sparse attention approximates exact attention? your attention is naturally $n^c - sparse$. *arXiv preprint arXiv:2404.02690*, 2024. 2

[9] Aosong Feng, Irene Li, Yuang Jiang, and Rex Ying. Diffuser: efficient transformers with multi-hop attention diffusion for long sequences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12772–12780, 2023. 6

[10] Hang Guo, Yawei Li, Tao Dai, Shu-Tao Xia, and Luca Benini. Intlora: Integral low-rank adaptation of quantized diffusion models. *arXiv preprint arXiv:2410.21759*, 2024. 8

[11] Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024. 2

[12] Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. Ptqd: Accurate post-training quantization for diffusion models. *Advances in Neural Information Processing Systems*, 36:13237–13249, 2023. 8

[13] Weizhe Hua, Zihang Dai, Hanxiao Liu, and Quoc Le. Transformer quality in linear time. In *International conference on machine learning*, pages 9099–9117. PMLR, 2022. 2

[14] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21807–21818, 2024. 6

[15] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024. 2

[16] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024. 1, 2, 3, 6

[17] Seunghoon Lee, Jeongwoo Choi, Byunggwan Son, Jaehyeon Moon, Jeimin Jeon, and Bumsub Ham. Q-diffusion: Quantizing diffusion models. *arXiv preprint arXiv:2302.04304*, 2023. 8

[18] Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024. 8

[19] Quanhao Li, Zhen Xing, Rui Wang, Hui Zhang, Qi Dai, and Zuxuan Wu. Magicmotion: Controllable video generation with dense-to-sparse trajectory guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12112–12123, 2025. 2

[20] Xingyang Li, Muyang Li, Tianle Cai, Haocheng Xi, Shuo Yang, Yujun Lin, Lvmin Zhang, Songlin Yang, Jinbo Hu, Kelly Peng, Maneesh Agrawala, Ion Stoica, Kurt Keutzer, and Song Han. Radial attention: $o(n \log n)$ sparse attention with energy decay for long video generation, 2025. 8

[21] Yanjing Li, Sheng Xu, Xianbin Cao, Xiao Sun, and Baochang Zhang. Q-dm: An efficient low-bit quantized diffusion model. *Advances in neural information processing systems*, 36:76680–76691, 2023. 8

[22] Zongyi Li, Shujie Hu, Shujie Liu, Long Zhou, Jeongsoo Choi, Lingwei Meng, Xun Guo, Jinyu Li, Hefei Ling, and Furu Wei. Arlon: Boosting diffusion transformers with autoregressive models for long video generation. *arXiv preprint arXiv:2410.20502*, 2024. 2

[23] Liu Liu, Zheng Qu, Zhaodong Chen, Yufei Ding, and Yuan Xie. Transformer acceleration with dynamic sparse attention. *arXiv preprint arXiv:2110.11299*, 2021. 2

[24] Xin Ma, Yaohui Wang, Xinyuan Chen, Gengyun Jia, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*, 2024. 2

[25] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. *Advances in neural information processing systems*, 34:14200–14213, 2021. 2

[26] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, 2023. 1

[27] Wenqi Shao, Yixiao Ge, Zhaoyang Zhang, Xuyuan Xu, Xiaogang Wang, Ying Shan, and Ping Luo. Dynamic token normalization improves vision transformers, 2022. 1

[28] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539, 2021. 2

[29] Austin Silveria, Soham V Govande, and Daniel Y Fu. Chipmunk: Training-free acceleration of diffusion transformers with dynamic column-sparse deltas. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025. 2

[30] Steven-SWZhang. Wan-video/wan2.2. GitHub repository. 6

[31] Xin Tan, Yuetao Chen, Yimin Jiang, Xing Chen, Kun Yan, Nan Duan, Yibo Zhu, Daxin Jiang, and Hong Xu. DSV: Exploiting dynamic sparsity to accelerate large-scale video dit training. *arXiv preprint arXiv:2502.07590*, 2025. 2, 8

[32] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024. 5

[33] Hunyuan Foundation Model Team. Tencent-hunyuan/hunyuanvideo, . GitHub repository. 6

[34] Hunyuan Foundation Model Team. Penguin video benchmark, . GitHub repository. 6

[35] tengjiayan20. zai-org/cogvideo. GitHub repository. 6

[36] Philippe Tillet, Hsiang-Tsung Kung, and David 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. 2

[37] John K Tsotsos. *A computational perspective on visual attention*. MIT press, 2021. 2

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017. 1

[39] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 2, 3, 6

[40] Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110, 2021. 1

[41] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023. 1

[42] Kai Wang, Shijian Deng, Jing Shi, Dimitrios Hatzinakos, and Yapeng Tian. Av-dit: Efficient audio-visual diffusion transformer for joint audio and video generation. *arXiv preprint arXiv:2406.07686*, 2024. 2

[43] Xiang Wang, Shiwei Zhang, Hangjie Yuan, Zhiwu Qing, Biao Gong, Yingya Zhang, Yujun Shen, Changxin Gao, and Nong Sang. A recipe for scaling up text-to-video generation with text-free videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6572–6582, 2024. 1

[44] Yimu Wang, Xuye Liu, Wei Pang, Li Ma, Shuai Yuan, Paul Debevec, and Ning Yu. Survey of video diffusion models: Foundations, implementations, and applications, 2025. 2

[45] Zijie Wu, Chaohui Yu, Yanqin Jiang, Chenjie Cao, Fan Wang, and Xiang Bai. Sc4d: Sparse-controlled video-to-4d generation and motion transfer. In *European Conference on Computer Vision*, pages 361–379. Springer, 2024. 2

[46] 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

[47] Yifei Xia, Suhan Ling, Fangcheng Fu, Yujie Wang, Huixia Li, Xuefeng Xiao, and Bin Cui. Training-free and adaptive sparse attention for efficient long video generation. *arXiv preprint arXiv:2502.21079*, 2025. 2

[48] Zhen Xing, Qijun Feng, Haoran Chen, Qi Dai, Han Hu, Hang Xu, Zuxuan Wu, and Yu-Gang Jiang. A survey on video diffusion models. *ACM Computing Surveys*, 57(2):1–42, 2024. 2

[49] Ning Yang, Mingrui Fan, Wentao Wang, and Haijun Zhang. Decision-making large language model for wireless communication: A comprehensive survey on key techniques. *IEEE Communications Surveys & Tutorials*, 2025. 2

[50] Shuo Yang, Ying Sheng, Joseph E Gonzalez, Ion Stoica, and Lianmin Zheng. Post-training sparse attention with double sparsity. *arXiv preprint arXiv:2408.07092*, 2024. 2

[51] Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, et al. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *arXiv preprint arXiv:2505.18875*, 2025. 2, 3, 6

[52] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024. 1, 2, 6

[53] Jiaojiao Ye, Zhen Wang, and Linnan Jiang. Pqd: Post-training quantization for efficient diffusion models. *arXiv preprint arXiv:2501.00124*, 2024. 8

[54] Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod

Grover, Arvind Krishnamurthy, et al. Flashinfer: Efficient and customizable attention engine for llm inference serving. *arXiv preprint arXiv:2501.01005*, 2025. 2, 6

[55] Bokyeong Yoon, Ah-Hyun Lee, Jinsung Kim, and Gordon Euhyun Moon. Exploring attention sparsity to accelerate transformer training on gpus. *IEEE Access*, 2024. 2

[56] Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattn: Accurate sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025. 1, 2, 3, 6

[57] 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, 8

[58] Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention. *arXiv preprint arXiv:2502.04507*, 2025. 1

[59] Yiming Zhang, Zhuokai Zhao, Zhaorun Chen, Zenghui Ding, Xianjun Yang, and Yining Sun. Beyond training: Dynamic token merging for zero-shot video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22046–22055, 2025. 1

[60] Zhenghao Zhang, Junchao Liao, Menghao Li, Zuozhuo Dai, Bingxue Qiu, Siyu Zhu, Long Qin, and Weizhi Wang. Tora: Trajectory-oriented diffusion transformer for video generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2063–2073, 2025. 2

[61] Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588*, 2024. 2

[62] Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai. An empirical study of spatial attention mechanisms in deep networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6688–6697, 2019. 1