INTRA: INTERLEAVED NON-CONTIGUOUS TOKEN SPARSE ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformers achieve strong performance across modalities but are bottlenecked by the quadratic cost of attention. Sparse attention can reduce this cost, but effective sparsity must preserve useful long-range communication and map well to GPU hardware. We introduce **INTRA**–**I**nterleaved **N**on-contiguous **T**oken spa**R**se **A**ttention–a hardware-aware static sparse attention design based on a relay-communication view under Computational Query Set (CQS) constraints. Rather than directly covering all token-to-token interactions within each sparse layer, INTRA uses complementary sparse groups to provide short mediated communication paths across layers. Scatter forms relay groups that sample broadly across local neighborhoods, while Gather reconnects these groups in subsequent layers. Under a partition model, this structure guides CQS-compatible sparse patterns with global reachability. Our CQS-aware FlashAttention-style kernel computes structured indices at runtime, loads selected non-contiguous tokens, and packs them densely for computation. On an A100, INTRA achieves a $1.53\times$ speedup in 2K `FLUX.1-dev` generation, with better measured image quality than evaluated CLEAR configurations after LoRA self-distillation. On LLaMA-3.1-8B, it improves long-context prefill efficiency with strong teacher-forced retrieval performance under matched LoRA adaptation. INTRA achieves significant speedups over MoBA and density-constrained NSA in terms of attention computation.

1 INTRODUCTION

Transformers excel across domains such as NLP and image generation, but their attention cost scales quadratically with sequence length, creating a bottleneck for long contexts and high-resolution generation. For instance, `FLUX.1-dev` (Black Forest Labs, 2024) requires nearly a minute to produce a 2048×2048 image on a single A100, precluding real-time generation.

Sparse attention reduces full-attention overhead (Liu et al., 2025; Jiang et al., 2024). We distinguish input-dependent (**dynamic**) from input-independent (**static**) connectivity separately from whether a method requires training or adaptation. INTRA uses static connectivity to avoid runtime pattern selection and evaluates adaptation of pretrained models to GPU-friendly sparse patterns.

Sparsity changes communication as well as FLOPs. Many modalities have coherent local neighborhoods, such as image patches, spatial regions, or short contiguous token segments. Local attention can mix nearby information within these neighborhoods; the remaining challenge is communication between neighborhoods under hardware constraints. A partition-based sparse layer permits interactions only within groups. Covering all dense edges would require many such layers, motivating relay paths that connect neighborhoods through successive groups.

Efficient execution constrains this design space. CLEAR (Liu et al., 2025) uses circular local sparsity, and NATTEN (Hassani et al., 2023) uses sliding windows. Fine-grained masks can leave partially used computational blocks (Figure 2). Computing whole blocks before masking unused scores diminishes realized speedups relative to nominal FLOP reductions. We formalize a hardware-oriented design class through Computational Query Sets (CQSs): query groups with structured key sets and local-to-global index mappings aligned with FlashAttention-style query blocks.

INTRA combines two CQS-compatible patterns. Under equal-size partitions, Scatter groups same-position tokens across neighborhoods, giving each group one representative per neighborhood. Gather

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.

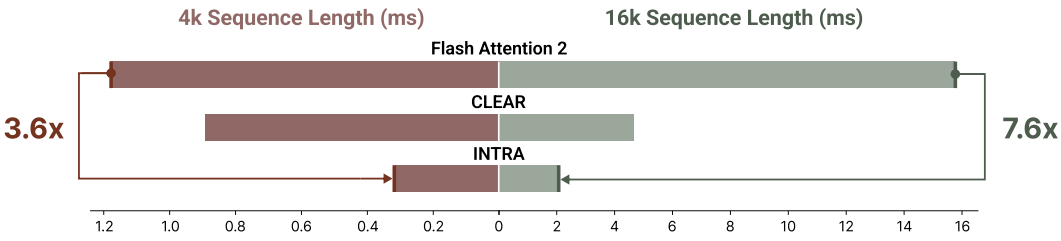


Figure 1: The figure reports the relative speedup achieved by INTRA compared to CLEAR and FlashAttention 2 on 4K and 16K input lengths, evaluated on an A100 GPU. Note that the axis density differs between the left and right sides of the figure to accommodate varying inference times.

reconnects these channels. If every Scatter group intersects every Gather group, two layers permit graph-theoretic reachability between any source and target. This motivates complementary patterns for efficient sparse communication.

Our contribution is the joint study of structured static attention, GPU execution, and pretrained-model adaptation. Building on factorized and local–global attention (Child et al., 2019; Tu et al., 2022), we (1) formulate coverage and overlap objectives within the Intra Sparse Pattern Design (ISPD) Principle and its CQS constraints; (2) implement selected Scatter/Gather patterns using runtime indexing, dense shared-memory packing, and scatter-back; and (3) evaluate their quality–latency trade-offs after adaptation.

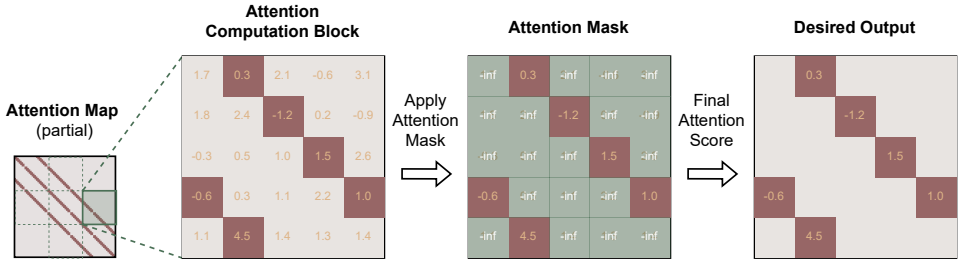


Figure 2: Low-level implementation of blockwise sparsity for handling mixed blocks. A mix block is the block that partially preserves attention scores, as shown on the left side of the figure.

INTRA delivers $1.53\times$ faster 2K `FLUX.1-dev` generation after LoRA self-distillation with better measured image metrics than evaluated CLEAR configurations. Wan2.2-TI2V-5B generation is $1.43\times$ faster end to end at 704×1280 with 121 frames (Table 19). On LLaMA-3.1-8B, INTRA accelerates SCROLLS (Shaham et al., 2022) prefill relative to MInference (Jiang et al., 2024) and XAttention (Xu et al., 2025). Separate 32-layer attention timings at 4K–64K show 3.6–$8.4\times$ speedups over MoBA and 7.3–$10.4\times$ over density-constrained NSA (Table 4b).

2 Related Works

2.1 Static Sparse Attention

Static sparse attention uses fixed, input-independent patterns. Sparse Transformer (Child et al., 2019) combines local and strided factorizations with short communication paths and interleaves patterns across layers. MaxViT (Tu et al., 2022) combines block and grid attention for local and distributed global interactions. These are close precedents for Scatter/Gather. Sliding-window methods (Jiang et al., 2023; Hassani et al., 2023; Liu et al., 2025), Longformer, and BigBird (Beltagy et al., 2020; Zaheer et al., 2020) provide other local/global constructions. INTRA focuses on CQS-compatible indexing, dense-tile execution, and adaptation of complementary patterns in pretrained models.

2.2 Dynamic Sparse Attention

Dynamic sparse methods adapt connectivity to the input or model state. MInference (Jiang et al., 2024), FlexPrefill (Lai et al., 2025), and XAttention (Xu et al., 2025) select heads, blocks, or patterns at runtime. NSA, MoBA, and H2O (Yuan et al., 2025; Lu et al., 2025a; Zhang et al., 2023) also use input-dependent selection or routing, with different training requirements. INTRA evaluates adapted static patterns that avoid runtime selection and target simpler GPU kernels.

2.3 Other Image Acceleration Methods

Token pruning and merging methods reduce the effective sequence length for vision transformers by removing or combining tokens (Rao et al., 2021; Kong et al., 2022; Bolya et al., 2023; Lu et al., 2025b). These methods are orthogonal to INTRA, which keeps the token set but changes the sparse communication pattern and kernel implementation.

2.4 Sparse Video Generation

Sparse video methods reduce spatiotemporal attention through hardware-aware regions, shifted windows, or spatial/temporal head specialization (Zhang et al., 2025; Liu et al., 2022; Xi et al., 2025). INTRA formulates a CQS-compatible communication principle applicable across architectures. Our image and language workloads cover high-resolution generation and long-context prefill; the 16K-token setting is also comparable to smaller video models, such as Wan 1.3B at 480p (Team Wan et al., 2025). We additionally evaluate video transfer on Wan2.2-TI2V-5B (Appendix G.8).

3 Hardware-Efficient Sparse Communication

3.1 Computational Query Sets and ISPD Constraints

We study a class of sparse attention patterns compatible with FlashAttention-style execution. A **Computational Query Set** (CQS) is a group of query tokens evaluated together, with structured local-to-global mappings for selecting the corresponding key/value tokens. A CQS may use query-dependent key indices, as long as they are generated by structured mappings that can be evaluated uniformly for the group. We call a sparse pattern **ISPD-compatible** if it satisfies:

1. **Coverage:** The union of all CQSs covers the query-token set. A query token may belong to more than one CQS.
2. **Index Mapping:** There exist structured functions f_q and f_k such that, for a given CQS index i^{cqs},
$$i_g^q = f_q(i_l^q, i^{\text{cqs}}), \quad i_g^k = f_k(i_l^k, i_l^q, i^{\text{cqs}}). \tag{1}$$
3. **Block Alignment:** Each CQS contains a number of queries approximately divisible by the query block size B; we use $B = 64$ for FlashAttention-style kernels.

The "approximately" qualifier only accounts for a final partial block when the sequence length is not divisible by B. Within this design class, intra-CQS computation avoids cross-thread-group synchronization: queries from the same CQS can be grouped, and the required non-contiguous K/V tokens can be packed contiguously for FlashAttention-style computation. Our Scatter and Gather patterns satisfy these constraints; their explicit f_q and f_k mappings are in Appendix C.

3.2 Sparse Attention as Communication

We model a sparse attention layer as a directed information-flow graph over tokens:

$$G^{(l)} = (V, E^{(l)}), \qquad (j, i) \in E^{(l)} \iff \text{query token } i \text{ attends to key token } j.$$

A path follows the direction in which token information can be used by later queries. A *partition-based* sparse layer divides tokens into groups and computes attention within each group. We use this

abstraction to reason about Scatter/Gather communication paths. We assume tokens are partitioned into local neighborhoods

$$\mathcal{P} = \{P_1, \ldots, P_T\}, \qquad P_t \cap P_{t'} = \varnothing \ (t \neq t'), \qquad \bigcup_{t=1}^{T} P_t = V.$$

Here P_t may be an image patch, spatial neighborhood, or short text segment. Our modeling premise is that local attention already provides useful intra-neighborhood mixing; the sparse-design problem is communication between neighborhoods while staying inside the ISPD-compatible design class. We call the sparse groups used to move information between neighborhoods *relay groups*.

4 Scatter-Gather Relay Design

4.1 Static Sparse Attention Under Unknown Long-Range Demand

Dynamic sparse attention can select important keys after observing the input or attention scores. This adaptivity incurs runtime pattern-selection costs and can introduce irregular memory access. INTRA instead fixes connectivity in advance, enabling simple local-to-global index mappings and structured token loading. A static pattern cannot know which long-range token pairs matter for a given input, head, or layer. It should therefore allocate communication opportunities across possible long-range interactions, rather than specialize to a single assumed attention pattern. We express this goal through a worst-case neighborhood-coverage objective.

4.2 From Direct Edges to Relay Paths

A sparse layer with per-query budget g can preserve at most ng directed token-token interactions, while dense attention contains n^2 possible interactions. When $g \ll n$, a static sparse mask cannot directly preserve all potentially important dense edges.

Two sparse layers can provide a mediated path for a missing direct interaction through a relay token:

$$j \to r \to i.$$

Our analysis counts the relay opportunities connecting source and target neighborhoods.

This observation guides INTRA: the sparse pattern should provide broad and balanced relay opportunities while remaining compatible with efficient CQS execution.

4.3 Locality as a Design Motivation

Nearby image patches often share visual structure, while short text segments provide coherent local context. This motivates separating two roles: local attention mixes nearby tokens, and non-local groups provide paths between neighborhoods. When an interaction involves distant neighborhoods, their local representatives are plausible relay carriers. The combinatorial objective formalizes this intuition by balancing coverage across neighborhood pairs.

4.4 Why Scatter Is Uniform and Non-Contiguous

Let $N = TR$ tokens be divided into $T \geq 2$ local neighborhoods, with integer $R \geq 1$:

$$P_1, \ldots, P_T, \qquad |P_t| = R.$$

A group of T tokens covers all T neighborhoods only by taking one representative from each; repeated samples from one neighborhood leave another uncovered.

Scatter avoids this failure mode. Writing tokens as (t, r), where t is the neighborhood index and r is the local index, Scatter forms

$$S_r = \{(t, r) : t = 1, \ldots, T\}, \qquad r = 1, \ldots, R.$$

Each Scatter group contains one token from every local neighborhood. Thus, every group is a global low-resolution view of the sequence or image.

Sampling distant neighborhoods makes Scatter non-contiguous. The R groups of size T compute $RT^2 = N^2/R$ attention pairs, retaining a $1/R$ fraction of dense pairs. At fixed R, the complexity is $O(N^2)$. Gather 0, the image variant using 16×16 local neighborhoods, instead uses $O(Nw)$ pairs for a fixed window of w keys per query.

4.5 Why Gather Is Needed

Scatter provides global relay groups but separates communication by local index. Tokens in S_r can exchange information across neighborhoods with other tokens in S_r, while different channels S_r and $S_{r'}$ remain disconnected. This matters because useful source-side and target-side representatives need not share the same local index: broad coverage within a channel alone cannot connect every source–target pair.

Gather is introduced to reconnect these Scatter channels. Let

$$\mathcal{G} = \{G_b\}_{b=1}^{R}$$

be the second partition, and define the Scatter-to-Gather overlap

$$X_{rb} = |S_r \cap G_b|.$$

If $X_{rb} = 0$, then Scatter channel S_r has no relay witness into Gather group G_b. A good Gather pattern should therefore make these overlaps as balanced as possible, avoiding weak or missing channel connections.

A simple balanced construction is cyclic:

$$G_b = \{(t, r) : b \equiv t + r \pmod{R}\}.$$

Its minimum overlap is $\lfloor T/R \rfloor$. Thus every Scatter group intersects every Gather group, enabling universal two-layer reachability in this abstract construction, only when $T \geq R$.

4.6 Design Summary

Within equal-size partitions, uniform Scatter maximizes minimum neighborhood-pair coverage, and balanced Gather maximizes minimum inter-partition overlap. These objectives motivate complementary patterns under a static communication budget.

Thus INTRA implements the following communication structure:

$$\text{local mixing} \;\rightarrow\; \text{uniform non-contiguous relay} \;\rightarrow\; \text{balanced relay reconnection.}$$

Appendix D proves the coverage and overlap results for the partition model. We next describe its image and language instantiations.

4.7 Mapping Back to INTRA

For `FLUX.1-dev`, INTRA interleaves Scatter and Gather 0 (Figure 3): Scatter groups same-position tokens across local neighborhoods, and Gather 0 reconnects groups through 16×16 neighborhoods. Each layer remains structured while successive patterns provide complementary relay paths. For LLaMA-3.1-8B retrieval, 4 of 32 layers use block-level sparse attention, one every eight layers. These layers select every second contiguous 256-token block, subject to causal masking. The remaining layers use sliding-window attention with a span of 575 tokens and 64 sink tokens (Xiao et al., 2024). Sparse attention is applied during prefill, followed by dense attention during autoregressive decoding.

For image generation, interleaved Scatter/Gather provides mediated global communication with ISPD-compatible layers. The next section describes the kernel's non-contiguous token loading and contiguous packing.

5 CQS-Aware Token-Sparse FlashAttention Kernel

For token-level Scatter/Gather patterns, INTRA implements a CQS-aware variant of FlashAttention (Dao, 2024). Each CQS is scheduled in query blocks of 64 tokens. Structured mappings f_q and

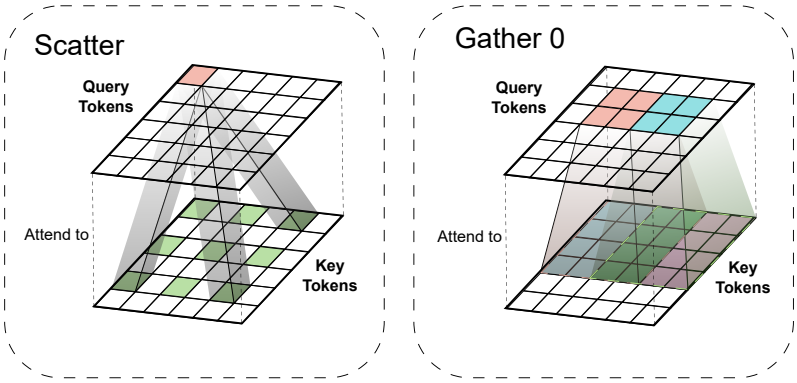


Figure 3: The Scatter and Gather 0 attention patterns used for image generation. Query and key tokens are arranged in a 2D feature map. Scatter groups tokens at the same local position across neighborhoods, while Gather 0 uses a 16×16 neighborhood spanning multiple Scatter groups.

f_k compute global positions at runtime; the kernel loads the selected Q, K, and V tokens, packs them into dense shared-memory tiles, and applies the FlashAttention-style online-softmax core. Outputs are scattered back using the query mapping. This avoids computing and masking unused scores in mixed blocks. Masking within the packed tiles supports full and causal attention.

Scatter and Gather 0 use strided cross-neighborhood access and two-dimensional local offsets, respectively. Gather 1 is the repeated-segment token grouping illustrated in Appendix A; all three share the kernel with different index mappings and underlie the token-level benchmarks. Language retrieval uses the block-level extension in Section 4.7, with selected 256-token blocks in global layers and sliding windows with sinks elsewhere. Appendix E profiles the token-level kernels' indexing, memory traffic, and computation.

6 Evaluation

We evaluate whether the CQS-compatible relay design is practical in both the `FLUX.1-dev` (Black Forest Labs, 2024) image generation model and the LLaMA 3 (Dubey et al., 2024) language model. INTRA changes how the model exchanges information between tokens, requiring adaptation; LoRA (Hu et al., 2022) suffices in our evaluated settings.

6.1 Experiments

Baseline INTRA provides broad communication through fixed sparse patterns. For image generation, we compare with CLEAR, an adapted static sparse-attention method, to evaluate pattern design and execution in the same pretrained-model setting.

For text generation, we compare with MInference (Jiang et al., 2024) and XAttn (Xu et al., 2025) on the long-context SCROLLS benchmark (Shaham et al., 2022).

Experiment Setup We fine-tune `FLUX.1-dev` using CLEAR's 10K self-generated image-text dataset. Following CLEAR, we sample 5k COCO (Lin et al., 2014) validation prompts and evaluate generated images using FID (Heusel et al., 2017; Seitzer, 2020), LPIPS (Zhang et al., 2018), and CLIPI (Radford et al., 2021). For LLaMA 3, we fine-tune and evaluate on SCROLLS (Shaham et al., 2022), covering GovReport (Huang et al., 2021), Qasper (Dasigi et al., 2021), QMSum (Zhong et al., 2021), QuALITY (Pang et al., 2022), SummScreen-FD (Chen et al., 2022), and ContractNLI (Koreeda & Manning, 2021). These tasks fit our 16K-token fine-tuning budget and provide reliable validation sets.

Resource Usage and Practicality All fine-tuning and evaluation use NVIDIA A100 GPUs. `FLUX.1-dev` fine-tuning takes 43 GPU hours. The matched language retrieval protocol uses rank-16 LoRA and takes about one hour on eight GPUs per run (approximately 8 GPU hours).

Table 1: Single full-attention latency and speedup on A100. INTRA interleaves Scatter/Gather for image generation.

Methods	Seq Len	Config	GFLOPS	Latency(ms)	Speedup	FID ↓
FlashAttention 2	4K	-	260.9	1.181	-	-
CLEAR		r=8	63.5	0.893	1.32×	11.70
INTRA		Interleave Scatter/Gather	50.75	0.324	3.64×	11.74
FlashAttention 2	16K	-	3507.9	15.764	-	-
CLEAR		r=8	246.2	4.625	3.41×	32.46
INTRA		Interleave Scatter/Gather	345.5	2.056	7.67×	15.67

Table 2: End-to-end latency and image quality at 1K/2K. CLEAR (r=4–8) uses 1/4-resolution K/V and radius 8; CLEAR (r=48) is slower than dense `FLUX.1-dev`.

	1024×1024 Image Generation				2048×2048 Image Generation			
Method	**FID (↓)**	**LPIPS (↓)**	**CLIPI (↑)**	**Gen Time(s)**	**FID (↓)**	**LPIPS (↓)**	**CLIPI (↑)**	**Gen Time(s)**
Original Model	-	-	-	13.42	-	-	-	66.78
CLEAR (r=8)	**11.70**	49.96	0.8995	12.85	32.46	61.62	0.8046	47.81
CLEAR (r=4-8)	-	-	-	13.73	19.27	54.60	0.8467	52.07
INTRA	11.74	**49.46**	**0.9045**	**11.97**	**15.67**	**51.99**	**0.8559**	**43.72**

Appendix G.2 gives the adaptation recipe and distinguishes it from the earlier SCROLLS fine-tuning setup.

Results Table 1 reports image attention-kernel speedups; token-level language timings are in Table 3(a) and Appendix E. At 2K, INTRA achieves the best measured image metrics and lowest generation latency among evaluated sparse configurations: 43.72 seconds and FID 15.67 versus CLEAR ($r = 8$)'s 32.46 (Table 2). Table 3 reports single attention operation timings alongside separate SCROLLS quality results: INTRA has the lowest latency at all three context lengths and the highest QuALITY score. Appendix F gives group-count sweeps and the summary calculations.

For video transfer, INTRA reduces Wan2.2-TI2V-5B (Team Wan et al., 2025; Team Wan, 2025) generation time from 340.9 to 238.4 seconds ($1.43\times$) at 704×1280 with 121 frames on A100-SXM4-80GB. At the VBench (Huang et al., 2024) setting (480×832, 49 frames), latency falls from 44.1 to 41.2 seconds; evaluation on 944 prompts with five samples each yields a 13-dimension mean of 0.4677 versus dense attention's 0.4745. Table 19 combines end-to-end, forward, and attention timings; Appendix G.8 gives per-dimension quality results.

Appendix G.1 compares with a smaller dense diffusion model; Appendix H provides visual comparisons.

Needle Retrieval and Attention Efficiency. We evaluate LLaMA-3.1-8B on needle-in-a-haystack (NIAH) (Kamradt, 2023) retrieval at nine depths and 4K–32K contexts. Dense attention, MoBA (Lu et al., 2025a), NSA (Yuan et al., 2025), and INTRA share the LoRA budget and teacher-forced exact-match protocol; each model reuses one adapter trained on disjoint data. INTRA scores 0.923 on average and 0.918 at 32K. Table 4 presents retrieval scores alongside separate attention-only timings: across 4K–64K, INTRA is 3.6–$8.4\times$ faster than MoBA and 7.3–$10.4\times$ faster than tuned NSA.

Cross-Model Evaluation. On Qwen3-8B (Yang et al., 2025), INTRA achieves a mean NIAH score of 0.912 versus dense attention's 1.000, using the same evaluation protocol and LoRA recipe and budget. Qwen3-8B uses five block-level sparse layers among 36, with sparse-layer retained density tuned for each model family.

Table 3: Attention operation speed and separate SCROLLS quality. INTRA achieves **2.23× the speed of XAttention** and **14.94× the speed of MInference** (geometric means, 8K–32K). Respective unweighted mean-score gaps are 0.87 and 0.42 points.

(a) Single causal-attention operation latency (ms ↓) on A100. INTRA uses token-level Scatter/Gather.

Method	**8K**	**16K**	**32K**
XAttention ($S = 16$)	1.91	4.22	8.03
MInference	14.30	26.42	51.20
INTRA	**0.51**	**1.75**	**6.50**

(b) SCROLLS task scores and unweighted six-task mean (↑).

Method	**GovReport**	**Qasper**	**QMSum**	**QuALITY**	**SummScreen -FD**	**Contract NLI**	**Unweighted mean**
XAttention	31.16	39.94	24.93	27.5	20.27	**77.5**	**36.88**
MInference	**31.44**	**39.99**	**25.18**	24.5	**20.40**	77.1	36.44
INTRA	27.77	39.64	24.45	**29.0**	19.71	75.5	36.01

Table 4: LLaMA-3.1-8B retrieval quality and separate attention timing experiments.

(a) Mean NIAH with matched LoRA adaptation (9 depths; 4K–32K).

Method	Dense + LoRA	MoBA	NSA	INTRA
Mean NIAH ↑	1.000	0.194	0.325	**0.923**

(b) Attention latency (ms ↓) on A100-SXM4-80GB.

Context	**Dense** (SDPA)	**INTRA** (`g2s4`)	**MoBA**	**NSA (tuned)** $\rho \leq 0.125$
4K	27.4	**11.3**	95.2	117.7
16K	381.0	**59.7**	349.7	569.5
64K	6043.3	**577.2**	2073.0	4235.6

Attention only: sum of 32 per-layer medians (50 repetitions), bfloat16, batch 1. INTRA `g2s4` uses four Scatter $G = 2$ layers and 28 width-576 windows; NSA is tuned under density cap $\rho \leq 0.125$. Retrieval and timing configurations are specified in Section 4.7 and Appendix G.7, respectively.

6.2 Ablation Study

Global-Layer Ablation. We fix LLaMA-3.1-8B, training data, LoRA rank and steps, and four layer positions; the other 28 layers retain the same windows and sink tokens. Only attention at the selected positions changes. Sliding-window, INTRA block-sparse, and dense fillings obtain mean NIAH scores of 0.028, 0.923, and 1.000 at overall retained densities of 0.069, 0.123, and 0.185 (Table 5). INTRA thus retains 92.3% of the dense-filled control's retrieval score with about two-thirds of its retained attention entries. This measures the quality–density trade-off with fixed layer placement and adaptation.

Table 5: LLaMA-3.1-8B attention-pattern ablation at four fixed layers with matched training budgets. The other 28 layers use windows with sinks; density covers all 32 layers.

Pattern in selected layers	**Overall retained density**	**Mean NIAH ↑**
Sliding window	0.069	0.028
INTRA block-level sparse attention	0.123	0.923
Dense attention	0.185	1.000

Local Windows and Global Reach. To test whether increasing local capacity suffices, Appendix G.3 varies window width from 575 to 4096 in all 32 layers. At 32K, width 4096 scores 0.111 despite reaching 1.000 at 4K; INTRA's four global sparse layers retain 0.918 at 32K. Together with

the configuration comparison in Appendix G.4, this supports allocating attention to global reach rather than only widening local windows.

Image Pattern and Sparsity. Table 6 separately tests using either Scatter or Gather alone and varying the Scatter group count during interleaving. Scatter alone yields FID 213.47, Gather 0 improves this to 13.23, and interleaving reaches 11.75, with consistent LPIPS and CLIPI improvements. This contrast is consistent with the complementary roles in Section 4: Scatter supplies cross-neighborhood communication, while Gather reconnects otherwise isolated Scatter channels.

With interleaving retained, increasing Scatter groups from 4 to 8 to 16 reduces attended tokens but raises FID from 10.99 to 11.75 to 12.85; eight groups is an intermediate quality–sparsity choice. Switching every two layers gives the best measured metrics among intervals of one to four layers (Appendix G.1). Appendix G collects the supplementary experiments and adaptation studies.

Table 6: Ablation study: pattern choice and number of Scatter groups during interleaving.

(a) Use Scatter/Gather or both

	FID (↓)	LPIPS (↓)	CLIPI (↑)
Scatter	213.47	64.91	0.67
Gather 0	13.23	54.10	0.88
Scatter + Gather 0	**11.75**	**49.46**	**0.90**

(b) Number of groups in Scatter

	FID (↓)	LPIPS (↓)	CLIPI (↑)
4 groups	**10.99**	**48.29**	**0.908**
8 groups	11.75	49.46	0.905
16 groups	12.85	49.88	0.896

6.3 Limitations

Our partition analysis optimizes coverage and overlap, not attention approximation or downstream quality. The deployed geometric and causal masks impose additional constraints; causal connectivity is restricted to preceding tokens. Comparisons with other factorized patterns and larger models remain future work.

Quality is task-dependent: INTRA trails both SCROLLS baselines on five of six tasks. Relative to adapted dense attention, GovReport falls from 31.05 to 27.77, while Qasper changes from 40.09 to 39.64 (Table 14). Image comparisons do not establish dense-quality parity. Wan's aesthetic and imaging scores decrease by 0.0194 and 0.0564; shared zeros remain unexplained. Video quality is evaluated only at 49 frames.

Retrieval measures teacher-forced exact match after shared-budget adaptation, rather than free-generation or zero-shot performance or general rankings of MoBA/NSA. Layer ablations compare quality and density, not matched FLOPs; configuration sweeps vary both density and layer allocation. The SD 1.5 comparison also differs in architecture and training data. Quality scores lack confidence intervals; timing repetitions do not measure quality uncertainty.

Runtime indexing and non-contiguous loads add overhead; unchanged components limit end-to-end gains (Appendix E). Token kernels and block-sparse retrieval use distinct configurations. MoBA/NSA timing comparisons cover attention only, not full-model prefill. Reshaping changes scheduling and occupancy, preventing isolated overhead measurement.

7 Conclusion

We introduced INTRA as a sparse communication design under ISPD/CQS constraints. In a partition-based relay model, Scatter gives max-min local-neighborhood coverage and Gather gives balanced two-layer reconnection. Together, they provide a communication criterion for complementary static patterns.

The CQS-aware kernel turns these patterns into dense-tile GPU computation through runtime indexing and non-contiguous token packing. After adaptation, INTRA delivers $1.53\times$ faster 2K `FLUX.1-dev` generation and better measured image metrics than evaluated CLEAR configurations. Language experiments demonstrate faster prefill and effective long-range retrieval with a small allocation of global sparse layers, supporting structured static attention as a practical acceleration strategy.

AI USE STATEMENT

Generative AI assistants were used for polishing and reference checking. Our adaptation protocols also use model-generated data and dense-teacher outputs, as described in Section 6.1 and Appendix G.2. AI-assisted revisions were checked against cited primary sources and reported tables where applicable; these checks do not constitute independent reruns of the experiments. The authors proposed and determined the research questions, central ideas, methodology, and final technical claims. The authors take responsibility for the final text, claims, citations, and artifacts, including material prepared with AI assistance.

REFERENCES

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint 2004.05150*, 2020.

Black Forest Labs. FLUX.1 [dev]. Model card, 2024. URL https://huggingface.co/black-forest-labs/FLUX.1-dev.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token Merging: Your ViT But Faster. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JroZRaRw7Eu.

Mingda Chen, Zewei Chu, Sam Wiseman, and Kevin Gimpel. SummScreen: A dataset for abstractive screenplay summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8602–8615, 2022. doi: 10.18653/v1/2022.acl-long.589. URL https://aclanthology.org/2022.acl-long.589/.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating Long Sequences with Sparse Transformers. *arXiv preprint arXiv:1904.10509*, 2019. URL https://arxiv.org/abs/1904.10509.

Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610, 2021. doi: 10.18653/v1/2021.naacl-main.365. URL https://aclanthology.org/2021.naacl-main.365/.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783v1.

Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *arXiv preprint arXiv:1706.08500*, 2017. URL https://arxiv.org/abs/1706.08500.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1419–1436, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.112. URL https://aclanthology.org/2021.naacl-main.112.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive Benchmark Suite for Video Generative Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21807–21818, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Huang_VBench_Comprehensive_Benchmark_Suite_for_Video_Generative_Models_CVPR_2024_paper.html.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating Pre-filling for Long-Context LLMs via Dynamic Sparse Attention. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.nips.cc/paper_files/paper/2024/hash/5dfbe6f5671e82c76841ba687a8a9ecb-Abstract-Conference.html.

Greg Kamradt. Needle In A Haystack. GitHub repository, 2023. URL https://github.com/gkamradt/needle-in-a-haystack.

Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, Minghai Qin, and Yanzhi Wang. SPViT: Enabling Faster Vision Transformers via Latency-Aware Soft Token Pruning. In *Computer Vision – ECCV 2022*, 2022. URL https://www.ecva.net/papers/eccv_2022/papers_ECCV/papers/136710618.pdf.

Yuta Koreeda and Christopher Manning. ContractNLI: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 1907–1919, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.findings-emnlp.164.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. FlexPrefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. URL https://arxiv.org/abs/1405.0312.

Songhua Liu, Zhenxiong Tan, and Xinchao Wang. CLEAR: Conv-Like Linearization Revs Pre-Trained Diffusion Transformers Up. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.nips.cc/paper_files/paper/2025/hash/e36da7acd188c6655792270b38830124-Abstract-Conference.html.

Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video Swin Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3202–3211, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Liu_Video_Swin_Transformer_CVPR_2022_paper.html.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Neo Y. Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025a. URL https://arxiv.org/abs/2502.13189.

Wenbo Lu, Shaoyi Zheng, Yuxuan Xia, and Shengjie Wang. ToMA: Token merge with attention for diffusion models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 40930–40951, 2025b. URL https://proceedings.mlr.press/v267/lu25v.html.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5336–5358, 2022. doi: 10.18653/v1/2022.naacl-main.391. URL https://aclanthology.org/2022.naacl-main.391/.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 8748–8763, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification, 2021. URL https://arxiv.org/abs/2106.02034.

Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.3.0.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. SCROLLS: Standardized CompaRison Over Long Language Sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 12007–12021, 2022. URL https://aclanthology.org/2022.emnlp-main.823/.

Team Wan. Wan2.2-TI2V-5B. Model card, 2025. URL https://huggingface.co/Wan-AI/Wan2.2-TI2V-5B.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. URL https://arxiv.org/abs/2503.20314.

Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. MaxViT: Multi-Axis Vision Transformer. In *European Conference on Computer Vision*, 2022. URL https://arxiv.org/abs/2204.01697.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity, 2025. URL https://arxiv.org/abs/2502.01776.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. XAttention: Block Sparse Attention with Antidiagonal Scoring. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025. URL https://proceedings.mlr.press/v267/xu25ag.html.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin

Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention, 2025. URL https://arxiv.org/abs/2502.11089.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast video generation with sliding tile attention. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. arXiv 2502.04507.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2o: Heavy-hitter oracle for efficient generative inference of large language models, 2023. URL https://arxiv.org/abs/2306.14048.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. QMSum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5905–5921, 2021. doi: 10.18653/v1/2021.naacl-main.472. URL https://aclanthology.org/2021.naacl-main.472/.

A Token-Level Scatter and Gather 1 Patterns

Figure 4 illustrates the token-level Scatter and Gather 1 variants used in the kernel benchmarks. The language retrieval experiments instead use the block-level global layers and sliding-window layers with sink tokens described in Section 4.7.

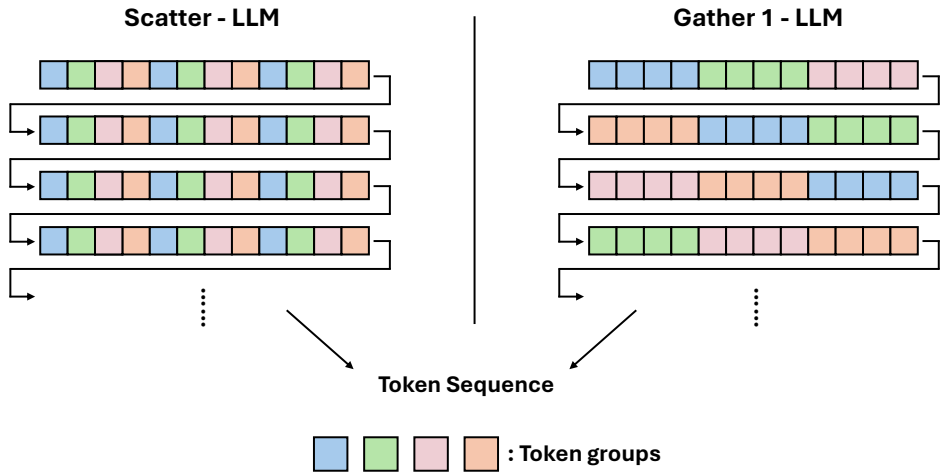


Figure 4: Token-level Scatter and Gather 1 patterns with a grouping factor of 4. Tokens are partitioned into 4 groups, shown by color, and both patterns compute attention within each group. These illustrate kernel variants rather than the block-level mask used in the retrieval experiments.

B GPU Memory Hierarchy

NVIDIA GPUs use a memory hierarchy consisting primarily of global memory, shared memory, and registers. Global memory is large but high-latency; shared memory is faster and shared within a thread block; registers are fastest and private to individual threads. INTRA changes which global-memory tokens are loaded, then packs the selected tokens contiguously in shared memory before applying FlashAttention-style computation.

C Sparse Pattern Index Function

C.1 Scatter

For the LLM Scatter pattern with query block size 64 and 8 groups, let i^{cqs} be the CQS index. The local-to-global mappings are

$$f_q(i_l^q, i^{\text{cqs}}) = i^{\text{cqs}} + 8 \cdot i_l^q, \tag{2}$$

$$f_k(i_l^k, i_l^q, i^{\text{cqs}}) = i^{\text{cqs}} + 8 \cdot i_l^k. \tag{3}$$

C.2 Gather 0

Assume an image feature map with width W and height H, both divisible by 8. The query and key block sizes are both $8 \times 8 = 64$, and the neighborhood region is 16×16. We use zero-based, row-major indexing for tokens and query blocks. For CQS index i^{cqs} and local query index $0 \le i_l^q < 64$,

$$\begin{aligned} f_q(i_l^q, i^{\text{cqs}}) = &\left(8 \left\lfloor \frac{i^{\text{cqs}}}{W/8} \right\rfloor + \left\lfloor \frac{i_l^q}{8} \right\rfloor\right) W \\ &+ 8\,(i^{\text{cqs}} \bmod (W/8)) + (i_l^q \bmod 8). \end{aligned} \tag{4}$$

$$f_k(i_l^k, i_l^q, i^{\text{cqs}}) = \begin{cases} f_q(i_l^q, i^{\text{cqs}}) - 4W + \frac{i_l^k}{8}W + i_l^k \ \% \ 8 & \text{if } i^{\text{cqs}} \ \% \ (W/8) = 0, \\ f_q(i_l^q, i^{\text{cqs}}) - 4W - 8 + \frac{i_l^k}{8}W + i_l^k \ \% \ 8 & \text{if } i^{\text{cqs}} \ \% \ (W/8) = (W/8) - 1, \\ f_q(i_l^q, i^{\text{cqs}}) - 4W - 4 + \frac{i_l^k}{8}W + i_l^k \ \% \ 8 & \text{otherwise.} \end{cases} \quad (5)$$

D Formal Relay Analysis for Static CQS Design

This appendix formalizes the relay-design intuition used in Section 4. The mask-level analysis establishes uniform Scatter and balanced Gather as minimax allocations for neighborhood coverage and inter-partition overlap under CQS-style partition constraints.

D.1 Setup

Let $T \geq 2$ and $R \geq 1$ be integers. Partition $N = TR$ tokens into T local neighborhoods:

$$P_1, \ldots, P_T, \qquad |P_t| = R.$$

We consider static partition-based CQS groups. Scatter consists of R relay groups

$$\mathcal{S} = \{S_a\}_{a=1}^{R}, \qquad |S_a| = T,$$

which partition the token set. Define

$$c_{a,t} = |S_a \cap P_t|.$$

The proof uses token counts and intersections of fixed sets. The local neighborhoods specify the partition over which coverage is evaluated.

D.2 Uniform Scatter as Minimax Neighborhood-Pair Coverage

For a pair of neighborhoods (P_s, P_t), define the relay opportunity count

$$K_{s,t} = \sum_{a=1}^{R} c_{a,s} c_{a,t}.$$

This counts how many ordered source-target representative pairs from P_s and P_t are co-grouped in some Scatter relay group.

Let $W_{s,t} \geq 0$ be an arbitrary normalized long-range demand matrix over ordered neighborhood pairs:

$$\sum_{s \neq t} W_{s,t} = 1.$$

Define the demand-weighted coverage of $\mathcal{S}$ as

$$\Phi(\mathcal{S}; W) = \sum_{s \neq t} W_{s,t} K_{s,t}.$$

Since the static mask does not know W in advance,

$$\min_{W} \Phi(\mathcal{S}; W) = \min_{s \neq t} K_{s,t}.$$

Thus robustness to arbitrary neighborhood-level demand reduces to maximizing the weakest relay opportunity count.

Proposition D.1. For any partition into R relay groups of size T,

$$\min_{s \neq t} K_{s,t} \leq R.$$

Uniform Scatter achieves equality by taking exactly one token from every neighborhood in every relay group:

$$|S_a \cap P_t| = 1 \qquad \forall a, t.$$

Consequently,

$$\max_{\mathcal{S}} \min_{W} \Phi(\mathcal{S}; W) = R.$$

Proof. For each relay group S_a,

$$\sum_{t=1}^{T} c_{a,t} = T.$$

Therefore,

$$\sum_{s \neq t} c_{a,s} c_{a,t} = \left(\sum_{t} c_{a,t} \right)^2 - \sum_{t} c_{a,t}^2 \leq T^2 - T,$$

because $\sum_t c_{a,t}^2 \geq T$ for nonnegative integers summing to T. Summing over all R groups gives

$$\sum_{s \neq t} K_{s,t} \leq RT(T-1).$$

There are $T(T-1)$ ordered neighborhood pairs, so the average value of $K_{s,t}$ is at most R. Hence

$$\min_{s \neq t} K_{s,t} \leq R.$$

Uniform Scatter has $c_{a,t} = 1$ for all a, t, so

$$K_{s,t} = \sum_{a=1}^{R} 1 \cdot 1 = R$$

for every ordered pair $s \neq t$, attaining the upper bound. □

Interpretation. Any non-uniform relay group that samples one neighborhood multiple times must under-sample another. This can improve some neighborhood pairs only by weakening others. If long-range demand is unknown, the weakened pair becomes the worst case. Uniform Scatter is therefore the no-prior robust static allocation of relay opportunities.

D.3 Balanced Gather as Minimax Channel Reconnection

Scatter creates global relay groups but separates them by local index. Write

$$S_r = \{(t, r) : t = 1, \ldots, T\}, \qquad r = 1, \ldots, R.$$

Let Gather be another partition

$$\mathcal{G} = \{G_b\}_{b=1}^{R}, \qquad |G_b| = T,$$

and define the overlap matrix

$$X_{rb} = |S_r \cap G_b|.$$

The entry X_{rb} counts the number of relay witnesses from Scatter channel S_r into Gather group G_b. If $X_{rb} = 0$, then that channel-to-group connection is absent.

For arbitrary reconnection demand, the weakest entry controls the worst case. Since each Scatter group has size T,

$$\sum_{b=1}^{R} X_{rb} = T.$$

Therefore,

$$\min_{b} X_{rb} \leq \left\lfloor \frac{T}{R} \right\rfloor.$$

Thus no Gather partition can make the weakest overlap larger than $\lfloor T/R \rfloor$.

A cyclic Gather construction achieves this value:

$$G_b = \{(t, r) : b \equiv t + r \pmod{R}\}.$$

For fixed r and b,

$$|S_r \cap G_b| = |\{t : b \equiv t + r \pmod{R}\}| \in \left\{ \left\lfloor \frac{T}{R} \right\rfloor, \left\lceil \frac{T}{R} \right\rceil \right\}.$$

Thus cyclic Gather maximizes the weakest Scatter-to-Gather overlap under the same equal-size partition constraint. Universal two-layer connectivity additionally requires $T \geq R$, so that $\lfloor T/R \rfloor \geq 1$. If $T < R$, zero overlaps remain and the construction does not provide universal two-layer connectivity.

D.4 Relation to Implemented Patterns

The results optimize neighborhood-pair coverage and inter-partition overlap within equal-size partitions. Uniform Scatter distributes representatives across neighborhoods, and cyclic Gather balances connections between the resulting channels. These objectives guide the implementations in Section 4.7: image generation combines Scatter with geometric Gather 0 neighborhoods, while language retrieval combines causal global block attention with local windows and sink tokens. The global-layer and image-pattern ablations evaluate these complementary roles after adaptation.

Balanced Gather reconnection. Given fixed Scatter groups and fixed Gather group sizes, the row and column sums of X_{ab} are fixed by the partition sizes. Maximizing $\min_{a,b} X_{ab}$ therefore favors balancing the overlap matrix and avoiding zero entries. When every entry is positive, the two-layer reachability result applies.

E Overhead Analysis

Table 7: Supplementary single causal-attention operation latency (ms) on A100. The FlashAttention 2 baseline and INTRA 4K result complement Table 3(a), which reports the 8K–32K sparse-method timings. INTRA uses token-level Scatter/Gather.

Method	**4K**	**8K**	**16K**	**32K**
FlashAttention 2	0.82	2.86	10.57	40.96
INTRA	0.16	See Table 3(a)		

Table 8: GFLOPS entries accompanying the single causal-attention timings in Tables 3(a) and 7. Dashes indicate unreported settings.

Method	**4K**	**8K**	**16K**	**32K**
FlashAttention 2	137.4	549.8	2199.1	8796.3
MInference	–	329.8	1275.4	4398.1
XAttn S=16	–	153.6	614.2	2615.1
INTRA	17.2	68.7	274.9	1099.5

This appendix profiles the token-level Scatter/Gather kernels. The language-sequence patterns are illustrated in Appendix A. With 8 groups, the ideal attention speedup is $8\times$. Actual Scatter/Gather speedups approach this value as sequence length grows, indicating that fixed overheads become less significant for longer contexts. The block-level mask construction cost for the language retrieval configuration is reported separately in Appendix G.6.

Table 9: Ideal vs. actual speedups for the token-level Scatter/Gather kernels on A100.

Pattern	**Seq. Len.**	**Ideal**	**Actual**
Scatter	4K	$8\times$	$4.81\times$
Gather 1	4K	$8\times$	$5.84\times$
Scatter	8K	$8\times$	$5.43\times$
Gather 1	8K	$8\times$	$5.83\times$
Scatter	16K	$8\times$	$5.90\times$
Gather 1	16K	$8\times$	$6.15\times$
Scatter	32K	$8\times$	$6.22\times$
Gather 1	32K	$8\times$	$6.37\times$

Nsight profiling in Table 10 characterizes the balance between memory traffic and computation after reducing the $QK^\top$ workload. INTRA has higher memory throughput and lower compute throughput than dense FlashAttention, consistent with its reduced arithmetic workload. Gather 0's

Table 10: Nsight profiling at 16K in the image setting. Throughput numbers are reported by NVIDIA profiling tools at the kernel level.

Method	Memory Throughput	Compute Throughput	Time (ms)
FlashAttention	49.95	70.69	15.80
INTRA Scatter	86.31	56.64	3.04
INTRA Gather 0	79.48	51.19	1.07

more contiguous loads produce fewer uncoalesced global-memory transactions than Scatter's strided loads.

Table 11 compares INTRA with an equally sparse dense-FlashAttention simulation. The simulation reshapes inputs to match sparsity, providing a computational reference for the Scatter and Gather 1 timings.

Table 11: INTRA compared with an equally sparse dense-FlashAttention simulation.

Pattern	Seq. Len.	INTRA (ms)	Equally Sparse (ms)
Scatter	4K	0.17	0.19
Scatter	8K	0.53	0.52
Scatter	16K	1.79	1.65
Scatter	32K	6.58	5.73
Gather 1	4K	0.29	0.22
Gather 1	16K	1.07	0.79

Table 12 isolates the attention contribution by replacing attention with a dummy operation. The left subtable reports full `FLUX.1-dev` generation latency at 4K and 16K, while the right subtable reports LLaMA 3 prompt-prefill latency from 4K to 32K. INTRA therefore targets the attention-heavy generation and prefill stages; autoregressive decoding is more memory-bound, and batch parallelism remains orthogonal because the sparse pattern is applied independently within each sequence.

Table 12: End-to-end latency with and without attention.

(a) `FLUX.1-dev` generation

Setting	4K	16K
Original	13.4s	66.8s
Remove attention	11.6s	41.2s

(b) LLaMA 3 prefill

Setting	4K	8K	16K	32K
Original	0.31s	0.64s	1.41s	3.47s
Remove attention	0.27s	0.53s	1.05s	2.07s

The latency measurements in this overhead analysis are averaged over 2,000 runs. Model-quality scores are reported as point estimates.

F Speed–Quality Trade-off

Figure 5 varies the group count as INTRA's sparsity control knob: more groups increase sparsity and reduce computation. For LLaMA 3, the group counts are 8, 12, and 16; for `FLUX.1-dev`, they are 4, 8, and 16. These are numbers of groups, not tokens per group. The curves show the trade-off between relative speed and Qasper F1 or image FID as sparsity changes.

Main-table summary. Separately, we average the six task scores reported in Table 3, $\bar{s}_m = \frac{1}{6}\sum_{j=1}^{6} s_{m,j}$. This is an unweighted mean of task-specific metrics in score points. The means are 36.88 for XAttention, 36.44 for MInference, and 36.01 for INTRA. Differences are computed before rounding: INTRA's mean-score gaps are 0.87 and 0.42 points, respectively.

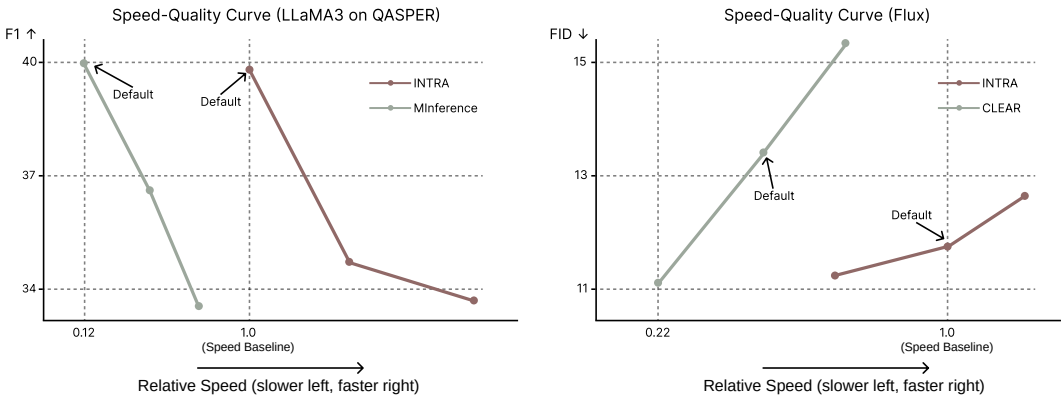


Figure 5: Speed–quality trade-offs for LLaMA 3 on Qasper (left) and `FLUX.1-dev` (right). Horizontal axes show relative speed with the INTRA default at 1: points nearer the origin are slower, and points farther right are faster.

The attention operation summary uses the geometric mean of latency ratios over $\mathcal{L} = \{8\text{K}, 16\text{K}, 32\text{K}\}$: $g_B = \left(\prod_{L\in\mathcal{L}} t_B(L)/t_{\text{INTRA}}(L)\right)^{1/3}$. This gives $2.23\times$ against XAttention ($S = 16$) and $14.94\times$ against MInference. These summaries use Table 3 directly and are separate from the group-count sweeps in Figure 5.

G Additional Experiments and Ablations

This appendix collects image-generation ablations, language adaptation and retrieval studies, implementation costs, and video-generation results.

G.1 Image Generation Ablations

Table 13a compares INTRA with Stable Diffusion 1.5 under the same FID protocol. INTRA obtains FID 11.75, compared with 30.02 for the smaller dense diffusion model.

We also ablate how often INTRA switches between Scatter and Gather. The number of layers in Table 13b indicates how many consecutive layers share the same sparse pattern before switching. Interleaving every 2 layers performs best across metrics.

Table 13: Additional image-generation ablations. Left: comparison with a smaller dense diffusion model. Right: ablation on interleaving frequency.

(a) Smaller dense model

Method	FID (↓)
SD 1.5	30.02
INTRA	**11.75**

(b) Interleaving frequency

Frequency	FID (↓)	LPIPS (↓)	CLIPI (↑)
1 layer	12.6	51.7	0.895
2 layers	**11.7**	**49.4**	**0.904**
3 layers	13.1	51.0	0.892
4 layers	12.8	53.2	0.890

G.2 Language Adaptation Protocols

SCROLLS fine-tuning. Table 14 compares the original LLaMA 3.1 8B model without fine-tuning, the original model with fine-tuning, and INTRA with fine-tuning. The Qasper result uses LoRA rank 32 and closes most of the gap to the fine-tuned dense model. The earlier language fine-tuning setup was reported as 320 GPU hours. This cost and the rank-32 Qasper setting are separate from the matched retrieval adaptation protocol below.

Table 14: LLaMA 3.1 8B performance on SCROLLS with and without fine-tuning.

	GovReport	Qasper	QMSum	QuALITY	SummScreen-FD
Original model (no fine-tune)	24.13	34.91	21.17	0.0	16.77
Original model	**31.05**	**40.09**	**25.37**	26.0	**20.51**
INTRA	27.77	39.64	24.45	**29.0**	19.71

Matched retrieval adaptation. For the NIAH comparisons and language ablations, all methods, including the dense control, use the same data and adaptation budget. The training mixture contains 60% public train-split conversations in the evaluation format, 25% synthetic retrieval examples, and 15% plain continuation. The data are packed into 400 blocks of 16,384 tokens and are disjoint from all evaluation documents. We train for 1,200 steps using rank-16 LoRA on the attention $q/k/v/o$ and MLP gate/up/down projections. Supervision comes from each model's own dense outputs, following knowledge distillation (Hinton et al., 2015), without ground-truth task labels. One adapter is trained per model and reused across its evaluation tasks. For LLaMA-3.1-8B, the adapters contain 41.9M trainable parameters (approximately 0.5% of the model), and each run takes about one hour on eight A100 GPUs, or approximately 8 GPU hours. Qwen3-8B uses the same adaptation recipe and budget, with the number and retained density of its global sparse layers configured as described in the main text.

G.3 Window-Width Ablation

We further evaluate LLaMA-3.1-8B models using sliding-window attention in all 32 layers under the same adaptation budget. Increasing the window width from 575 to 4096 tokens raises the NIAH score at 4K from 0.111 to 1.000, whereas the score at 32K increases only from 0.000 to 0.111. These results show that wider windows alone remain insufficient for reliable retrieval at 32K under the evaluated protocol. Table 15 reports the measured scores.

Table 15: Window-width ablation on LLaMA-3.1-8B with sliding-window attention in all 32 layers. Scores are measured NIAH results at 4K and 32K context lengths.

Window width	NIAH at 4K ↑	NIAH at 32K ↑
575	0.111	0.000
2048	0.556	0.000
4096	1.000	0.111

G.4 Attention-Configuration Comparison

Table 16 compares alternative attention configurations of LLaMA-3.1-8B under a fixed adaptation budget. The configuration with sparse attention in half the layers retains a density of 0.125 within each selected layer, while the four-layer configuration retains 0.5 within each selected layer. The four-layer configuration achieves the highest NIAH score among the sparse configurations shown: 0.923 at overall density 0.123, compared with 0.222 at density 0.234 for wider windows. Across these density and layer-allocation choices, concentrating global attention in four layers provides the strongest retrieval result among the tested sparse configurations.

Table 16: Language attention configurations under the same adaptation budget. Retained density refers to the overall attention configuration, normalized to dense attention at 1.000.

Configuration	Retained density	Mean NIAH ↑
Sparse attention in 16 layers (per-layer density 0.125)	0.098	0.130
Wider windows	0.234	0.222
Four global sparse layers (per-layer density 0.5)	0.123	0.923
Dense + LoRA	1.000	1.000

G.5 LoRA Adaptation Ablation

Without LoRA adaptation, NIAH scores range from 0.02 to 0.09 across the sparse methods and configurations tested in this study. These unadapted runs provide the reference for measuring the effect of LoRA adaptation on retrieval.

G.6 Mask Construction and Caching

For the block-level language configuration, the block mask is constructed once per configuration and cached for reuse. Mask construction takes 3.4 ms at a context length of 16K. This is a one-time mask-construction cost; repeated calls with the same cached configuration reuse the mask.

G.7 Attention Timing Comparison with MoBA and NSA

Table 17 compares INTRA, MoBA (Lu et al., 2025a), and NSA (Yuan et al., 2025) on one A100-SXM4-80GB, measured sequentially with LLaMA-3.1-8B attention shapes: 32 query heads, eight key/value heads (GQA-4), head dimension 128, bfloat16, batch size one, and causal prefill. We time attention only, taking the median of 50 repetitions for each layer. Panel (a) sums these medians across 32 layers; panel (b) reports single-layer latency. For method A and reference B, speedup is t_B/t_A.

Configurations and density. The fixed `g2s4` schedule uses four Scatter layers with $G = 2$ and 28 sliding-window layers with width 576. Its window contribution to overall retained density varies with context length. The separate single-layer comparison uses Scatter with $G = 8$ and a target retained density of 0.125. MoBA uses the official implementation, selecting two of 16 blocks (density 0.125), with the fastest tested chunk setting at each length. NSA's paper setting is $(l, d, l', n, w) = (32, 16, 64, 16, 512)$; its measured densities at 4K, 16K, and 64K are 0.578, 0.220, and 0.104, respectively. For density-constrained NSA, we sweep 18 combinations of (l, d), l', and w, binary-search the largest n satisfying density ≤ 0.125, and select the fastest configuration. The selected (l, d, l', n, w) tuples are $(64, 32, 64, 3, 128)$ at 4K, $(64, 32, 64, 6, 512)$ at 16K, and $(128, 64, 128, 27, 512)$ at 64K.

Reading the comparison. We label density-constrained NSA as *NSA (tuned, $\rho \leq 0.125$)*, where ρ denotes retained attention density. The 0.125 cap constrains connectivity; the speedup columns report $t_{\text{baseline}}/t_{\text{INTRA}}$. For example, at 16K, $569.5/59.7 \approx 9.5\times$ corresponds to an 89.5% reduction in the measured attention time. Table 4b brings the 32-layer comparison into the main text; Table 17 additionally reports NSA's paper configuration and the separate single-layer experiment.

Table 17: Causal prefill attention latency (ms) on A100-SXM4-80GB. (a) Sum of 32 per-layer medians for the fixed INTRA `g2s4` schedule and baselines. (b) Single-layer comparison with target density 0.125. NSA (tuned) is the fastest tested setting satisfying $\rho \leq 0.125$; NSA (paper) uses its paper hyperparameters. INTRA speedup over a baseline is $t_{\text{baseline}}/t_{\text{INTRA}}$.

(a) 32-layer attention totals; lower latency is better.

	Attention latency (ms ↓)					INTRA speedup over	
Context	Dense (SDPA)	INTRA (`g2s4`)	MoBA	NSA (paper)	NSA (tuned) $\rho \leq 0.125$	MoBA	NSA (tuned)
4K	27.4	**11.3**	95.2	238.8	117.7	8.4×	10.4×
16K	381.0	**59.7**	349.7	936.3	569.5	5.9×	9.5×
64K	6043.3	**577.2**	2073.0	4390.6	4235.6	3.6×	7.3×

(b) Single-layer latency with Scatter $G = 8$.

Context	Dense	INTRA Scatter	MoBA	NSA (tuned) $\rho \leq 0.125$	INTRA speedup over NSA (tuned)
4K	0.856	**0.223**	2.976	3.677	16.5×
16K	11.906	**1.955**	10.927	17.797	9.1×
64K	188.852	**28.541**	64.782	132.363	4.6×

Comparison. INTRA has the lowest attention latency in every reported setting. Across 4K–64K, the fixed `g2s4` schedule is 3.6–8.4× faster than MoBA and 7.3–10.4× faster than density-constrained NSA, reaching 10.47× the dense attention speed at 64K. The single-layer comparison further isolates the Scatter kernel: at target density 0.125, it is 3.84×, 6.09×, and 6.62× faster than dense attention at 4K, 16K, and 64K, and 4.6–16.5× faster than density-constrained NSA. The advantage thus persists both for a sparse-layer/window schedule and for a single global sparse layer. INTRA's static indexing avoids input-dependent selection and routing, providing an efficient execution path for the evaluated workloads. These timing measurements complement the separate matched-adaptation retrieval comparison in Table 4.

G.8 Video Generation Results

Table 18 reports all 13 evaluated VBench dimensions at 480×832 with 49 frames, covering temporal consistency, perceptual quality, style, and object and spatial criteria. The summary in the main text is the unweighted arithmetic mean of these 13 per-dimension scores. We report both dense and INTRA scores together with their absolute differences.

Table 18: All 13 evaluated VBench (Huang et al., 2024) dimensions on Wan2.2-TI2V-5B (Team Wan et al., 2025; Team Wan, 2025), using 944 prompts and five samples per prompt. Higher scores are better; the final column reports absolute dense–INTRA differences.

VBench dimension	Dense	INTRA	Absolute difference
Subject consistency	0.9427	0.9405	0.0022
Background consistency	0.9587	0.9551	0.0036
Aesthetic quality	0.5362	0.5168	0.0194
Imaging quality	0.6036	0.5472	0.0564
Temporal flickering	0.9886	0.9874	0.0012
Motion smoothness	0.9664	0.9689	0.0025
Dynamic degree	0.4583	0.4667	0.0084
Overall consistency	0.2435	0.2400	0.0035
Appearance style	0.2277	0.2222	0.0055
Temporal style	0.2427	0.2356	0.0071
Object class	0.0000	0.0000	0.0000
Multiple objects	0.0000	0.0000	0.0000
Spatial relationship	0.0000	0.0000	0.0000

Video latency and attention scaling. Table 19 measures Wan2.2-TI2V-5B on A100-SXM4-80GB at three sequence lengths. End-to-end timing includes text encoding, 50 denoising steps with two classifier-free-guidance evaluations per step (100 DiT forwards), and VAE decoding. Standalone forward timing uses batch size one and all 30 DiT blocks; self-attention timing sums the 30 self-attention layers. Pipeline DiT totals and standalone forward times come from separate measurements.

At 5,070 tokens, self-attention accounts for 12% of dense forward time; accelerating it by 2.12× yields a 1.07× end-to-end speedup. At 27,280 tokens, attention's share rises to 44%, and its 4.16× speedup translates into 1.48× faster standalone DiT execution and 1.43× faster generation, saving 102.5 seconds per video. At 53,680 tokens, attention occupies 60% of dense forward time: INTRA reduces self-attention from 5528 to 1129 ms and the complete forward from 9143 to 4855 ms. The increasing attention share explains why structured sparsity gives larger overall gains as video sequence length grows.

Table 19: Wan2.2-TI2V-5B video timing on A100-SXM4-80GB. Arrows show dense → INTRA; speedups are dense time divided by INTRA time. End-to-end and pipeline components are in seconds, while standalone forward and self-attention times are in milliseconds. The 49-frame setting matches VBench; 121 frames corresponds to the native five-second setting at 24 fps. The 241-frame setting measures standalone speed only; dashes denote unmeasured pipeline timings.

Resolution **Frames / video tokens**	480×832 49 / 5,070	704×1280 121 / 27,280	704×1280 241 / 53,680
End-to-end generation and pipeline components (s)			
Total generation	$44.1 \rightarrow 41.2$	$340.9 \rightarrow 238.4$	—
End-to-end speedup	$1.07\times$	$\mathbf{1.43}\times$	—
DiT total (100 forwards)	$40.9 \rightarrow 38.0$	$326.9 \rightarrow 224.2$	—
VAE decoding	2.6	13.4	—
Standalone DiT forward: batch size one, all 30 blocks			
DiT forward (ms)	$399 \rightarrow 371$	$3257 \rightarrow 2198$	$9143 \rightarrow 4855$
Forward speedup	$1.07\times$	$1.48\times$	$\mathbf{1.88}\times$
Self-attention (ms)	$49.3 \rightarrow 23.2$	$1423 \rightarrow 342$	$5528 \rightarrow 1129$
Self-attention speedup	$2.12\times$	$4.16\times$	$\mathbf{4.90}\times$
Attention share of dense forward	12%	44%	60%

H QUALITATIVE RESULTS

Figure 6 provides an overview; Figures 7 and 8 provide additional visual comparisons between the original `FLUX.1-dev` model, INTRA, and CLEAR.



Figure 6: `FLUX.1-dev` outputs: original FlashAttention 2, INTRA, and CLEAR (left to right).

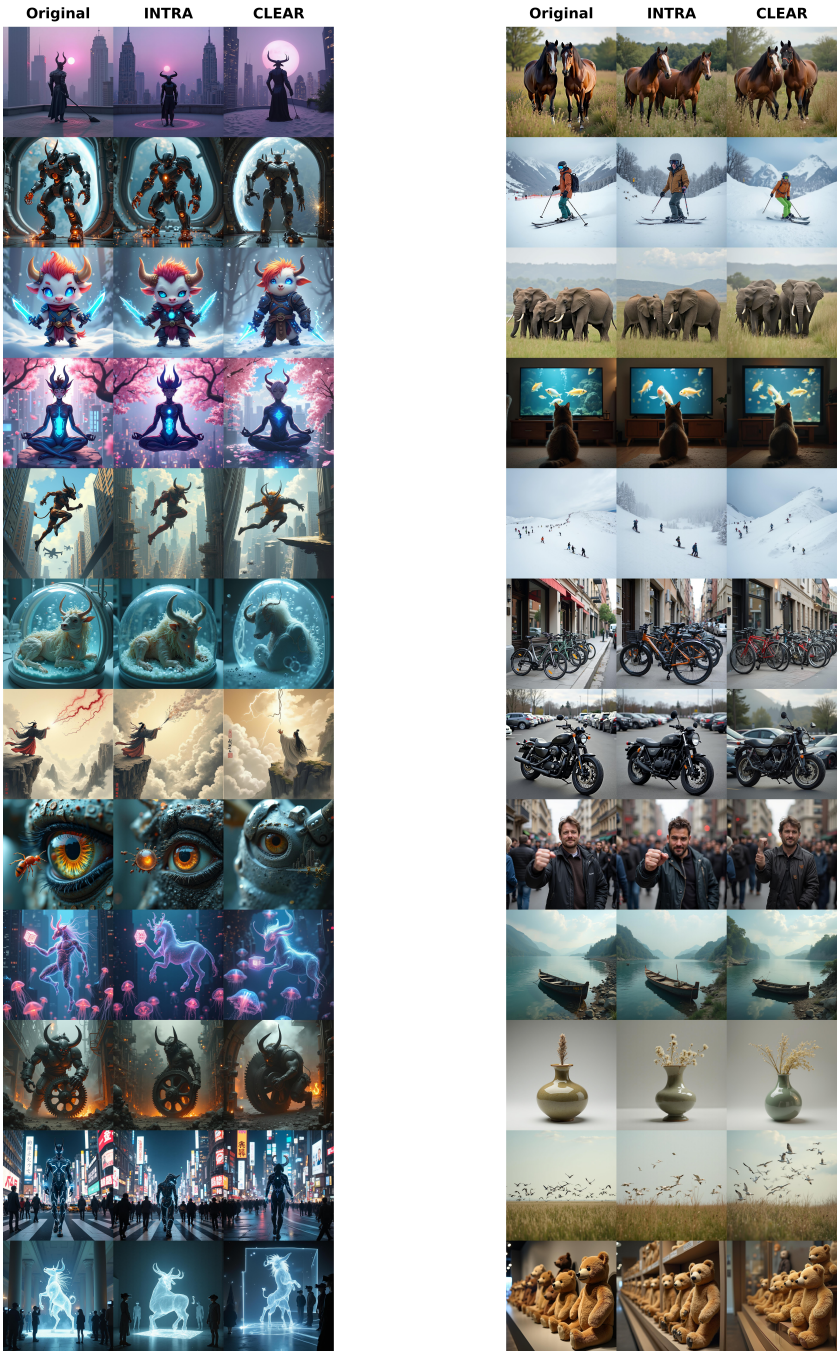


Figure 7: Visual examples comparing original `FLUX.1-dev`, INTRA, and CLEAR.

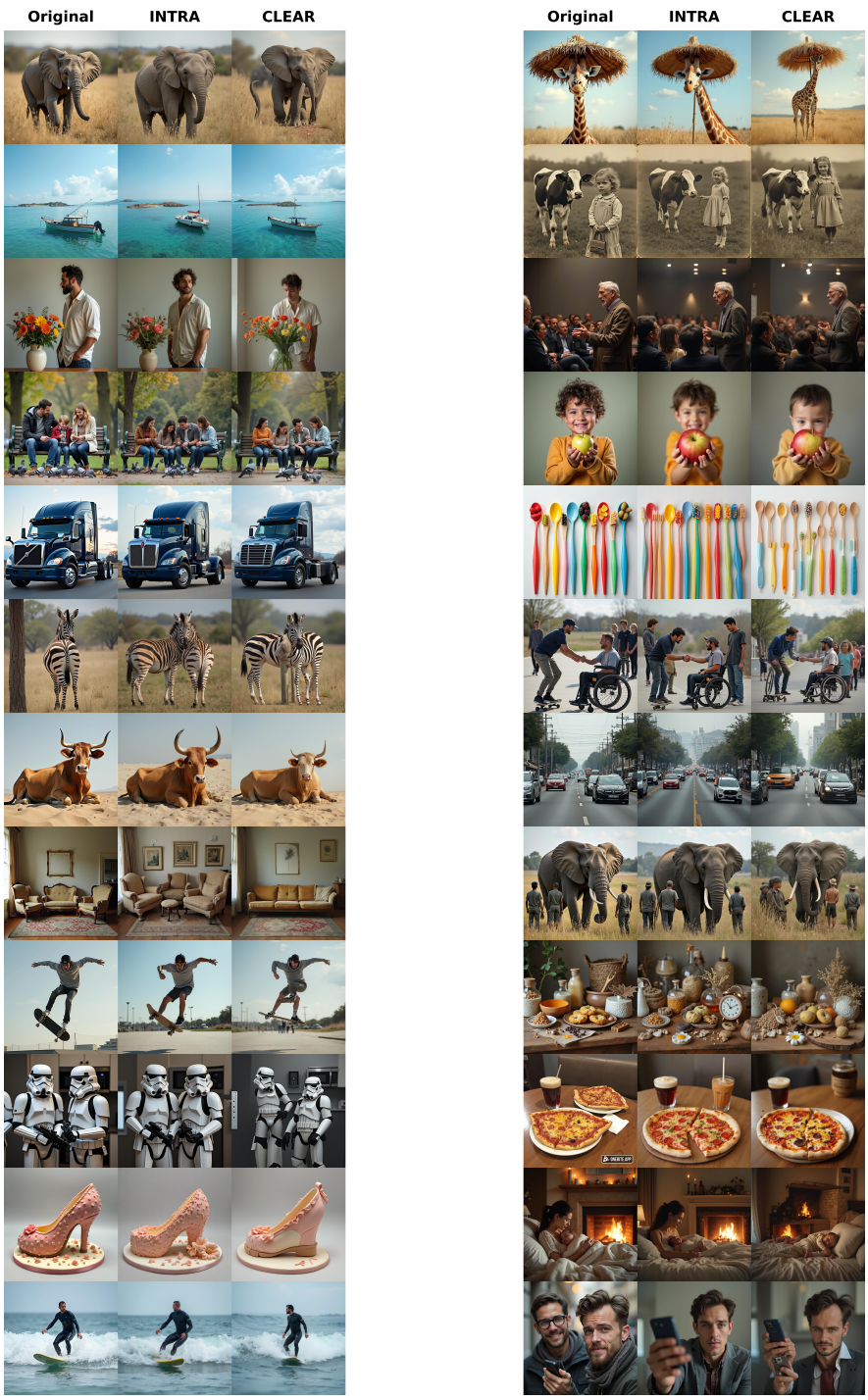


Figure 8: Additional visual examples with the same column order as Figure 7: original `FLUX.1-dev`, INTRA, and CLEAR within each triplet.