SQUAD: SUB-QUADRATIC ATTENTION DISTILLATION FOR EFFICIENT VIDEO GENERATION

Anonymous authors
Paper under double-blind review

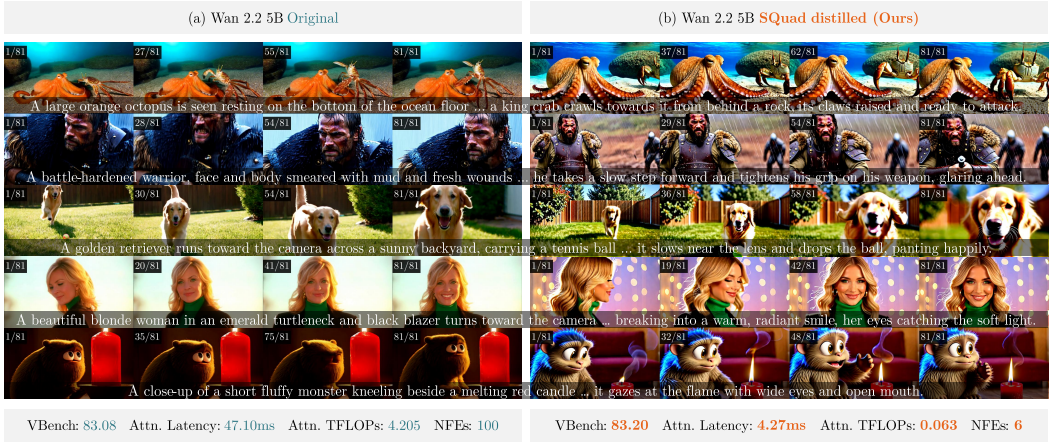


Figure 1: **Efficient video generation with SQuad.** Given the same prompts, the original Wan 2.2 5B model (a) produces high-quality videos but at significant computational cost. Our SQuad-distilled model (b) preserves the quality while substantially reducing the computational cost.

ABSTRACT

Video Diffusion Transformers (DiTs) spend most of their compute inside the Self-Attention operation, whose cost grows quadratically, $\mathcal{O}(n^2)$, with the number of latent tokens n. For the task of video generation, the token count is large, so this term dominates runtime and memory, and thereby caps the resolution and duration we can generate. Linear $\mathcal{O}(n)$ and low-rank $\mathcal{O}(nk)$ surrogates of Self-Attention trade the full softmax QK^T for cheaper kernels, but rarely recover the original's expressivity, leaving a stubborn quality gap. Motivated by this, we propose SQuad, a Sub-Quadratic Attention Distillation framework that achieves a complexity of $\mathcal{O}(n\sqrt{n})$ in the resulting distilled Attention, naturally balancing the efficiency v/s expressivity trade-off. Instead of training our own Video DiT from scratch, which is prohibitively expensive, we fit a pretrained full softmax Self-Attention DiT into our proposed SQuad-Attention one by distilling the former in two stages: Flow-Matching Supervised Fine-Tuning (SFT), followed by improved Distribution Matching Distillation (DMD2) which additionally makes the sampling more efficient. On the Wan 2.2 5B text-to-video model, SQuad matches the quadratic teacher on VBench (83.20 v/s 83.08) while cutting the per-step per-block attention FLOPs by $\sim$67$\times$ and attention latency by $\sim$11$\times$, and end-to-end DiT latency by 2$\times$, all while also generating a video in only 6 Neural Functional Evaluations (NFEs) instead of the default 100.

1 INTRODUCTION

Video generation has quickly become one of the most exciting frontiers of generative modeling. DiTs (Peebles & Xie, 2023) now produce high-resolution, temporally coherent clips of remarkable visual quality (Brooks et al., 2024; Blattmann et al., 2023; Wan Team, 2025). On today's hardware, however, these models generate only a few seconds of video in a single pass (Wan Team, 2025). Longer videos are stitched together by generating short clips one after another, each conditioned on the last—a process that ultimately drifts and loses coherence as the video grows (Li et al., 2026; Cui

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.

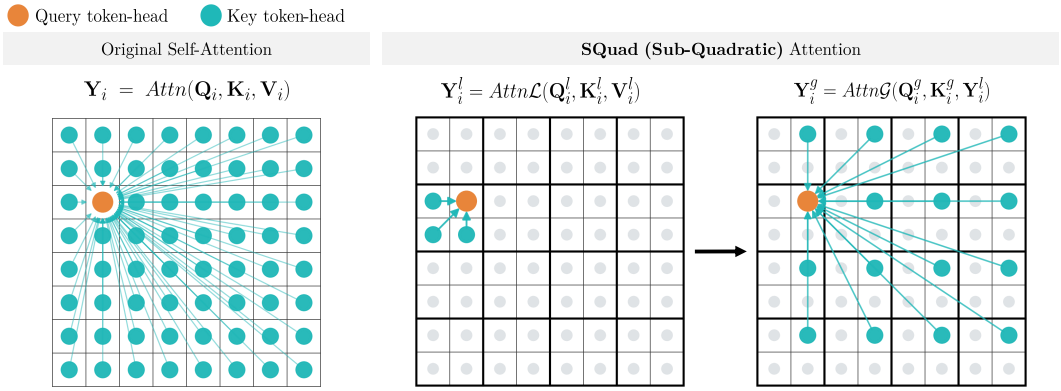


Figure 2: **SQuad attention.** *Left:* full softmax Self-Attention at $\mathcal{O}(n^2)$ complexity. *Right:* SQuad factorizes it into a *local* pass *within* $\mathcal{O}(\sqrt{n})$ windows followed by a *global* pass *across* the windows, giving a full receptive field at $\mathcal{O}(n\sqrt{n})$ complexity with a true softmax throughout.

et al., 2026; Liu et al., 2026; Chen et al., 2026; Huang et al., 2026; Yang et al., 2026). Generating longer videos with higher resolution natively, therefore remains a central goal of the field. Underlying all of these models is the softmax Self-Attention mechanism (Vaswani et al., 2017), which lets every token attend to every other token and is widely seen as the main driver of the expressivity and scalability of modern transformers. Softmax Self-Attention matters far beyond video: it is the backbone of nearly every domain that transformers now dominate. Studying how our proposed modification generalizes across all of these settings would be fascinating, but in favor of a concrete and carefully evaluated research, we focus on video generation with a well-defined scope, where the cost of attention is the highest and the potential payoff is perhaps one of the greatest, if not the greatest.

The biggest challenge is computational cost. Softmax Self-Attention scales quadratically with the number of tokens, $\mathcal{O}(n^2)$, in both time and memory (Vaswani et al., 2017). In video, a single token indexes a point in space and time, and their number easily reaches tens of thousands per frame-volume. At this scale the quadratic term dominates the entire generation budget and caps the resolution, duration, and throughput the GPU hardware can afford. The obvious fix is to make the attention cheaper. Linear $\mathcal{O}(n)$ attention and related $\mathcal{O}(nk)$ approximations (Katharopoulos et al., 2020; Choromanski et al., 2021; Wang et al., 2020) replace the softmax with a kernelized or low-rank surrogate and bring the cost down. But these surrogates rarely match the expressivity of the full softmax Self-Attention. The non-linearity and the sharp, input-dependent selectivity that make softmax Self-Attention so effective are exactly what the approximations give up. The result is a stubborn quality gap, and it is widest in the high-fidelity, detail-sensitive regime of video generation. Faced with this trade-off, prior works have looked for a middle ground. Whether it is linearizing a pretrained model or distilling its attention into kernel-based approximations (Katharopoulos et al., 2020; Choromanski et al., 2021), or even replacing the attention altogether with structured state-space models such as S4 and Mamba (Gu et al., 2022; Gu & Dao, 2024), on their own, neither substitution quite matches the performance and scalability of full softmax Self-Attention. Hence the field has increasingly turned to *hybrid* designs that interleave a few expensive quadratic full softmax Self-Attention blocks with many cheaper efficient ones (Lieber et al., 2024). This idea has recently reached video generation as well (Ghafoorian et al., 2025; Ghafoorian & Habibian, 2026; Zhang et al., 2025a; Li et al., 2025; Yang et al., 2025c; Zhang et al., 2025b; Ghafoorian et al., 2026). These hybrids recover much of the lost quality, yet they pay for it with complex-heterogeneous architectures, delicate choices about which layers stay quadratic, and more-exacerbatingly, a lingering dependence on the very softmax Self-Attention operation they set out to replace.

Considering this scenario, we take a different route with **SQuad**, a *Sub-Quadratic Attention Distillation* framework for efficient video generation. Rather than abandoning softmax Self-Attention, we lean into one of its well-known properties: in video DiTs, the attention maps are sparse and heavy-tailed. Almost all of the attention mass falls on a small set of critical tokens, while the rest of the $\mathrm{softmax}(QK^\top/\sqrt{d})$ entries stay close to zero (Zhang et al., 2025a; Xi et al., 2025; Yang et al., 2025b; Chen et al., 2025). We turn this observation into a framework that lowers the attention

cost from $\mathcal{O}(n^2)$ to $\mathcal{O}(n\sqrt{n})$ while keeping a genuine softmax throughout, unlike the linear variants (glance over fig. 2). Our proposed version of attention is rather simple but surprisingly effective. We propose a fixed, reduced communication pattern with two steps: locally, tokens mix within $\mathcal{O}(\sqrt{n})$-sized windows; while globally, tokens at the same position in each window mix across all windows. On the Wan 2.2 5B text-to-video generation model, as well as 2.1 1.3B, SQuad distills the quadratic full softmax Self-Attention into our proposed sub-quadratic SQuad-Attention with almost no loss in quality, while cutting latency and TFLOPs substantially (see fig. 1).

2 Related Work

We cover the more relevant related works on efficient attention methods (section 2.1), and Distillation of Video Generators (section 2.2) here, while defer the broader coverage on Text-to-Video generation models to the supplementary (section C).

2.1 Efficient Attention / Token Merging

Because this is such an important problem, a large body of work has sought to solve it. One line replaces or approximates the softmax. Attention Surgery (Ghafoorian et al., 2025) linearizes a pretrained video diffusion transformer by distilling its softmax attention into a hybrid softmax/linear form while preserving quality. ReHyAt (Ghafoorian & Habibian, 2026) blends the two as well, but computes the softmax component as a chunk-wise recurrence while using linear attention for farther-away chunks, allowing constant runtime memory. A second, larger line keeps the exact softmax but evaluates it only over a sparse subset of token pairs, exploiting the observation that video attention maps are highly sparse and structured. VSA (Zhang et al., 2025a) makes this sparsity *trainable*: a coarse stage pools tokens into tiles to locate critical regions, and a fine stage computes token-level attention only within them, as a single differentiable kernel usable in both training and inference. Radial Attention (Li et al., 2025) instead fixes a static $\mathcal{O}(n \log n)$ mask motivated by "spatio-temporal energy decay", shrinking each token's attention window as temporal distance grows. Jenga (Zhang et al., 2025b) is likewise training-free, combining dynamic attention carving with space-filling curve-based token flattening at inference time. We compare with these methods as our primary efficiency baselines, but note that SQuad differs from all of these in two ways. Unlike the linear/hybrid surrogates, it retains a genuine softmax throughout; and unlike the sparse-mask methods, its reduced communication pattern is *fixed* and structured, which lowers the complexity to $\mathcal{O}(n\sqrt{n})$ without the complex data-dependent token-selection steps.

2.2 Distilling Video Generators

We find that for SQuad, a simple Flow-Matching (Lipman et al., 2022) SFT is *not* enough to recover the original's quality after swapping the softmax for SQuad-Attention; some form of teacher distillation is required. For T2V generation, such distillation is currently done in two complementary ways.

Model (capability) distillation matches what happens *inside* the teacher, not just what comes out of it. The idea is to pull the student's intermediate features towards those of a stronger network: REPA (Yu et al., 2025) aligns a diffusion transformer's hidden states with a self-supervised encoder like DINO, REPA-E (Leng et al., 2025) carries this through the VAE end-to-end, and Wang et al. (2025) study when such alignment helps and when to switch it off. All of this leans on a now-familiar observation — that diffusion backbones already learn good, transferable features on their own (Xiang et al., 2023). Neodragon (Karnewar et al., 2025) takes the same idea to video, matching a teacher's activations to shrink a model down for on-device generation.

Step distillation instead aims to reduce the NFEs needed at inference. Progressive schemes halve the sampler's step count (Salimans & Ho, 2022; Meng et al., 2023); consistency-based methods map any trajectory point to its endpoint (Song et al., 2023; Luo et al., 2023; Kim et al., 2024; Lu & Song, 2024); adversarial schemes add a GAN objective for one-to-four-step generation (Sauer et al., 2024; Lin et al., 2025); distribution-matching methods minimize a reverse-KL between student and teacher via a score difference (Yin et al., 2024b;a), extended to video by CausVid (Yin et al., 2025), Self Forcing (Huang et al., 2025b), and the Motion Consistency Model (Zhai et al., 2024); and recent flow-map methods learn the two-time map of the flow directly (Geng et al., 2025; Sabour et al., 2025; Boffi et al., 2024). Rather than inventing yet another distillation method for our proposed SQuad-Attention, we simply adopt DMD2 (Yin et al., 2024a), which strikes a good balance between the two objectives above: as a distribution-matching method it allows us to recover the capability lost

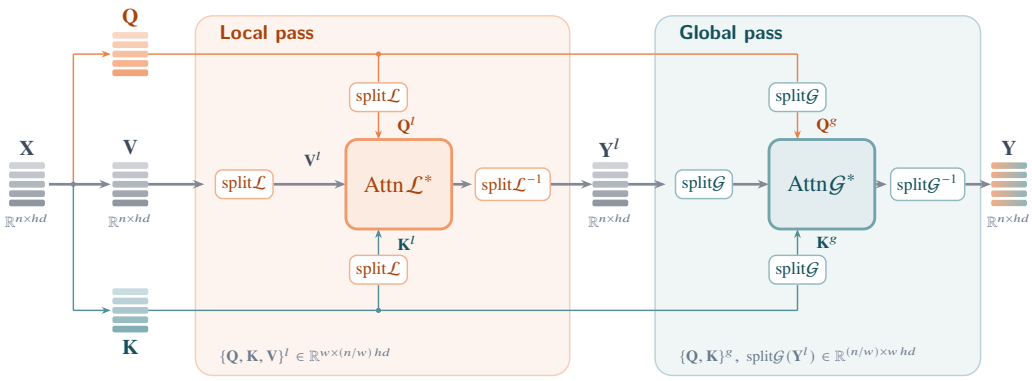


Figure 3: **Illustration of the SQuad Attention operation.** The starred operators $\mathrm{Attn}\mathcal{L}^*$ and $\mathrm{Attn}\mathcal{G}^*$ apply the attention operations per-head and concatenate the outputs.

to SQuad-Attention, while also reduces the step-count, allowing us to generate videos with far fewer NFEs.

3 Method

We first begin with full softmax Self-Attention, and fix the notation we build on (section 3.1) and then define the SQuad-Attention operator (section 3.2). The broader discussion on latent video diffusion setup, and the details of how we train, are deferred to sections A and B of supplementary respectively. We recommend reading section A prior to continuing.

3.1 Preliminaries: Full softmax Self-Attention

The DiT $\mathcal{D}_\theta$ is a composition of L identical blocks (with $L = 30$ for Wan 2.2 5B). The latent z_σ is first flattened and patchified into a sequence of n token features, which the blocks then refine in tandem:

$$\begin{aligned} \mathbf{X}^{(0)} &= \mathrm{embed}(z_\sigma), \\ \mathbf{X}^{(\ell)} &= \mathcal{B}_\theta^{(\ell)}\big(\mathbf{X}^{(\ell-1)}, \sigma, \mathbf{p}\big), \quad \ell = 1, \dots, L, \end{aligned} \tag{1}$$

where σ is the current noise-level and $\mathbf{p}$ is the text prompt. The flow-velocity is read out from $\mathbf{X}^{(L)}$ by a final projection. Each block $\mathcal{B}_\theta^{(\ell)}$ applies three residual sub-layers — a self-attention over the n video tokens, a cross-attention to the text prompt $\mathbf{p}$, and a token-wise feed-forward network — each modulated by the timestep σ (omitted in following for brevity):

$$\begin{aligned} \mathbf{X} &\leftarrow \mathbf{X} + \mathrm{SelfAttn}(\mathbf{X}), \\ \mathbf{X} &\leftarrow \mathbf{X} + \mathrm{CrossAttn}(\mathbf{X}, \mathbf{p}), \\ \mathbf{X} &\leftarrow \mathbf{X} + \mathrm{FFN}(\mathbf{X}). \end{aligned} \tag{2}$$

SQuad modifies only the self-attention while the cross-attention and the feed-forward network are left untouched. In what follows we therefore drop the block index and write $\mathbf{X} \in \mathbb{R}^{n \times hd}$ for the token features entering a single block's self-attention. This so-called Self-Attention operation that forms the backbone of essentially every modern Transformer, diffusion or otherwise, is remarkably simple. A sequence of n input tokens is first projected into three sequences queries, keys, and values as

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V, \tag{3}$$

with learned projection matrices $\mathbf{W}_{\{Q,K,V\}} \in \mathbb{R}^{hd \times hd}$. Here n is the number of tokens; for video latents it is the size of a spatio-temporal volume, $n = T \times H \times W$, as described earlier. Each of $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ is then split along its feature dimension into h heads of dimension d, giving per-head sequences $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i \in \mathbb{R}^{n \times d}$ for $i \in \{1, \dots, h\}$. A rotary position embedding (RoPE) (Su et al., 2021) is applied to the queries and keys only,

$$\mathbf{Q}_i \leftarrow \mathrm{RoPE}(\mathbf{Q}_i), \qquad \mathbf{K}_i \leftarrow \mathrm{RoPE}(\mathbf{K}_i), \tag{4}$$

while the values $\mathbf{V}_i$ are left unchanged. The full softmax self-attention for the i-th head is then

$$\mathbf{Y}_i \;=\; \mathrm{Attn}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) \;=\; \mathrm{softmax}\!\left(\frac{\mathbf{Q}_i \mathbf{K}_i^\top}{\sqrt{d}}\right)\mathbf{V}_i \tag{5}$$

Such that, $\mathrm{softmax}$ is a row-wise operator and $\mathbf{Y}_i \in \mathbb{R}^{n\times d}$. The per-head outputs are concatenated back along the feature dimension,

$$\mathbf{Y} \;=\; \mathrm{concat}(\mathbf{Y}_1, \ldots, \mathbf{Y}_h) \;\in\; \mathbb{R}^{n\times hd}, \tag{6}$$

yielding an output sequence $\mathbf{Y}$ with the same shape as the input $\mathbf{X}$. The $n \times n$ score matrix $\mathbf{Q}_i\mathbf{K}_i^\top$ inside the softmax is the source of the quadratic cost: forming and applying it takes $\mathcal{O}(n^2)$ work per head.

The Cross-Attention sub-layer is mechanically identical, except that the keys and values are projected from the text-prompt embedding $\mathbf{p}$ rather than from $\mathbf{X}$. Its cost is therefore cheap: the prompt length (512 tokens for Wan 2.2 5B) is typically far smaller than the number of visual tokens n.

3.2 SQUAD

Our proposal is a simple change. Where a standard DiT block computes $\mathbf{Y} = \mathrm{SelfAttn}(\mathbf{X})$ via the full softmax Attn of eq. (5), we compute $\mathbf{Y} \;=\; \mathrm{SQuadAttn}(\mathbf{X})$. leaving every other part of the DiT blocks as it is.

A composition of two attentions. $\mathrm{SQuadAttn}$ is not a new kind of attention. It is the composition of two ordinary softmax attentions, a *local* pass $\mathrm{Attn}\mathcal{L}$ and a *global* pass $\mathrm{Attn}\mathcal{G}$. Writing the composition as a nesting makes the data flow explicit,

$$\mathbf{Y}_i \;=\; \underbrace{\mathrm{Attn}\mathcal{G}\Big(\mathbf{Q}_i,\; \mathbf{K}_i,\; \underbrace{\mathrm{Attn}\mathcal{L}\big(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i\big)}_{\mathbf{Y}_i^l}\Big)}_{\mathbf{Y}_i^g}. \tag{7}$$

Figure 2 gives the conceptual picture: the local pass mixes tokens inside a window, and the global pass then mixes across windows. Note that the two passes are not fused, and neither are their outputs pooled or summed or aggregated. Instead the *value stream acts as a shared scratch* between them. As eq. (7) shows, the local output enters the global pass in the value slot. The global mixing therefore operates on tokens that have already been locally mixed, and a full receptive field is recovered even though neither pass alone is dense (see supplementary section D).

Construction of the local and global token views. Both $\mathrm{Attn}\mathcal{L}$ and $\mathrm{Attn}\mathcal{G}$ are formally the Attn of eq. (5), while they differ only in how the queries and keys are laid out. Crucially, this re-viewing happens *before* the split into heads. Given a window size w, we first rearrange the projected $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n\times hd}$ into a local and a global view,

$$\begin{aligned} \{\mathbf{Q}, \mathbf{K}, \mathbf{V}\}^l &= \mathrm{split}\mathcal{L}\left(\{\mathbf{Q}, \mathbf{K}, \mathbf{V}\}\right) \in \mathbb{R}^{w\times (n/w)\, hd}, \\ \{\mathbf{Q}, \mathbf{K}\}^g &= \mathrm{split}\mathcal{G}\left(\{\mathbf{Q}, \mathbf{K}\}\right) \in \mathbb{R}^{(n/w)\times w\, hd}, \end{aligned} \tag{8}$$

and only then split into heads exactly as in section 3.1. The head dimension d is unchanged; the number of heads absorbs the axis that is no longer attended over. So the local view yields $(n/w)\, h$ heads over sequences of length w, and the global view yields wh heads over sequences of length (n/w). Both rearrangements are pure re-indexings. They carry no parameters and no arithmetic, and each is exactly invertible. We thus refer to their inverses $\mathrm{split}\mathcal{L}^{-1}$ and $\mathrm{split}\mathcal{G}^{-1}$ which simply revert the rearrangement of the tokens.

Concretely, for a video latent with token grid of $T \times H \times W$ size and a window of $w = w_t \times w_h \times w_w$ size, the two views are the `einops` rearrangements

$$\begin{aligned} \mathrm{split}\mathcal{L}:&\ \texttt{b (pt wt) (ph wh) (pw ww) h d} \\ &\to \texttt{b (wt wh ww) (pt ph pw h) d}, \\ \mathrm{split}\mathcal{G}:&\ \texttt{b (pt wt) (ph wh) (pw ww) h d} \\ &\to \texttt{b (pt ph pw) (wt wh ww h) d}, \end{aligned} \tag{9}$$

where `b` is the batch size, and $\texttt{pt} = T/w_t$, $\texttt{ph} = H/w_h$, $\texttt{pw} = W/w_w$ count the windows along each axis. In both cases the second axis of the result is the one attended over and the third collects

Table 1: VBench evaluation comparing our SQuad to the original baseline, DMD distilled original, and various trained as well as training-free efficient attention methods. We show selected representative dimensions here due to space constraints.

Method	VBench			Quality Dimensions							Semantic Dimensions					
	Tot.	Qual.	Sem.	Subj.	Backg.	T.Flick.	Motion	Dyn.	Aesth.	Imag.	Obj.	M-Obj.	Act.	Color	Spat.	Scene
Base DiT: Wan 2.2 5B **Generation resolution:** `81x704x1280` **Token length:** $n = 18480$																
Original	83.08	83.98	79.48	92.56	96.19	99.49	98.11	65.28	66.33	67.75	89.08	71.02	97.20	83.68	81.83	54.10
DMD	82.54	83.36	79.28	94.70	94.69	96.73	96.89	75.83	66.94	69.00	92.85	75.37	98.00	83.50	79.18	52.82
VSA	84.14	84.93	81.00	95.02	95.66	98.36	98.34	66.94	68.62	70.91	94.53	81.14	98.80	84.77	85.71	53.65
Jenga																
– Training free	83.08	84.14	78.82	91.88	94.52	97.47	97.14	87.22	66.73	69.29	90.49	71.57	98.20	84.93	79.06	51.12
– With training	84.36	85.07	81.54	94.44	95.84	99.01	97.72	71.94	68.60	70.18	94.43	79.56	97.60	83.41	87.87	54.99
Radial Attention	84.56	85.46	80.96	94.44	95.85	98.77	97.08	84.17	68.03	69.98	92.53	76.11	98.40	88.90	83.10	53.91
Attention Surgery	83.39	84.55	78.74	94.15	95.25	98.38	97.39	75.00	68.01	69.79	92.04	72.56	97.60	83.12	82.50	51.21
ReHyAt																
– 20 Blocks	83.70	84.46	80.66	96.31	96.52	98.08	97.87	61.94	69.05	69.63	94.87	81.31	97.40	84.69	82.10	53.94
– 30 Blocks	83.22	83.62	81.61	97.06	97.22	98.66	98.18	42.22	68.79	69.84	95.73	84.45	97.00	83.80	84.90	55.99
SQuad (Ours)																
– 20 Blocks	82.90	83.45	80.73	95.12	95.64	97.31	97.27	65.28	68.21	68.93	94.95	82.26	98.20	82.89	85.06	53.71
– 30 Blocks	83.20	83.78	80.88	95.77	96.41	97.79	98.16	57.22	68.70	68.47	95.87	83.23	97.40	82.98	84.35	54.43
Base DiT: Wan 2.1 1.3B **Generation resolution:** `81x480x832` **Token length:** $n = 32760$																
Original	83.26	84.25	79.30	93.05	96.11	99.02	97.92	71.94	67.74	66.27	90.41	74.39	96.80	86.35	77.87	52.71
DMD	83.04	85.07	74.94	93.94	94.78	97.29	97.77	85.28	68.51	70.06	86.41	69.70	97.20	80.88	68.85	48.20
SQuad (Ours)																
– 20 Blocks	82.74	84.43	76.00	93.09	95.03	97.64	97.58	85.56	66.85	67.78	89.21	75.09	97.20	82.45	70.44	44.30
– 30 Blocks	82.70	84.27	76.44	93.21	95.32	98.39	97.77	82.50	65.44	66.50	91.61	75.00	95.20	85.76	70.48	45.62

the remaining positions into the head count, leaving the head dimension d untouched. When the token extents are not divisible by the window shape, our implementation pads the grid and masks the padded positions in the attention.

Given the definitions of the local and global split operations, both $\mathrm{Attn}\mathcal{L}$ and $\mathrm{Attn}\mathcal{G}$ have each the same three steps: split, attend, and unsplit,

$$\begin{aligned} \mathrm{Attn}\mathcal{L}(\cdot) &= \mathrm{split}\mathcal{L}^{-1}\Big(\mathrm{Attn}\left(\mathbf{Q}_i^l, \mathbf{K}_i^l, \mathbf{V}_i^l\right)\Big), \\ \mathrm{Attn}\mathcal{G}(\cdot) &= \mathrm{split}\mathcal{G}^{-1}\Big(\mathrm{Attn}\left(\mathbf{Q}_i^g, \mathbf{K}_i^g, \mathrm{split}\mathcal{G}(\mathbf{Y}_i^l)\right)\Big). \end{aligned} \tag{10}$$

Both operators therefore consume and return tensors of the original shape $\mathbb{R}^{n \times hd}$. This is what makes them composable in eq. (7). Figure 3 illustrates the flow of the operations mathematically defined in eq. (10). We analyze the resulting complexity in section E, and derive the optimal window size in section F of the supplementary material. But even without the derivation, it can be intuitively observed that $w = \sqrt{n}$ is the optimal choice for the window size balancing the number of tokens in the local and the global attention passes, ultimately giving us $\mathcal{O}(n\sqrt{n})$ Sub-Quadratic complexity. Hence the name SQuad.

Distillation. Replacing $\mathrm{SelfAttn}$ with $\mathrm{SQuadAttn}$ changes the function each block computes, so the pretrained weights no longer fit. Rather than train a SQuad model from scratch, we demonstrate that a pretrained quadratic DiT can be fitted into our modified one by distillation, in two stages. We first perform a simple Flow-Matching SFT, which re-seats the network under the new attention. We then apply DMD2 step distillation, which recovers the teacher's generation quality and reduces the sampling NFEs at the same time. We defer the details of both stages to section B of the supplementary.

4 Experiments

4.1 Experimental Setup

Models. We conduct all our experiments on the Wan 2.2 5B as well as the older Wan 2.1 1.3B (Wan Team, 2025).

Metrics. Our setup is designed to evaluate the efficacy of our proposed method over two axes: preservation of the generation quality, and improvement in the efficiency. We evaluate our experiments towards the first axis with the VBench (Huang et al., 2024) suite of metrics as well as human user

Table 2: User study. Base model for all is Wan 2.2 5B.

Baseline Method	Human Preference %		
	SQuad	No pref.	Baseline
Original 100 NFEs	41%	33%	26%
DMD 6 NFEs	33%	37%	30%
DMD Rad. Attn. 6 NFEs	35%	31%	34%

Table 3: Ablation of SFT and DMD for SQuad on 30 Blocks.

SFT	DMD	NFE	VBench		
			Tot.	Qual.	Sem.
✔		100	73.03	75.66	62.48
	✔	6	80.91	82.23	75.65
✔	✔	6	**82.99**	**83.69**	**80.19**

preference study, and along the second using the typical efficiency metrics such as TFLOPs, GPU latency, and NFEs.

Human preference study. To compare two methods, we randomly sampled long gpt-enhanced prompts from VBench and showed the users the prompts with a varying randomly ordered left/right videos and asked them to pick their preferred video or mark the comparison as a tie. This process resulted in 1,179 paired comparisons, on 648 distinct prompts and 1,036 distinct videos, some with the same prompt but varying random seeds.

Datasets. For our Stage-1 SFT we use the videos from the VIPE 1M dataset (Huang et al., 2025a), and the descriptive captions for those generated by us using the Qwen3-8B-Instruct (Yang et al., 2025a) model. We will release these captions for further research. For the Stage-2 DMD2 step distillation, we only use the captions, since this version of DMD bootstraps the student and does not require video data.

4.2 Main Experiments

We first present our evaluation of our proposed method SQuad in comparison to the original baseline models Wan 2.2 5B as well as the older Wan 2.1 1.3B since they operate on different resolutions natively, thus producing token lengths of different sizes. Specifically the token length for the 5B model at 704p resolution is $n = 18480$ while for the 1.3B model at 480p resolution is $n = 32760$. We experiment on these two settings to demonstrate the generalization and robustness of our proposed change, while present a broader comparison against various efficient attention methods on the 5B model.

As presented in table 1 On Wan 2.2 5B, SQuad attains a VBench Total score of 83.20, effectively matching the full softmax Self-Attention of the original DiT 83.08. Crucially, it does so while introducing **zero additional parameters** and with a simple implementation which is hardware device friendly by design, not requiring any specialized GPU fused kernels or on-device optimizations. The competing efficient-attention method, Radial Attention (Li et al., 2025), that posts higher VBech total (84.56) than ours proposes an $\mathcal{O}(n\log(n))$ method, yet, as demonstrated in the table 4, we surpass their latency as well as TFLOPs without requiring specialized GPU kernels in the implementation. Lastly, as presented in table 2, 31% of the users couldn't distinguish between our and their generations while 35% preferred ours. This strengthens the fact that our proposed method holds up to the compared efficient attention methods when it comes to video generation quality.

Table 4 shows that SQuad decisively improves efficiency from the original baseline. On Wan 2.2 5B it reduces the per-block latency from 62.01 to 19.04 ms (3.3× faster), and compute from 9.681 to 5.548 TFLOPs (1.7× fewer), all with zero added parameters, unlike VSA (Zhang et al., 2025a), Attention Surgery (Ghafoorian et al., 2025), and ReHyAt (Ghafoorian & Habibian, 2026), which each carry 72–283M extra weights. We note that the added learnable parameters is presented here merely as a representation of their method's complex nature. Our gains grow with model scale and sequence length. Although we couldn't perform distillation experiments, we still run the efficiency evaluations on the larger Wan 2.1 14B backbone, and find that SQuad reduces compute from 41.959 to 20.223 TFLOPs and latency from 300.29 to 59.24 ms (ref table 4). This is the expected signature of trading an $\mathcal{O}(n^2)$ operation for an $\mathcal{O}(n\sqrt{n})$ one: the deeper the token grid, the larger the payoff. Supplementary section H explores the FLOPs scaling curves in more details. Lastly, beyond the primary Wan 2.2 5B model, we apply the identical SQuad distillation to Wan 2.1 1.3B to test that the method is not tied to a single backbone or single token-length setting. The efficiency results (Table 4) transfer directly; the corresponding Wan 2.1 quality evaluations are reported in Table 1.

Table 4: Efficiency comparisons

Method	NFE	#Par.	Lat.	TFLOPs
Base DiT: Wan 2.2 5B (81x704x1280)				
Original DiT	100	0	62.01ms	9.7
VSA	6	283M	24.13ms	6.2
Jenga	6	0	22.63ms	6.2
Radial Attention	6	0	25.50ms	7.0
Attention Surgery	6	72M	33.53ms	6.8
ReHyAt	6	72M	58.56ms	6.5
SQuad (Ours)	6	0	19.04ms	5.5
Other variants (81x480x832)				
Wan 2.1 1.3B Orig.	100	0	88.36ms	9.4
– **SQuad (Ours)**	6	0	15.22ms	2.9
Wan 2.1 14B Orig.	100	0	300.29ms	41.9
– **SQuad (Ours)**	6	0	59.24ms	20.2

Table 5: End-to-end latency of a single DiT forward pass, measured in PyTorch eager mode and under `torch.compile` (`max-autotune`). All rows are the Wan 2.2 5B backbone at `81x704x1280`. SQuad is the fastest in both modes.

Method	NFE	End-to-End DiT Latency	
		Eager	Compiled
Original DiT	100	870ms	667ms
VSA	6	724ms	434ms
Jenga	6	680ms	427ms
Radial Attention	6	765ms	619ms
Attention Surgery	6	1006ms	537ms
ReHyAt	6	1757ms	544ms
SQuad (Ours)	6	**520ms**	**314ms**

4.3 Hardware Compilation and Performance

The complexity analysis of section E establishes that SQuad reduces the attention cost from $\mathcal{O}(n^2)$ to $\mathcal{O}(n\sqrt{n})$, and the Block-level TFLOPs counts as well as the Block-level Latencies of table 4 confirm that this translates into a real reduction in arithmetic work. Neither quantity, however, is what a practitioner actually waits for. Asymptotic complexity ignores constants, memory traffic, and kernel efficiency; FLOP counts credit every multiply-add equally, even though a large fused GEMM and a bandwidth-bound softmax over an $n \times n$ score matrix reach wildly different fractions of peak throughput. An attention operator can therefore look cheap on paper and still fail to deliver, either because the work it removes was never the bottleneck or because the reshaping it introduces costs more than the multiplications it saves. The only way to settle the question is to measure wall-clock time on the device. We report that measurement here, in both of the execution modes a user is likely to deploy: PyTorch *eager* mode and a `torch.compile` *compiled* mode in table 5.

What we time. We time a *single forward pass of the DiT* — the quantity that the sampler invokes once per function evaluation — rather than the whole generation pipeline. Text encoding and VAE decoding are excluded: they are shared verbatim by every method we compare and would otherwise dilute the effect under study. Timing uses pairs of `torch.cuda.Event(enable_timing=True)` objects registered as forward-pre and forward hooks on the transformer module. Because the hooks sit on the outermost module wrapper and only *record* events, they measure the compiled region without entering it, and so remain valid under `fullgraph=True` compilation; the same instrumentation is used unchanged in both modes. Each measured interval is bracketed on the CUDA stream and read back only after a `torch.cuda.synchronize()`, so the reported numbers are true device latencies and contain no host-side asynchrony artifacts. Classifier-free guidance is disabled ($\omega = 0$; recall from section B that CFG is distilled *into* the student), so each denoising step issues exactly one transformer forward and our NFE count equals the number of timed forwards. More details in supplementary section G.

Protocol and statistics. We profile a fixed list of 30 prompts spanning varied scenes, subjects, and camera motion. The first 5 prompts are executed but discarded: they populate the allocator caches, trigger cuDNN/cuBLAS autotuning, and—in compiled mode—absorb the one-off graph capture and `max-autotune` kernel search, none of which reflect steady-state cost. The remaining 25 prompts are *counted*, and with NFE $= 6$ this yields $25 \times 6 = 150$ timed forward passes per mode, which we pool before aggregating. We report the mean over this pooled sample. The RNG is re-seeded before every prompt, so a given prompt sees identical noise in the eager and compiled passes and the two modes are compared on exactly the same work. Every method in table 5 is measured with this identical harness, on the same machine, in the same session-level environment.

4.4 Ablation Experiments

We now finally detail the process of experimentation through which we found the best configuration of our proposed SQuad method reported in table 1. In principle, all design choices following the $\mathcal{O}(n\sqrt{n})$ framework work reasonably, but since we are not training from scratch and distilling a

Table 7: Ablation of local/global attention and their ordering on SQuad applied to all 30 Blocks. Attention latencies are in ms and × columns are improvements relative to the Original Attention.

Method	Local	Global	Order	Block TFLOPs	Block Lat.	Attn. TFLOPs	Attn. Lat.	TFLOPs ×	Lat. ×	VBench		
										Tot.	Qual.	Sem.
Original	–	–	–	9.680	61.80	4.205	47.10	1.0	1.0	83.08	83.98	79.48
SQuad	✔		–	5.523	19.46	0.038	3.61	110.7	13.0	62.62	73.65	18.50
		✔	–	5.509	17.86	0.025	2.88	168.2	16.4	62.63	71.48	27.24
(Ours)	✔	✔	L→G	5.548	19.55	0.063	4.27	66.7	11.0	83.20	83.78	80.88
	✔	✔	G→L	5.548	19.93	0.063	4.25	66.7	11.1	82.99	83.69	80.19
	✔	✔	alternate	–	–	–	–	–	–	82.87	83.57	80.06

pretrained model, some choices work better. Since it made more conceptual sense to us, we actually started the experiments with the global → local order of the proposed attentions.

Both training stages are necessary. For now. Thus, on the global → local order of application, Table 3 disentangles Stage-1 SFT from Stage-2 DMD2. It is clear that when all 30 blocks of the model are replaced with SQuad-Attention: SFT alone collapses (VBench Tot. 73.03), and DMD2 alone reaches only 80.91, but the two stages *together* recover the full 82.99 VBench Tot.

Table 6: Window characterization ablation. With $N = 18480$ tokens the window target is $\lceil \sqrt{N} \rceil = 136$; "Target diff." is each window's #Tok. minus that target.

Window shape	Dims	#Tok.	Target diff.	VBench Tot.
AR-preserving	4×5×7	140	+4	81.99
Spat.	1×9×16	144	+8	82.31
Spat. isotropic	1×12×12	144	+8	81.55
Temp.	21×2×4	168	+32	**82.99**
Temp. isotropic	21×3×3	189	+53	81.80
Isotropic (small)	5×5×5	125	−11	81.55
Isotropic (large)	6×6×6	216	+80	81.69

Window characterization. Then the next question is what shape or characterization of the windows should we use. Table 6 studies how the choice of window geometry, at a roughly fixed token budget near the sub-quadratic target $\lceil \sqrt{n} \rceil = 136$, for $n = 18480$ affects quality, contrasting aspect-ratio preserving, spatial-only, temporal-only, and isotropic windows. We find that the temporal windows are the easiest to train and produce the best quality videos. This could be attributed to the asymetric nature of the DiT patchification (1x temporal while 2x spatial), but further analysis is required to reason properly.

The need for and the order of the two passes. Finally, we attempt to experimentally answer the question: what order should the two passes run in. Table 7 shows that, used in isolation, the local-only and global-only passes are the cheapest but don't work properly as the restricted receptive field limits their generation quality strongly. Composing the two restores full quality, and a uniform local → global pass is the best with VBench total of 83.20 We also experimented with blockwise alternating order of the two passes, and found it to perform worse than the uniform local → global pass.

5 Conclusion

We introduced SQuad, a Sub-Quadratic Attention Distillation framework that makes the attention in a pretrained video DiT sub-quadratic *without* abandoning the softmax that gives it its expressivity. Beyond the concrete efficiency gains, our broader aim is to shine light on a simple hypothesis: the full $\mathcal{O}(n^2)$ complexity of token-to-token communication may not be necessary. Cheaper *factorizations that keep the softmax intact*, rather than replace it with linear or low-rank surrogates, are an under-explored middle ground. We hope SQuad encourages this direction not only as a distillation target for existing models, but also as a candidate attention mechanism for pretraining large models from scratch.

Limitations and future work. First, our distillation couples SQuad attention with *step* distillation (DMD2); a purely Flow-Matching instantiation would isolate the attention change from the sampling speedup. Second, composing *more* than two reduced-complexity passes could trade a little depth for a cheaper, more expressive operator. Finally, SQuad is not specific to video; applying it to other modalities is a promising avenue.

AI USE STATEMENT

We used AI assistants for writing and presentation: drafting, refining, and copy-editing the text of this paper, and preparing the supplementary project webpage. The core contribution, the SQuad Attention operator itself, was implemented entirely by hand, validated against a suite of developer tests, and peer-reviewed by the authors; no AI system was involved in writing that code. Some of the downstream experiments built on top of that implementation were carried out with the assistance of coding agents. All results, figures, and claims reported in this paper were verified by the authors.

ETHICS STATEMENT

SQuad makes an existing video diffusion transformer cheaper to run rather than extending what such a model can generate. It therefore inherits the characteristics of the pretrained model it distills, including any biases in that model's training data, and it introduces no new data collection. Two considerations are worth naming explicitly. First, substantially reducing the cost of video generation lowers the barrier to both beneficial and harmful use; the same efficiency that makes these models practical inside creative tools also makes large-scale generation of misleading or non-consensual synthetic video cheaper. We believe efficiency research of this kind is better conducted openly, where the resulting models can be studied and where provenance and watermarking tools can be developed against them, but the trade-off is real and we do not claim it away.
Second, our human preference study involved voluntary participants comparing pairs of generated videos. No personal or identifying information was collected, and all prompts were drawn from the public VBench suite, so participants saw no content beyond what that benchmark already contains. Most importantly, the reduction in attention cost translates directly into lower energy consumption per generated video, which we regard as a humble positive outcome of our work.

REPRODUCIBILITY STATEMENT

The change we propose is deliberately simple. It replaces the full attention in each transformer block with two successive reduced attentions, one over local windows and one over global window summaries, and requires no operator beyond the standard scaled dot-product attention already present in the base model. Additionally, in order to support reproduction and follow-up work we will release the full inference and training code, the distilled model weights, and the ViPE captions used for training upon acceptance.

REFERENCES

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *arXiv preprint arXiv:2406.07507*, 2024.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. *OpenAI Technical Report*, 2024.

Junsong Chen, Yuyang Zhao, Jincheng Yu, Ruihang Chu, Junyu Chen, Shuai Yang, Xianbang Wang, Yicheng Pan, Daquan Zhou, Huan Ling, Haozhe Liu, Hongwei Yi, Hao Zhang, Muyang Li, Yukang Chen, Han Cai, Sanja Fidler, Ping Luo, Song Han, and Enze Xie. Sana-video: Efficient video generation with block linear diffusion transformer. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.24695.

Pengtao Chen, Xianfang Zeng, Maosen Zhao, Peng Ye, Mingzhu Shen, Wei Cheng, Gang Yu, and Tao Chen. Sparse-vdit: Unleashing the power of sparse attention to accelerate video diffusion transformers. *arXiv preprint arXiv:2506.03065*, 2025. arXiv:2506.03065.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations (ICLR)*, 2021.

Justin Cui, Jie Wu, Ming Li, Tao Yang, Xiaojie Li, Rui Wang, Andrew Bai, Yuanhao Ban, and Cho-Jui Hsieh. Self-forcing++: Towards minute-scale high-quality video generation. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.02283.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Mohsen Ghafoorian and Amirhossein Habibian. Rehyat: Recurrent hybrid attention for video diffusion transformers. *arXiv preprint arXiv:2601.04342*, 2026. arXiv:2601.04342.

Mohsen Ghafoorian, Denis Korzhenkov, and Amirhossein Habibian. Attention surgery: An efficient recipe to linearize your video diffusion transformer. *arXiv preprint arXiv:2509.24899*, 2025.

Mohsen Ghafoorian, Denis Korzhenkov, Adil Karjauv, Ioannis Lelekas, Noor Fathima, Spyridon Stasis, Hanno Ackermann, Boris van Breugel, Markus Nagel, Fatih Porikli, Animesh Karnewar, and Amirhossein Habibian. Mobilewan: Closing the quality gap for mobile video diffusion. *arXiv preprint arXiv:2607.06173*, 2026.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *Conference on Language Modeling (COLM)*, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*, 2022.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

Jiahui Huang, Qunjie Zhou, Hesam Rabeti, Aleksandr Korovko, Huan Ling, Xuanchi Ren, Tianchang Shen, Jun Gao, Dmitry Slepichev, Chen-Hsuan Lin, Jiawei Ren, Kevin Xie, Joydeep Biswas, Laura Leal-Taixé, and Sanja Fidler. ViPE: Video pose engine for 3d geometric perception. *arXiv preprint arXiv:2508.10934*, 2025a. doi: 10.48550/arXiv.2508.10934. URL https://arxiv.org/abs/2508.10934.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint arXiv:2506.08009*, 2025b. NeurIPS 2025 (Spotlight); arXiv:2506.08009.

Yufei Huang, Liangyu Yuan, Changxi Chi, Yunfan Liu, Cheng Tan, Siyuan Li, Jingbo Zhou, Haitao Lin, Chang Yu, and Stan Z. Li. Steinsgate: Adding causality to diffusions for long video generation via path integral. In *International Conference on Learning Representations (ICLR)*, 2026. OpenReview:8WS5nDWIWE.

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. doi: 10.1109/CVPR52733.2024.02060.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.05954.

Animesh Karnewar, Denis Korzhenkov, Ioannis Lelekas, Adil Karjauv, Noor Fathima, Hanwen Xiong, Vancheeswaran Vaidyanathan, Will Zeng, Rafael Esteves, Tushar Singhal, Fatih Porikli, Mohsen Ghafoorian, and Amirhossein Habibian. Neodragon: Mobile video generation using diffusion transformer. *arXiv preprint arXiv:2511.06055*, 2025.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning (ICML)*, 2020.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.02279.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. Repa-e: Unlocking vae for end-to-end tuning with latent diffusion transformers. *arXiv preprint arXiv:2504.10483*, 2025.

Wuyang Li, Wentao Pan, Po-Chien Luan, Yang Gao, and Alexandre Alahi. Stable video infinity: Infinite-length video generation with error recycling. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.09212.

Xingyang Li, Muyang Li, Tianle Cai, Haocheng Xi, Shuo Yang, Yujun Lin, Lvmin Zhang, Songlin Yang, Jinbo Hu, Kelly Peng, Maneesh Agrawala, Ion Stoica, Kurt Keutzer, and Song Han. Radial attention: $o(n \log n)$ sparse attention with energy decay for long video generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.19852.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avshalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Shanchuan Lin, Xin Xia, Yuxi Ren, Ceyuan Yang, Xuefeng Xiao, and Lu Jiang. Diffusion adversarial post-training for one-step video generation. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2501.08316.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Jinxiu Liu, Xuanming Liu, Kangfu Mei, Yandong Wen, Ming-Hsuan Yang, and Weiyang Liu. Streaming autoregressive video generation via diagonal distillation. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2603.09488.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024. ICLR 2025 (Oral); arXiv:2410.11081.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. arXiv:2210.03142.

NVIDIA. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. arXiv:2112.10752.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation. *arXiv preprint arXiv:2506.14603*, 2025.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2202.00512.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision (ECCV)*, 2024. arXiv:2311.17042.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2303.01469.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Ziqiao Wang, Wangbo Zhao, Yuhao Zhou, Zekai Li, Zhiyuan Liang, Mingjia Shi, Xuanlei Zhao, Pengfei Zhou, Kaipeng Zhang, Zhangyang Wang, Kai Wang, and Yang You. Repa works until it doesn't: Early-stopped, holistic alignment supercharges diffusion training. *arXiv preprint arXiv:2505.16792*, 2025.

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. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2502.01776.

Weilai Xiang, Hongyu Yang, Di Huang, and Yunhong Wang. Denoising diffusion autoencoders are unified self-supervised learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. arXiv:2303.09769.

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*, 2025a. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, Song Han, and Yukang Chen. Longlive: Real-time interactive long video generation. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.22622.

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, Jianfei Chen, Song Han, Kurt Keutzer, and Ion Stoica. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *arXiv preprint arXiv:2505.18875*, 2025b. NeurIPS 2025 (Spotlight); arXiv:2505.18875.

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, Jianfei Chen, Song Han, Kurt Keutzer, and Ion Stoica. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *arXiv preprint arXiv:2505.18875*, 2025c. NeurIPS 2025 (Spotlight); arXiv:2505.18875.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024a. arXiv:2405.14867.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b. arXiv:2311.18828.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. arXiv:2412.07772.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.06940.

Yuanhao Zhai, Kevin Lin, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Chung-Ching Lin, David Doermann, Junsong Yuan, and Lijuan Wang. Motion consistency model: Accelerating video diffusion with disentangled motion-appearance distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.06890.

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. Vsa: Faster video diffusion with trainable sparse attention. *arXiv preprint arXiv:2505.13389*, 2025a. arXiv:2505.13389.

Yuechen Zhang, Jinbo Xing, Bin Xia, Shaoteng Liu, Bohao Peng, Xin Tao, Pengfei Wan, Eric Lo, and Jiaya Jia. Training-free efficient video generation via dynamic token carving. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b. arXiv:2505.16864.

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

SQuad: Sub-Quadratic Attention Distillation for Efficient Video Generation

Supplementary Material

A Preliminaries: Latent Video Diffusion Transformers

Modern text-to-video generators are, almost universally, latent video Diffusion Transformers (DiTs) trained in two stages (Rombach et al., 2022): a variational autoencoder is first learned as a *continuous tokenizer*, and a DiT is then trained to generate in that continuous tokenized latent space with a flow-matching objective.

Stage 1 — continuous tokenizer. A causal 3D variational autoencoder is trained to compress and reconstruct videos. An encoder $\mathcal{E}_\psi$ maps an RGB clip $Z \in \mathbb{R}^{3\times F\times H_p\times W_p}$ (with F frames of spatial size $H_p \times W_p$) to the parameters of a Gaussian latent posterior,

$$(\boldsymbol{\mu}, \boldsymbol{s}) \;=\; \mathcal{E}_\psi(Z), \qquad z \;=\; \boldsymbol{\mu} + \boldsymbol{s} \odot \boldsymbol{\epsilon}', \;\; \boldsymbol{\epsilon}' \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{A1}$$

where the reparameterized latent $z \in \mathbb{R}^{c\times T\times H\times W}$ is far smaller than Z, and a decoder $\mathcal{G}_\psi$ reconstructs the video, $\hat{Z} = \mathcal{G}_\psi(z)$. The pair is trained end-to-end with a reconstruction loss and a KL term that pulls the posterior towards a unit Gaussian,

$$\mathcal{L}_{\text{VAE}} \;=\; \underbrace{\big\|Z - \mathcal{G}_\psi(z)\big\|}_{\text{reconstruction}} + \beta\, \underbrace{D_{\text{KL}}\big(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{s}^2)\,\big\|\,\mathcal{N}(\mathbf{0}, \mathbf{I})\big)}_{\text{latent regularization}}. \tag{A2}$$

In practice $\mathcal{L}_{\text{VAE}}$ is often augmented with perceptual and adversarial losses to train more robust tokenizers. For Wan 2.2 5B the VAE compresses by $\sim 4\times$ temporally and $16\times$ spatially into $c = 48$ latent channels. Once trained, it is frozen and serves as the tokenizer for Stage 2: the latent grid $T \times H \times W$ is flattened into a sequence of $n = THW$ tokens that the DiT operates on.

Stage 2 — flow-matching DiT. A DiT $\mathcal{D}_\theta$ is trained over the frozen latents with the rectified-flow (flow-matching) objective (Lipman et al., 2022). Taking a clean latent z_0 from the Stage-1 encoder and Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the noisy latent at level $\sigma \in [0, 1]$ is the straight-line interpolant

$$z_\sigma \;=\; (1 - \sigma)\, z_0 + \sigma\, \boldsymbol{\epsilon}, \tag{A3}$$

and $\mathcal{D}_\theta$ is trained to predict the constant velocity $\boldsymbol{\epsilon} - z_0$ of this path which points from the data towards the unit Gaussian by minimizing

$$\mathcal{L}_{\text{FM}} \;=\; \mathbb{E}_{z_0,\,\boldsymbol{\epsilon},\,\sigma}\big\|\,\mathcal{D}_\theta(z_\sigma, \sigma, \mathbf{p}) - (\boldsymbol{\epsilon} - z_0)\,\big\|_2^2, \tag{A4}$$

where $\mathbf{p}$ is the text-prompt embedding. In practice z_0 is a reparameterized sample z (eq. (A1)) for robustness, though using the posterior mean $\boldsymbol{\mu}$ instead also works comparably.

Sampling. At inference, generation runs the flow in reverse. Starting from pure noise $z_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and conditioned on a prompt $\mathbf{p}$, we integrate the learned velocity field from $\sigma = 1$ to $\sigma = 0$ with iterative Euler steps,

$$z_{\sigma-\Delta\sigma} \;=\; z_\sigma - \Delta\sigma\, \mathcal{D}_\theta(z_\sigma, \sigma, \mathbf{p}), \tag{A5}$$

over a schedule of Neural Functional Evaluations (NFEs) to obtain the clean latent z_0. The decoder $\mathcal{G}_\psi$ finally maps it back to an RGB video, $\hat{Z} = \mathcal{G}_\psi(z_0)$. We note that several further ingrained details such as patchification of the latent grid, and classifier-free guidance (which doubles the number of Euler-step NFEs) are standard de-facto practice; we omit them here for lucidity, but note that they are present and duly handled in all our experiments.

B Method: Two-stage distillation

Replacing full attention with SQuad changes the function the network computes, so the pretrained weights no longer fit. We recover —and then accelerate— the model in two stages. Stage 1 re-fits the SQuad network to the teacher's behaviour with the original flow-matching objective. Stage 2 distills the many-step model into a few-step generator with distribution-matching distillation. Throughout, the frozen pretrained Wan 2.2 model serves as the teacher.

Stage 1: flow-matching supervised fine-tuning. We instantiate the SQuad transformer $\mathcal{D}_\theta$ from the pretrained weights and fine-tune it with the same rectified-flow loss used to train the backbone. For a clean latent z_0, noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and a sampled noise level σ, we form the interpolant z_σ of eq. (A3) and regress the velocity:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) \;=\; \mathbb{E}_{z_0,\,\boldsymbol{\epsilon},\,\sigma}\Big\| \mathcal{D}_\theta(z_\sigma, \sigma, \mathbf{p}) \;-\; \big(\boldsymbol{\epsilon} - z_0\big) \Big\|_2^2. \tag{A6}$$

The noise level σ is drawn from a logit-normal schedule. This stage adapts the query/key/value and output projections — and the rest of the network — to the new attention pattern, restoring generation quality under a standard 50-step sampler. We train Stage 1 for 8k iterations.

Stage 2: distribution-matching distillation. Stage 1 yields a competitive but still multi-step model. To make it fast we distill it into a few-step generator with DMD2. Three networks are involved: the few-step *student* generator G_θ (initialized from the Stage 1 SQuad model), a frozen *teacher* $\mathcal{D}_\phi^{\mathrm{tea}}$ (the pretrained model, giving the real score), and a trainable *critic* $\mathcal{D}_\psi^{\mathrm{fake}}$ (a copy of the SQuad model that tracks the student's own distribution).

The student maps noise to a clean latent in a few steps; denote a generated sample $\hat{z}_0$. Following the flow parametrization, from a prediction at level σ we read off the implied clean latent $\hat{z}_0 = z_\sigma - \sigma\, \mathcal{D}_\theta(z_\sigma, \sigma, \mathbf{p})$. DMD matches the distribution of $\hat{z}_0$ to the teacher's by pushing the student down the gradient of the KL divergence between the two, which reduces to the difference of two scores evaluated on a re-noised sample $\hat{z}_\tau = (1-\tau)\hat{z}_0 + \tau\, \boldsymbol{\epsilon}'$ at a randomly sampled level τ:

$$\nabla_\theta \mathrm{KL} \;\propto\; \mathbb{E}\Big[\big(s_{\mathrm{fake}}(\hat{z}_\tau, \tau) \;-\; s_{\mathrm{real}}(\hat{z}_\tau, \tau)\big)\, \frac{\partial \hat{z}_0}{\partial \theta} \Big]. \tag{A7}$$

Here s_{fake} is the critic's velocity/score on the student distribution and s_{real} is the teacher's. The teacher score uses classifier-free guidance,

$$s_{\mathrm{real}} \;=\; \mathcal{D}_\phi^{\mathrm{tea}}(\hat{z}_\tau, \tau, \varnothing) \;+\; \omega\, \big(\mathcal{D}_\phi^{\mathrm{tea}}(\hat{z}_\tau, \tau, \mathbf{p}) - \mathcal{D}_\phi^{\mathrm{tea}}(\hat{z}_\tau, \tau, \varnothing)\big), \tag{A8}$$

with guidance scale ω and $\varnothing$ the null prompt. In practice we normalize the distribution-matching gradient by its magnitude and inject it through a stop-gradient surrogate loss, so that $-\nabla_\theta \mathcal{L}_{\mathrm{DMD}}$ equals the gradient in eq. (A7).

The critic is what makes eq. (A7) usable: it must estimate the score of the *current* student distribution, which drifts as θ updates. We therefore train $\mathcal{D}_\psi^{\mathrm{fake}}$ online, by flow-matching on the student's own samples,

$$\mathcal{L}_{\mathrm{critic}}(\psi) \;=\; \mathbb{E}\Big\| \mathcal{D}_\psi^{\mathrm{fake}}(\hat{z}_\tau, \tau, \mathbf{p}) \;-\; \big(\boldsymbol{\epsilon}' - \hat{z}_0\big) \Big\|_2^2. \tag{A9}$$

Generator and critic are optimized on a two-timescale schedule: the critic is updated several times per generator update so that its score estimate stays accurate. We run Stage 2 for 15–30k iterations, distilling down to a 6-step generator. Because classifier-free guidance is itself distilled into the student, the reported NFEs are literal forward passes of the network.

C Related Work: Text-to-Video Generation

Diffusion models have rapidly become the dominant paradigm for text-to-video generation. The transition from U-Net backbones to the Diffusion Transformer (DiT) (Peebles & Xie, 2023) architecture, which replaces convolutions with a scalable stack of self-attention blocks operating on spatio-temporal latent tokens, has been central to the recent leap in visual quality and temporal coherence. Building on this method, a broad family of large-scale video generators has emerged, including HunyuanVideo (Kong et al., 2024), Open-Sora (Zheng et al., 2024), Pyramidal Flow (Jin et al., 2025), LTX-Video (HaCohen et al., 2024), Cosmos (NVIDIA, 2025), Neodragon (Karnewar et al., 2025), and the Wan 2.1/2.2 suite (Wan Team, 2025). We conduct our experiments primarily on the Wan 2.2 model, chosen for its architectural relevance, simplicity, and, above all, its popularity as an open video foundation model.

D Full Receptive Field in a Single SQuad Layer

In section 3.2 we claimed that although neither the local pass $\mathrm{Attn}\mathcal{L}$ nor the global pass $\mathrm{Attn}\mathcal{G}$ is dense on its own, their composition recovers a full receptive field within a single layer. We prove that here.

Indexing. It is convenient to name tokens by their position in the windowed view rather than by their flat index. Let $p = n/w$ be the number of windows and index every token by the pair (c, j), where $c \in \{1, \dots, p\}$ is its window and $j \in \{1, \dots, w\}$ is its slot inside that window. The two rearrangements of eq. (8) attend over exactly one of these indices each: $\mathrm{split}\mathcal{L}$ attends over j at fixed c, and $\mathrm{split}\mathcal{G}$ attends over c at fixed j. Writing the resulting softmax weights for head h as

$$\beta^{c,h}_{j\to j'} \;=\; \underset{j'}{\mathrm{softmax}}\Big(\tfrac{1}{\sqrt{d}}\,\langle \mathbf{Q}_{c,j,h},\, \mathbf{K}_{c,j',h}\rangle\Big) \tag{A10}$$

for the local pass, and

$$\alpha^{j,h}_{c\to c'} \;=\; \underset{c'}{\mathrm{softmax}}\Big(\tfrac{1}{\sqrt{d}}\,\langle \mathbf{Q}_{c,j,h},\, \mathbf{K}_{c',j,h}\rangle\Big) \tag{A11}$$

for the global pass, the two passes read

$$\begin{aligned} \big(\mathbf{Y}_l\big)_{c,j,h} &= \sum_{j'=1}^{w} \beta^{c,h}_{j\to j'}\, \mathbf{V}_{c,j',h}, \\ \big(\mathbf{Y}_g\big)_{c,j,h} &= \sum_{c'=1}^{p} \alpha^{j,h}_{c\to c'}\, \big(\mathbf{Y}_l\big)_{c',j,h}. \end{aligned} \tag{A12}$$

Each is a proper softmax attention: the weights are non-negative and $\sum_{j'} \beta^{c,h}_{j\to j'} = \sum_{c'} \alpha^{j,h}_{c\to c'} = 1$.

Unrolling the composition. Substituting the first line of eq. (A12) into the second gives the output of a single SQuad layer at token (c, j),

$$\begin{aligned} \big(\mathbf{Y}_g\big)_{c,j,h} &= \sum_{c'=1}^{p} \alpha^{j,h}_{c\to c'} \sum_{j'=1}^{w} \beta^{c',h}_{j\to j'}\, \mathbf{V}_{c',j',h} \\ &= \sum_{c'=1}^{p} \sum_{j'=1}^{w} \underbrace{\alpha^{j,h}_{c\to c'}\, \beta^{c',h}_{j\to j'}}\, \gamma^{h}_{(c,j)\to(c',j')}\, \mathbf{V}_{c',j',h}. \end{aligned} \tag{A13}$$

The double sum ranges over *all* $p \cdot w = n$ tokens. So the output at (c, j) is a weighted average of every value in the sequence, with effective weights $\gamma^{h}_{(c,j)\to(c',j')} = \alpha^{j,h}_{c\to c'}\, \beta^{c',h}_{j\to j'}$. These form a valid distribution as well, since

$$\sum_{c'} \sum_{j'} \gamma^{h}_{(c,j)\to(c',j')} \;=\; \sum_{c'} \alpha^{j,h}_{c\to c'} \sum_{j'} \beta^{c',h}_{j\to j'} \;=\; 1. \tag{A14}$$

Reachability. Concretely, information travels from any source token (c', j') to any target token (c, j) along a two-hop path. The local pass first carries (c', j') to (c', j), moving within the source window to the slot j that the target occupies. The global pass then carries (c', j) to (c, j), moving across windows at that shared slot. Both hops exist for every choice of indices, so every source reaches every target.

Two remarks follow. First, the effective weight $\gamma^{h}_{(c,j)\to(c',j')}$ is a product of two softmax weights, so it is strictly positive whenever both hops are, meaning the layer has no structural zeros in its receptive field. Second, the reachability is achieved in one layer, not accumulated over depth as in strictly local attention such as sliding windows. SQuad is therefore a drop-in replacement for full attention rather than a local approximation of it, which is what allows a pretrained quadratic model to be fitted to it by distillation alone.

On the ordering. The argument is symmetric in the two passes. Running global before local gives $\gamma^{h}_{(c,j)\to(c',j')} = \beta^{c,h}_{j\to j'}\, \alpha^{j',h}_{c\to c'}$, which is again a full receptive field, differing only in which intermediate token the two hops route through: (c, j') instead of (c', j). The two orders are therefore equally dense but not identical as functions, which is what motivates the ordering ablation of table 7.

E Complexity of SQuad Attention

We derive here the cost of a single SQuad layer as a function of the window size w, counting the work spent forming and applying the softmax score matrices. Throughout, n is the sequence length, h the number of heads, d the head dimension, and $p = n/w$ the number of windows.

Cost of one softmax attention. For a batch of B independent attentions, each over a sequence of length s with head dimension d, the two matrix products inside eq. (5) cost

$$\underbrace{B \cdot s^2 \cdot d}_{\mathbf{Q}\mathbf{K}^\top} + \underbrace{B \cdot s^2 \cdot d}_{\text{applying to } \mathbf{V}} = \mathcal{O}\big(B\, s^2\, d\big). \tag{A15}$$

We use this as the unit of accounting. The full softmax attention of section 3.1 is the case $B = h$, $s = n$, giving the familiar $\mathcal{O}(h\, n^2\, d)$.

Cost of the two passes. The rearrangements of eq. (9) determine B and s for each pass. Recall that the attended axis becomes the sequence, and the remaining positions are folded into the head count. For the *local* pass, the sequence is the within-window position, so $s = w$, and the head count absorbs the p windows, so $B = p\, h$. Substituting into eq. (A15),

$$\mathcal{C}_{\mathcal{L}} = \underbrace{(p\, h)}_{\text{attentions}} \cdot \underbrace{w^2}_{\text{length}^2} \cdot \underbrace{d}_{\text{feat.}} = h\, d\, \frac{n}{w}\, w^2 = h\, d\, n\, w. \tag{A16}$$

For the *global* pass, the sequence is the window index, so $s = p = n/w$, and the head count absorbs the w within-window slots, so $B = w\, h$. This gives

$$\mathcal{C}_{\mathcal{G}} = \underbrace{(w\, h)}_{\text{attentions}} \cdot \underbrace{\Big(\frac{n}{w}\Big)^2}_{\text{length}^2} \cdot \underbrace{d}_{\text{feat.}} = h\, d\, \frac{n^2}{w}. \tag{A17}$$

Total. Since SQuad composes the two passes, their costs add, and the total attention cost of a SQuad layer is

$$\mathcal{C}(w) = \mathcal{C}_{\mathcal{L}} + \mathcal{C}_{\mathcal{G}} = h\, d \left(n\, w + \frac{n^2}{w} \right). \tag{A18}$$

The two terms pull in opposite directions. Enlarging the window makes the local pass more expensive (longer sequences within a window) and the global pass cheaper (fewer windows to attend across); shrinking it does the reverse. Note also the two extremes: $w = n$ places every token in a single window and recovers full quadratic attention through the local pass, while $w = 1$ does the same through the global pass. The interesting regime is in between.

F The Optimal Window Size

Because eq. (A18) is a sum of one term increasing in w and one decreasing in w, it has an interior minimum. Treating w as continuous and differentiating,

$$\frac{\mathrm{d}\mathcal{C}}{\mathrm{d}w} = h\, d \left(n - \frac{n^2}{w^2} \right), \tag{A19}$$

and setting this to zero gives

$$n = \frac{n^2}{w^2} \quad \Longrightarrow \quad w^2 = n \quad \Longrightarrow \quad w^\star = \sqrt{n}. \tag{A20}$$

This is a minimum rather than a maximum, since $\mathrm{d}^2\mathcal{C}/\mathrm{d}w^2 = 2\, h\, d\, n^2/w^3 > 0$ for all $w > 0$. Equivalently, $w^\star$ is the point at which the two terms of eq. (A18) balance: substituting $w = \sqrt{n}$ makes both $n\, w$ and n^2/w equal to $n^{3/2}$.

Resulting complexity. Evaluating eq. (A18) at the optimum,

$$\mathcal{C}(w^\star) = h\, d \left(n\sqrt{n} + \frac{n^2}{\sqrt{n}} \right) = 2\, h\, d\, n^{3/2} = \mathcal{O}\big(n\sqrt{n}\big). \tag{A21}$$

Setting the window size to $\sqrt{n}$ is therefore exactly the regime in which SQuad attains sub-quadratic $\mathcal{O}(n\sqrt{n})$ complexity, at the cost of only two softmax passes instead of one. This is the configuration of our best model. In the video setting w must additionally factor into a valid window shape $w_t w_h w_w$, so we pick the admissible shape closest to $\sqrt{n}$ while preserving the aspect ratio of the latent volume; table 7 and our window-shape ablation study the effect of departing from this choice.

Table A1: Ablation of local/global attention and their ordering (SQuad 20 Blocks). Best results are in **bold**. Attention TFLOPs/Latency are residuals (attn1 minus its projection/norm submodules); latencies are in ms and × columns are improvements relative to the Original.

Method	Local	Global	Order	Block TFLOPs	Block Lat.	Attn. TFLOPs	Attn. Lat.	TFLOPs ×	Lat. ×	VBench		
										Tot.	Qual.	Sem.
Original	–	–	–	9.680	61.80	4.205	47.10	1.0	1.0	83.08	83.98	79.48
SQuad 20 Blocks	✔		–	5.523	19.46	0.038	3.61	110.7	13.0	81.86	83.64	74.74
		✔	–	5.509	17.86	0.025	2.88	168.2	16.4	82.31	82.98	79.61
	✔	✔	L→G	5.548	19.55	0.063	4.27	66.7	11.0	82.90	83.44	80.73
	✔	✔	G→L	5.548	19.93	0.063	4.25	66.7	11.1	82.73	83.46	79.76
	✔	✔	alternate	–	–	–	–	–	–	82.86	83.39	80.73

G Additional details for Hardware Compilation and Performance

Hardware and software environment. All latencies are measured on a single **NVIDIA H100 80 GB HBM3** (compute capability 9.0, 132 streaming multiprocessors, 79.2 GiB of usable HBM3), with one GPU visible to the process so that no measurement is perturbed by co-tenancy. The software stack is PyTorch 2.9.1 (CUDA build 12.8, cuDNN 9.10.2) with `diffusers` 0.36.0, and the caching allocator is configured with `expandable_segments:True` to avoid fragmentation stalls across the 30-prompt sweep. Weights and all attention computation are in `bfloat16`. The compiled mode wraps the transformer with `torch.compile(dynamic=False, fullgraph=True, mode="max-autotune")`; static shapes are appropriate here because the latent grid is fixed by the generation resolution. Measurements are taken at our headline generation setting of $81 \times 704 \times 1280$ (i.e. $n = 18480$ latent tokens) with NFE $= 6$, and peak memory is recorded alongside each run (≈ 60 GB in both modes for SQuad).

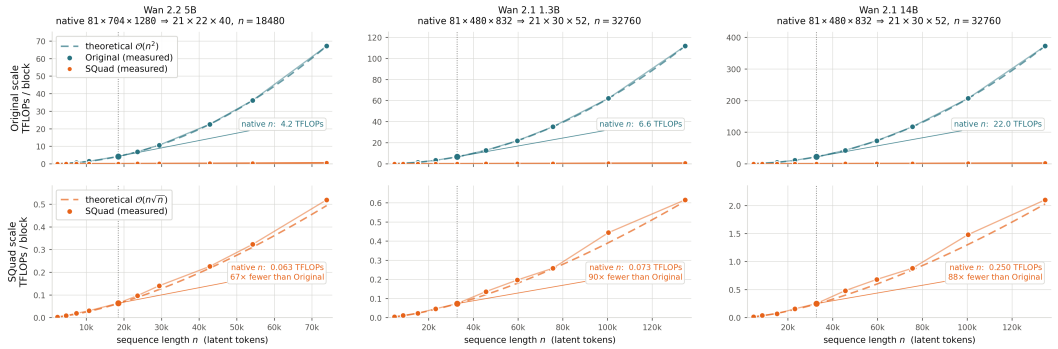
Analyzing the results. Table 5 reports the mean single-forward latency for every method in both modes, and the two columns tell complementary stories. SQuad is the fastest method in *both*: at 520 ms eager and 314 ms compiled it is $1.67\times$ and $2.12\times$ faster than the full-attention Original, and it still leads the closest efficient baseline (Jenga, at 680 and 427 ms) by $1.31\times$ and $1.36\times$ respectively. That the advantage holds in both columns is the point: a ranking established in one execution mode need not survive the other, since compilation fuses pointwise work and removes Python dispatch overhead *unevenly* across methods.

The eager column is where this unevenness becomes diagnostic. Two of the baselines are actually *slower* than the 870 ms full-attention Original that they set out to accelerate, despite reporting lower FLOP counts in table 4: Attention Surgery costs 1006 ms and ReHyAt 1757 ms. Their speedups materialize only after compilation, at 537 and 544 ms respectively, which is to say that their theoretical savings are not realizable by the stock PyTorch executor. They depend instead on specialized, hand-tuned GPU kernels, such as custom sparse-attention or chunked-recurrence implementations, to convert a FLOP reduction into a wall-clock reduction. This is a real deployment cost. Such kernels must be written, maintained, and re-tuned for each new accelerator generation, and until they exist for a given target the advertised efficiency simply does not appear.

SQuad requires none of this. Its two passes are ordinary softmax attentions separated by pure re-indexing (eq. (7)), so the modified transformer is handed *unchanged* to the most vanilla compilation setting PyTorch offers, a single `torch.compile` call, with no custom CUDA, no bespoke attention kernel, and no operator registration. The $1.66\times$ that compilation adds on top of the architectural gain therefore comes for free, from a compiler that knows nothing about SQuad. We take this as the practical counterpart to the method's conceptual simplicity: an efficiency gain that is available to anyone who can call `torch.compile` is far more likely to be adopted than one gated behind a kernel-engineering effort, and it transfers to new hardware the moment the compiler does.

H FLOPs Scaling Analysis

The complexity result of section E is asymptotic, and the latency study of section 4.3 is a single operating point. This section sits between the two: it measures the actual attention FLOPs of a single block, as a function of sequence length n, across the full range of resolutions each backbone could



(a) Wan 2.2 5B (b) Wan 2.1 1.3B (c) Wan 2.1 14B

Figure A1: Attention FLOPs per block vs. sequence length n, across three backbones. Top: full-softmax Original against its $\mathcal{O}(n^2)$ reference, with SQuad overlaid. Bottom: SQuad on its own scale against the $\mathcal{O}(n\sqrt{n})$ reference. Markers are measured; dashed lines are the theoretical curves.

plausibly generate. The question it answers is whether the $\mathcal{O}(n\sqrt{n})$ promise is realized in practice, or whether constants erode it.

What is measured. For each backbone we build one real transformer block in each of its two forms, Original and SQuad, and profile the Self-Attention cost with DeepSpeed's FLOPs profiler. We sweep ten real video shapes per model at the native aspect ratio, spanning roughly a $30\times$ range in n, and repeat the sweep for all three backbones. Crucially, the window is re-solved at every resolution rather than held fixed, since SQuad is $\mathcal{O}(n\sqrt{n})$ only when the window volume m tracks $\sqrt{n}$. We keep the two constraints the method itself imposes: every window spans the full temporal extent ($w_t = T$), so temporal mixing stays exact and only the spatial extents grow with n; and $w_h : w_w$ follows the latent grid's aspect ratio, so windows remain geometrically similar to the frame instead of degenerating into strips. Reassuringly, this rule independently recovers the trained $21 \times 2 \times 4$ window at both backbones' native resolutions, which confirms that the sweep and the trained models describe one method. Figure A1 plots the result; every marker is a genuine measurement, and the dashed line through each provides the theoretical function curve for reference.

The measured curves match the theory. The Original attention overlays its $\mathcal{O}(n^2)$ reference almost exactly across the entire range, on all three backbones. This is a useful sanity check on the measurement itself: the profiled quadratic cost is the quadratic cost the analysis predicts, with no hidden constant drift. The SQuad curve, plotted on its own scale in the lower panel of each figure, follows the $\mathcal{O}(n\sqrt{n})$ reference closely across the whole sweep and grows far more slowly than the quadratic curve everywhere. Read on the shared Original scale (upper panels), SQuad is a nearly flat line pinned to zero, which is the visual form of the reduction: at the native operating point it costs 0.063, 0.073, and 0.250 TFLOPs for the 5B, 1.3B, and 14B backbones, against 4.2, 6.6, and 22.0 TFLOPs for full attention, i.e. $67\times$, $90\times$, and $88\times$ fewer.

Conformance to the theoretical curves is quantitative. Because both references are absolute rather than fitted, the agreement can be stated as a ratio instead of read off a plot. Across all thirty measured points the Original cost lands within 0.4% of $4hdn^2$, so the quadratic baseline is matched essentially exactly. The SQuad measurements sit within 21% of the idealized $\mathcal{O}(n\sqrt{n})$ optimum and 8% above it on average, with no systematic drift as n grows: the largest and smallest sequence lengths conform equally well, which is the signature of a genuinely sub-quadratic operator rather than a quadratic one with a small constant. The residual gap is fully explained by integrality, and not by any missing term in the analysis. The optimum $m^\star = \sqrt{n}$ is generally irrational, whereas a realizable window must satisfy $m = T \cdot w_h \cdot w_w$ for integer extents, and a window that does not divide the latent grid forces a little reflect padding. Scoring the measurements against the closed-form cost $4hd(nm + n^2/m)$ of the window actually used, rather than against the unreachable optimum, removes this discrepancy: the measured values then agree to within 0.7% on average, with 26 of 30 points inside 3%. The theory therefore predicts the measured FLOPs of every configuration, and the only price paid in practice is that of rounding a window to whole tokens.

Table A2: Ablation of SFT and DMD for SQuad 20 Blocks. Best results are in **bold**.

Method	SFT	DMD	NFE	VBench		
				Tot.	Qual.	Sem.
SQuad	✔		100	82.58	83.70	78.12
20 Blocks		✔	6	82.73	83.50	79.65
(Ours)	✔	✔	**6**	**82.73**	**83.46**	**79.76**

The gain grows with both sequence length and model scale. Because the two curves have different asymptotic order, their ratio is not a constant: the reduction widens as the grid grows. On the 5B backbone the attention-FLOPs reduction rises from $67\times$ at the native $n = 18480$ to $129\times$ at $n = 73920$; on Wan 2.1 1.3B it rises from $90\times$ to $181\times$, and on 14B from $88\times$ to $177\times$ at $n = 134820$. The effect also strengthens with model width, since wider models spend a larger share of a block inside attention. This is the regime video generation is moving toward, higher resolutions and longer clips, and it is exactly where a sub-quadratic operator compounds: the larger and longer the generation, the more of the quadratic term SQuad removes.

I Additional Ablation Experiments

The ablations in the main paper are run at the SQuad 30-block configuration, the setting we ultimately adopt. Here we repeat both of them at 20 blocks, where only the middle two-thirds of the network is converted and ten blocks retain full softmax attention. The 20-block results are interesting less for their absolute numbers than for what the *contrast* between the two block counts reveals: the design choices that look optional under a shallow replacement become decisive under a deep one.

Local and global passes at 20 blocks. Table A1 repeats the local/global ablation of table 7 at 20 blocks. The striking observation is how little separates the configurations. Every variant, including the single-pass ones, lands within 1.22 points of the full-attention Original on VBench Total, and local-only attention alone reaches 81.86. Compare this with the 30-block case, where dropping either pass is catastrophic: local-only falls to 62.62 Total and 18.50 Semantic, and global-only to 62.63 and 27.24. Ten unmodified full-attention blocks are evidently enough to paper over an impoverished communication pattern in the other twenty. They can supply the global mixing that a local-only replacement removes, so the network never has to rely on the reduced pattern for long-range interaction.

This is precisely why the 20-block setting is the weaker place to study the method. It understates the importance of composing the two passes, and it makes the ordering look like a matter of taste: L→G leads G→L by 0.17 Total here, against 0.21 at 30 blocks, and the alternating variant is within noise of both. The ordering preference we report in the main paper is therefore not an artifact of the 20-block regime; it survives, and slightly widens, when the replacement is deep enough that the reduced pattern must carry the model on its own.

SFT and DMD at 20 blocks. Table A2 repeats the two-stage training ablation. The same pattern appears, even more sharply. At 20 blocks the three rows are essentially indistinguishable, spanning 0.15 points of Total (82.58 to 82.73): SFT alone, DMD2 alone, and the two together all land in the same place, and one could reasonably conclude that either stage suffices. At 30 blocks that conclusion would be badly wrong. The same three rows span 9.96 points, SFT alone collapses to 73.03, DMD2 alone reaches only 80.91, and just their combination recovers the full 82.99.

The reading we take from this is that the two stages are not redundant but load-bearing, and that their necessity scales with how much of the network the SQuad replacement disturbs. A shallow replacement leaves enough of the pretrained function intact that either stage can re-seat it; a deep one perturbs the network far enough that step distillation and flow-matching supervised fine-tuning each recover something the other cannot. Reporting only the 20-block ablation would have hidden this entirely, which is why the deeper configuration is the one we ablate in the main paper.