PARTITION THE SUPPORT, RECONSTRUCT THE RESIDUAL: TRAINING-FREE SPARSE ATTENTION FOR VIDEO GENERATION AND WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Block-sparse attention accelerates video transformers by grouping queries and keys and computing attention only between selected groups. However, it faces two challenges: (1) queries sharing a block route may attend to different keys; and (2) retaining most attention mass does not necessarily preserve the attention output. We find that partition choice affects both the overlap among grouped queries' preferred supports and how well an affine function of the sparse output can represent the difference between dense and sparse outputs. We introduce SparsePR, a training-free method combining Response-Coupled Partitioning with Probe-Fitted Residual Reconstruction. We first group paired key/value tokens by how their keys respond to sampled queries. We then group queries by their responses to the resulting key-group centroids for shared sparse routing. We compute exact attention on a small set of query rows and fit an affine correction to the differences between their dense and sparse outputs using weighted ridge regression. The fitted map predicts a residual for each unprobed query, which we add to its sparse output. We measure attention-output error as the relative L_2 error with respect to each query's dense output. Across four video generation and world models, SparsePR reduces this error, averaged over unprobed queries, by 61.6–90.5% compared with semantic block-sparse attention without residual correction, at 22% executed-pair density including exact probes. Ablations show that response-coupled partitioning reduces error before correction and improves reconstruction with the same probe budget, while probe fitting provides the largest overall reduction. SparsePR achieves $1.48\times$–$2.61\times$ end-to-end speedups over dense attention, with dense-reference PSNR of 24.42–31.84 dB at 21.9–26.0% realized executed-pair density.

1 INTRODUCTION

Video generation models (Kong et al., 2024; Wan et al., 2025; Yang et al., 2025) and video world models (Ali et al., 2025; Agarwal et al., 2026) process long spatiotemporal token sequences, making quadratic self-attention a major inference bottleneck. Attention often concentrates on a subset of tokens (Zhang et al., 2025b; Chen et al., 2026a), which training-free methods exploit by selectively computing query–key interactions (Xi et al., 2025; Zhang et al., 2025a; Luo et al., 2026). For hardware efficiency, block-sparse methods organize queries and keys into blocks and compute selected block pairs. These blocks can be organized by spatial and temporal position (Zhang et al., 2025b; Xi et al., 2025) or be formed by separately clustering query and key token representations by semantic similarity (Yang et al., 2026). Queries within a group share the same selected key/value groups, even when individual queries may favor different keys. This shared selection can therefore omit interactions important to some queries and change their attention outputs. Grouping queries with similar attention patterns can reduce this mismatch. However, skipped interactions can still cause the sparse output to differ from the dense output. Estimating this difference and adding it to the sparse output can improve accuracy without computing every interaction.

We make three observations that further motivate our approach. First, individual queries may concentrate attention on small but different sets of keys. Per-query concentration alone therefore does not describe the overlap among these sets within a query group. Second, retained attention mass alone does not determine attention-output error. The error also depends on the value vectors asso-

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.



Figure 1: Qualitative results for the Cosmos-Predict2.5-14B model.

ciated with the skipped interactions. Similar retained mass can therefore produce different output errors. Third, partition choice affects how much of the difference between dense and sparse attention outputs can be represented as an affine function of the sparse output, a linear map with an offset. We assess this representability through the best fit using dense outputs on all evaluated queries. Prediction from a small probe set is evaluated separately. Together, these observations motivate designing token partitions that group queries with overlapping attention supports and make the resulting output error easier to approximate with an affine function of the sparse output.

We introduce SparsePR, a training-free sparse-attention framework that combines *Response-Coupled Partitioning* and *Probe-Fitted Residual Reconstruction*. Response-Coupled Partitioning groups paired key/value tokens using low-dimensional coordinates derived from their keys' responses to sampled queries. The resulting key group centroids define response coordinates for grouping queries that share a sparse route. We construct the K/V groups first, then the query groups without alternating between the two partitions in a single pass. Probe-Fitted Residual Reconstruction computes exact attention for a small set of query rows and measures the differences between their dense and sparse outputs. We use weighted ridge regression to fit an affine map from the sparse outputs to these differences, then apply it to unprobed queries and add the predicted corrections to their sparse outputs. Both components operate online and require no offline sparsity profiling.

We make three contributions. First, we examine how token partitioning affects shared attention support and the output error that an affine correction can represent, and show why retained attention mass alone does not determine attention output accuracy. Second, we introduce SparsePR, a training-free method combining response-based token grouping with correction of sparse attention outputs. Third, across four video generation and world models, SparsePR reduces mean relative attention output error on unprobed queries by 61.6–90.5% compared with semantic block-sparse attention without correction at 22% executed-pair density including probes, and achieves $1.48\times$–$2.61\times$ end-to-end speedups over dense attention at 21.9–26.0% realized executed-pair density.

2 Related Work

Sparse Attention for Video Generation Models. Sliding Tile Attention (Zhang et al., 2025b), SVG (Xi et al., 2025), Sparse-vDiT (Chen et al., 2026a), Radial Attention (Li et al., 2026c), and LVSA (Glorian et al., 2026) exploit spatial, temporal, or head-specific attention structure. SpargeAttention (Zhang et al., 2025a) and XAttention (Xu et al., 2025) estimate block importance online. AdaSpa (Xia et al., 2025) combines online search with cached statistics across denoising steps, while HEART (Zheng et al., 2026) reuses sparse masks based on head-specific changes in queries and keys. ScalingAttention (Zhou et al., 2026a) extracts sparse topology priors offline. Among trained methods, VORTA (Sun et al., 2026) learns to route between attention variants, VSA (Zhang et al., 2026) trains sparse attention, and Veda (Han et al., 2026) uses distillation. VMoBA (Wu et al., 2026) supports sparse training and training-free inference. SparsePR combines online token partitioning with an output correction fitted from exact query rows in the current attention call, addressing shared block selection and error left by skipped interactions.

Token Partitioning for Executable Block Sparsity. Earlier methods organize tokens through hashing, clustering, and learned permutations (Kitaev et al., 2020; Roy et al., 2021; Vyas et al., 2020; Daras et al., 2020; Tay et al., 2020). SVG2 (Yang et al., 2026) separately clusters query and

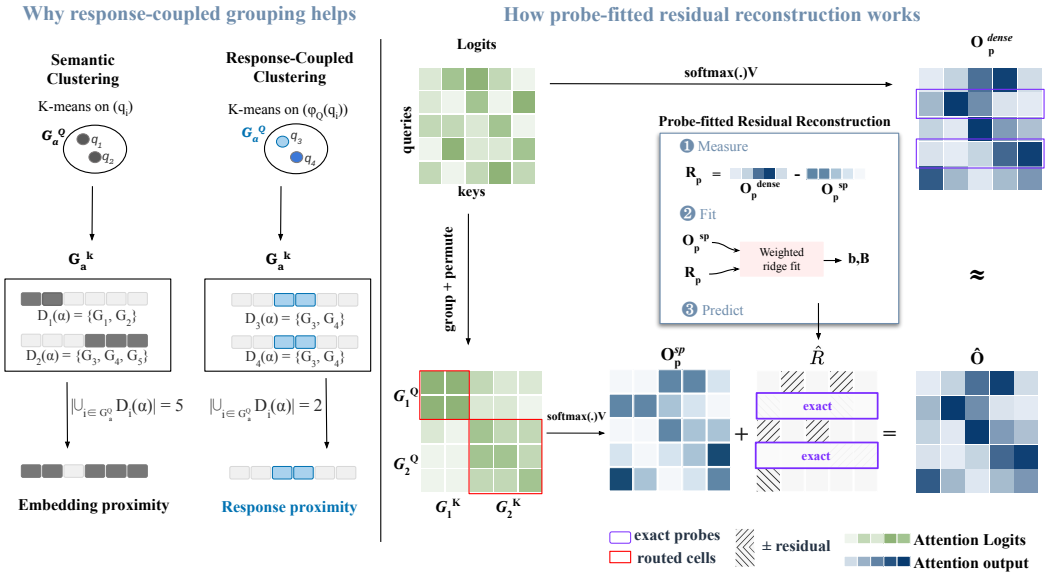

Figure 2: Overview of SparsePR. Response-Coupled Partitioning builds executable K/V and query groups, and Probe-Fitted Residual Reconstruction uses exact probe rows to correct the sparse output.

keys into contiguous groups. DraftAttention (Shen et al., 2025) uses low-resolution attention to guide sparse selection, while DFSAttn (Hu et al., 2026) combines Hilbert curve token reordering with hierarchical block scoring. AdaCluster (Tan et al., 2026) uses angular similarity for queries and Euclidean similarity for keys. SVOO (Luo et al., 2026) alternates key clustering based on query-centroid affinities and query clustering based on key-centroid affinities, using sparsity budgets calibrated offline. Response-Coupled Partitioning derives low-dimensional key coordinates from a metric built from sampled queries, then uses key group centroids to define query coordinates. It requires neither alternating partition refinement nor offline sparsity profiling, operating under a specified executed-pair budget.

Attention Approximation and Residual Reconstruction. Attention approximations use random features, landmarks, or sparse and low-rank approximations (Choromanski et al., 2020; Xiong et al., 2021; Chen et al., 2021). Re-ttention (Chen et al., 2026b) reshapes attention using past softmax statistics. Rectified SpaAttn (Liu et al., 2025) adjusts attention allocation using pooled attention estimates. PSA (Li et al., 2025) represents K/V blocks at multiple pooling levels. SVG-EAR (Zhou et al., 2026b) combines centroid-based compensation for skipped blocks with error-aware routing. PISA (Li et al., 2026b) approximates unselected interactions through blockwise Taylor expansions, while Sol-Attn (Li et al., 2026a) uses shared token to block scores for routing and approximate correction. SparsePR fits a call-specific affine map from sparse outputs to the differences between dense and renormalized sparse outputs measured on exact probe rows. Weighted ridge regression fits this map, which corrects unprobed queries directly in output space. This correction models the combined effect of omitted interactions and softmax renormalization directly in output space.

3 Sparse Attention Formulation and Observations

We formalize how token partitioning, route selection, and the treatment of skipped interactions define the sparse attention output. We then examine three observations that motivate SparsePR.

Dense and block-sparse attention. For one attention head, let Q, K, and V denote the query, key, and value matrices. Q contains N_q query tokens, while K and V contain N_k paired key and value tokens, all with per-head dimension d. We use row vectors q_i, k_j, and v_j for their individual token

representations. Dense attention computes

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right), \qquad O^{\mathrm{dense}} = AV. \tag{1}$$

A block-sparse operator partitions the query indices into $\mathcal{G}^Q = \{G_a^Q\}_{a=1}^{C_q}$ and the key indices into $\mathcal{G}^K = \{G_b^K\}_{b=1}^{C_k}$. The value paired with each key inherits its group assignment. The Cartesian product $G_a^Q \times G_b^K$ defines a cell of query–key interactions. Let $z_{ab} \in \{0, 1\}$ indicate whether this cell is selected for exact computation. All queries in G_a^Q share the same selected key groups and their corresponding values. We define routing density ρ_{route} as the number of query and key interactions evaluated in selected cells divided by $N_q N_k$. This definition counts individual interactions, so it accounts for unequal group sizes. Selecting all cells recovers dense attention and gives $\rho_{\mathrm{route}} = 1$.

Under renormalized hard drop, unselected keys are omitted and softmax is normalized over the retained keys. Let K_a and V_a collect the key and value rows from groups with $z_{ab} = 1$. For each query $i \in G_a^Q$,

$$O_i^{\mathrm{sp}} = \mathrm{softmax}\left(\frac{q_i K_a^\top}{\sqrt{d}}\right) V_a. \tag{2}$$

Collecting these rows gives O^{sp}. We define the residual left by this sparse operator as

$$R = O^{\mathrm{dense}} - O^{\mathrm{sp}}.$$

3.1 Structural Observations on Sparse Attention

We use semantic clustering as a reference partition, applying k-means separately to the current query and key activations, with each value token inheriting the assignment of its paired key (Yang et al., 2026). We use this partition to make the following three observations:

O1: Per-Query Sparsity Does Not Determine Shared-Route Support.

For retained mass $\alpha \in (0, 1]$, let $D_i(\alpha)$ denote the smallest set of highest-weight key indices whose cumulative dense attention mass for query i is at least α. For m queries $i_1, \ldots, i_m \in G_a^Q$, define

$$\tau_i(\alpha) = \frac{|D_i(\alpha)|}{N_k}, \qquad \gamma_a^{(m)}(\alpha) = \frac{|\bigcup_{r=1}^m D_{i_r}(\alpha)|}{N_k}.$$

Here, $\tau_i(\alpha)$ measures individual support density, while $\gamma_a^{(m)}(\alpha)$ measures pooled support density, defined by the union of individual top-mass supports. This union retains at least α attention mass for every sampled query, although a smaller jointly selected shared support may suffice. By operating on individual keys, the diagnostic isolates query-side support overlap from the effects of K/V-block granularity. Under the semantic partition, Fig. 3(a) compares both densities at $\alpha = 0.9$. Pooling eight queries increases the median support density from 6.2% to 22.9% for Wan2.2 and from 56.5% to 77.7% for Cosmos-Predict2.5. Pooled support therefore depends on both individual attention concentration and support overlap within query groups.

O2: Retained Attention Mass Alone Does Not Determine Output Error. For $i \in G_a^Q$, let U_a be the omitted key indices and $p_{U,i} = \sum_{j \in U_a} A_{ij}$ its dense attention mass. For $p_{U,i} > 0$, let $O_{U,i}$ be the output formed by renormalizing the dense attention weights over U_a. The residual then satisfies:

$$R_i = O_i^{\mathrm{dense}} - O_i^{\mathrm{sp}} = p_{U,i}\,(O_{U,i} - O_i^{\mathrm{sp}})\,. \tag{3}$$

The derivation is provided in Appendix A.1. Thus, high retained mass makes $p_{U,i}$ small but does not constrain $\|O_{U,i} - O_i^{\mathrm{sp}}\|_2$. Figure 3(b) demonstrates this for Cosmos3-Nano. Each point summarizes one attention head and reports the mean retained mass and p99 normalized output error across its queries. Cases A and B are two heads evaluated for the same input, layer, and generation step. Their retained masses differ by only 0.006 percentage points, whereas their p99 errors differ by $6.4\times$. Appendix D.1 reports the same comparison for the other three models.

O3: Partition Choice Affects Affine Residual Representability. We examine whether changing the partition affects how well an affine function of the sparse output can represent the attention

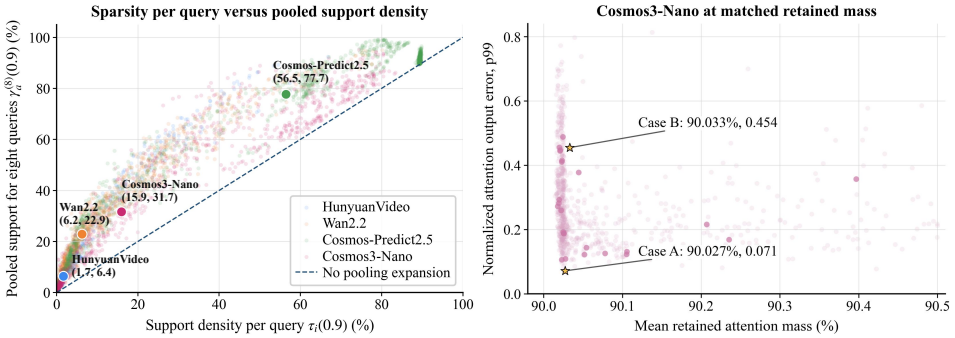

Figure 3: Structural observations under the semantic reference partition. (a) Pooling eight queries expands the support required to retain 90% attention mass. Markers denote model medians. (b) Nearly identical retained mass can yield substantially different output errors.

output error. Let g index a query and K/V partition configuration, with sparse output O_g^{sp} and residual $R_g = O^{\mathrm{dense}} - O_g^{\mathrm{sp}}$. We fit an affine estimate:

$$\widehat{R}_g^{\mathrm{oracle}} = O_g^{\mathrm{sp}} B_g + \mathbf{1} b_g. \tag{4}$$

Here, B_g maps sparse output channels to residual channels, and the row vector b_g is an offset shared across queries. We choose the coefficients to minimize the squared fitting error summed over all evaluated queries and output channels. We call this an oracle fit because it uses dense outputs on all evaluated queries, rather than only a small probe set.

Across four evaluated models, response-coupled partitioning increases the fraction of residual squared norm explained by the best affine fit by 3.70–15.52 percentage points and reduces the remaining squared error by 36.6–70.8% relative to semantic partitioning. Thus, the fit captures a larger fraction of the residual and leaves less error. The smaller remaining error reflects both the initial residual magnitude and the fraction explained by the fit. Appendix D.5 reports results for each model. Section 4.2 describes estimating a correction within the same affine family from a small probe set using weighted ridge regression, with accuracy evaluated separately on unprobed queries.

4 Method

Following observations in Section 3, SparsePR combines token grouping based on responses between queries and keys with an affine correction fitted from exact probe rows. Figure 2 summarizes the two components of SparsePR.

4.1 Response-Coupled Partitioning

An executable partition should group queries with similar routing behavior and paired K/V tokens whose keys induce similar response profiles. Proximity in the activation space does not guarantee either property. Nearby queries can require different K/V supports, while nearby keys can respond differently under the same queries. We construct partitions in low-dimensional response coordinates derived from the current attention call, capturing the dominant response variations while reducing the cost of clustering. The construction is coupled and asymmetric. Sampled queries define the coordinates for K/V grouping, and the resulting groups define the query coordinates.

Key-response coordinates for paired K/V groups.

Let $Q_s \in \mathbb{R}^{M_s \times d}$ contain M_s query rows selected deterministically at evenly spaced token positions from the current attention call. We group keys by the similarity of their responses to these queries using the metric

$$M_K = \frac{1}{M_s} Q_s^\top Q_s.$$

For any two keys k_j and k_ℓ,

$$(k_j - k_\ell)M_K(k_j - k_\ell)^\top = \frac{1}{M_s}\left\|Q_s(k_j - k_\ell)^\top\right\|_2^2.$$

Thus, distance under M_K measures the average squared response difference between keys: keys are close when they respond similarly across the sampled queries. These responses are pre-softmax logits up to the common $1/\sqrt{d}$ attention scale. Appendix B.1 provides the derivation.

To obtain compact clustering coordinates, we retain the leading r_K eigenvectors of M_K. We normalize their eigenvalues to unit mean and weight each eigenvector by the square root of its normalized eigenvalue, forming $F_K \in \mathbb{R}^{d\times r_K}$. We represent key k_j by

$$\phi_K(k_j) = k_j F_K \in \mathbb{R}^{r_K}. \tag{5}$$

Squared Euclidean distances in these coordinates reproduce the rank-r_K truncated response metric up to a common positive scale. Applying k-means with C_k requested clusters yields the paired K/V groups $\mathcal{G}^{KV}$, with each value inheriting its key's assignment. In the centroid-based construction below, C_k denotes the number of nonempty K/V groups.

Query-response coordinates from K/V groups. The resulting K/V groups define the response space used to compare queries. For each nonempty group, we compute its centroid in the original key space. We subtract the uniform mean of these centroids and stack the centered vectors as the rows of $\widetilde{K}$. Centering removes a common offset from each query's logits across groups while preserving their relative differences. The query-response metric is

$$M_Q = \frac{1}{C_k d}\widetilde{K}^\top\widetilde{K},$$

Two queries are close under this metric when they produce similar relative pre-softmax logit profiles over these groups. Applying the same eigenvalue weighted construction to the leading r_Q eigendirections of M_Q gives $F_Q \in \mathbb{R}^{d\times r_Q}$. Each projected query is then RMS-normalized:

$$\phi_Q(q_i) = \mathrm{RMSNorm}(q_i F_Q). \tag{6}$$

RMS normalization divides each projected vector by its root mean square magnitude, emphasizing response direction and reducing the influence of magnitude. Numerical stabilization is specified in Appendix B.2. The original queries are retained for routing and attention. Applying k-means with C_q requested clusters to $\phi_Q(q_i)$ yields the query partition $\mathcal{G}^Q$ used for shared routing. The construction proceeds once from K/V grouping to query grouping, without alternating refinement between the two partitions.

4.2 Probe-Fitted Residual Reconstruction

For each attention call, Probe-Fitted Residual Reconstruction uses $M \ll N_q$ exact query rows to fit a residual estimator. Motivated by the affine diagnostic in Section 3.1, we model the residual as an affine function of the sparse output:

$$x_i = O_i^{\mathrm{sp}} \in \mathbb{R}^d, \qquad R_i \approx b + x_i B.$$

Exact probe calibration. We select M probe indices $\mathcal{P} \subseteq \{1,\dots,N_q\}$ using the query groups. Exact attention on each probe gives $R_p = O_p^{\mathrm{dense}} - O_p^{\mathrm{sp}}$. Let $X_\mathcal{P}, R_\mathcal{P} \in \mathbb{R}^{M\times d}$ stack the probe sparse outputs and residuals, respectively, and let W be the diagonal matrix of group-coverage weights normalized to unit mean. These weights reflect the populations of the represented query groups. Groups without probes contribute no residual observations to the fit, but their queries use the same fitted correction. The selection and coverage-weight rules are specified in Appendix A.2; weight normalization is given in Appendix A.3. Using weighted probe statistics, we standardize $X_\mathcal{P}$ and center $R_\mathcal{P}$, yielding $\bar{X}_\mathcal{P}$ and $\bar{R}_\mathcal{P}$. We denote the weighted residual mean by μ_R.

Weighted ridge fit. We estimate the feature-to-residual map by weighted ridge regression:

$$B_\lambda = \arg\min_B \left\|W^{1/2}\left(\bar{R}_\mathcal{P} - \bar{X}_\mathcal{P}B\right)\right\|_F^2 + \lambda\|B\|_F^2. \tag{7}$$

Here, $B_\lambda \in \mathbb{R}^{d\times d}$ maps standardized sparse-output features to centered residual channels, and $\lambda > 0$ controls the regularization strength. The weights determine each probe's contribution to the fit, while the ridge penalty shrinks the coefficients and stabilizes estimation from a small probe set. Standardization and the solution procedure are detailed in Appendix A.3.

Residual prediction. For an unprobed query, let $\bar{x}_i$ denote its sparse output standardized using the probe statistics. We predict $\widehat{R}_i = \mu_R + \bar{x}_i B_\lambda$, without output-subspace truncation (Appendix D.3). The final attention output is

$$\widehat{O}_i = \begin{cases} O_i^{\text{dense}}, & i \in \mathcal{P}, \\ O_i^{\text{sp}} + \widehat{R}_i, & i \notin \mathcal{P}. \end{cases} \tag{8}$$

4.3 Sparse Execution and Exact-Pair Accounting

Routing and GQA. After reserving the cost of exact probes, we convert the target routed density into a common per-query key-token budget. For each query group, we rank nonempty K/V groups in descending order of the value-error score defined in Appendix B.3 and retain the shortest ranked prefix whose cumulative key count reaches this budget. Each selected cell contributes $|G_a^Q||G_b^{KV}|$ query–key pairs. Whole-cell selection can cause realized density to differ from the target, so we report it for each configuration. For grouped-query attention (GQA), the key-response metric and K/V partition are shared across query heads associated with the same native K/V head. Query partitions and routing decisions are computed separately for each query head.

Sparse execution and pair accounting. We permute Q, K, and V into contiguous group-major layouts, keeping each value token with its paired key. FlashInfer's variable-block sparse-attention primitive (Ye et al., 2025) takes the selected cells and ragged group offsets, evaluates the selected interactions exactly, and renormalizes each query row over its retained K/V groups as in Eq. (2). We then restore the output to the original query order. After sparse attention, we evaluate M exact probe rows per head against all keys, giving $\rho_{\text{exec}} = \rho_{\text{route}} + M/N_q$, where N_q is the query count per head. This counts pairs recomputed during probe evaluation.

5 Experiments

Implementation details. We evaluate all models on a single NVIDIA H100 with BF16 activations. SparsePR uses $C_q = 300$, $C_k = 1000$, $r_K = 48$, $r_Q = 64$, $M = 64$, and $\lambda = 1$. The total executed-pair target, including probes, is 22% for HunyuanVideo, Wan2.2, and Cosmos-Predict2.5, and 26% for Cosmos3-Nano. Timings include all online overheads. Appendix C details numerical precision, query sampling, clustering, GQA, and timing.

Models and benchmarks. We evaluate HunyuanVideo-13B for text-to-video generation using prompts from the Penguin suite and VBench metrics. We evaluate Wan2.2-I2V-A14B for image-to-video generation using VBench-I2V, the image-to-video component of VBench++ Huang et al. (2025). For Cosmos-Predict2.5-14B and Cosmos3-Nano-16B, we report selected VBench-I2V metrics and the PBench Quality score NVIDIA (2025) on their respective evaluation subsets.

Metrics. We measure fidelity to dense outputs using PSNR, SSIM, and LPIPS on corresponding frames. Generation quality is measured by VBench/VBench-I2V imaging quality (*ImgQual*), subject consistency, and PBench Quality. We report realized executed-pair density, end-to-end FLOPs, and end-to-end generation speedup. Dense, sparse, and reproduced baseline runs use matched inputs, seeds, and generation settings.

Quality and efficiency. Table 1 compares dense-reference fidelity, generation quality, and efficiency across four models. Figure 1 shows matched dense and SparsePR outputs, with additional comparisons in Appendix E.

5.1 Ablations

Partition and reconstruction. Table 2 compares partitioning and residual reconstruction.

Table 1: Quality–efficiency comparison across content-generation and physical video world models.

		Dense-Reference Fidelity			VBench / VBench++		PBench	Efficiency		
Model	**Method**	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**ImgQual↑**	**SubCons↑**	**Quality↑**	**Density↓**	**PFLOPs↓**	**E2E Speedup↑**
Content-generation video models										
HunyuanVideo 13B, 720p Text-to-Video	Dense	–	–	–	0.850	0.976	–	100.0%	612.38	1.00×
	SpargeAttn Zhang et al. (2025a)	25.31	0.832	0.217	0.763	0.928	–	40.19%	389.76	1.38×
	SVG2 Yang et al. (2026)	29.87	0.907	0.121	**0.850**	0.927	–	25.45%	299.02	2.30×
	SVOO Luo et al. (2026)	24.87	0.843	0.224	0.679	**0.976**	–	33.26%	345.81	2.17×
	SVG-EAR Zhou et al. (2026b)	30.54	0.918	0.098	0.845	0.903	–	22.17%	281.86	1.93×
	SparsePR	**31.84**	**0.932**	**0.087**	**0.850**	**0.976**	–	**21.92%**	**255.95**	**2.61×**
Wan2.2-I2V-A14B 14B active, 720p Image-to-Video	Dense	–	–	–	0.689	0.974	–	100.0%	658.46	1.00×
	SpargeAttn Zhang et al. (2025a)	26.74	0.871	0.116	0.680	0.953	–	30.15%	396.83	1.58×
	SVG2 Yang et al. (2026)	28.18	0.878	0.105	0.668	0.970	–	31.28%	393.95	1.59×
	SVOO Luo et al. (2026)	29.67	0.913	0.095	**0.689**	**0.973**	–	31.67%	389.62	1.61×
	SVG-EAR Zhou et al. (2026b)	29.75	**0.921**	0.086	0.680	0.970	–	23.64%	378.88	1.61×
	SparsePR	**30.66**	0.907	**0.044**	0.687	**0.973**	–	**21.97%**	**336.55**	**1.80×**
Physical video world models										
Cosmos-Predict2.5 14B, 720p Image-to-World	Dense	–	–	–	0.714	0.976	77.76	100.0%	526.87	1.00×
	SVG2 Yang et al. (2026)	20.07	0.624	0.330	0.678	0.896	76.14	28.81%	286.51	1.24×
	SVOO Luo et al. (2026)	22.06	0.685	0.289	0.701	0.909	76.03	37.63%	315.38	1.03×
	SVG-EAR Zhou et al. (2026b)	25.54	0.908	**0.062**	0.710	**0.976**	**77.76**	29.75%	289.69	1.10×
	SparsePR	**26.33**	**0.942**	0.068	**0.714**	**0.976**	77.75	**22.14%**	**253.61**	**1.51×**
Cosmos3-Nano 16B, 720p Image-to-World	Dense	–	–	–	0.700	0.950	77.31	100.0%	127.61	1.00×
	SVG2 Yang et al. (2026)	22.45	0.735	0.216	0.677	0.915	75.03	37.29%	96.19	1.16×
	SVOO Luo et al. (2026)	16.64	0.573	0.381	**0.707**	**0.962**	**77.59**	67.32%	108.01	1.02×
	SVG-EAR Zhou et al. (2026b)	21.16	0.709	0.261	0.658	0.872	72.85	37.18%	96.14	1.10×
	SparsePR	**24.42**	**0.801**	**0.176**	0.699	0.949	77.30	**25.96%**	**81.79**	**1.48×**

Table 2: Partition and reconstruction ablations at 22% executed-pair density, including exact probes. Entries report mean/p99 normalized attention output error on unprobed rows. Lower is better. Repair comparisons fix probe indices and estimator settings. The key-response K/V variant changes only K/V grouping, retaining semantic query groups.

Configuration	**HunyuanVideo**	**Wan2.2**	**Cosmos-Predict2.5**	**Cosmos3-Nano**
Partition construction, repair disabled				
Semantic partition	0.089 / 0.714	0.164 / 1.734	0.791 / 7.532	0.359 / 3.356
Key-response K/V partition	0.085 / 0.812	0.156 / 1.659	0.769 / 7.270	0.341 / 3.174
Response-coupled partition	**0.073 / 0.696**	**0.108 / 0.647**	**0.761 / 7.227**	**0.331 / 3.150**
Partition–repair interaction				
Semantic partition + probe repair	0.053 / 0.356	0.105 / 0.819	0.262 / 0.826	0.124 / 0.668
SparsePR	**0.030 / 0.207**	**0.063 / 0.323**	**0.075 / 0.489**	**0.076 / 0.447**

With repair disabled, response-coupled partitioning lowers mean and p99 error relative to semantic partitioning across all four models. Both components reduce error, and their combination achieves the lowest mean and p99 error across all four models. With the same probes and estimator, SparsePR reduces mean error by 38.7–71.4% and p99 error by 33.1–60.6% relative to semantic partitioning with repair (Fig. 4(a)).

Density sweep. Table 3 varies the total executed-pair budget for Cosmos3-Nano from 12% to 35%, holding remaining settings fixed within each method. Exact probes count toward the repaired methods' budgets. SparsePR achieves the lowest values among the compared methods at all five budgets.

Table 3: Cosmos3-Nano density sweep. Within each method, only the executed-pair budget varies, including exact probes where repair is used. Entries report mean/p99 normalized attention-output error on unprobed rows. Lower is better.

Method	$\rho = 12\%$	$\rho = 17\%$	$\rho = 22\%$	$\rho = 28\%$	$\rho = 35\%$
Semantic + hard drop	0.547 / 4.993	0.435 / 4.010	0.359 / 3.356	0.293 / 2.778	0.235 / 2.266
Response-coupled + hard drop	0.506 / 4.634	0.402 / 3.749	0.331 / 3.150	0.270 / 2.616	0.217 / 2.142
Semantic + probe repair	0.172 / 0.865	0.148 / 0.838	0.124 / 0.668	0.076 / 0.484	0.063 / 0.418
SparsePR	**0.127 / 0.724**	**0.099 / 0.577**	**0.076 / 0.447**	**0.068 / 0.422**	**0.056 / 0.366**

Probe and estimator design. In Table 4, query-group-stratified probes achieve the lowest mean and p99 errors across four models, tying random probes in mean error on Cosmos-Predict2.5. With sparse outputs, probes, and evaluation rows fixed, full weighted ridge outperforms mean-only, scalar

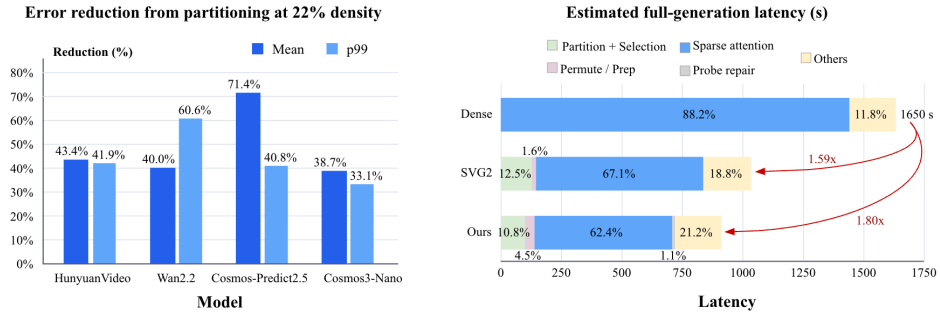

Figure 4: Partitioning and runtime analysis. (a) Error reduction from response-coupled vs. semantic partitioning at 22% density. (b) Wan2.2 full generation latency for Dense, SVG2, and SparsePR.

affine, and diagonal affine corrections in both metrics, supporting cross-channel residual prediction. Further appendices D.2 and D.3 examine query RMS normalization and sensitivity to probe count, output-rank truncation, and ridge strength.

Table 4: Probe-selection and correction-family ablations with response-coupled partitioning, $M = 64$, $\lambda = 1$. Within each panel, only the indicated component varies. Entries report mean/p99 normalized attention-output error on a shared set of unprobed query rows. Lower is better. Bold row labels mark the reference configuration, repeated in both panels.

Configuration	HunyuanVideo	Wan2.2	Cosmos-Predict2.5	Cosmos3-Nano
Probe selection: full weighted ridge fixed				
Random probes	0.039 / 0.235	0.080 / 0.356	**0.075** / 0.503	0.083 / 0.467
Uniform spatiotemporal probes	0.037 / 0.246	0.089 / 0.394	0.090 / 0.543	0.096 / 0.549
Query group stratified probes	**0.030 / 0.207**	**0.063 / 0.323**	**0.075 / 0.489**	**0.076 / 0.447**
Correction family: query group stratified probes fixed				
Mean-only	0.043 / 0.362	0.089 / 0.455	0.350 / 3.538	0.191 / 1.376
Scalar affine	0.034 / 0.281	0.068 / 0.365	0.195 / 2.069	0.131 / 0.880
Diagonal affine	0.032 / 0.246	0.065 / 0.352	0.082 / 0.543	0.099 / 0.604
Full weighted ridge	**0.030 / 0.207**	**0.063 / 0.323**	**0.075 / 0.489**	**0.076 / 0.447**

5.2 Runtime Analysis

Figure 4(b) decomposes estimated full-generation latency on Wan2.2. SparsePR reduces latency from 1650 s to 917 s, corresponding to a 1.80× speedup, compared with 1038 s and 1.59× for SVG2. Probe-Fitted Residual Reconstruction accounts for only 1.1% of latency. Thus, the primary fidelity gain from probe fitting adds little end-to-end overhead.

6 Conclusion

We study training-free block-sparse attention as an executable operator in which partitioning shapes both shared support and the output residual left by skipped interactions. SparsePR uses Response-Coupled Partitioning to improve support overlap under shared routing and Probe-Fitted Residual Reconstruction to correct the output residual left by skipped interactions. Across four video generation and world models, SparsePR reduces mean attention-output error on unprobed queries by 61.6–90.5% relative to semantic block-sparse attention without correction at 22% executed-pair density, including probes. Ablations show that response-coupled partitioning reduces error before correction and improves reconstruction under the same probe budget. SparsePR achieves 1.48×–2.61× end-to-end speedups while maintaining benchmark quality close to dense attention at 21.9–26.0% realized executed-pair density. These results show that effective sparse execution depends on both partition geometry and correction of the output error caused by skipped interactions.

Reproducibility Statement

We provide the methodological and implementation details needed to reproduce SparsePR and its evaluation. Section 4 specifies response-coupled partitioning, probe-fitted residual reconstruction, sparse routing, and executed-pair accounting. Appendices A and B give the probe-selection algorithm, weighted ridge formulation, and routing derivations, while Appendix C reports numerical precision, GPU implementation, shared hyperparameters, sequence lengths, and the timing protocol. Dense, sparse, and reproduced baseline runs use matched checkpoints, conditioning inputs, random seeds, sampling schedules, inference steps, guidance settings, resolutions, and frame counts. We report requested and realized executed-pair densities, including exact probe evaluations. Appendix D provides sensitivity analyses for query normalization, probe count, output rank, ridge strength, correction family, and affine residual representability.

Ethics Statement

This work does not involve human subjects, private or sensitive data, or the collection of personally identifiable information. The experiments use publicly available models, datasets, and benchmarks. We are not aware of additional ethical concerns specific to the proposed sparse attention methodology beyond those associated with the underlying video generation and world models.

AI Use Statement

Generative AI tools were used to assist with the clarity and presentation of the manuscript, as well as with limited code debugging. All research decisions were made by the authors, who independently verified the implementations, experiments, analyses, and reported results. The authors reviewed all AI-assisted content and take responsibility for the final manuscript and its claims.

References

Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, et al. World simulation with video foundation models for physical ai. *arXiv preprint arXiv:2511.00062*, 2025.

Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems*, 34: 17413–17426, 2021.

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, 2026a.

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, 2026b.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Giannis Daras, Nikita Kitaev, Augustus Odena, and Alexandros G Dimakis. Smyrf-efficient attention using asymmetric clustering. *Advances in Neural Information Processing Systems*, 33: 6476–6489, 2020.

Gael Glorian, Ioannis Lamprou, Zhen Zhang, Yujie Yuan, and Hongsheng Liu. Lvsa: Training-free sparse attention for long video diffusion. *arXiv preprint arXiv:2605.31057*, 2026.

Shihao Han, Hao Yang, Xinting Hu, Xiaofeng Mei, Yi Jiang, and Xiaojuan Qi. Veda: Scalable video diffusion via distilled sparse attention. *arXiv preprint arXiv:2605.30325*, 2026.

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, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, et al. Vbench++: Comprehensive and versatile benchmark suite for video generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

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.

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.

Xiaolong Li, Youping Gu, Xi Lin, Weijie Wang, and Bohan Zhuang. Psa: Pyramid sparse attention for efficient video understanding and generation. *arXiv preprint arXiv:2512.04025*, 2025.

Xingyang Li, Muyang Li, Tianle Cai, Haocheng Xi, Shuo Yang, Yujun Lin, Lvmin Zhang, Songlin Yang, Jinbo Hu, Kelly Peng, et al. Radial attention: $o(n \log n)$ sparse attention with energy decay for long video generation. *Advances in Neural Information Processing Systems*, 38:16822–16852, 2026c.

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.

NVIDIA. Pbench: A physical ai benchmark for world models, 2025. URL `https://huggingface.co/datasets/nvidia/PBench`.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Xuan Shen, Chenxia Han, Yufa Zhou, Yanyue Xie, Yifan Gong, Quanyi Wang, Yiwei Wang, Yanzhi Wang, Pu Zhao, and Jiuxiang Gu. Draftattention: Fast video diffusion via low-resolution attention guidance. *arXiv preprint arXiv:2505.14708*, 2025.

Wenhao Sun, Rong-Cheng Tu, Yifu Ding, Jingyi Liao, Zhao Jin, Shunyu Liu, and Dacheng Tao. Vorta: Efficient video diffusion via routing sparse attention. *Advances in Neural Information Processing Systems*, 38:7837–7863, 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.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International conference on machine learning*, pp. 9438–9447. PMLR, 2020.

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems*, 33:21665–21674, 2020.

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.

Jianzong Wu, Liang Hou, Haotian Yang, Ye Tian, Pengfei Wan, Di ZHANG, and Yunhai Tong. Vmoba: Mixture-of-block attention for video diffusion models. In *International Conference on Learning Representations*, volume 2026, pp. 132534–132548, 2026.

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.

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. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15982–15993. IEEE, 2025.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 14138–14148, 2021.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. *arXiv preprint arXiv:2503.16428*, 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, 2026.

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, 2025.

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.

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. Faster video diffusion with trainable sparse attention. *Advances in Neural Information Processing Systems*, 38:152509–152534, 2026.

Xuzhe Zheng, Yuexiao Ma, Jing Xu, Xiawu Zheng, Rongrong Ji, and Fei Chao. HASTE: Training-free video diffusion acceleration via head-wise adaptive sparse attention. *arXiv preprint arXiv:2605.14513v1*, 2026. URL https://arxiv.org/abs/2605.14513v1.

Ruiliang Zhou, Xuecheng Wu, Kang He, Guangyun Han, Bin Liu, Qinqin Chen, Wende Xu, Qingjie Zhao, and Chengru Song. Scalingattention: Discovering intrinsic sparse attention topology for video diffusion transformers. *arXiv preprint arXiv:2606.23019*, 2026a.

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*, 2026b.

TABLE OF CONTENTS

A Residual Reconstruction Details **14**

A.1 Hard Drop Residual and Affine Model 14

A.2 Exact Probe Selection Stratified by Query Group 14

A.3 Weighted Affine Fitting and Residual Prediction 15

B Response Coupled Partitioning and Sparse Execution **16**

B.1 Key Response Coordinates for Paired K/V Groups 16

B.2 Query Response Coordinates from K/V Groups 17

B.3 Routing Score and Executed Pair Budget 17

C Implementation and Evaluation Details **18**

C.1 GPU Implementation 18

C.2 Shared Hyperparameters 18

C.3 Timing Protocol 18

D Additional Ablations and Sensitivity Analyses **19**

D.1 Matched Retained Mass and Output Error 19

D.2 Query RMS Normalization 19

D.3 Probe and Estimator Sensitivity 20

D.4 Correction Family Comparison 20

D.5 Oracle Affine Residual Representability 21

D.6 Further Discussion 21

E More Qualitative Results **22**

A Residual Reconstruction Details

We derive the residual left by renormalized hard drop and describe the estimator fitted for each attention call. The estimator uses only sparse outputs as regression features. Query response coordinates determine query groups and probe selection but do not enter the regression.

A.1 Hard Drop Residual and Affine Model

For query row i let $\mathcal{S}_i$ and $\mathcal{U}_i$ denote the retained and omitted key sets. The attention logit is

$$\ell_{ij} = \frac{q_i k_j^\top}{\sqrt{d}}. \tag{A.1}$$

The retained and omitted partition sums and value numerators are

$$\begin{aligned} Z_{S,i} &= \sum_{j\in\mathcal{S}_i} \exp(\ell_{ij}), & N_{S,i} &= \sum_{j\in\mathcal{S}_i} \exp(\ell_{ij}) v_j, \\ Z_{U,i} &= \sum_{j\in\mathcal{U}_i} \exp(\ell_{ij}), & N_{U,i} &= \sum_{j\in\mathcal{U}_i} \exp(\ell_{ij}) v_j. \end{aligned} \tag{A.2}$$

Dense attention and renormalized hard drop therefore compute

$$O_i^{\text{dense}} = \frac{N_{S,i} + N_{U,i}}{Z_{S,i} + Z_{U,i}}, \qquad O_i^{\text{sp}} = \frac{N_{S,i}}{Z_{S,i}}. \tag{A.3}$$

Let

$$p_{U,i} = \frac{Z_{U,i}}{Z_{S,i} + Z_{U,i}}, \qquad O_{U,i} = \frac{N_{U,i}}{Z_{U,i}} \quad (p_{U,i} > 0) \tag{A.4}$$

denote the omitted mass under dense softmax and the output obtained by renormalizing over the omitted keys. Then

$$O_i^{\text{dense}} = (1 - p_{U,i}) O_i^{\text{sp}} + p_{U,i} O_{U,i}, \tag{A.5}$$

and the exact attention output residual is

$$\boxed{R_i = O_i^{\text{dense}} - O_i^{\text{sp}} = p_{U,i} \left(O_{U,i} - O_i^{\text{sp}} \right).} \tag{A.6}$$

Sparse execution provides O_i^{sp} but does not evaluate the omitted interactions needed for $p_{U,i}$ and $O_{U,i}$. We fit an affine map from sparse outputs to measured residuals within the current call.

$$x_i = O_i^{\text{sp}} \in \mathbb{R}^d, \qquad R_i \approx b + x_i B. \tag{A.7}$$

This empirical model does not assume that O_i^{sp} uniquely determines the omitted statistics. The oracle analysis measures how much of the residual this affine family can represent under each partition.

A.2 Exact Probe Selection Stratified by Query Group

We fit the affine map for each query head in the current attention call. The fit therefore adapts to the input and routing pattern at that layer and denoising step without an offline calibration dataset. We evaluate $M = 64 \ll N_q$ probe rows against all keys. Probes add M/N_q to the executed pair density. This fraction ranges from 0.054% to 0.145% for the evaluated sequence lengths.

Probe selection uses the query groups and their response coordinates without inspecting dense residuals. For each nonempty group G_a^Q define

$$c_a = \frac{1}{|G_a^Q|} \sum_{i\in G_a^Q} \phi_Q(q_i), \qquad d_i = \left\| \phi_Q(q_i) - c_a \right\|_2. \tag{A.8}$$

We order rows within each group by increasing d_i and break ties by original row index. We order nonempty groups by decreasing size and break ties by group index. Each traversal takes the next row from every group in this order and skips exhausted groups. Selection stops after M distinct probes. With at least M nonempty groups this selects the row nearest the centroid from each of the M largest groups.

Algorithm A.1 Exact Probe Selection Stratified by Query Group

Require: Query partition $\mathcal{G}^Q$ of $\{1, \dots, N_q\}$ with coordinates $\{\phi_Q(q_i)\}_{i=1}^{N_q}$ and budget $1 \le M \le N_q$
Ensure: M distinct probe indices $\mathcal{P}$ and diagonal weights W with unit mean

$\mathcal{A} \leftarrow \{a : |G_a^Q| > 0\}$ ordered by decreasing $|G_a^Q|$ then increasing a
for all $a \in \mathcal{A}$ **do**
  $c_a \leftarrow |G_a^Q|^{-1} \sum_{i \in G_a^Q} \phi_Q(q_i)$
  $d_i \leftarrow \|\phi_Q(q_i) - c_a\|_2$ for each $i \in G_a^Q$
  $\pi_a \leftarrow$ indices in G_a^Q sorted lexicographically by (d_i, i)
end for
$\mathcal{P} \leftarrow \varnothing$ and $t \leftarrow 1$
while $|\mathcal{P}| < M$ **do**
  for all $a \in \mathcal{A}$ in the specified order **do**
    if $t \le |G_a^Q|$ **then**
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{\pi_a(t)\}$
      if $|\mathcal{P}| = M$ **then**
        break
      end if
    end if
  end for
  $t \leftarrow t + 1$
end while
for all $a \in \mathcal{A}$ with $m_a = |\mathcal{P} \cap G_a^Q| > 0$ **do**
  $w_p \leftarrow |G_a^Q|/m_a$ for each $p \in \mathcal{P} \cap G_a^Q$
end for
$\bar{w} \leftarrow M^{-1} \sum_{p \in \mathcal{P}} w_p$
$W \leftarrow \mathrm{diag}(w_p/\bar{w})_{p \in \mathcal{P}}$ in the probe matrix row order
return $\mathcal{P}, W$

Exact attention on each probe $p \in \mathcal{P}$ provides

$$R_p = O_p^{\mathrm{dense}} - O_p^{\mathrm{sp}}. \tag{A.9}$$

The estimator is fitted from the pairs $\{(x_p, R_p)\}_{p \in \mathcal{P}}$.

Group coverage weights. Let $m_a = |\mathcal{P} \cap G_a^Q|$ be the number of probes selected from query group a. Each probe $p \in G_a^Q$ receives weight

$$w_p = \frac{|G_a^Q|}{m_a}. \tag{A.10}$$

Each represented group has $m_a > 0$ and total raw weight $\sum_{p \in \mathcal{P} \cap G_a^Q} w_p = |G_a^Q|$. Selecting more probes from a group therefore does not increase its total raw weight. Groups without probes contribute no observations. We normalize the weights to unit mean to form W as in Section 4.2. These deterministic coverage weights do not correct for random sampling.

A.3 Weighted Affine Fitting and Residual Prediction

For the probe statistics we normalize coverage weights to unit sum.

$$\alpha_p = \frac{w_p}{\sum_{t \in \mathcal{P}} w_t}. \tag{A.11}$$

The weighted feature and residual means are

$$\mu_x = \sum_{p \in \mathcal{P}} \alpha_p x_p, \qquad \mu_R = \sum_{p \in \mathcal{P}} \alpha_p R_p. \tag{A.12}$$

Define the elementwise feature scale

$$s_x = \left[\sum_{p \in \mathcal{P}} \alpha_p (x_p - \mu_x)^{\odot 2} + \epsilon \right]^{1/2}, \tag{A.13}$$

where $\odot 2$ denotes elementwise squaring. The standardized feature and centered residual are

$$\bar{x}_p = (x_p - \mu_x) \oslash s_x, \qquad \bar{R}_p = R_p - \mu_R, \tag{A.14}$$

where $\oslash$ denotes elementwise division. Stacking the probe rows gives $\bar{X}_{\mathcal{P}} \in \mathbb{R}^{M\times d}$ and $\bar{R}_{\mathcal{P}} \in \mathbb{R}^{M\times d}$.

For the regression objective we normalize coverage weights to unit mean.

$$\widetilde{w}_p = \frac{M w_p}{\sum_{t\in\mathcal{P}} w_t}, \qquad W = \operatorname{diag}\left(\{\widetilde{w}_p\}_{p\in\mathcal{P}}\right). \tag{A.15}$$

This preserves relative weights and removes their common scale from the balance between the fitting loss and ridge penalty.

Weighted ridge fit. We fit the centered residual from the standardized sparse output.

$$B_\lambda = \arg\min_B \left\| W^{1/2}\left(\bar{R}_{\mathcal{P}} - \bar{X}_{\mathcal{P}} B\right)\right\|_F^2 + \lambda \|B\|_F^2. \tag{A.16}$$

Its closed form is

$$B_\lambda = \left(\bar{X}_{\mathcal{P}}^\top W \bar{X}_{\mathcal{P}} + \lambda I\right)^{-1} \bar{X}_{\mathcal{P}}^\top W \bar{R}_{\mathcal{P}}. \tag{A.17}$$

The implementation solves the regularized normal equations with a linear solver rather than explicitly forming the inverse.

Residual prediction. For an unprobed row i we use the same probe statistics to obtain $\bar{x}_i = (x_i - \mu_x) \oslash s_x$. The predicted residual is

$$\widehat{R}_i = \mu_R + \bar{x}_i B_\lambda. \tag{A.18}$$

The centered prediction lies in the span of the centered probe residuals. The default estimator uses no output truncation or correction norm cap. It requires neither a residual SVD nor an output projection. We evaluate output rank truncation only in Appendix D.3. The final output is

$$\widehat{O}_i = \begin{cases} O_i^{\text{dense}}, & i \in \mathcal{P}, \\ O_i^{\text{sp}} + \widehat{R}_i, & i \notin \mathcal{P}. \end{cases} \tag{A.19}$$

B Response Coupled Partitioning and Sparse Execution

B.1 Key Response Coordinates for Paired K/V Groups

For each K/V head we form $Q_s \in \mathbb{R}^{M_s\times d}$ from query rows at deterministic evenly spaced token positions in the current call. The key response metric is

$$M_K = \frac{1}{M_s} Q_s^\top Q_s. \tag{B.1}$$

Let $\Delta k = k_j - k_\ell$ for two keys and let q_i denote row i of Q_s. Their average squared response difference is

$$\begin{aligned} \frac{1}{M_s}\sum_{i=1}^{M_s} (q_i \Delta k^\top)^2 &= \Delta k \left(\frac{1}{M_s}\sum_{i=1}^{M_s} q_i^\top q_i\right) \Delta k^\top \\ &= \Delta k \left(\frac{Q_s^\top Q_s}{M_s}\right) \Delta k^\top = \Delta k M_K \Delta k^\top. \end{aligned} \tag{B.2}$$

The metric therefore compares responses to the sampled queries before softmax. Dividing this quantity by d gives the average squared difference between logits scaled by $1/\sqrt{d}$. We retain the leading r_K eigendirections of M_K and normalize their eigenvalues to unit mean. Scaling each direction by the square root of its normalized eigenvalue gives $F_K \in \mathbb{R}^{d\times r_K}$. The key response coordinate is

$$\phi_K(k_j) = k_j F_K \in \mathbb{R}^{r_K}. \tag{B.3}$$

Euclidean distance in these coordinates is equivalent to a low-rank Mahalanobis-type distance in the original key space, with response-induced metric $F_K F_K^\top$. Clustering $\phi_K(k_j)$ with k means produces paired K/V groups. Each value v_j inherits the assignment of its paired key k_j. Values do not enter the clustering features.

B.2 Query Response Coordinates from K/V Groups

Here C_k counts nonempty K/V groups and may be smaller than the requested cluster count. We exclude empty groups from the centroid mean and the query response metric. We compute each centroid $\bar{k}_b$ in the original key space and subtract the uniform centroid mean μ_K. The centered centroids form the rows of $\widetilde{K} \in \mathbb{R}^{C_k \times d}$. For query q_i we have

$$\frac{q_i(\bar{k}_b - \mu_K)^\top}{\sqrt{d}} = \frac{q_i \bar{k}_b^\top}{\sqrt{d}} - \frac{q_i \mu_K^\top}{\sqrt{d}}. \tag{B.4}$$

The subtracted term is shared by every group. Centering therefore preserves relative logits and their softmax over groups. The query response metric is

$$M_Q = \frac{\widetilde{K}^\top \widetilde{K}}{C_k d}. \tag{B.5}$$

For two queries q_i and q_j we obtain

$$(q_i - q_j) M_Q (q_i - q_j)^\top = \frac{1}{C_k d} \left\| \widetilde{K}(q_i - q_j)^\top \right\|_2^2. \tag{B.6}$$

Distance under M_Q is the average squared difference between the centered logit profiles over groups. Each nonempty group receives equal weight regardless of its token count.

Let $F_Q \in \mathbb{R}^{d \times r_Q}$ be obtained from the leading r_Q eigendirections of M_Q using the same eigenvalue weighting as above. The normalized query response coordinate is

$$\begin{aligned} \phi_Q(q_i) &= \mathrm{RMSNorm}(q_i F_Q) \\ &= \frac{q_i F_Q}{\max\left(\sqrt{\|q_i F_Q\|_2^2 / r_Q}, \epsilon\right)} \in \mathbb{R}^{r_Q}. \end{aligned} \tag{B.7}$$

Before RMS normalization, Euclidean distance in the projected coordinates defines a low-rank Mahalanobis-type metric in query space. RMS normalization then makes clustering depend primarily on response direction. The floor $\epsilon = 10^{-6}$ stabilizes the denominator. RMS normalization has no learned gain or bias. It does not preserve softmax concentration. We retain the original queries for routing and attention. Clustering $\phi_Q(q_i)$ gives the query partition. These coordinates also stratify probe selection but do not enter residual regression. The construction proceeds from K/V groups to query groups without alternating refinement.

B.3 Routing Score and Executed Pair Budget

Values enter the routing score but not the clustering features. Let $\bar{q}_a$ be the query group centroid and let $\bar{k}_b$ and $\bar{v}_b$ be the key and value centroids of G_b^{KV} in their original spaces. We use the SVG-EAR value error score (Zhou et al., 2026b).

$$s_{ab} = \frac{1}{|G_b^{KV}|} \sum_{j \in G_b^{KV}} \left\| e^{\bar{q}_a \bar{k}_b^\top / \sqrt{d}} \bar{v}_b - e^{\bar{q}_a k_j^\top / \sqrt{d}} v_j \right\|_2^2. \tag{B.8}$$

For each query group we rank nonempty K/V groups by decreasing score. We retain the shortest prefix that reaches the common key token budget per query from Section 4.3. Each selected cell costs $|G_a^Q||G_b^{KV}|$ exact interactions. We evaluate selected cells exactly and renormalize attention over their keys. Centroids determine routing scores but do not supply approximate output contributions.

After routed sparse attention we evaluate probes against all keys. Writing the probe fraction as $m = M/N_q$ gives total executed pair density

$$\rho_{\mathrm{exec}} = \rho_{\mathrm{route}} + m. \tag{B.9}$$

For target $\rho^\star$ the router reserves probe cost and uses routed budget $\rho^\star_{\mathrm{route}} = \rho^\star - m$. Selecting whole cells can make realized density differ from the target. We report realized density for each configuration.

C Implementation and Evaluation Details

C.1 GPU Implementation

We cluster on the GPU and fuse query response projection with RMS normalization. Initial k means centers are sampled uniformly with replacement from feature rows. Later calls initialize from the previous centroids in the original space projected into the current response coordinates. An FP32 eigendecomposition initializes the metric factors. Later calls update them with two subspace iterations initialized from the previous factors and using Cholesky factorization. At the evaluated production shape the fused query projection reduces latency from 1.6170 ms to 0.5064 ms with relative feature error 3.68×10^{-4} against the reference implementation. Small batched GPU linear algebra operations fit and apply the residual correction. The default estimator uses no residual SVD or output subspace projection.

C.2 Shared Hyperparameters

All experiments use one NVIDIA H100 GPU with BF16 model activations. Metric construction and centroid statistics use FP32 as does factorization. The optimized query projection uses TF32 multiplication with FP32 accumulation and normalization. Models share the method settings except for the total executed pair target.

Table C.1: Method configuration used in the evaluated models.

Parameter	**Value**
Requested query clusters C_q	300
Requested K/V clusters C_k	1000
Metric query sampling budget per query head	2048
Initial / subsequent k means iteration limits	$25/2$
k means convergence tolerance	10^{-4}
Key response rank r_K	48
Query response rank r_Q	64
Exact probes per query head M	64
Output subspace truncation	None
Ridge coefficient λ	1
Target ρ_{exec} for HunyuanVideo / Wan2.2 / Cosmos-Predict2.5	0.22
Target ρ_{exec} for Cosmos3-Nano	0.26

For GQA we concatenate query rows from the G heads sharing a K/V head. We select up to $2048G$ evenly spaced rows to construct the shared key response metric and use all rows when fewer are available.

Table C.2: Probe fractions for the evaluated sequence lengths with $M = 64$.

Model	N_q	M/N_q
HunyuanVideo	118,800	0.054%
Wan2.2	75,600	0.085%
Cosmos-Predict2.5	84,480	0.076%
Cosmos3-Nano	44,160	0.145%

C.3 Timing Protocol

Dense and sparse runs use the same checkpoints and conditioning inputs. Random seeds and sampling schedules also match. We hold inference steps and guidance settings fixed along with resolution and frame count. Executed pair density counts all exact interactions between queries and keys including probes. Reported latency includes response coordinate construction and clustering together with routing and permutation. It also includes sparse attention and exact probes as well as residual fitting and correction followed by output restoration.

Table D.1: **Query RMS normalization ablation.** Mean normalized attention output error $\|O_i^{\text{sp}} - O_i^{\text{dense}}\|_2 / (\|O_i^{\text{dense}}\|_2 + 10^{-8})$. Lower is better. Bold entries mark the lowest error in each row.

Model	Per query RMS	Global RMS	Per query RMS + log magnitude
HunyuanVideo	**0.073**	0.074	0.083
Wan2.2	**0.108**	0.109	0.166
Cosmos-Predict2.5	**0.761**	0.762	0.777
Cosmos3-Nano	**0.331**	0.332	0.337

D Additional Ablations and Sensitivity Analyses

D.1 Matched Retained Mass and Output Error

Figure D.1 extends O2 to HunyuanVideo, Wan2.2, and Cosmos-Predict2.5. Each point summarizes one attention head at a fixed input, layer, and generation step. Its coordinates are the mean retained attention mass and p99 normalized output error across queries.

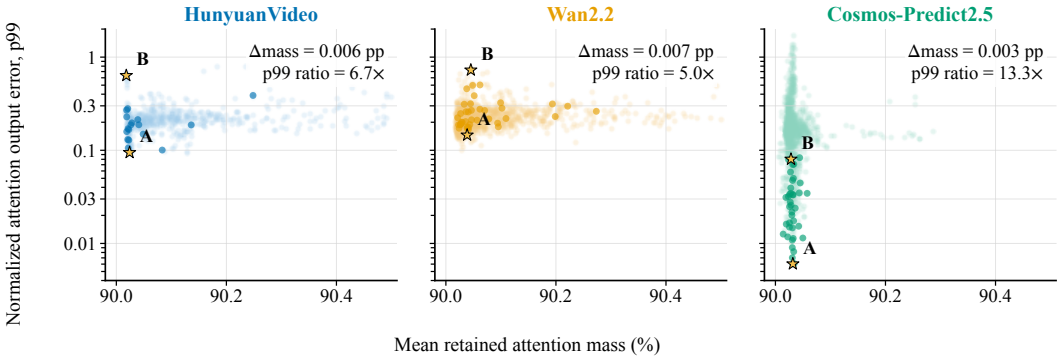

Figure D.1: **Matched retained mass across models.** Dark points are heads from the same input, layer, and generation step as Cases A and B. Stars mark two heads with nearly identical retained mass. The logarithmic y axis shows multiplicative differences in p99 normalized output error.

Within each panel, Cases A and B differ by at most 0.007 percentage points in retained mass, while their p99 errors differ by $5.0\times$ to $13.3\times$.

D.2 Query RMS Normalization

We compare RMS normalization per query with a shared RMS scale per head and with RMS normalization per query plus a log magnitude coordinate. The shared scale preserves relative query magnitudes. For the added coordinate we center and standardize the log RMS magnitude across query rows and multiply it by $\sqrt{r_Q}$ with magnitude weight one.

All variants share K/V groups and query projections as well as clustering seeds and evaluation rows. We use 32 captured calls per model with all query heads and 2048 evenly spaced evaluation queries per head. Requested group counts are $C_q = 300$ and $C_k = 1000$ with coordinate ranks $r_K = 48$ and $r_Q = 64$. We load activations in BF16 and replay attention in FP32. Clustering starts from fresh centers with at most 25 iterations. Each variant uses the same routing rule with a budget calibrated toward 22% executed pair density. We pool query errors across calls and heads for renormalized hard drop without residual repair.

RMS normalization per query gives the lowest mean error across all four models in Table D.1 with a small improvement over global RMS. The table includes all cases even when discrete routing budgets prevent the target density from being reached. For Cosmos-Predict2.5 we also evaluate 1247 of the 1280 cases where all variants lie within 0.2 percentage points of the target and of each other. The ordering is unchanged. Mean error is 0.8227 for RMS per query and 0.8247 for global RMS. Adding log magnitude gives 0.8423.

Table D.2: **Sensitivity to probe count and estimator settings.** Entries report mean normalized attention output error on the same unprobed query rows across settings. Lower is better. Each panel varies one parameter while holding the others fixed.

(a) Probe count with fixed $\lambda = 1$ and no output truncation

Model	$M = 16$	$M = 32$	$\mathbf{M = 64}$	$M = 128$
HunyuanVideo	0.037	0.034	**0.030**	0.027
Wan2.2	0.080	0.071	**0.063**	0.057
Cosmos-Predict2.5	0.118	0.092	**0.075**	0.065
Cosmos3-Nano	0.097	0.082	**0.076**	0.074

(b) Output rank truncation with fixed $M = 64$ and $\lambda = 1$

Model	$r = 4$	$r = 8$	$r = 16$	$r = 32$	**No truncation**
HunyuanVideo	0.034	0.032	0.031	0.031	**0.030**
Wan2.2	0.072	0.068	0.066	0.065	**0.063**
Cosmos-Predict2.5	0.113	0.089	0.079	0.077	**0.075**
Cosmos3-Nano	0.089	0.081	0.078	0.077	**0.076**

(c) Ridge strength with fixed $M = 64$ and no output truncation

Model	$\lambda = 0.001$	$\lambda = 0.01$	$\lambda = 0.1$	$\boldsymbol{\lambda = 1}$
HunyuanVideo	0.039	0.038	0.034	**0.030**
Wan2.2	0.086	0.080	0.072	**0.063**
Cosmos-Predict2.5	0.095	0.086	0.080	**0.075**
Cosmos3-Nano	0.095	0.090	0.084	**0.076**

Bold headers and entries mark the reference configuration. Requested ranks in panel (b) are capped at the observed centered probe residual rank of at most $M - 1$. No truncation denotes ridge regression without output subspace truncation. Increasing M in panel (a) also increases total executed pair density.

D.3 Probe and Estimator Sensitivity

Table D.2 evaluates the affine estimator using sparse outputs as its only features. It covers 32 captured calls per model and all query heads for a total of 4352 cases. The reference uses $M = 64$ and $\lambda = 1$ without output truncation. For the rank ablation we project the centered ridge prediction onto the leading r right singular vectors of $W^{1/2}\bar{R}_{\mathcal{P}}$. The residual mean μ_R is added separately without truncation. We use response coupled partitions and nested probes drawn from full query population. Each head has 2048 deterministic evaluation candidates. We exclude all 128 candidate probes to keep same unprobed evaluation rows across settings and pool errors across captures and heads.

We load BF16 activations and replay attention in FP32. Regression and residual subspace fitting use FP64. Feature standardization uses $\sqrt{\text{weighted variance} + 10^{-6}}$. Sparse base remains fixed for each model at the route calibrated to a 22% total executed pair target with $M = 64$. Varying M therefore changes total cost.

D.4 Correction Family Comparison

Table 4 compares correction by the weighted mean residual with three affine variants. Each affine variant predicts $\mu_R + \bar{x}_i B$. The scalar variant uses $B = aI$ and the diagonal variant uses $B = \mathrm{diag}(b)$. Full weighted ridge leaves B unrestricted. All three minimize the same weighted squared error with penalty $\lambda\|B\|_F^2$ and $\lambda = 1$. The scalar penalty is therefore $\lambda d a^2$. We do not penalize the residual mean or truncate the output.

All variants share sparse outputs and unprobed evaluation rows. They use the same $M = 64$ probes with identical coverage weights and feature standardization. Evaluation rows and numerical precision follow Appendix D.3.

D.5 Oracle Affine Residual Representability

We test O3 by comparing semantic and response coupled partitions using the same affine features $X_g = [\mathbf{1}, O_g^{\mathrm{sp}}]$. For each captured call and head we compute dense and sparse outputs on 2048 evenly spaced queries and fit the predictor to all of them. This oracle fit measures representability on the fitted rows before estimating a correction from a limited probe set.

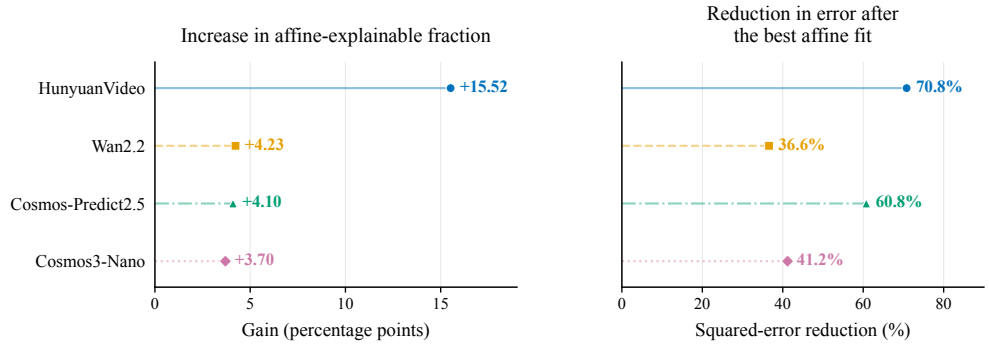

Figure D.2: Oracle affine residual representability under response coupled versus semantic partitioning. The left panel shows the increase in the fraction of residual squared norm explained by the best affine fit. The right panel shows the reduction in squared error remaining after fitting. We fit all 2048 evaluated queries per head and sum energies across cases before computing ratios. Cosmos-Predict2.5 uses the subset matched by density.

We use 32 captured calls per model with all query heads. This gives 768 cases for HunyuanVideo and 1280 for Wan2.2. Cosmos-Predict2.5 has 1280 cases and Cosmos3-Nano has 1024. Both partitions request $C_q = 300$ and $C_k = 1000$ and use the same routing rule with renormalized hard drop. Routing targets 22% total executed pair density and reserves the cost of 64 exact probes as in the repair configuration. This reservation does not restrict the oracle fit. The Cosmos-Predict2.5 figure uses 1248 cases matched by density and excludes cases where discrete routing budgets prevent matching. The explained fraction increases by 4.10 percentage points in this subset and by 4.08 across all 1280 cases. The reduction in remaining squared error rounds to 60.8% for both populations. Other models use all cases.

We load activations in BF16 and compute attention outputs in FP32. The oracle projection uses FP64 and an SVD of centered sparse outputs with scaled columns. The relative singular value tolerance is 10^{-10}. The residual mean supplies the affine intercept.

Figure D.2 reports the gain in explained residual energy in percentage points and the reduction in remaining squared error relative to semantic partitioning. We sum squared errors across the same cases before computing fractions and reductions.

Interpretation. Response coupled partitioning increases the fraction explained by the affine fit and reduces remaining squared error across all four models. The remaining error depends on both the initial residual magnitude and the fraction explained. These results support the capacity of the affine correction. Table 2 separately evaluates prediction on unprobed rows after fitting from a small probe set.

D.6 Further Discussion

We examine repair reliability and alternative estimators with $M = 64$ and $\lambda = 1$. Comparisons use identical sparse outputs and unprobed evaluation rows within each case.

Does repair ever worsen individual heads? Repair improves aggregate accuracy but can increase error for an individual head on a particular call. Mean normalized error increases in 13.05% of HunyuanVideo cases and 11.41% of Wan2.2 cases. The rates are 1.41% for Cosmos-Predict2.5

and 13.98% for Cosmos3-Nano. Each case is one head at a sampled layer and diffusion step. For Cosmos3-Nano layer 20 the mean error increases by 2.09% after pooling evaluated heads and steps. These are rates of worsening attention error and do not measure generation failures.

Does a small nonlinear predictor improve reconstruction? We compare full weighted ridge with a GELU predictor with one hidden layer of width 32. We train it for 200 Adam steps under five initialization seeds using the same probes and weights. The nonlinear predictor has higher mean error on all four models. For Cosmos3-Nano it lowers p99 error from 0.447 to 0.430 but increases mean error by 8.26%. Fitting takes 136 to 139 ms per head in the FP32 H100 benchmark compared with approximately 0.46 ms for ridge. These results favor ridge over the tested nonlinear configuration with 64 probes.

E More Qualitative Results

Figures E.1 through E.4 provide additional qualitative comparisons between dense attention and SparsePR.

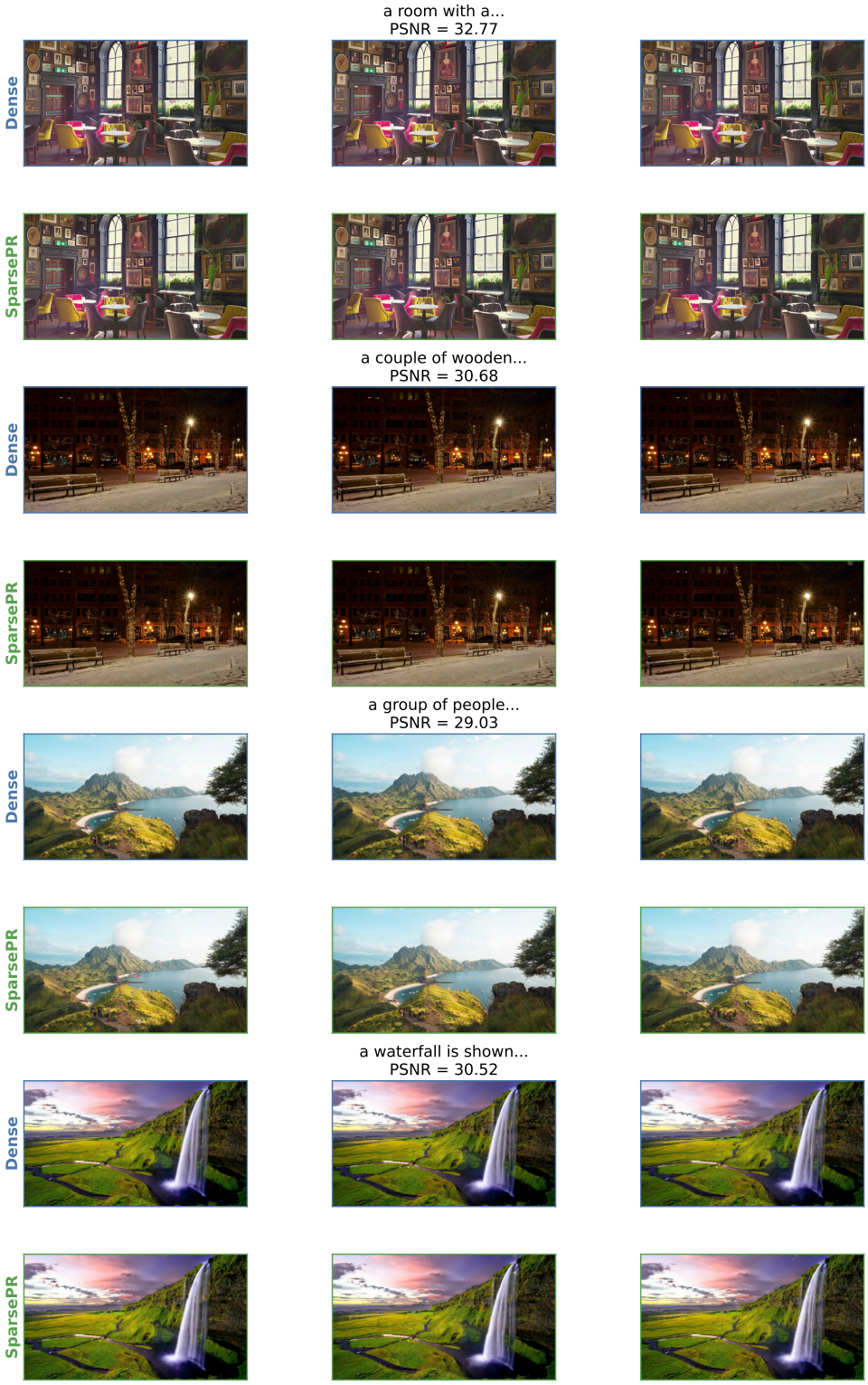

Figure E.1: Dense and SparsePR outputs for Cosmos3-Nano-16B on VBench++.

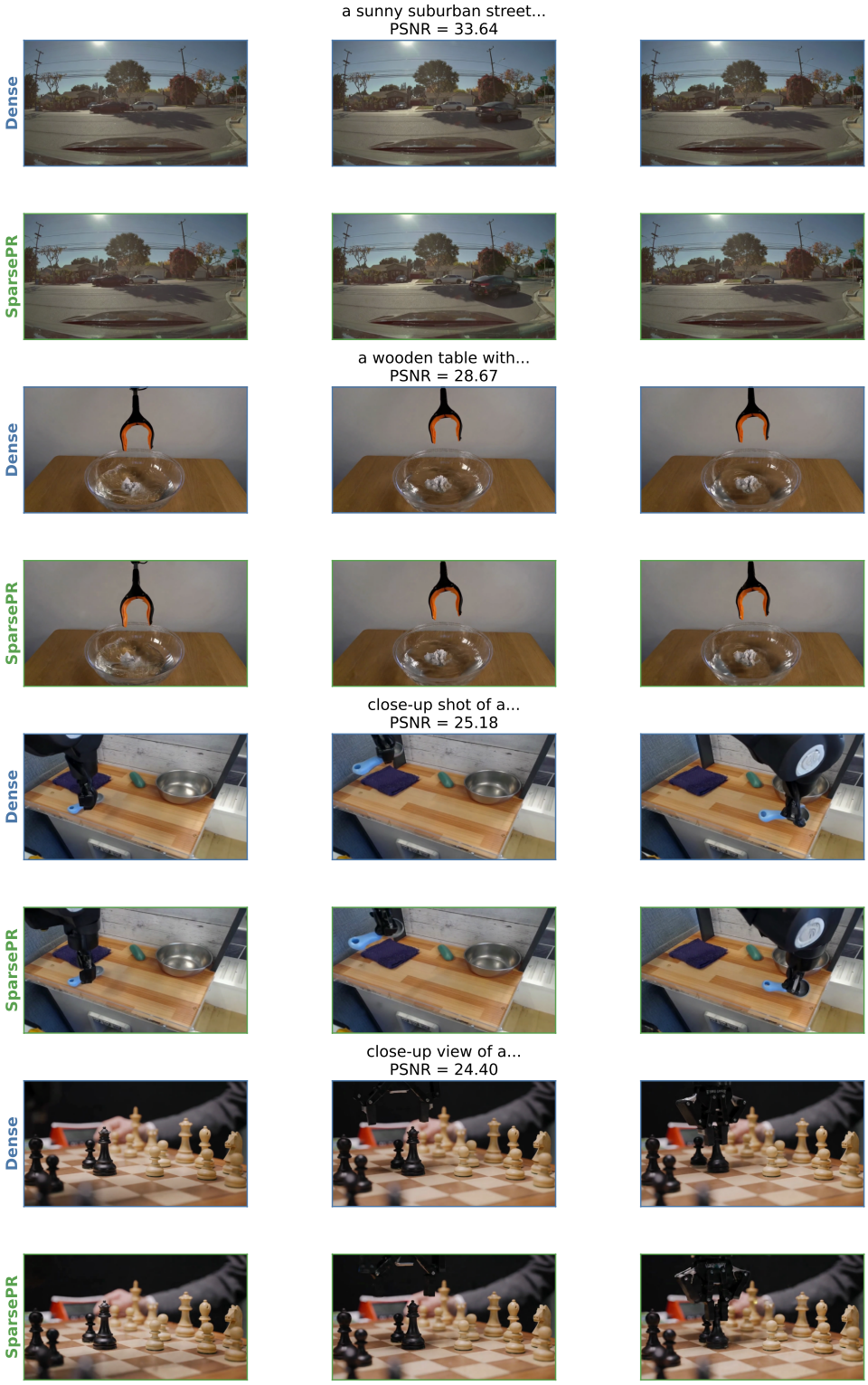

Figure E.2: Dense and SparsePR outputs for Cosmos3-Nano-16B on PBench.



Figure E.3: Qualitative comparison on HunyuanVideo-13B.

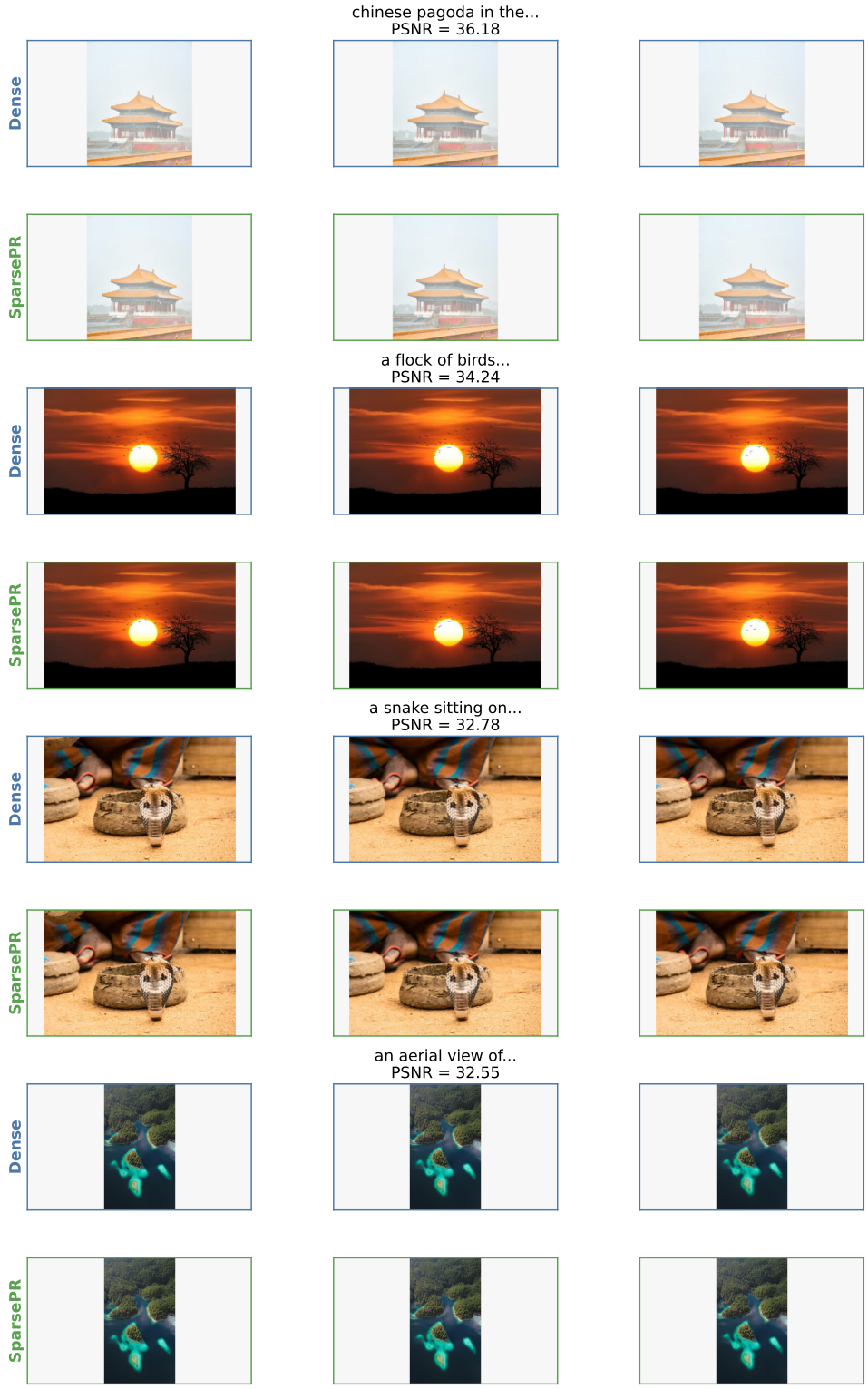

Figure E.4: Qualitative comparison on Wan2.2-I2V-A14B.