RESISPARSE: RESIDUAL-INFORMED EXACT-OR-APPROXIMATE SPARSE ATTENTION FOR EFFICIENT VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Video diffusion transformers repeatedly apply spatiotemporal self-attention, making denoising expensive as video sequences grow. Training-free sparse attention reduces computation, but discarding unselected interactions can make model outputs deviate from their dense-attention counterparts. A recent paradigm computes selected interactions exactly and approximates the rest. Accurate approximation remains essential: replacing individual keys and values with cluster centroids can still distort the output. We introduce ResiSparse, a training-free sparse-attention method guided by a theoretical analysis of centroid-based approximation. This analysis motivates three components: (i) Query-Induced Key Clustering uses query information to reweight key channels before clustering, reducing centroid-induced logit distortion; (ii) Unified Mass–Dispersion Routing combines centroid logits with within-cluster key and value variation to prioritize exact computation using a cost-normalized score; and (iii) Query-Prioritized Taylor Compensation uses a second-order expansion to account for logit variation within approximated blocks, correcting their centroid-based contributions to both the attention numerator and softmax denominator. Across Wan2.1-1.3B, Wan2.1-14B, and HunyuanVideo-13B, ResiSparse outperforms the evaluated training-free sparse-attention methods and achieves a stronger quality–efficiency trade-off, with PSNRs of 25.49, 21.49, and 27.46 at $2.18\times$, $1.59\times$, and $1.71\times$ denoising speedups, respectively.



Figure 1: Qualitative comparison with dense attention. Red boxes highlight SVG-EAR's speckle-like artifacts, which are subtle in a single frame but become more apparent as flickering in the video; ResiSparse largely suppresses them and more closely matches the dense output. PSNR measures dense-reference fidelity, while latency and speedup report denoising efficiency. Additional qualitative comparisons appear in Appendix D.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Recent video generation models produce increasingly high-resolution and long-form outputs (Ho et al., 2022; Ma et al., 2024; Yang et al., 2025b; Kong et al., 2024; Wan et al., 2025). This progress expands the latent sequence along both space and time, while diffusion transformers (Peebles & Xie, 2023) repeatedly apply spatiotemporal self-attention throughout denoising. For N_{tok} video tokens and head dimension d, each dense attention layer costs $O(N_{\mathrm{tok}}^2 d)$; across many layers and denoising steps, spatiotemporal self-attention consequently becomes a dominant computational bottleneck. Reducing this cost without modifying model weights is especially valuable because modern video generators are expensive to train.

Training-free sparse attention addresses this bottleneck by exploiting the fact that only a subset of token interactions is strongly activated for a particular layer and denoising step (Xi et al., 2025; Zhang et al., 2025b; Yang et al., 2025a). Most existing methods follow an *exact-or-drop* strategy: they identify important query–key blocks, compute those blocks exactly, and omit the rest. This strategy is efficient and maps naturally to block-sparse kernels, but it irreversibly discards all unselected blocks, whose aggregate contribution can remain non-negligible. In particular, dropping a block removes both its weighted-value contribution from the attention numerator and its contribution to the softmax denominator.

These observations motivate an *exact-or-approximate* alternative (Li et al., 2026b; Zhou et al., 2026; Liu et al., 2025; Li et al., 2026a). Important blocks are evaluated exactly, whereas the remaining blocks are approximated rather than discarded. PISA (Li et al., 2026b) approximates the unselected blocks through a blockwise Taylor expansion. However, when tokens within a block are highly heterogeneous, a block mean may inaccurately reproduce their tokenwise interactions.

SVG-EAR (Zhou et al., 2026) mitigates this limitation by adopting the semantic grouping and permutation strategy of SVG2 (Yang et al., 2025a). It groups similar query and key tokens into clusters and reorders them into contiguous computational blocks, thereby reducing within-block heterogeneity. SVG-EAR also uses a tokenwise estimate of compensation error to prioritize cluster pairs for exact computation. For each approximated block, the key and value tokens are replaced by their respective cluster centroids, and the aggregate contribution of the key cluster is computed separately for each query token.

Despite these advances in cluster construction and error-aware routing, we argue that preserving the dense-attention output still depends critically on how accurately the non-exact blocks are approximated. Semantic clustering makes tokens within a block more similar, and error-aware routing moves difficult cluster pairs to exact computation, but the remaining interactions are still approximated using key and value centroids. This raises a central question:

How can centroid-based approximation of the remaining interactions be made more accurate without sacrificing sparse-attention efficiency?

To answer this question, we develop a theoretical framework for analyzing centroid-based approximation. For a given query, replacing a cluster's keys with their centroid assigns the same logit to all tokens in that cluster. This affects the sum of exponentiated logits within the cluster, which enters both the softmax denominator and the value-centroid contribution to the numerator. The weighted-value sum also depends on how the exponentiated logits vary with the values within the cluster. We derive exact expressions for these effects in terms of centroid-induced logit deviations and within-cluster value variation, providing a basis for improving centroid-based compensation.

ResiSparse uses this analysis to improve both the construction and compensation of approximated blocks. **Query-Induced Key Clustering (QIKC)** reweights key channels according to their influence on query-dependent logits, targeting the logit deviations that affect both softmax normalization and value weighting. **Unified Mass–Dispersion Routing (UMDR)** bounds the additional within-cluster value contribution using the centroid logit and key and value dispersions, then normalizes this bound by the cost of exact computation to rank candidate pairs. **Query-Prioritized Taylor Compensation (QPTC)** estimates a second-order correction to each remaining cluster's sum of exponentiated logits from query-prioritized key-channel variances, updating both numerator and denominator. QIKC thus improves the clustering, UMDR selects pairs for exact computation, and QPTC applies the estimated second-order corrections.

Figure 1 illustrates the resulting qualitative improvement: ResiSparse suppresses the speckle-like artifacts visible with SVG-EAR and more closely matches the dense-attention output. Across Wan2.1-1.3B, Wan2.1-14B, and HunyuanVideo-13B, ResiSparse consistently improves both sides of the quality–efficiency trade-off: it achieves the highest dense-reference fidelity among the evaluated sparse methods while also reducing denoising latency. On Wan2.1-14B, for example, ResiSparse increases PSNR from 20.63 to 21.49 and reduces latency from 1064 to $992\,\mathrm{s}$. Across all three configurations, it delivers $1.59\times$–$2.18\times$ speedups over dense attention while keeping the VBench average within 0.45 points of the dense result. The mechanism evaluations examine logit distortion, routing quality, and QPTC's correction accuracy. Our contributions are:

- We develop **an exact cluster-level analysis** of centroid approximation, characterizing its effects on the attention numerator and softmax denominator through key-induced logit deviations and within-cluster value variation.
- Guided by this analysis, we develop **Query-Induced Key Clustering**, **Unified Mass–Dispersion Routing**, and **Query-Prioritized Taylor Compensation** to reduce centroid-induced logit deviations, prioritize exact computation using a cost-normalized bound, and estimate the second-order correction for the remaining approximated pairs.
- We develop **efficient GPU implementations** of the three components, reusing clustering results for routing and compensation and fusing compensation with the output merge to limit additional overhead. Across Wan2.1-1.3B, Wan2.1-14B, and HunyuanVideo-13B, ResiSparse achieves the highest dense-reference fidelity among the evaluated sparse methods while providing $1.59\times$–$2.18\times$ denoising speedups.

2 Related Work

Training-free exact-or-drop sparse attention. Training-free methods construct sparse attention patterns for pretrained video diffusion models without updating model parameters. Some methods exploit predefined or profiled spatiotemporal structure (Xi et al., 2025; Zhang et al., 2025b; Chen et al., 2026), whereas others estimate block importance online (Zhang et al., 2025a; Hu et al., 2026). Among the latter, several methods group semantically similar queries and keys before scoring the resulting blocks (Yang et al., 2025a; Luo et al., 2026; Tan et al., 2026; Lee et al., 2026). Although these methods differ in how they form and select blocks, they largely follow an exact-or-drop strategy: selected blocks are computed exactly, and the remaining blocks contribute neither to the weighted-value sum nor to softmax normalization.

Training-free exact-or-approximate sparse attention. A newer line preserves information outside the exactly computed blocks through approximation or reconstruction. PISA (Li et al., 2026b) applies a blockwise Taylor approximation; Rectified SpaAttn (Liu et al., 2025) corrects sparse attention using pooled weights; Sol-Attn (Li et al., 2026a) reuses proxy scores within online softmax to approximate unselected blocks; Re-ttention (Chen et al., 2025) reshapes sparse attention using softmax statistics from earlier diffusion steps; and SparsePR (Taghavi et al., 2026) fits an output-space residual correction from sampled exact query rows. Most closely related, SVG-EAR (Zhou et al., 2026) combines semantic clustering with centroid-based compensation and routes blocks according to a tokenwise estimate of compensation error. ResiSparse analyzes how centroid-induced logit deviations affect each cluster's sum of exponentiated logits and within-cluster value contribution to guide clustering, low-cost routing, and second-order compensation.

3 Problem Formulation and Analysis

3.1 Centroid-Based Exact-or-Approximate Attention

For one attention head with dimension d, scaled dot-product attention (Vaswani et al., 2017) maps a query q and N_{k} key–value pairs (k_t, v_t) to

$$O(q) = \frac{\sum_{t=1}^{N_{\mathrm{k}}} \exp(q^\top k_t/\sqrt{d})v_t}{\sum_{t=1}^{N_{\mathrm{k}}} \exp(q^\top k_t/\sqrt{d})}. \tag{1}$$

Following SVG2 (Yang et al., 2025a), we cluster query and key tokens and permute the clusters into contiguous blocks; paired values follow the key assignments. This produces C_{q} query clusters and C_{k} key–value clusters. Let s and t index their tokens, i and j index their clusters, m_i and n_j denote cluster sizes, and $\bar{q}_i$, $\bar{k}_j$, and $\bar{v}_j$ denote the corresponding centroids. We use $\mathbb{E}_s[\cdot]$ and $\mathbb{E}_j[\cdot]$ for averages over query tokens and the key–value tokens in cluster j, respectively. Following SVG-EAR (Zhou et al., 2026), $\mathcal{S}_i$ contains the key clusters computed exactly for query cluster i, whereas $\mathcal{D}_i$ contains those approximated by their key and value centroids.

Each query cluster shares one routing pattern, while centroid logits for approximated clusters are evaluated separately for its query tokens. Following SVG-EAR's use of query centroids for routing-error estimation (Zhou et al., 2026), we analyze cluster contributions at $\bar{q}_i$. Appendix A.5 separately relates attention-output errors at the query centroid and individual query tokens. At $\bar{q}_i$, the exact sum of exponentiated logits and weighted-value sum for pair (i, j) are

$$Z_{ij} := \sum_{t \in j} \exp(\bar{q}_i^\top k_t / \sqrt{d}), \qquad Y_{ij} := \sum_{t \in j} \exp(\bar{q}_i^\top k_t / \sqrt{d}) v_t. \tag{2}$$

For a non-exact pair $j \in \mathcal{D}_i$, define

$$\widehat{Z}_{ij} := n_j \exp(\bar{q}_i^\top \bar{k}_j / \sqrt{d}), \qquad \widehat{Y}_{ij} := \widehat{Z}_{ij} \bar{v}_j. \tag{3}$$

The exact output is obtained by summing Y_{ij} over clusters and dividing by $Z_i := \sum_j Z_{ij}$. Exact-or-approximate attention replaces (Y_{ij}, Z_{ij}) with $(\widehat{Y}_{ij}, \widehat{Z}_{ij})$ for $j \in \mathcal{D}_i$ before summation and normalization.

3.2 Effects of Centroid Substitution

We analyze how centroid substitution changes Z_{ij} and Y_{ij}, the two quantities accumulated before attention normalization.

For each non-exact pair $j \in \mathcal{D}_i$, write

$$k_t = \bar{k}_j + \Delta k_t, \qquad v_t = \bar{v}_j + \Delta v_t, \qquad \delta \ell_{ij}(t) := \frac{\bar{q}_i^\top \Delta k_t}{\sqrt{d}}. \tag{4}$$

Proposition 1 (Cluster-level decomposition of the attention numerator and denominator)**.** *For each approximated query–key cluster pair (i, j), the exact quantities in Equation 2 satisfy*

$$\begin{aligned} Z_{ij} &= \widehat{Z}_{ij} \mathbb{E}_j[e^{\delta \ell_{ij}(t)}], \\ Y_{ij} &= Z_{ij} \bar{v}_j + A_{ij}, \end{aligned} \tag{5}$$

where the within-cluster value-weighting term is

$$A_{ij} := \widehat{Z}_{ij} \mathbb{E}_j[(e^{\delta \ell_{ij}(t)} - 1) \Delta v_t]. \tag{6}$$

The factor $e^{\delta \ell_{ij}(t)}$ is the ratio of the exponentiated token logit to the exponentiated centroid logit. Its cluster average scales $\widehat{Z}_{ij}$ to give Z_{ij}, while its variation across tokens weights the value deviations in A_{ij}. Thus, the exact weighted-value sum contains the value centroid weighted by Z_{ij} and the additional contribution A_{ij}. **Centroid approximation replaces Z_{ij} by $\widehat{Z}_{ij}$ and omits A_{ij}.** In the numerator, Z_{ij} weights the value centroid $\bar{v}_j$; the softmax denominator is the sum of Z_{ij} over clusters. Although value deviations average to zero, their contributions need not cancel under token-specific exponential weights. Appendix A.1 proves these identities.

3.3 Motivation from the Cluster-Level Analysis

Proposition 1 identifies how logit deviations and value variation affect centroid approximation. When $\delta \ell_{ij}(t)$ is close to zero, $e^{\delta \ell_{ij}(t)}$ approaches one, reducing the discrepancy between Z_{ij} and $\widehat{Z}_{ij}$ and weakening the unequal weighting of value deviations in A_{ij}. This motivates constructing clusters that preserve query-dependent logits. For the remaining deviations, improving the estimate of Z_{ij} refines both the value-centroid contribution to the numerator and the softmax denominator. Meanwhile, pairs with large $\|A_{ij}\|$ are natural candidates for exact computation, since centroid compensation omits this contribution. Identifying these pairs requires considering value variation together with logit variation.

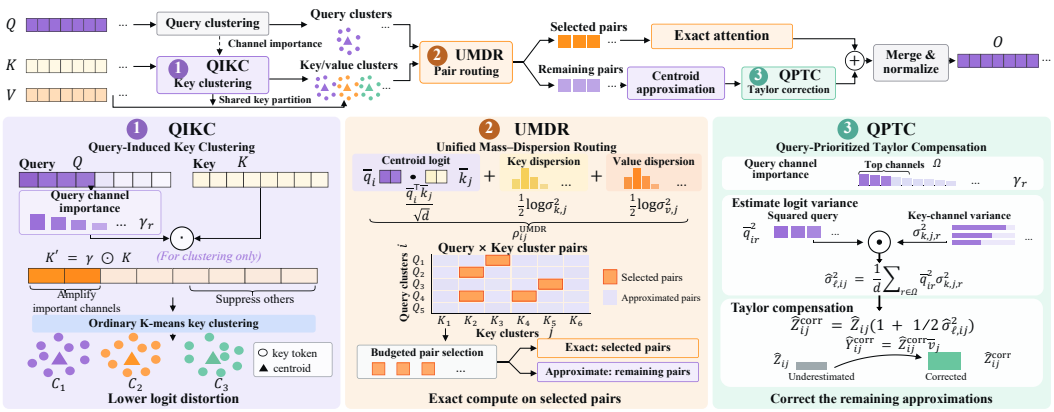


Figure 2: Overview of ResiSparse. The top shows the sparse-attention workflow, and the bottom illustrates the three proposed improvements. QIKC uses query channel energies to scale keys before clustering, reducing centroid-induced logit distortion. UMDR combines centroid logits with key and value dispersions to prioritize cluster pairs for exact computation. QPTC estimates a second-order denominator correction from query-prioritized key-channel variances and updates the numerator accordingly.

4 METHOD

Figure 2 summarizes the three mechanisms in ResiSparse.

4.1 QUERY-INDUCED KEY CLUSTERING (QIKC)

Recent video sparse-attention methods organize queries and keys into semantically coherent clusters so that important interactions form contiguous computational blocks (Yang et al., 2025a; Luo et al., 2026; Tan et al., 2026; Lee et al., 2026). ResiSparse also uses the key and value centroids to approximate pairs outside the exact region: each query token is scored against $\bar{k}_j$, and the aggregate weight multiplies $\bar{v}_j$. The key partition should therefore minimize the logit distortion introduced by replacing individual keys with $\bar{k}_j$, rather than Euclidean key variation alone. For query q_s, replacing k_t with $\bar{k}_j$ changes the logit by $q_s^\top(k_t - \bar{k}_j)/\sqrt{d}$, whose mean squared value is

$$\mathbb{E}_s\left[\left(\frac{q_s^\top(k_t - \bar{k}_j)}{\sqrt{d}}\right)^2\right] = \frac{1}{d}(k_t - \bar{k}_j)^\top \mathbb{E}_s[q_s q_s^\top](k_t - \bar{k}_j). \tag{7}$$

The mean squared query value in each channel, its query channel energy, determines the corresponding diagonal weight in this expression. Forming the complete query second-moment matrix $\mathbb{E}_s[q_s q_s^\top] \in \mathbb{R}^{d\times d}$ requires $O(N_\mathrm{q} d^2)$ work. One way to use this matrix with ordinary K-means is to factorize the $d \times d$ matrix to obtain a dense linear transformation, costing $O(d^3)$, and apply it to all N_k keys, costing $O(N_\mathrm{k} d^2)$. This dense per-key preprocessing is costly for long video sequences. Computing the exact diagonal would still require $O(N_\mathrm{q} d)$ work over all query tokens. QIKC therefore estimates these channel energies from the already-available C_q query centroids, weighting their squared values by cluster size. Let $r \in \{1, \ldots, d\}$ index a feature channel, let m_i be the number of queries in cluster i, and let $N_\mathrm{q} = \sum_i m_i$. The estimated query channel energy $\widetilde{\omega}_r$, the normalization factor α_q, and the resulting key-channel scale γ_r are

$$\widetilde{\omega}_r = \sum_{i=1}^{C_\mathrm{q}} \frac{m_i}{N_\mathrm{q}} \bar{q}_{ir}^2, \qquad \alpha_\mathrm{q} = \frac{1}{d}\sum_{p=1}^{d} \sqrt{\widetilde{\omega}_p}, \qquad \gamma_r = \frac{\sqrt{\widetilde{\omega}_r}}{\alpha_\mathrm{q}}. \tag{8}$$

The square root converts the estimated energy $\widetilde{\omega}_r$ into a coordinate scale: after scaling, Euclidean distance weights channel r by γ_r^2, which is proportional to $\widetilde{\omega}_r$. The factor α_q normalizes the channel scales to have mean one, leaving only their relative weighting to change. Estimating these energies and transforming the keys require $O(C_\mathrm{q} d)$ and $O(N_\mathrm{k} d)$ work, respectively.

QIKC then runs ordinary K-means on the element-wise scaled keys $\mathrm{diag}(\gamma)k_t$. Channels with higher estimated query energy therefore contribute more strongly to the cluster assignments. After assignment, the key centroids are recomputed in the original feature space; the scaling changes the partition but not the keys or logits used by attention. Appendix A.2 derives the full-matrix-to-diagonal approximation.

4.2 Unified Mass–Dispersion Routing (UMDR)

SVG-EAR (Zhou et al., 2026) estimates compensation error by comparing each value token weighted by its exponentiated logit with the value centroid weighted by the corresponding centroid-based exponentiated logit. Using query centroids as proxies, this estimate probes every key–value token pair and costs $O(C_{\mathrm{q}}N_{\mathrm{k}}d)$. UMDR instead bounds the value-weighting term A_{ij} in Equation 6 using scalar key and value dispersions. Exact computation recovers this within-cluster contribution for selected pairs.

For each key–value cluster j, we compute scalar key and value dispersions:

$$\sigma^2_{\mathrm{v},j} := \frac{1}{n_j d}\sum_{t\in j}\|v_t - \bar{v}_j\|^2, \qquad \sigma^2_{\mathrm{k},j} := \frac{1}{n_j d}\sum_{t\in j}\|k_t - \bar{k}_j\|^2. \tag{9}$$

These quantities are the mean squared key and value deviations per feature channel within cluster j. Bounding logit variation with the scalar key dispersion avoids the $O(C_{\mathrm{k}}d^2)$ storage of full key-cluster covariances. Let $\kappa_i = \max_{j,t\in j}|\delta\ell_{ij}(t)|$ over the candidate clusters in row i. The mean-value theorem gives $|e^{\delta\ell} - 1| \le e^{\kappa_i}|\delta\ell|$. Applying Cauchy–Schwarz to Equation 6 then separates logit deviation from value deviation. Their root-mean-square magnitudes are at most $\|\bar{q}_i\|\sigma_{\mathrm{k},j}$ and exactly $\sqrt{d}\,\sigma_{\mathrm{v},j}$, respectively, yielding

$$\|A_{ij}\| \le \sqrt{d}\,e^{\kappa_i}\|\bar{q}_i\|\,n_j \exp\!\left(\frac{\bar{q}_i^\top \bar{k}_j}{\sqrt{d}}\right)\sqrt{\sigma^2_{\mathrm{k},j}\sigma^2_{\mathrm{v},j}}. \tag{10}$$

The factor $\sqrt{d}\,e^{\kappa_i}\|\bar{q}_i\|$ is identical for all candidate clusters in row i, so routing does not require computing it. Appendix A.3 provides the derivation.

UMDR ranks the bound on the omitted value-weighting contribution per unit of exact computation. It divides the right-hand side of Equation 10 by $m_i n_j$, the number of token interactions in pair (i, j). This cancels n_j, while m_i is identical across candidates in row i. Taking the logarithm of the remaining factors and flooring the dispersions for numerical stability yields

$$\boxed{\rho^{\mathrm{UMDR}}_{ij} := \frac{\bar{q}_i^\top \bar{k}_j}{\sqrt{d}} + \frac{1}{2}\log\max(\sigma^2_{\mathrm{k},j}, \epsilon) + \frac{1}{2}\log\max(\sigma^2_{\mathrm{v},j}, \epsilon)}. \tag{11}$$

Here, $\epsilon > 0$ is a small numerical floor. The centroid logit comes from the exponential factor in the cost-normalized bound, while the key and value dispersions bound the within-cluster value-weighting term. UMDR selects the highest-scoring cluster pairs for exact computation. The required key and value moments are accumulated during token permutation, and scoring all $C_{\mathrm{q}}C_{\mathrm{k}}$ pairs costs $O(C_{\mathrm{q}}C_{\mathrm{k}}d)$.

4.3 Query-Prioritized Taylor Compensation (QPTC)

For pairs that remain approximated, QPTC estimates the sum of exponentiated logits, Z_{ij}, in Equation 5. This sum contributes to the denominator and multiplies $\bar{v}_j$ in the numerator. Computing it exactly for all remaining pairs would revisit their individual keys, costing up to $O(C_{\mathrm{q}}N_{\mathrm{k}}d)$ across query centroids. The logit deviations around a key centroid have zero mean, so Jensen's inequality gives $Z_{ij} \ge \widehat{Z}_{ij}$: centroid approximation underestimates Z_{ij}. A second-order expansion shows that the leading relative correction is determined by the logit variance,

$$\frac{Z_{ij}}{\widehat{Z}_{ij}} \approx 1 + \frac{1}{2}\sigma^2_{\ell,ij}, \qquad \sigma^2_{\ell,ij} = \frac{1}{d}\bar{q}_i^\top \Sigma^{\mathrm{k}}_j \bar{q}_i. \tag{12}$$

The linear term vanishes after averaging because the key deviations have zero mean. The quadratic term therefore captures the leading effect of within-cluster logit variation on the sum of exponentiated logits.

Table 1: Main quality and efficiency comparison under matched generation settings. Complete VBench results are provided in Table 4. Higher PSNR, SSIM, SC, IQ, AVG, and speedup are better; lower LPIPS, density, and latency are better. Bold indicates the best result among the evaluated sparse methods.

	Similarity			VBench			Efficiency		
Method	PSNR ↑	SSIM ↑	LPIPS ↓	SC ↑	IQ ↑	AVG ↑	Density ↓	Latency ↓	Speedup ↑
Wan2.1 *(1.3B, text-to-video, 81 frames, 720p)*									
Dense	–	–	–	92.60%	66.59%	76.80%	100%	774 s	1.00×
SVG2	23.38	0.8065	0.1529	91.84%	65.99%	**76.46%**	31.65%	401 s	1.93×
SVG-EAR	24.83	0.8363	0.1244	91.99%	65.89%	76.30%	26.01%	379 s	2.04×
ResiSparse	**25.49**	**0.8456**	**0.1132**	**92.36%**	**66.16%**	76.35%	**25.99%**	**355 s**	**2.18×**
Wan2.1 *(14B, text-to-video, 81 frames, 540p)*									
Dense	–	–	–	92.29%	66.37%	77.24%	100%	1579 s	1.00×
SVG2	18.07	0.6314	0.2552	92.12%	65.12%	77.03%	39.49%	1121 s	1.41×
SVG-EAR	20.63	0.7292	0.1869	92.29%	65.99%	76.96%	32.39%	1064 s	1.48×
ResiSparse	**21.49**	**0.7511**	**0.1674**	**92.71%**	**66.22%**	**77.15%**	**31.99%**	**992 s**	**1.59×**
HunyuanVideo *(13B, text-to-video, 129 frames, 540p)*									
Dense	–	–	–	95.15%	65.90%	77.15%	100%	1251 s	1.00×
SVG2	26.34	0.8524	0.0986	94.89%	**65.59%**	77.03%	35.54%	817 s	1.53×
SVG-EAR	27.08	0.8618	0.0923	94.92%	65.51%	**77.09%**	29.92%	770 s	1.62×
ResiSparse	**27.46**	**0.8693**	**0.0840**	**95.09%**	65.55%	77.08%	**28.75%**	**733 s**	**1.71×**

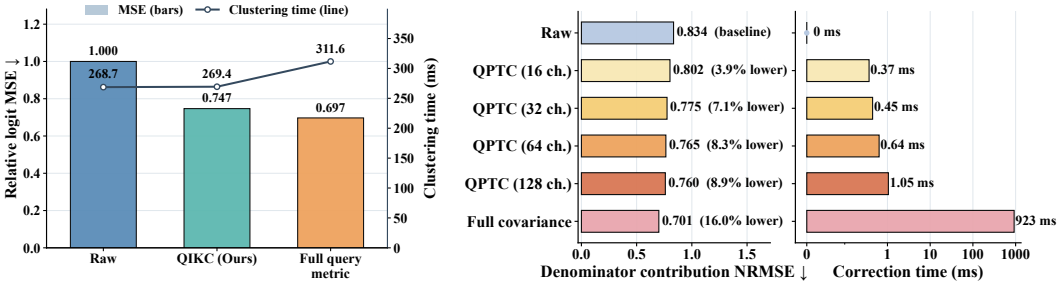
Computing this variance for every pair requires a full $d \times d$ key covariance per cluster: accumulating these matrices costs $O(N_{\mathrm{k}} d^2)$ work and $O(C_{\mathrm{k}} d^2)$ storage, and evaluating all quadratic forms costs $O(C_{\mathrm{q}} C_{\mathrm{k}} d^2)$ work. QPTC first discards cross-channel covariance and uses the key variance of each channel, weighted by the squared query-centroid coordinate. It then restricts this sum to query-prioritized channels. This replaces each dense quadratic form with a short weighted sum and limits the key variances that must be accumulated and stored. Specifically, Ω contains the d_{sel} channels with the highest estimated query energy $\widetilde{\omega}_r$ already computed by QIKC. Using these estimated query channel energies for channel selection prioritizes coordinates that queries weight more strongly on average. The selected key-channel variances then describe how much variation remains within each cluster along these coordinates. Let $\sigma^2_{\mathrm{k},j,r}$ denote the variance of key channel r within cluster j. The selected-channel estimate and the resulting corrected contributions are

$$\begin{aligned} \widehat{\sigma}^2_{\ell,ij} &= \frac{1}{d} \sum_{r \in \Omega} \bar{q}^2_{ir} \sigma^2_{\mathrm{k},j,r}, \\ \widehat{Z}^{\mathrm{corr}}_{ij} &= \widehat{Z}_{ij} \left(1 + \frac{\widehat{\sigma}^2_{\ell,ij}}{2} \right), \qquad \widehat{Y}^{\mathrm{corr}}_{ij} = \widehat{Z}^{\mathrm{corr}}_{ij} \bar{v}_j. \end{aligned} \tag{13}$$

One channel set is reused across all query clusters, while the variance estimate remains specific to pair (i, j) through $\bar{q}_i$ and the key-channel variances of cluster j. The corrected estimate $\widehat{Z}^{\mathrm{corr}}_{ij}$ contributes to the softmax denominator and scales $\bar{v}_j$ in the numerator. Each selected-channel estimate requires $O(d_{\mathrm{sel}})$ work rather than a full $d \times d$ quadratic form. Appendix A.4 provides the derivation, error decomposition, and complexity analysis. We use $d_{\mathrm{sel}} = 32$ unless otherwise specified.

4.4 Efficient Implementation

ResiSparse integrates the three components into the sparse-attention computation. QIKC scales the keys before K-means, while the K/V permutation kernel accumulates the centroids and dispersions used by UMDR together with the selected-channel key variances used by QPTC. The scalar moments reuse key and value data loaded for permutation, while selected-channel variances are accumulated in the same pass, avoiding separate token traversals to collect these quantities. FlashInfer (Ye et al., 2025) computes the selected pairs exactly, and a fused Triton kernel applies centroid approximation, QPTC correction, and the log-sum-exp merge to the remaining pairs without materializing dense



(a) Relative logit MSE and clustering time.

(b) Denominator-contribution NRMSE and correction time.

Figure 3: Clustering and compensation analysis. (a) Bars compare ordinary key K-means (Raw), QIKC, and clustering with tokenwise query moments (Oracle). Logit MSE is reported relative to Raw; the line shows clustering time with 50 K-means iterations. (b) Bars compare denominator-contribution NRMSE for uncorrected centroids (Raw), QPTC at different channel counts, and Full covariance, which includes all channels and their cross-channel covariance in the same second-order correction; the adjacent plot shows correction time. Lower error and time are better.

attention. During compensation, the kernel accumulates the corrected sums of exponentiated logits into the softmax denominator and their products with value centroids into the numerator. Keeping these intermediate contributions inside the kernel avoids writing a separate approximate output to global memory before the merge. Appendix B provides the GPU dataflow and pseudocode.

5 Experiments

5.1 Setup

We evaluate ResiSparse on three text-to-video settings: Wan2.1-1.3B at 720p with 81 frames, Wan2.1-14B at 540p with 81 frames, and HunyuanVideo-13B at 540p with 129 frames. We compare ResiSparse with dense attention, SVG2 (Yang et al., 2025a), and SVG-EAR (Zhou et al., 2026).

To measure how faithfully sparse attention preserves the dense model's output, we report framewise PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) against the paired dense video. To assess the quality of the generated videos, we use VBench (Huang et al., 2024); the main table reports subject consistency (SC), imaging quality (IQ), and the official overall score (AVG). We measure efficiency using end-to-end denoising latency, speedup over dense attention, and realized attention density. Appendix C provides the remaining VBench dimensions and experimental settings.

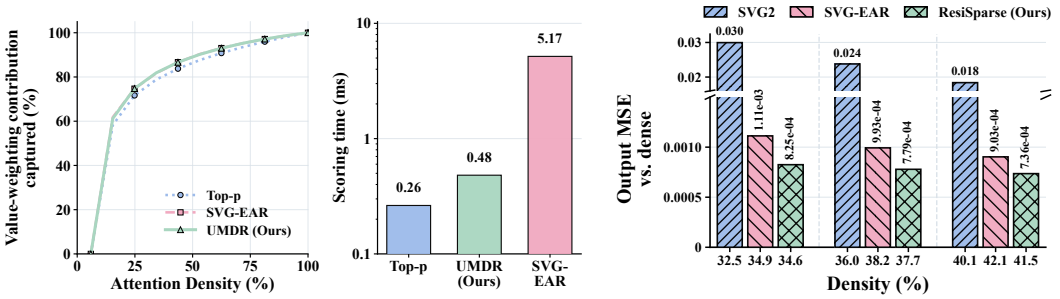
5.2 Quality and Efficiency Evaluation

Table 1 shows that ResiSparse consistently produces the closest match to dense attention among the evaluated sparse methods. Compared with SVG-EAR, PSNR increases from 24.83 to 25.49 on Wan2.1-1.3B, from 20.63 to 21.49 on Wan2.1-14B, and from 27.08 to 27.46 on HunyuanVideo-13B. SSIM and LPIPS follow the same trend across all three models.

ResiSparse combines this fidelity advantage with near-dense VBench quality and substantial acceleration. Its VBench average remains within 0.45 points of dense attention across all three settings. On Wan2.1-14B, ResiSparse achieves the highest average among the sparse methods. It also reduces denoising latency from 774 to $355\,\mathrm{s}$ on Wan2.1-1.3B, from 1579 to $992\,\mathrm{s}$ on Wan2.1-14B, and from 1251 to $733\,\mathrm{s}$ on HunyuanVideo-13B, corresponding to $1.59\times$–$2.18\times$ speedups. Together, the similarity and latency results place ResiSparse at the strongest quality–efficiency operating point among the evaluated sparse methods across all three models.

5.3 Mechanism analysis

We average results over inputs sampled across layers and denoising timesteps of Wan2.1-1.3B (720p).



(a) Value-weighting capture and scoring time.

(b) Layer-output MSE.

Figure 4: Routing and layer-output error analysis. (a) Curves show the fraction of candidate blocks' total $\|A_{ij}\|$ covered by selected exact blocks at their realized attention density. UMDR closely matches SVG-EAR's coverage with lower scoring time. (b) Bars show layer-output MSE against dense attention at top-p values 0.80, 0.85, and 0.90 from left to right.

Query-Induced Key Clustering (QIKC). Figure 3(a) evaluates whether QIKC forms key clusters whose centroids preserve tokenwise logits. With 50 K-means iterations, QIKC reduces relative logit MSE from 1.000 to 0.747, compared with the oracle's 0.697. Clustering takes $269.4\,\mathrm{ms}$, versus $268.7\,\mathrm{ms}$ for Raw and $311.6\,\mathrm{ms}$ for Oracle. Thus, centroid-based query channel energies retain most of the full metric's benefit with little overhead over ordinary K-means.

Unified Mass–Dispersion Routing (UMDR). On fixed query clusters and QIKC key clusters, Figure 4(a) compares routing methods by the fraction of candidate blocks' total $\|A_{ij}\|$ covered by exact computation. UMDR closely follows SVG-EAR across the measured attention densities. Its improvement over Top-p supports using key and value dispersions to identify blocks whose token-specific value weighting is poorly captured by the centroid. These cluster-level quantities retain much of the routing benefit of token-level probing while reducing scoring time from $5.17\,\mathrm{ms}$ for SVG-EAR to $0.48\,\mathrm{ms}$ for UMDR. Appendix C gives the metric and measurement details.

Query-Prioritized Taylor Compensation (QPTC). Figure 3(b) evaluates compensation accuracy and cost on approximated pairs, with QIKC key clusters and UMDR-selected exact blocks fixed. Using 32 query-prioritized channels reduces denominator-contribution NRMSE from 0.834 to 0.775 in $0.45\,\mathrm{ms}$. Increasing the channel count to 64 or 128 yields smaller additional gains at higher cost, making 32 channels an effective accuracy–cost trade-off. Full covariance reaches 0.701 but requires $923\,\mathrm{ms}$, highlighting the efficiency of selected-channel correction.

Layer-output MSE. Figure 4(b) compares sparse and dense self-attention outputs after the output projection. Across matched top-p settings, ResiSparse achieves the lowest MSE despite using a slightly lower realized density than SVG-EAR, indicating that more accurate approximation rather than more exact token interactions drives the gain. Appendix C provides the evaluation details.

6 Conclusion

We presented ResiSparse, a training-free exact-or-approximate attention method for accelerating video diffusion transformers. Exact cluster-level expressions relate centroid-induced logit deviations to sums of exponentiated logits and within-cluster value contributions. Guided by this analysis, QIKC targets logit distortion through query-induced clustering, UMDR prioritizes exact computation using a cost-normalized bound on the omitted value-weighting contribution, and QPTC estimates a second-order correction to these sums for the remaining approximated pairs. Across Wan2.1-1.3B, Wan2.1-14B, and HunyuanVideo-13B, ResiSparse achieves the highest dense-reference fidelity among the evaluated sparse methods, maintains VBench quality close to dense attention, and provides $1.59\times$–$2.18\times$ denoising speedups. Future work could develop more accurate compensation for approximated blocks to preserve output fidelity at higher sparsity.

AI USE STATEMENT

In this work, we used generative AI tools to assist with literature and information retrieval, improve the readability and clarity of the manuscript, brainstorm research ideas, provide translation assistance, support code writing and editing, and assist with mathematical derivations and proofs. All AI-assisted text, code, and mathematical content were reviewed and verified by the authors. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, propose or refine research hypotheses, design or evaluate the research methodology or experiments, implement the core methods, clean or reformat datasets, conduct qualitative or thematic data analysis, or interpret experimental results. The core research ideas, theoretical contributions, and scientific conclusions were developed and led by the authors, who take full responsibility for the final content of this work.

ETHICS STATEMENT

This work focuses on accelerating video generation using publicly available pretrained models and benchmarks. Mitigating biases and harmful-generation risks in the underlying models is outside the scope of this study. As faster generation may also lower the cost of misuse, applications should retain appropriate content-safety and privacy safeguards.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 present the cluster-level approximation analysis and the ResiSparse method, while Section 5 describes the evaluation setup and results. Appendix A provides detailed derivations and proofs, Appendix B presents the GPU implementation and pseudocode, and Appendix C specifies the experimental settings and evaluation metrics. We provide our implementation code and qualitative video results in the supplementary material, with additional frame-level comparisons in Appendix D.

REFERENCES

Pengtao Chen, Xianfang Zeng, Maosen Zhao, Mingzhu Shen, Wei Cheng, Gang Yu, and Tao Chen. Sparse-vdit: Unleashing the power of sparse attention to accelerate video diffusion transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 2957–2965, 2026.

Ruichen Chen, Keith Mills, Liyao Jiang, Chao Gao, and Di Niu. Re-ttention: Ultra sparse visual generation via attention statistical reshape. *Advances in Neural Information Processing Systems*, 38:58029–58055, 2025.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Jie Hu, Zixiang Gao, Yutong He, and Kun Yuan. Dfsattn: Dynamic fine-grained sparse attention for efficient video generation. *arXiv preprint arXiv:2605.23445*, 2026.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024.

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.

Dongyeun Lee, Amir Zandieh, Vahab Mirrokni, Junmo Kim, and Insu Han. Hypervattention: Efficient sparse attention with spatio-temporal clustering for video diffusion. *arXiv preprint arXiv:2607.03012*, 2026.

Haopeng Li, Yitong Li, Junsong Chen, Tian Ye, Haozhe Liu, Jincheng Yu, Duomin Wang, Ruihua Zhang, Zeke Xie, Enze Xie, et al. Sol-attn: Accelerating video generation inference via on-the-fly attention sparsification. *arXiv preprint arXiv:2607.24027*, 2026a.

Haopeng Li, Shitong Shao, Wenliang Zhong, Zikai Zhou, Lichen Bai, Hui Xiong, and Zeke Xie. Pisa: Piecewise sparse attention is wiser for efficient diffusion transformers. *arXiv preprint arXiv:2602.01077*, 2026b.

Xuewen Liu, Zhikai Li, Jing Zhang, Mengjuan Chen, and Qingyi Gu. Rectified spaattn: Revisiting attention sparsity for efficient video generation. *arXiv preprint arXiv:2511.19835*, 2025.

Jiayi Luo, Jiayu Chen, Jiankun Wang, Cong Wang, Hanxin Zhu, Qingyun Sun, Chen Gao, Zhibo Chen, and Jianxin Li. Attention sparsity is input-stable: Training-free sparse attention for video generation via offline sparsity profiling and online qk co-clustering. *arXiv preprint arXiv:2603.18636*, 2026.

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.

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

Pardis Taghavi, Reza Langari, and Gaurav Pandey. Partition the support, reconstruct the residual: Training-free sparse attention for video generation and world models. *arXiv preprint arXiv:2608.18484*, 2026.

Haoyue Tan, Shengnan Wang, Yulin Qiao, Juncheng Zhang, Youhui Bai, Ping Gong, Zewen Jin, and Cheng Li. Adacluster: Adaptive query-key clustering for sparse attention in video generation. *arXiv preprint arXiv:2604.18348*, 2026.

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*, 30, 2017.

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.

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

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.

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. *Advances in Neural Information Processing Systems*, 38:96965–96991, 2025a.

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. In *International Conference on Learning Representations*, volume 2025, pp. 83048–83077, 2025b.

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. *Proceedings of Machine Learning and Systems*, 7, 2025.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattention: Accurate and training-free sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025a.

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*, 2025b.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Xuanyi Zhou, Qiuyang Mang, Shuo Yang, Haocheng Xi, Jintao Zhang, Huanzhi Mao, Joseph E Gonzalez, Kurt Keutzer, Ion Stoica, and Alvin Cheung. Svg-ear: Parameter-free linear compensation for sparse video generation via error-aware routing. *arXiv preprint arXiv:2603.08982*, 2026.

A Theoretical Analysis

This appendix first derives exact expressions for each cluster's sum of exponentiated logits and weighted-value sum. It then derives the QIKC clustering objective, the UMDR bound on the within-cluster value-weighting term, and the second-order correction used by QPTC. The final subsection gives a separate bound relating attention-output errors at a query centroid and its individual query tokens.

Fix one attention head with head dimension d. Let s and t index query and key–value tokens, with corresponding vectors $q_s \in \mathbb{R}^d$ and $k_t, v_t \in \mathbb{R}^d$. Let $i \in [C_{\mathrm{q}}]$ and $j \in [C_{\mathrm{k}}]$ index query and key clusters, where $[C] = \{1, \dots, C\}$, with m_i and n_j tokens, and let $N_{\mathrm{q}} = \sum_i m_i$ and $N_{\mathrm{k}} = \sum_j n_j$. We write $\bar{q}_i$, $\bar{k}_j$, and $\bar{v}_j$ for the corresponding centroids; $\mathbb{E}_s[\cdot]$ for an average over query tokens; and $\mathbb{E}_j[\cdot]$ for the average over key tokens in cluster j. We write $s \in i$ and $t \in j$ for membership in query and key clusters, respectively, and r denotes a channel. All vector norms below are Euclidean norms.

Unless an argument q_s is shown explicitly, all attention quantities below are evaluated at $\bar{q}_i$. Let Z_{ij} denote key cluster j's exact contribution to the softmax denominator and let Y_{ij} denote its exact weighted-value numerator. Their sums are $Z_i = \sum_j Z_{ij}$ and $Y_i = \sum_j Y_{ij}$, and the exact output is $O_i = Y_i / Z_i$. Exact quantities have no accent, centroid approximations carry a hat, and the superscript corr marks quantities associated with individual query–key cluster pairs after QPTC correction. The sets $\mathcal{S}_i$ and $\mathcal{D}_i$ contain the key clusters evaluated exactly and approximately for query cluster i.

A.1 Cluster-level decomposition of the attention numerator and denominator

We derive the identities used to analyze centroid approximation before attention normalization.

Proof of Proposition 1. For a token $t \in j$, use the deviations in Equation 4. Since $\bar{k}_j$ and $\bar{v}_j$ are cluster means,

$$\mathbb{E}_j[\Delta k_t] = 0, \qquad \mathbb{E}_j[\Delta v_t] = 0, \qquad \mathbb{E}_j[\delta \ell_{ij}(t)] = 0.$$

Substituting $k_t = \bar{k}_j + \Delta k_t$ into the exponentiated logit gives

$$\exp\left(\frac{\bar{q}_i^\top k_t}{\sqrt{d}}\right) = \exp\left(\frac{\bar{q}_i^\top \bar{k}_j}{\sqrt{d}}\right) e^{\delta \ell_{ij}(t)}. \tag{14}$$

Summing the exponentiated logits over the cluster and using $\widehat{Z}_{ij} = n_j \exp(\bar{q}_i^\top \bar{k}_j / \sqrt{d})$ yields

$$\begin{aligned} Z_{ij} &= \exp\left(\frac{\bar{q}_i^\top \bar{k}_j}{\sqrt{d}}\right) \sum_{t \in j} e^{\delta \ell_{ij}(t)} \\ &= \widehat{Z}_{ij} \mathbb{E}_j[e^{\delta \ell_{ij}(t)}]. \end{aligned} \tag{15}$$

For the weighted-value sum, substitute $v_t = \bar{v}_j + \Delta v_t$ and separate the constant centroid from its deviations:

$$\begin{aligned} Y_{ij} &= \widehat{Z}_{ij} \mathbb{E}_j[e^{\delta \ell_{ij}(t)} (\bar{v}_j + \Delta v_t)] \\ &= \widehat{Z}_{ij} \mathbb{E}_j[e^{\delta \ell_{ij}(t)}] \bar{v}_j + \widehat{Z}_{ij} \mathbb{E}_j[e^{\delta \ell_{ij}(t)} \Delta v_t] \\ &= Z_{ij} \bar{v}_j + \widehat{Z}_{ij} \mathbb{E}_j[e^{\delta \ell_{ij}(t)} \Delta v_t]. \end{aligned} \tag{16}$$

The final expectation can be centered at $e^0 = 1$ because the value deviations have zero mean:

$$\begin{aligned} \mathbb{E}_j[e^{\delta \ell_{ij}(t)} \Delta v_t] &= \mathbb{E}_j[(e^{\delta \ell_{ij}(t)} - 1) \Delta v_t] + \mathbb{E}_j[\Delta v_t] \\ &= \mathbb{E}_j[(e^{\delta \ell_{ij}(t)} - 1) \Delta v_t]. \end{aligned}$$

Multiplying this expectation by $\widehat{Z}_{ij}$ gives A_{ij} in Equation 6, proving $Y_{ij} = Z_{ij} \bar{v}_j + A_{ij}$. □

The two identities distinguish the sum of exponentiated logits, Z_{ij}, from the weighted contribution of within-cluster value deviations, A_{ij}. Replacing Z_{ij} by an estimate changes both the denominator and the numerator term multiplying $\bar{v}_j$. QPTC estimates a correction to $\widehat{Z}_{ij}$; the additional term A_{ij} motivates UMDR's choice of pairs for exact computation.

A.2 QIKC: QUERY-INDUCED CLUSTERING METRIC

For a key token $t \in j$, let $\Delta k_t = k_t - \bar{k}_j$. Replacing k_t by $\bar{k}_j$ changes the logit for query token s by $q_s^\top \Delta k_t/\sqrt{d}$, whose average squared error over the query tokens is

$$\mathbb{E}_s\left[\left(\frac{q_s^\top \Delta k_t}{\sqrt{d}}\right)^2\right] = \frac{1}{d}\mathbb{E}_s[\Delta k_t^\top q_s q_s^\top \Delta k_t] \tag{17}$$

$$= \frac{1}{d}\Delta k_t^\top G_{\mathrm{q}} \Delta k_t, \qquad G_{\mathrm{q}} = \mathbb{E}_s[q_s q_s^\top]. \tag{18}$$

This quadratic form defines the ideal key-clustering metric $G_{\mathrm{q}} = \mathbb{E}_s[q_s q_s^\top]$ for minimizing average squared logit error. Implementing it through ordinary K-means would require forming the $d \times d$ matrix in $O(N_{\mathrm{q}} d^2)$ work, factorizing it in $O(d^3)$ work, and applying the resulting dense transformation to all keys in $O(N_{\mathrm{k}} d^2)$ work. QIKC makes two successive approximations. First, it replaces this full metric with a diagonal one. Let $r, p \in [d]$ index channels, with q_{sr} denoting channel r of query token s and $k_{tr}, \bar{k}_{jr}$ the corresponding key coordinates. Keeping only diagonal entries gives

$$\mathbb{E}_s\left[\left(\frac{q_s^\top \Delta k_t}{\sqrt{d}}\right)^2\right] \approx \frac{1}{d}\sum_{r=1}^{d} \omega_r^\star (k_{tr} - \bar{k}_{jr})^2, \qquad \omega_r^\star = \mathbb{E}_s[q_{sr}^2]. \tag{19}$$

Here $\omega_r^\star$ is the exact query channel energy, the mean squared value of channel r over query tokens. This is the first approximation. Second, by the law of total second moments, $\omega_r^\star$ consists of the weighted squared query-centroid coordinates and the within-cluster query variance. Computing the exact channel energies requires an $O(N_{\mathrm{q}} d)$ pass over all query tokens. QIKC retains the centroid contribution and omits the within-cluster variance, reducing this step to $O(C_{\mathrm{q}} d)$ using the already-available query centroids and cluster sizes. Define

$$\widetilde{\omega}_r = \frac{1}{N_{\mathrm{q}}}\sum_i m_i \bar{q}_{ir}^2, \qquad \widetilde{G}_{\mathrm{q}} := \mathrm{diag}(\widetilde{\omega}_1, \ldots, \widetilde{\omega}_d). \tag{20}$$

Thus, $\widetilde{\omega}_r$ estimates the query channel energy $\omega_r^\star$ using its centroid contribution, and $\widetilde{G}_{\mathrm{q}}$ is the resulting diagonal metric. We implement it with ordinary Euclidean K-means by defining

$$\alpha_{\mathrm{q}} := \frac{1}{d}\sum_{p=1}^{d} \sqrt{\widetilde{\omega}_p}, \qquad \gamma_r := \frac{\sqrt{\widetilde{\omega}_r}}{\alpha_{\mathrm{q}}}. \tag{21}$$

For any key displacement Δk, these definitions give

$$\frac{1}{d}\Delta k^\top \widetilde{G}_{\mathrm{q}} \Delta k = \frac{\alpha_{\mathrm{q}}^2}{d}\left\|\mathrm{diag}(\gamma_1, \ldots, \gamma_d)\Delta k\right\|_2^2. \tag{22}$$

Because α_{q}^2/d is shared by every key displacement, scaled Euclidean distance yields the same Lloyd assignments as the $\widetilde{G}_{\mathrm{q}}$ objective. In summary, QIKC discards cross-channel interactions, estimates query channel energies from centroids, and realizes the resulting diagonal metric through scaled Euclidean K-means. The normalization α_{q} keeps scaled-key magnitudes comparable without changing assignments. It is positive whenever at least one estimated query channel energy is nonzero; if all estimated energies vanish, we use identity scaling, $\gamma_r = 1$ for every channel.

Empirically, this approximation retains most of the oracle's reduction in logit MSE with little overhead over ordinary K-means (Figure 3(a)). The comparison uses 50 K-means iterations and includes the time to compute channel scales and scale keys.

A.3 UMDR: COST-NORMALIZED ROUTING BOUND

For an approximated cluster pair, the value-weighting term omitted by centroid compensation is

$$A_{ij} = \widehat{Z}_{ij}\mathbb{E}_j[(e^{\delta\ell_{ij}(t)} - 1)\Delta v_t]. \tag{23}$$

For each candidate pair, define its largest within-cluster logit deviation as $\kappa_{ij} := \max_{t\in j} |\delta\ell_{ij}(t)|$, and for a fixed query cluster i, let $\kappa_i := \max_{j\in[C_\mathrm{k}]} \kappa_{ij}$. Apply the mean-value theorem to $f(x) = e^x$ on the interval between 0 and $\delta\ell_{ij}(t)$. For some intermediate point ξ_{ijt}, the exponential difference equals $e^{\xi_{ijt}}\delta\ell_{ij}(t)$. Since $|\xi_{ijt}| \le |\delta\ell_{ij}(t)| \le \kappa_i$, its magnitude satisfies

$$|e^{\delta\ell_{ij}(t)} - 1| = e^{\xi_{ijt}}|\delta\ell_{ij}(t)| \le e^{\kappa_i}|\delta\ell_{ij}(t)|, \qquad |\xi_{ijt}| \le \kappa_i. \tag{24}$$

The triangle inequality first moves the norm inside the cluster average in Equation 23. After inserting Equation 24, Cauchy–Schwarz is applied to the two tokenwise factors $|\delta\ell_{ij}(t)|$ and $\|\Delta v_t\|$. This yields

$$\begin{aligned}\|A_{ij}\| &\le \widehat{Z}_{ij}\mathbb{E}_j\Big[|e^{\delta\ell_{ij}(t)} - 1|\,\|\Delta v_t\|\Big] \\ &\le \widehat{Z}_{ij}e^{\kappa_i}\sqrt{\mathbb{E}_j[\delta\ell_{ij}(t)^2]}\sqrt{\mathbb{E}_j[\|\Delta v_t\|^2]}.\end{aligned} \tag{25}$$

The first expectation is the query-dependent logit variance, while the second equals d times the within-cluster value dispersion defined below. In particular, define

$$\Sigma_j^\mathrm{k} := \mathbb{E}_j[(k_t - \bar{k}_j)(k_t - \bar{k}_j)^\top], \qquad \sigma_{\ell,ij}^2 := \mathbb{E}_j[\delta\ell_{ij}(t)^2] = \frac{\bar{q}_i^\top \Sigma_j^\mathrm{k} \bar{q}_i}{d}. \tag{26}$$

The value-side quantity is defined as

$$\sigma_{\mathrm{v},j}^2 := \frac{1}{n_j d}\sum_{t\in j}\|v_t - \bar{v}_j\|^2 = \frac{1}{d}\mathbb{E}_j[\|\Delta v_t\|^2]. \tag{27}$$

Substituting these identities into Equation 25 gives the following bound for cluster pair (i, j):

$$\|A_{ij}\| \le \widehat{Z}_{ij}e^{\kappa_i}\sqrt{\sigma_{\ell,ij}^2}\sqrt{d\sigma_{\mathrm{v},j}^2}. \tag{28}$$

This expression shows why both key-induced logit variation and value dispersion enter the routing criterion: if either vanishes, $A_{ij} = 0$ for that pair.

Evaluating $\sigma_{\ell,ij}^2$ for all cluster pairs directly over the key tokens costs $O(C_\mathrm{q}N_\mathrm{k}d)$. Computing it from reusable cluster statistics instead requires $O(N_\mathrm{k}d^2)$ work and $O(C_\mathrm{k}d^2)$ storage for the full covariances, followed by $O(C_\mathrm{q}C_\mathrm{k}d^2)$ work for the quadratic forms. UMDR avoids both costs by using the key dispersion, equivalently the average variance across feature channels,

$$\sigma_{\mathrm{k},j}^2 := \frac{1}{d}\operatorname{tr}(\Sigma_j^\mathrm{k}) = \frac{1}{n_j d}\sum_{t\in j}\|k_t - \bar{k}_j\|^2. \tag{29}$$

Because Σ_j^k is positive semidefinite, its Rayleigh quotient satisfies $\bar{q}_i^\top\Sigma_j^\mathrm{k}\bar{q}_i \le \|\bar{q}_i\|^2\lambda_{\max}(\Sigma_j^\mathrm{k})$, and its largest eigenvalue is at most the sum of its nonnegative eigenvalues, $\operatorname{tr}(\Sigma_j^\mathrm{k})$. Therefore,

$$\begin{aligned}\sigma_{\ell,ij}^2 &= \frac{\bar{q}_i^\top\Sigma_j^\mathrm{k}\bar{q}_i}{d} \\ &\le \frac{\|\bar{q}_i\|^2\lambda_{\max}(\Sigma_j^\mathrm{k})}{d} \le \frac{\|\bar{q}_i\|^2\operatorname{tr}(\Sigma_j^\mathrm{k})}{d} = \|\bar{q}_i\|^2\sigma_{\mathrm{k},j}^2.\end{aligned} \tag{30}$$

Substituting Equation 30 and $\widehat{Z}_{ij} = n_j\exp\Big(\frac{\bar{q}_i^\top\bar{k}_j}{\sqrt{d}}\Big)$ gives

$$\|A_{ij}\| \le \sqrt{d}\,e^{\kappa_i}\|\bar{q}_i\|\,n_j\exp\left(\frac{\bar{q}_i^\top\bar{k}_j}{\sqrt{d}}\right)\sqrt{\sigma_{\mathrm{k},j}^2\sigma_{\mathrm{v},j}^2}. \tag{31}$$

For a given query cluster i, the terms that distinguish candidate key clusters are the centroid estimate $n_j\exp\Big(\frac{\bar{q}_i^\top\bar{k}_j}{\sqrt{d}}\Big)$ of their denominator contribution, key dispersion, and value dispersion.

UMDR allocates a limited exact-computation budget separately for each query cluster. Exact computation of pair (i, j) forms the logits and weighted-value contributions for all $m_i n_j$ combinations of its query and key tokens. With a fixed head dimension d, each token interaction requires the same amount of arithmetic, so the block area $m_i n_j$ gives the relative computation required by the pair. Dividing Equation 31 by this cost gives a bound on the omitted value-weighting contribution per unit of exact computation:

$$\frac{\|A_{ij}\|}{m_i n_j} \le \frac{\sqrt{d}\, e^{\kappa_i} \|\bar{q}_i\|}{m_i} \exp\left(\frac{\bar{q}_i^\top \bar{k}_j}{\sqrt{d}}\right) \sqrt{\sigma_{\mathrm{k},j}^2 \sigma_{\mathrm{v},j}^2}. \tag{32}$$

Candidate key clusters are compared only within the same query-cluster row i. The factor preceding $\exp\left(\frac{\bar{q}_i^\top \bar{k}_j}{\sqrt{d}}\right)$ therefore has the same value for every candidate j in that row and does not change their ordering. UMDR takes the logarithm of the terms that vary across candidates:

$$\boxed{\rho_{ij}^{\mathrm{UMDR}} := \frac{\bar{q}_i^\top \bar{k}_j}{\sqrt{d}} + \frac{1}{2}\log\max(\sigma_{\mathrm{k},j}^2, \epsilon) + \frac{1}{2}\log\max(\sigma_{\mathrm{v},j}^2, \epsilon)}. \tag{33}$$

For positive dispersions above the numerical floor, taking the logarithm preserves the ordering of the bound. The floor $\epsilon > 0$ regularizes smaller dispersions and keeps the score finite at zero. Cluster size still determines the routing budget: the baseline centroid estimates $n_j \exp\left(\frac{\bar{q}_i^\top \bar{k}_j}{\sqrt{d}}\right)$ define a top-p key-token area for each query-cluster row, and selecting pair (i, j) consumes n_j key-token positions from that row's budget. UMDR fills this budget in descending order of Equation 33. The fused permutation pass provides the per-cluster mean squared key and value norms in Equation 59, from which the dispersions are recovered by subtracting the corresponding squared centroid norms. The bound concerns A_{ij}, the within-cluster value-weighting term in Equation 5. QPTC estimates Z_{ij} for the remaining approximated pairs.

A.4 QPTC: SECOND-ORDER DENOMINATOR CORRECTION

For an approximated pair (i, j), the exact and centroid-based denominator contributions satisfy

$$Z_{ij} = \widehat{Z}_{ij} \mathbb{E}_j[e^{\delta\ell_{ij}(t)}], \qquad \mathbb{E}_j[\delta\ell_{ij}(t)] = 0. \tag{34}$$

We first determine the direction of the resulting error. Applying Jensen's inequality to the convex exponential gives

$$\begin{aligned} \mathbb{E}_j[e^{\delta\ell_{ij}(t)}] &\ge e^{\mathbb{E}_j[\delta\ell_{ij}(t)]} = 1, \\ Z_{ij} &\ge \widehat{Z}_{ij}. \end{aligned} \tag{35}$$

Centroid substitution therefore underestimates Z_{ij}. In Equation 5, Z_{ij} enters both the denominator and the value-centroid term in the numerator.

To estimate the size of this underestimation, apply Taylor's theorem around zero to each key token:

$$e^{\delta\ell_{ij}(t)} = 1 + \delta\ell_{ij}(t) + \frac{\delta\ell_{ij}(t)^2}{2} + \frac{e^{\eta_{ijt}} \delta\ell_{ij}(t)^3}{6}, \tag{36}$$

where η_{ijt} lies between 0 and $\delta\ell_{ij}(t)$. Averaging over cluster j and using $\mathbb{E}_j[\delta\ell_{ij}(t)] = 0$ gives

$$\mathbb{E}_j[e^{\delta\ell_{ij}(t)}] = 1 + \frac{\sigma_{\ell,ij}^2}{2} + \mathcal{T}_{ij}^{(3)}, \qquad \mathcal{T}_{ij}^{(3)} := \frac{1}{6}\mathbb{E}_j[e^{\eta_{ijt}} \delta\ell_{ij}(t)^3]. \tag{37}$$

Thus, $\mathcal{T}_{ij}^{(3)}$ exactly collects the terms beyond the second-order approximation. Substituting this expression into Equation 34 yields

$$Z_{ij} - \widehat{Z}_{ij} = \widehat{Z}_{ij}\left(\frac{\sigma_{\ell,ij}^2}{2} + \mathcal{T}_{ij}^{(3)}\right). \tag{38}$$

The leading second-order correction is therefore $\widehat{Z}_{ij}\sigma_{\ell,ij}^2/2$.

Computing the exact logit variance $\sigma_{\ell,ij}^2 = \bar{q}_i^\top \Sigma_j^{\mathrm{k}} \bar{q}_i / d$ requires a full $d \times d$ key covariance for every key cluster and a quadratic form for every query–key cluster pair. Accumulating these matrices

costs $O(N_{\mathrm{k}}d^2)$ work and $O(C_{\mathrm{k}}d^2)$ storage, while evaluating all quadratic forms costs $O(C_{\mathrm{q}}C_{\mathrm{k}}d^2)$ work. This overhead would undermine the efficiency of online sparse attention. QPTC therefore approximates the logit variance in two steps: it first discards cross-channel covariance and then retains only the query-prioritized channels.

Let $\sigma^2_{\mathrm{k},j,r} := [\Sigma^{\mathrm{k}}_j]_{rr}$ denote the variance of key channel r within cluster j. Keeping all diagonal entries gives the estimate $d^{-1}\sum_{r=1}^{d}\bar{q}^2_{ir}\sigma^2_{\mathrm{k},j,r}$. To further reduce the per-pair computation, QPTC restricts this sum to the d_{sel} channels with the highest estimated query energy,

$$\Omega := \mathrm{Top}_{d_{\mathrm{sel}}}\{\widetilde{\omega}_1, \ldots, \widetilde{\omega}_d\}. \tag{39}$$

The selected-channel estimate and corrected contributions are

$$\widehat{\sigma}^2_{\ell,ij} = \frac{1}{d}\sum_{r\in\Omega}\bar{q}^2_{ir}\sigma^2_{\mathrm{k},j,r}, \qquad \widehat{Z}^{\mathrm{corr}}_{ij} = \widehat{Z}_{ij}\left(1 + \frac{\widehat{\sigma}^2_{\ell,ij}}{2}\right), \qquad \widehat{Y}^{\mathrm{corr}}_{ij} = \widehat{Z}^{\mathrm{corr}}_{ij}\bar{v}_j. \tag{40}$$

Accumulating the selected-channel variances requires $O(N_{\mathrm{k}}d_{\mathrm{sel}})$ work and $O(C_{\mathrm{k}}d_{\mathrm{sel}})$ storage, while evaluating the estimate for all cluster pairs costs $O(C_{\mathrm{q}}C_{\mathrm{k}}d_{\mathrm{sel}})$ work.

The selected-channel diagonal estimate differs from the exact logit variance in two ways:

$$\sigma^2_{\ell,ij} - \widehat{\sigma}^2_{\ell,ij} = \frac{1}{d}\left[\sum_{p\notin\Omega}\bar{q}^2_{ip}\sigma^2_{\mathrm{k},j,p} + 2\sum_{p<r}\bar{q}_{ip}\bar{q}_{ir}[\Sigma^{\mathrm{k}}_j]_{pr}\right]. \tag{41}$$

The first term is the variance carried by channels outside Ω, and the second is the contribution of cross-channel covariance. Combining this decomposition with Equation 37 gives

$$Z_{ij} - \widehat{Z}^{\mathrm{corr}}_{ij} = \widehat{Z}_{ij}\left[\mathcal{T}^{(3)}_{ij} + \tfrac{1}{2}\left(\sigma^2_{\ell,ij} - \widehat{\sigma}^2_{\ell,ij}\right)\right]. \tag{42}$$

The remaining denominator error therefore consists of the third-order Taylor remainder and the difference between the exact logit variance and its selected-channel diagonal estimate. QPTC retains the leading second-order effect while avoiding full covariance accumulation and quadratic forms for every query–key cluster pair. Since $Z_{ij} \geq \widehat{Z}_{ij} > 0$ and $\widehat{\sigma}^2_{\ell,ij} \geq 0$, dividing the raw and corrected errors by $\widehat{Z}_{ij}$ gives

$$\begin{aligned}
\left|Z_{ij} - \widehat{Z}^{\mathrm{corr}}_{ij}\right| < \left|Z_{ij} - \widehat{Z}_{ij}\right| \\
&\iff \left|\frac{Z_{ij}}{\widehat{Z}_{ij}} - 1 - \frac{\widehat{\sigma}^2_{\ell,ij}}{2}\right| < \frac{Z_{ij}}{\widehat{Z}_{ij}} - 1 \\
&\iff 0 < \frac{\widehat{\sigma}^2_{\ell,ij}}{2} < 2\left(\frac{Z_{ij}}{\widehat{Z}_{ij}} - 1\right).
\end{aligned} \tag{43}$$

The same logit variance $\sigma^2_{\ell,ij}$ enters UMDR and QPTC in different ways. UMDR bounds it by $\|\bar{q}_i\|^2\sigma^2_{\mathrm{k},j}$ for low-cost pair ranking, while QPTC estimates its query-dependent magnitude from selected channels to determine the denominator correction.

Section 5.3 evaluates the aggregate effect of this correction using denominator-contribution NRMSE.

A.5 Extension to individual query tokens

The preceding analysis uses the query centroid $\bar{q}_i$. We complement it with a sensitivity bound on the attention-output error when the query moves from $\bar{q}_i$ to a token in the same cluster. The comparison conditions on the approximation constructed for query cluster i: the key–value partition, its centroids, the routing sets $\mathcal{S}_i$ and $\mathcal{D}_i$, and the QPTC correction factors are held fixed. Define the query-cluster radius as

$$\mathrm{rad}_{\mathrm{q},i} := \max_{s\in i}\|q_s - \bar{q}_i\|. \tag{44}$$

Proposition 2 (Token-level residual bound from a query centroid)**.** *Fix a query cluster i and the approximation constructed for it as described above. Let $O^{\mathrm{exact}}(z)$ denote the exact attention output for a query vector z. For each $j \in \mathcal{D}_i$, let $\lambda_{ij} = n_j$ under centroid compensation and $\lambda_{ij} = n_j(1 + \widehat{\sigma}^2_{\ell,ij}/2)$ under QPTC. The corresponding exact-or-approximate output is*

$$O^{\mathrm{approx}}(z) := \frac{\sum_{j\in\mathcal{S}_i}\sum_{t\in j} e^{z^\top k_t/\sqrt{d}} v_t + \sum_{j\in\mathcal{D}_i} \lambda_{ij} e^{z^\top \bar{k}_j/\sqrt{d}} \bar{v}_j}{\sum_{j\in\mathcal{S}_i}\sum_{t\in j} e^{z^\top k_t/\sqrt{d}} + \sum_{j\in\mathcal{D}_i} \lambda_{ij} e^{z^\top \bar{k}_j/\sqrt{d}}}. \tag{45}$$

Let $R(z) := O^{\mathrm{exact}}(z) - O^{\mathrm{approx}}(z)$. If $\max_t \|k_t\| \le \beta_{\mathrm{k}}$ and $\max_t \|v_t\| \le \beta_{\mathrm{v}}$, then every query token $s \in i$ satisfies

$$\|R(q_s) - R(\bar{q}_i)\| \le \frac{2\beta_{\mathrm{k}}\beta_{\mathrm{v}}}{\sqrt{d}} \|q_s - \bar{q}_i\| \le \frac{2\beta_{\mathrm{k}}\beta_{\mathrm{v}}\,\mathrm{rad}_{\mathrm{q},i}}{\sqrt{d}}. \tag{46}$$

Consequently,

$$\|R(q_s)\| \le \|R(\bar{q}_i)\| + \frac{2\beta_{\mathrm{k}}\beta_{\mathrm{v}}\,\mathrm{rad}_{\mathrm{q},i}}{\sqrt{d}}. \tag{47}$$

Proof. We first bound the Jacobians of the exact and approximate outputs, then integrate their difference from $\bar{q}_i$ to q_s. For a vector output, ∇_z denotes its Jacobian; $\|\cdot\|_2$ denotes the matrix operator norm.

Define the exact attention weights, output, and weighted key mean by

$$\begin{aligned} p_t(z) &:= \frac{e^{z^\top k_t/\sqrt{d}}}{\sum_u e^{z^\top k_u/\sqrt{d}}}, \\ O^{\mathrm{exact}}(z) &:= \sum_t p_t(z) v_t, \qquad \mu_{\mathrm{k}}(z) := \sum_t p_t(z) k_t. \end{aligned} \tag{48}$$

Differentiating the numerator and denominator of $p_t(z)$ gives

$$\nabla_z p_t(z) = p_t(z)\left(\frac{k_t}{\sqrt{d}} - \sum_u p_u(z)\frac{k_u}{\sqrt{d}}\right) \tag{49}$$

$$= \frac{p_t(z)}{\sqrt{d}}\big(k_t - \mu_{\mathrm{k}}(z)\big). \tag{50}$$

Substituting this derivative into the exact output gives

$$\nabla_z O^{\mathrm{exact}}(z) = \sum_t v_t \nabla_z p_t(z)^\top \tag{51}$$

$$= \frac{1}{\sqrt{d}} \sum_t p_t(z) v_t \big(k_t - \mu_{\mathrm{k}}(z)\big)^\top. \tag{52}$$

To express this derivative in terms of value deviations from their weighted mean $O^{\mathrm{exact}}(z)$, first note that the weighted key deviations sum to zero:

$$\begin{aligned} \sum_t p_t(z)\big(k_t - \mu_{\mathrm{k}}(z)\big) &= \sum_t p_t(z) k_t - \mu_{\mathrm{k}}(z) \sum_t p_t(z) \\ &= \mu_{\mathrm{k}}(z) - \mu_{\mathrm{k}}(z) = 0. \end{aligned}$$

We can therefore subtract $O^{\mathrm{exact}}(z)$ from every v_t without changing the sum:

$$\begin{aligned} \nabla_z O^{\mathrm{exact}}(z) &= \frac{1}{\sqrt{d}} \sum_t p_t(z) v_t \big(k_t - \mu_{\mathrm{k}}(z)\big)^\top \\ &\quad - \frac{O^{\mathrm{exact}}(z)}{\sqrt{d}} \underbrace{\sum_t p_t(z)\big(k_t - \mu_{\mathrm{k}}(z)\big)^\top}_{=0} \\ &= \frac{1}{\sqrt{d}} \sum_t p_t(z)\big(v_t - O^{\mathrm{exact}}(z)\big)\big(k_t - \mu_{\mathrm{k}}(z)\big)^\top. \end{aligned}$$

For any unit direction h with $\|h\| = 1$, the triangle inequality bounds the norm of this sum by the sum of the termwise norms. Applying Cauchy–Schwarz to the factors $\sqrt{p_t(z)}\|v_t - O^{\text{exact}}(z)\|$ and $\sqrt{p_t(z)}|(k_t - \mu_{\text{k}}(z))^\top h|$ then gives

$$\begin{aligned}
\|\nabla_z O^{\text{exact}}(z)h\| &= \frac{1}{\sqrt{d}} \left\| \sum_t p_t(z) \big(v_t - O^{\text{exact}}(z)\big) \big(k_t - \mu_{\text{k}}(z)\big)^\top h \right\| \\
&\le \frac{1}{\sqrt{d}} \sum_t p_t(z) \|v_t - O^{\text{exact}}(z)\| \left| \big(k_t - \mu_{\text{k}}(z)\big)^\top h \right| \\
&\le \frac{1}{\sqrt{d}} \left(\sum_t p_t(z) \|v_t - O^{\text{exact}}(z)\|^2 \right)^{1/2} \\
&\quad \cdot \left(\sum_t p_t(z) \big((k_t - \mu_{\text{k}}(z))^\top h\big)^2 \right)^{1/2}.
\end{aligned}$$

For the value factor, expand the square and collect the terms using $\sum_t p_t(z) = 1$:

$$\begin{aligned}
&\sum_t p_t(z) \|v_t - O^{\text{exact}}(z)\|^2 \\
&= \sum_t p_t(z) \left[\|v_t\|^2 - 2 v_t^\top O^{\text{exact}}(z) + \|O^{\text{exact}}(z)\|^2 \right] \\
&= \sum_t p_t(z) \|v_t\|^2 - 2 \left(\sum_t p_t(z) v_t \right)^\top O^{\text{exact}}(z) + \|O^{\text{exact}}(z)\|^2 \\
&= \sum_t p_t(z) \|v_t\|^2 - 2 \|O^{\text{exact}}(z)\|^2 + \|O^{\text{exact}}(z)\|^2 \\
&= \sum_t p_t(z) \|v_t\|^2 - \|O^{\text{exact}}(z)\|^2 \\
&\le \sum_t p_t(z) \|v_t\|^2 \le \beta_{\text{v}}^2 \sum_t p_t(z) = \beta_{\text{v}}^2.
\end{aligned}$$

The third equality substitutes $\sum_t p_t(z) v_t = O^{\text{exact}}(z)$. The inequalities discard a nonnegative squared norm and use $\|v_t\| \le \beta_{\text{v}}$.

For the key factor, the weighted mean of $k_t^\top h$ is $\sum_t p_t(z) k_t^\top h = \mu_{\text{k}}(z)^\top h$. Expanding the scalar square gives

$$\begin{aligned}
&\sum_t p_t(z) \big((k_t - \mu_{\text{k}}(z))^\top h\big)^2 \\
&= \sum_t p_t(z) \left[(k_t^\top h)^2 - 2 (k_t^\top h)(\mu_{\text{k}}(z)^\top h) + (\mu_{\text{k}}(z)^\top h)^2 \right] \\
&= \sum_t p_t(z) (k_t^\top h)^2 - 2 (\mu_{\text{k}}(z)^\top h) \sum_t p_t(z) k_t^\top h + (\mu_{\text{k}}(z)^\top h)^2 \\
&= \sum_t p_t(z) (k_t^\top h)^2 - 2 (\mu_{\text{k}}(z)^\top h)^2 + (\mu_{\text{k}}(z)^\top h)^2 \\
&= \sum_t p_t(z) (k_t^\top h)^2 - (\mu_{\text{k}}(z)^\top h)^2 \\
&\le \sum_t p_t(z) (k_t^\top h)^2 \\
&\le \sum_t p_t(z) \|k_t\|^2 \|h\|^2 \le \beta_{\text{k}}^2.
\end{aligned}$$

Here, the last line uses $(k_t^\top h)^2 \le \|k_t\|^2 \|h\|^2$, $\|h\| = 1$, and $\|k_t\| \le \beta_{\text{k}}$. Substituting the two bounds above yields $\|\nabla_z O^{\text{exact}}(z)h\| \le \beta_{\text{k}} \beta_{\text{v}} / \sqrt{d}$ for every unit h. By the definition of the operator norm,

$$\|\nabla_z O^{\text{exact}}(z)\|_2 = \sup_{\|h\|=1} \|\nabla_z O^{\text{exact}}(z)h\| \le \frac{\beta_{\text{k}} \beta_{\text{v}}}{\sqrt{d}}. \tag{53}$$

For the approximate output, each fixed multiplier can be absorbed into a constant logit bias:

$$\lambda_{ij} e^{z^\top \bar{k}_j/\sqrt{d}} = e^{z^\top \bar{k}_j/\sqrt{d} + \log \lambda_{ij}}, \qquad \nabla_z \log \lambda_{ij} = 0. \tag{54}$$

Moreover, the centroid pairs satisfy

$$\|\bar{k}_j\| \le \frac{1}{n_j} \sum_{t \in j} \|k_t\| \le \beta_{\mathrm{k}}, \qquad \|\bar{v}_j\| \le \frac{1}{n_j} \sum_{t \in j} \|v_t\| \le \beta_{\mathrm{v}}. \tag{55}$$

The approximate output is therefore also a normalized exponential-weighted sum: it contains one term for each token in the exact clusters and one term for each centroid in the approximated clusters. The constant logit biases do not contribute to differentiation with respect to z, and the normalized weights still sum to one. Since both the original and centroid keys and values obey the same norm bounds, the centering, Cauchy–Schwarz, and squared-deviation steps above give

$$\|\nabla_z O^{\mathrm{approx}}(z)\|_2 \le \frac{\beta_{\mathrm{k}} \beta_{\mathrm{v}}}{\sqrt{d}}. \tag{56}$$

Since $R(z) = O^{\mathrm{exact}}(z) - O^{\mathrm{approx}}(z)$, the triangle inequality for the operator norm gives

$$\begin{aligned} \|\nabla_z R(z)\|_2 &= \|\nabla_z O^{\mathrm{exact}}(z) - \nabla_z O^{\mathrm{approx}}(z)\|_2 \\ &\le \|\nabla_z O^{\mathrm{exact}}(z)\|_2 + \|\nabla_z O^{\mathrm{approx}}(z)\|_2 \\ &\le \frac{2\beta_{\mathrm{k}} \beta_{\mathrm{v}}}{\sqrt{d}}. \end{aligned}$$

To relate this derivative bound to the two queries, consider the line segment $z_\tau = \bar{q}_i + \tau(q_s - \bar{q}_i)$, $0 \le \tau \le 1$. The chain rule gives $\frac{d}{d\tau} R(z_\tau) = \nabla_z R(z_\tau)(q_s - \bar{q}_i)$. Integrating from $\tau = 0$ to $\tau = 1$ therefore yields

$$R(q_s) - R(\bar{q}_i) = \int_0^1 \nabla_z R(z_\tau)(q_s - \bar{q}_i)\, d\tau.$$

Taking norms and applying the Jacobian bound along the segment gives

$$\begin{aligned} \|R(q_s) - R(\bar{q}_i)\| &\le \int_0^1 \|\nabla_z R(z_\tau)(q_s - \bar{q}_i)\|\, d\tau \\ &\le \int_0^1 \|\nabla_z R(z_\tau)\|_2 \|q_s - \bar{q}_i\|\, d\tau \\ &\le \int_0^1 \frac{2\beta_{\mathrm{k}} \beta_{\mathrm{v}}}{\sqrt{d}} \|q_s - \bar{q}_i\|\, d\tau \\ &= \frac{2\beta_{\mathrm{k}} \beta_{\mathrm{v}}}{\sqrt{d}} \|q_s - \bar{q}_i\|. \end{aligned}$$

The definition of $\mathrm{rad}_{\mathrm{q},i}$ gives the second inequality in Equation 46. Finally,

$$\|R(q_s)\| \le \|R(\bar{q}_i)\| + \|R(q_s) - R(\bar{q}_i)\| \tag{57}$$

$$\le \|R(\bar{q}_i)\| + \frac{2\beta_{\mathrm{k}} \beta_{\mathrm{v}}\, \mathrm{rad}_{\mathrm{q},i}}{\sqrt{d}}, \tag{58}$$

which proves Equation 47. □

Equation 47 bounds the token-level output residual by the residual at the query centroid and a term proportional to the query-cluster radius.

B Kernel Implementation Details

This section describes how the three components map to the GPU execution flow. For clarity, the batch and head dimensions are omitted from the pseudocode. Standard K-means, the top-p budget calculation, and FlashInfer exact attention are treated as existing primitives; the algorithms expand

Algorithm 1 Query-Induced Key Clustering (QIKC)

Require: Query centroids $\{\bar{q}_i\}_{i=1}^{C_\mathrm{q}}$, query-cluster sizes $\boldsymbol{m}$, query-token count N_q, keys K, key-cluster count C_k
Ensure: Key assignments a^k, key-cluster sizes $\boldsymbol{n}$, key permutation π_k
for all channels $r \in \{1, \dots, d\}$ in parallel **do**
  $\widetilde{\omega}_r \leftarrow \sum_i (m_i/N_\mathrm{q}) \bar{q}_{ir}^2$
end for
$\alpha_\mathrm{q} \leftarrow d^{-1} \sum_r \sqrt{\widetilde{\omega}_r}$
$\gamma_r \leftarrow \sqrt{\widetilde{\omega}_r}/\alpha_\mathrm{q}$ for all r
for all key tokens t in parallel **do**
  $k_t^{\mathrm{scaled}} \leftarrow \boldsymbol{\gamma} \odot k_t$
  $r_t^\mathrm{k} \leftarrow \|k_t^{\mathrm{scaled}}\|^2$
end for
$(a^\mathrm{k}, \boldsymbol{n}) \leftarrow \mathrm{KMeans}(K^{\mathrm{scaled}}, C_\mathrm{k}; \boldsymbol{r}^\mathrm{k})$
$\pi_\mathrm{k} \leftarrow \mathrm{argsort}(a^\mathrm{k})$
return $(a^\mathrm{k}, \boldsymbol{n}, \pi_\mathrm{k})$

the operations introduced by ResiSparse. Reductions used for routing and correction are accumulated in FP32.

The execution proceeds as follows. Query clustering supplies the query centroids used to construct the QIKC channel scale, after which the scaled keys are clustered. The resulting assignments drive a fused permutation pass that also computes the original-space key and value centroids, the scalar moments used by UMDR, and the selected-channel key variances used by QPTC. UMDR then constructs the exact block mask, FlashInfer evaluates the selected blocks, and the final kernel merges their output with QPTC-corrected centroid contributions from the remaining blocks.

QIKC. Standard query clustering produces the assignments a^q, centroids $\bar{Q}$, cluster sizes $\boldsymbol{m}$, and query permutation $\pi_\mathrm{q} = \mathrm{argsort}(a^\mathrm{q})$. Algorithm 1 begins from $\bar{Q}$ and $\boldsymbol{m}$ and expands only the QIKC-specific key preprocessing and clustering, while leaving the standard Lloyd iterations inside the existing K-means primitive. A fused key pass applies the channel scale and computes the squared norm reused by K-means assignment. The resulting key assignments are defined in the scaled space, but the original keys remain unchanged for subsequent attention computation.

The key permutation follows π_k and recomputes all quantities used by attention from the original, unscaled keys. In a single pass over each key cluster j, the fused permutation kernel writes the reordered K and V tensors and accumulates

$$\begin{aligned} \bar{k}_j &= \frac{1}{n_j} \sum_{t \in j} k_t, & \bar{v}_j &= \frac{1}{n_j} \sum_{t \in j} v_t, \\ \mu_{\mathrm{k},j}^{(2)} &= \frac{1}{n_j d} \sum_{t \in j} \|k_t\|^2, & \mu_{\mathrm{v},j}^{(2)} &= \frac{1}{n_j d} \sum_{t \in j} \|v_t\|^2. \end{aligned} \tag{59}$$

Before this pass, QPTC selects the channel set Ω using the highest estimated query channel energies $\widetilde{\omega}_r$. The same kernel accumulates the first and second key moments on these channels and returns

$$\sigma_{\mathrm{k},j,r}^2 = \frac{1}{n_j} \sum_{t \in j} k_{tr}^2 - \left(\frac{1}{n_j} \sum_{t \in j} k_{tr} \right)^2, \qquad r \in \Omega. \tag{60}$$

Thus, permutation, centroid construction, scalar moment accumulation, and selected-channel variance estimation are produced in the same fused pass over the original keys and values.

UMDR. The fused moments in Equation 59 yield the key and value dispersions without another token pass. Algorithm 2 then evaluates the UMDR score on the $C_\mathrm{q} \times C_\mathrm{k}$ cluster grid.

Because each centroid is the corresponding cluster mean, the identity

$$\frac{1}{n_j} \sum_{t \in j} \|x_t - \bar{x}_j\|^2 = \frac{1}{n_j} \sum_{t \in j} \|x_t\|^2 - \|\bar{x}_j\|^2$$

Algorithm 2 Unified Mass–Dispersion Routing (UMDR)

Require: Query centroids $\{\bar{q}_i\}_{i=1}^{C_\mathrm{q}}$, key/value centroids $\{(\bar{k}_j, \bar{v}_j)\}_{j=1}^{C_\mathrm{k}}$
Require: Key/value mean squared norms $(\boldsymbol{\mu}_\mathrm{k}^{(2)}, \boldsymbol{\mu}_\mathrm{v}^{(2)})$, numerical floor ϵ
Ensure: Routing scores $\rho^{\mathrm{UMDR}} \in \mathbb{R}^{C_\mathrm{q} \times C_\mathrm{k}}$
for all key clusters j in parallel **do**
  $\sigma_{\mathrm{k},j}^2 \leftarrow \max\Big(\mu_{\mathrm{k},j}^{(2)} - \|\bar{k}_j\|^2/d, 0\Big)$
  $\sigma_{\mathrm{v},j}^2 \leftarrow \max\Big(\mu_{\mathrm{v},j}^{(2)} - \|\bar{v}_j\|^2/d, 0\Big)$
end for
for all query–key cluster pairs (i, j) in parallel **do**
  $\rho_{ij}^{\mathrm{UMDR}} \leftarrow \bar{q}_i^\top \bar{k}_j/\sqrt{d} + \frac{1}{2}\log\max(\sigma_{\mathrm{k},j}^2, \epsilon) + \frac{1}{2}\log\max(\sigma_{\mathrm{v},j}^2, \epsilon)$
end for
return ρ^{UMDR}

Algorithm 3 Query-Prioritized Taylor Compensation (QPTC)

Require: Queries $\{q_s\}_{s=1}^{N_\mathrm{q}}$, query-cluster assignments $\{a_s^\mathrm{q}\}_{s=1}^{N_\mathrm{q}}$, key/value centroids $\{(\bar{k}_j, \bar{v}_j)\}_{j=1}^{C_\mathrm{k}}$, key-cluster sizes $\boldsymbol{n}$
Require: Approximated sets $\{\mathcal{D}_i\}$, channels Ω, variances $\{\sigma_{\mathrm{k},j,r}^2 : r \in \Omega\}$
Require: Exact outputs $\{O_s^{\mathrm{ex}}\}_{s=1}^{N_\mathrm{q}}$ and log-sum-exp values $\{\mathrm{LSE}_s^{\mathrm{ex}}\}_{s=1}^{N_\mathrm{q}}$
Ensure: Attention outputs $\{O_s\}_{s=1}^{N_\mathrm{q}}$
for all query–key cluster pairs (i, j) in parallel **do**
  $\widehat{\sigma}_{\ell,ij}^2 \leftarrow d^{-1}\sum_{r\in\Omega} \bar{q}_{ir}^2 \sigma_{\mathrm{k},j,r}^2$
  $B_{ij} \leftarrow \log(1 + \widehat{\sigma}_{\ell,ij}^2/2)$
end for
for all query clusters i and query tokens s with $a_s^\mathrm{q} = i$ in parallel **do**
  $m_s \leftarrow \mathrm{LSE}_s^{\mathrm{ex}}$, $\ell_s \leftarrow 1$, $\mathrm{acc}_s \leftarrow O_s^{\mathrm{ex}}$
  for each $j \in \mathcal{D}_i$ **do**
    $S_{s,j}^{\mathrm{corr}} \leftarrow q_s^\top \bar{k}_j/\sqrt{d} + \log n_j + B_{ij}$
    $m_s^{\mathrm{new}} \leftarrow \max(m_s, S_{s,j}^{\mathrm{corr}})$
    $\alpha \leftarrow \exp(m_s - m_s^{\mathrm{new}})$
    $p \leftarrow \exp(S_{s,j}^{\mathrm{corr}} - m_s^{\mathrm{new}})$
    $\ell_s \leftarrow \ell_s\alpha + p$
    $\mathrm{acc}_s \leftarrow \mathrm{acc}_s\alpha + p\bar{v}_j$
    $m_s \leftarrow m_s^{\mathrm{new}}$
  end for
  $O_s \leftarrow \mathrm{acc}_s/\ell_s$
end for
return O

allows the key and value dispersions in Equation 9 to be recovered from the accumulated mean squared norms by subtracting the corresponding squared centroid norms. The maximum with zero in Algorithm 2 only clamps small negative values caused by floating-point roundoff.

The top-p calculation determines the key-token budget for each query-cluster row, which is filled in descending order of $\rho_{ij}^{\mathrm{UMDR}}$. Computing the centroid logits costs $O(C_\mathrm{q}C_\mathrm{k}d)$ and avoids the token-level probe used by SVG-EAR (Zhou et al., 2026).

QPTC. After UMDR constructs the exact block mask, FlashInfer (Ye et al., 2025) evaluates those blocks and returns the exact output O_s^{ex} and log-sum-exp $\mathrm{LSE}_s^{\mathrm{ex}}$ for every query token. Algorithm 3 extends the online merge used by SVG-EAR (Zhou et al., 2026): it computes a pair-specific QPTC bias from the selected-channel variances and adds it to each approximated centroid logit before merging the resulting numerator and denominator contributions with the exact result.

Table 2: Generation settings used in the main evaluation.

Model	Resolution	Frames	Steps	Guidance	Shift	Precision
Wan2.1-1.3B	1280×720	81	50	5.0	–	BF16
Wan2.1-14B	960×544	81	50	5.0	–	BF16
HunyuanVideo-13B	960×544	129	50	6.0	7.0	BF16

Table 3: Sparse-attention parameters shared across the three evaluated models. "Warmup steps ratio" is the fraction of initial denoising steps evaluated with dense attention.

Method	C_{q}	C_{k}	Top-p	Min. key ratio	Warmup steps ratio	K-means iterations	d_{sel}
SVG2	300	500	0.90	0.15	10%	50 / 2	–
SVG-EAR	300	500	0.85	0.05	10%	50 / 2	–
ResiSparse	300	500	0.85	0.05	10%	50 / 2	32

The merge kernel processes query tokens and approximated key clusters in tiles, updates the numerator and denominator together in log-sum-exp form, and writes each query output directly to its original token position using the inverse query permutation. It merges the approximated contributions with the exact result without materializing tokenwise attention weights.

C Experimental Details

Evaluation setup. We evaluate the publicly released Wan2.1-T2V-1.3B, Wan2.1-T2V-14B, and HunyuanVideo-13B checkpoints using the official VBench prompts (Huang et al., 2024). Videos are generated at 16 fps using the scheduler released with each checkpoint; Table 2 lists the remaining generation settings. Sparse attention is applied to all transformer layers after the first 10% of denoising steps. The first K-means invocation uses at most 50 iterations, followed by two iterations initialized from the preceding denoising step; Table 3 gives the remaining sparse-attention parameters. PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) are computed frame by frame and averaged over frames and videos, while video quality is evaluated with the official VBench implementation (Huang et al., 2024). Table 4 reports the six displayed VBench dimensions and the official overall score, which is computed over the complete evaluation suite rather than as their arithmetic mean. We also optimize the implementations of FlashInfer and K-means. Latency is measured with batch size one and BF16 inference on a single NVIDIA A800 PCIe GPU after warm-up. Denoising latency includes the transformer denoising loop and excludes model loading, text encoding, VAE decoding, and video export; speedup is the ratio of paired dense to sparse denoising latency under the same generation setting.

Prompt for Figure 1. The qualitative comparison uses the following text prompt for all three attention methods:

> warm colors dominate the room, with a focus on the tabby cat sitting contently in the center. the scene captures the fluffy orange tabby cat wearing a tiny virtual reality headset. the setting is a cozy living room, adorned with soft, warm lighting and a modern aesthetic. a plush sofa is visible in the background, along with a few lush potted plants, adding a touch of greenery. the cat's tail flicks curiously, as if engaging with an unseen virtual environment. its paws swipe at the air, indicating a playful and inquisitive nature, as it delves into the digital realm. the atmosphere is both whimsical and futuristic, highlighting the blend of analog and digital experiences.

Table 4: Complete VBench results under the generation settings used in Table 1. We report subject consistency (SC), motion smoothness (MS), dynamic degree (DD), aesthetic quality (AQ), imaging quality (IQ), overall consistency (OC), and the official overall score (AVG). Higher is better. Bold indicates the best result among the evaluated sparse methods.

Method	SC ↑	MS ↑	DD ↑	AQ ↑	IQ ↑	OC ↑	AVG ↑
Wan2.1 *(1.3B, text-to-video, 81 frames, 720p)*							
Dense	92.60%	98.56%	56.94%	65.30%	66.59%	27.31%	76.80%
SVG2	91.84%	98.50%	**55.56%**	**65.33%**	65.99%	**27.44%**	**76.46%**
SVG-EAR	91.99%	**98.56%**	54.17%	65.20%	65.89%	27.34%	76.30%
ResiSparse	**92.36%**	**98.56%**	54.17%	65.13%	**66.16%**	27.22%	76.35%
Wan2.1 *(14B, text-to-video, 81 frames, 540p)*							
Dense	92.29%	97.56%	68.06%	65.65%	66.37%	27.50%	77.24%
SVG2	92.12%	97.55%	**69.44%**	65.35%	65.12%	27.47%	77.03%
SVG-EAR	92.29%	97.53%	65.28%	65.47%	65.99%	27.65%	76.96%
ResiSparse	**92.71%**	**97.57%**	65.28%	**65.61%**	**66.22%**	**27.67%**	**77.15%**
HunyuanVideo *(13B, text-to-video, 129 frames, 540p)*							
Dense	95.15%	99.16%	59.72%	65.46%	65.90%	26.04%	77.15%
SVG2	94.89%	99.08%	**59.72%**	65.30%	**65.59%**	26.16%	77.03%
SVG-EAR	94.92%	99.13%	**59.72%**	**65.47%**	65.51%	**26.17%**	**77.09%**
ResiSparse	**95.09%**	**99.15%**	**59.72%**	65.20%	65.55%	26.14%	77.08%

Metrics for mechanism analysis. For a key token k_t in cluster j, let $\Delta k_t = k_t - \bar{k}_j$, and define the token-level query second moment $G_{\mathrm{q}} = N_{\mathrm{q}}^{-1} \sum_{s=1}^{N_{\mathrm{q}}} q_s q_s^\top$. The query-induced key-logit MSE is

$$\mathcal{E}_{\mathrm{logit}} := \frac{1}{N_{\mathrm{q}} N_{\mathrm{k}}} \sum_{s=1}^{N_{\mathrm{q}}} \sum_{j=1}^{C_{\mathrm{k}}} \sum_{t \in j} \left(\frac{q_s^\top \Delta k_t}{\sqrt{d}} \right)^2 = \frac{1}{N_{\mathrm{k}} d} \sum_{j=1}^{C_{\mathrm{k}}} \sum_{t \in j} \Delta k_t^\top G_{\mathrm{q}} \Delta k_t. \tag{61}$$

We evaluate self-attention inputs recorded during inference. To compare improvements across inputs with different logit scales, we divide each method's MSE by Raw's MSE for the same input and report the mean ratio. This gives the relative logit MSE in Figure 3(a), with Raw equal to one. The exact-moment oracle performs clustering with the token-level query second moment G_{q}. All three variants use 500 key clusters and run 50 K-means iterations from the same initial key indices. The 300 query centroids are precomputed using 50 K-means iterations. Reported time includes 50 key K-means iterations and the preceding operations: QIKC computes channel scales and scales the keys, whereas Oracle constructs G_{q}, performs its eigendecomposition, and applies the resulting transformation to the keys. Query clustering is excluded. We report the mean time across recorded inputs.

To evaluate UMDR against its theoretical objective, we compute the magnitude of the value-weighting term $A_{ij} = Y_{ij} - Z_{ij} \bar{v}_j$ at each query centroid directly from the individual keys and values. Let $\mathcal{C}_i$ contain the candidate key clusters for query cluster i, excluding the same initially retained clusters for all methods. The fraction covered by the selected exact clusters $\mathcal{S}_i$ is

$$\mathrm{Capture}_i := \frac{\sum_{j \in \mathcal{S}_i \cap \mathcal{C}_i} \|A_{ij}\|}{\sum_{j \in \mathcal{C}_i} \|A_{ij}\|}. \tag{62}$$

Each method selects whole blocks in descending score order until their cumulative area $m_i n_j$ reaches the target budget. The horizontal axis reports each method's realized attention density from the actual selected block areas, including the initially retained blocks. All methods share the same query and key partitions, initialized independently with 50 K-means iterations for each recorded input.

QPTC estimates the sum of exponentiated logits Z_{ij} for each approximated cluster pair. We compare the corrected estimate with the exact sum to evaluate QPTC separately from within-cluster value weighting and routing quality. Query and key clustering are independently initialized with 50 iterations for each input; the resulting clusters and UMDR routing mask are fixed across channel counts. Correction time covers standalone channel selection, key-variance estimation, and log-bias

computation. The three estimates in Figure 3(b) are

$$\widetilde{Z}_{ij}^{\mathrm{raw}} = \widehat{Z}_{ij}, \qquad \widetilde{Z}_{ij}^{\mathrm{QPTC}} = \widehat{Z}_{ij}\left(1 + \frac{\widehat{\sigma}^2_{\ell,ij}}{2}\right), \qquad \widetilde{Z}_{ij}^{\mathrm{full}} = \widehat{Z}_{ij}\left(1 + \frac{\sigma^2_{\ell,ij}}{2}\right). \tag{63}$$

We compare each estimate with Z_{ij} using

$$\mathcal{E}_Z := \left(\frac{\sum_i m_i \sum_{j\in\mathcal{D}_i}\left(\widetilde{Z}_{ij} - Z_{ij}\right)^2}{\sum_i m_i \sum_{j\in\mathcal{D}_i} Z_{ij}^2}\right)^{1/2}. \tag{64}$$

This is a weighted normalized RMSE over approximated cluster pairs. The factor m_i accounts for the number of query tokens represented by $\bar{q}_i$, which gives each represented query token equal weight in the centroid-based evaluation. No additional n_j factor is needed because Z_{ij} already sums over all key tokens in cluster j. Normalization by the squared exact contributions makes the metric invariant to their overall scale and avoids the instability of computing a relative error separately for each cluster pair when Z_{ij} is close to zero. The restriction $j \in \mathcal{D}_i$ reflects that QPTC is applied only to approximated pairs. Raw applies no correction, QPTC uses the selected-channel variance, and Full covariance uses the complete within-cluster key covariance within the same second-order approximation. Lower values indicate more accurate denominator correction.

Layer-output MSE. For each recorded input and method, K-means is initialized independently with 50 iterations; the two-iteration warm-start update used during generation is not used in this analysis. We compute the mean-squared error between the sparse and dense self-attention outputs after the output projection and report its mean over the recorded inputs. Attention density is the fraction of token pairs assigned to exact computation, accounting for the realized query- and key-cluster sizes.

D Additional Qualitative Results

Figures 5–7 provide additional frame-level comparisons between dense attention and ResiSparse on the three evaluated video-generation models.

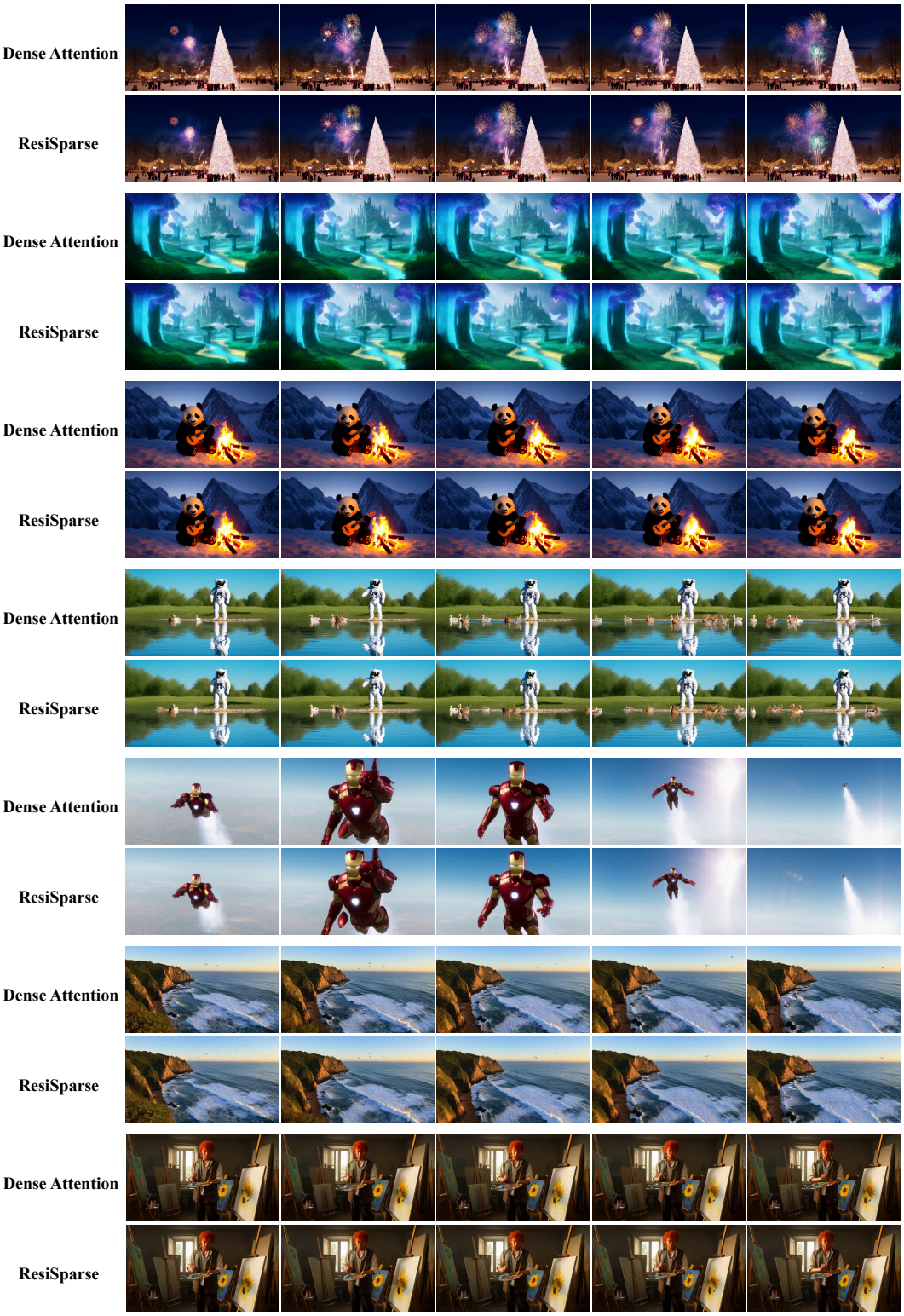


Figure 5: Additional qualitative results on Wan2.1-1.3B. Each example compares corresponding video frames generated with dense attention and ResiSparse under the same generation setting.

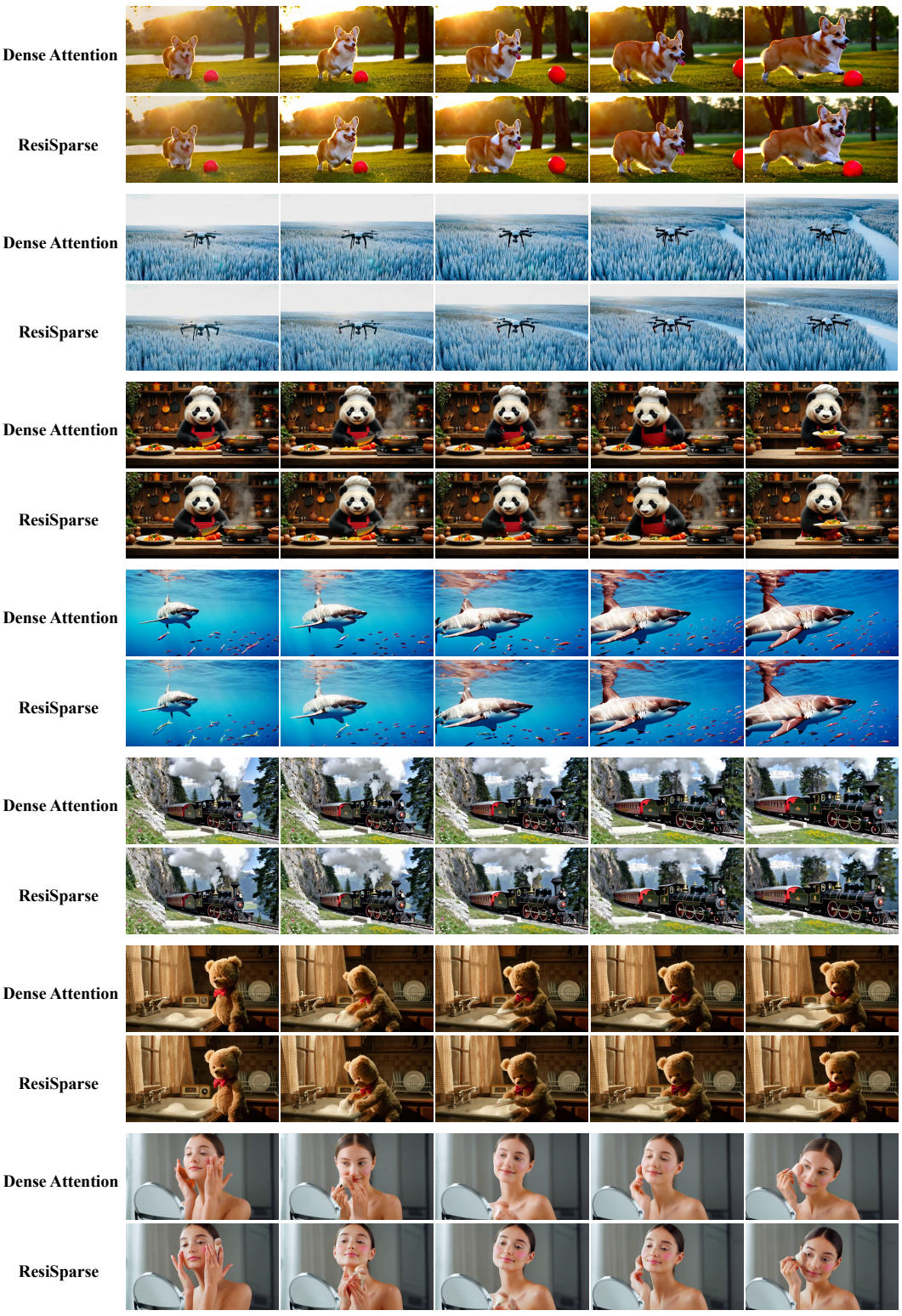


Figure 6: Additional qualitative results on Wan2.1-14B. Each example compares corresponding video frames generated with dense attention and ResiSparse under the same generation setting.

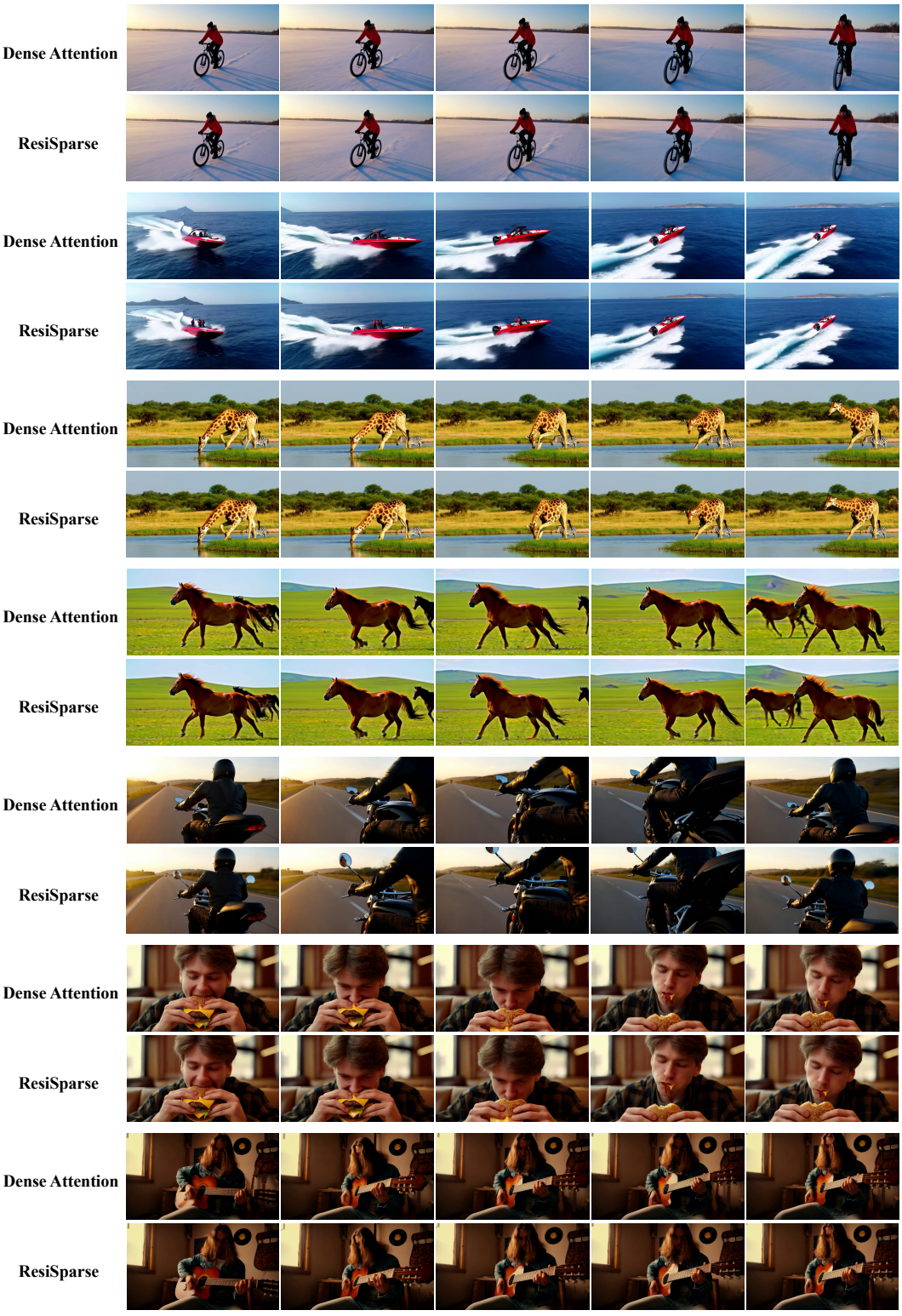


Figure 7: Additional qualitative results on HunyuanVideo-13B. Each example compares corresponding video frames generated with dense attention and ResiSparse under the same generation setting.