EFlow: Fast Few-Step Video Generator Training from Scratch via Efficient Solution Flow

Dogyun Park[1,2*], Yanyu Li[1], Sergey Tulyakov[1], and Anil Kag[1]

[1] Snap Inc.
[2] Korea University, Seoul, Republic of Korea

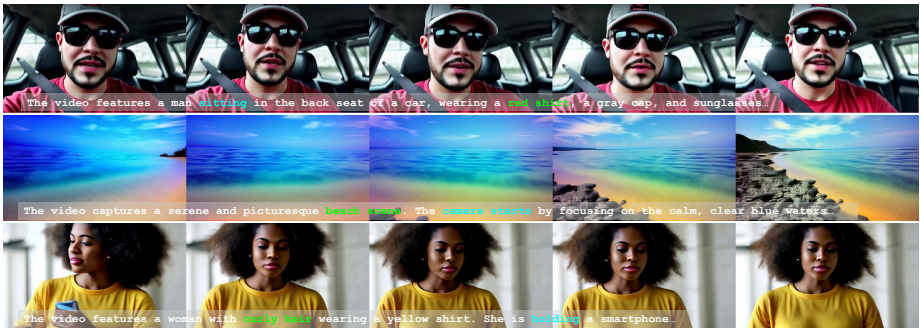
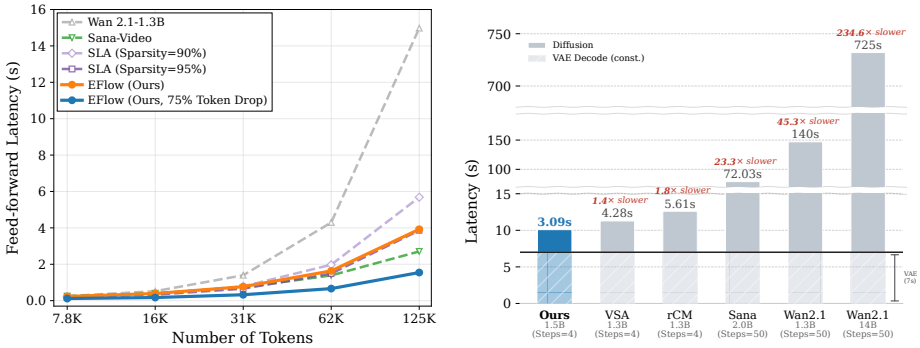


Fig. 1: Text-to-video results generated by our EFlow with 4 inference steps.

Abstract. Scaling video diffusion transformers is fundamentally bottlenecked by two compounding costs: the expensive quadratic complexity of attention per step, and the iterative sampling steps. In this work, we propose EFlow, a efficient few-step training framework, that tackles these bottlenecks simultaneously. To reduce sampling steps, we build on a solution-flow objective that learns a function mapping a noised state at time t to time s. Making this formulation computationally feasible and high-quality at video scale, however, demands two complementary innovations. First, we propose *Gated Local–Global Attention*, a token-droppable hybrid block which is efficient, expressive, and remains highly stable under aggressive random token-dropping, substantially reducing per-step compute. Second, we develop an efficient few-step training recipe. We propose *Path-Drop Guided* training to replace the expensive guidance target with a computationally cheap, weak path. Furthermore, we augment this with a *Mean-Velocity Additivity* regularizer to ensure high fidelity at extremely low step counts. Together, our EFlow enables a practical from-scratch training pipeline, achieving up to 2.5× higher training throughput over standard solution-flow, and 45.3× lower inference latency than standard iterative models with competitive performance on Kinetics and large-scale text-to-video datasets.

Keywords: Video diffusion · Efficient architecture · Few-step generation

*Work done during internship at Snap Inc.



(a) Latency vs. tokens (b) 480p inference latency

Fig. 2: (a) Feed-forward latency vs. token number for different attention designs. Softmax attention (Wan 2.1) scales poorly as the sequence length grows while our GLGA (EFlow) is significantly faster than softmax and remains competitive with sparse/linear baselines. Applying 75% token dropping on GLGA yields the lowest latency and the most favorable scaling. **(b) 480p inference latency** for our model and recent video generators. Our model achieves 1.4× and 45.3× faster diffusion inference over baselines by combining an efficient backbone with few-step sampling.

1 Introduction

Diffusion and flow-based models [1–5] enable high-fidelity video generation, yet scaling them to long, high-resolution clips remains computationally daunting. Video diffusion transformers (VideoDiTs) suffers from a compounding double bottleneck: (i) heavy per-step compute, as spatiotemporal self-attention scales quadratically with token count, and (ii) iterative sampling, requiring tens of denoising steps that multiply this per-step cost. Together, these constraints severely limit the practicality of training and deploying video models.

Most existing research attacks these challenges in isolation. Architectural efforts reduce per-step complexity via efficient linear attention [6–9], sparsity [10–13], or token pruning [14–16]. However, aggressive token removal often harms stability or expressivity—especially in the presence of locality mechanisms (like convolutions) that implicitly assume dense, grid-like spatial neighborhoods. Conversely, sampling-focused approaches reduce the number of steps through distillation [17–20] or consistency-style objectives [21–26], but frequently introduce substantial engineering complexity (*e.g.*, pre-trained teachers, multi-stage pipelines, or adversarial components). Furthermore, Classifier-Free Guidance (CFG)—a key ingredient for controllable generation—can effectively double training compute by requiring both conditional and unconditional network evaluations per step.

In this work, we propose EFlow, **fast few-step VideoDiT training** (see Tab. 1), by combining an efficient, token-droppable attention backbone with a scalable training recipe for solution-flow (SoFlow) [25]. Concretely, we build on the recent SoFlow [25] formulation, which learns a bi-time function mapping a noised state at time t to state at any time s. We select this framework for its unique advantages in few-step generation: it enables single-stage training from

Table 1: Comparison of our method with VideoDiT families. We summarize attention complexity, training pipeline requirements (reliance on teacher or auxiliary models like discriminator), and inference cost via number of function evaluations (NFEs). N, N' D, and W denote token count, sparse token number, dimension, and local window size, respectively. The "×2" indicates model evaluations for classifier-free guidance.

	Standard (Wan 2.1, Wan2.2)	**Efficient** (Sana, S^2DiT)	**Efficient** (VSA, SLA)	**Few-step** (DMD, rCM)	**Ours** (EFLOW)
Attention architecture					
Type	Softmax	Linear+Conv	Sparse softmax	Softmax	GLGA
Complexity	$\mathcal{O}(N^2)$	$\mathcal{O}(ND)$	$\mathcal{O}(N'^2)$	$\mathcal{O}(N^2)$	$\mathcal{O}(N(D+W))$
Token-Droppable	✓	✗	✗	✓	✓
Training pipeline					
No Teacher Model	✓	✓	✗	✗	✓
No Auxiliary Model	✓	✓	✓	✗	✓
Inference					
Inference NFEs	50×2	50×2	50×2	4	4

scratch, completely eliminating the need for pre-trained teachers [23] or auxiliary networks [27]. Furthermore, its hybrid flow-matching and solution-consistency objective avoids the memory-intensive Jacobian-Vector Products (JVPs) required by consistency-style models [24,28], permitting the use of highly optimized kernels like FlashAttention [29, 30]. However, scaling this formulation to video poses a severe computational challenge. Enforcing solution consistency across time pairs and applying CFG requires multiple full-network evaluations per training step. When compounded by the attention quadratic complexity, this multi-pass requirement makes video-scale training prohibitively expensive.

We introduce two synergistic innovations for scalable few-step video training: *(i) Efficient, Token-Droppable Hybrid Attention for VideoDiT.* Prior linear-attention DiTs [7, 9] use convolutional mixing for locality, which breaks under aggressive token dropping due to its reliance on a dense spatial grid. We instead propose *Gated Local–Global Attention* (GLGA), a hybrid block fusing global linear attention with local sliding-window attention. Because window attention operates over available tokens rather than fixed grids, it is inherently robust to severe token removal. This adaptive routing yields an expressive and highly stable VideoDiT even when dropping the vast majority of tokens. Empirically, on the Kinetics dataset, this hybrid design achieves a **13% improvement** in Fréchet Video Distance (FVD) compared to linear-attention [7] (Tab. 2).
(ii) Efficient Few-Step Training Recipe. Integrating CFG into solution-flow requires two network evaluations per sample—a steep computational cost for video. We propose *Path-Drop Guided* (PDG) training, which replaces the expensive unconditional branch with a computationally cheap "weak-path" forward pass obtained by skipping middle transformer layers. Furthermore, to preserve quality at few-step sampling, we introduce a global *Mean-Velocity Additivity* (MVA) regularizer that mitigates the integration errors in local consistency matching.

Integrating the *GLGA VideoDiT* with *Scalable Few-Step Recipe*, EFLOW compounds compute savings to make from-scratch, few-step video training practical. Our framework achieves **1.76× and 2.5× higher training throughput** on

480p video compared to flow matching and solution-flow matching, respectively (Fig. 4(a)). These gains extend to deployment, where EFlow achieves **1.4× to 45.3× lower inference latency** (Fig. 2) than recent few-step methods (VSA, rCM), efficient architectures (Sana), and standard models (Wan2.1 1.3B), while maintaining competitive VBench scores on 480p text-to-video generation.

In summary, our key contributions are:

- **Efficient, Droppable VideoDiT Architecture:** We propose Gated Local–Global Attention (GLGA), which maintains high fidelity under aggressive token dropping, resulting in substantial training and inference speedups.
- **Path-Drop Guided (PDG) Training:** We introduce a novel guidance mechanism that constructs guidance targets using a lightweight, block-skipped forward pass, drastically reducing the CFG compute overhead.
- **Large-Scale Few-Step Video Training:** We present a simple, teacher-free pipeline that incorporates a novel global long-jump consistency objective (MVA) to enable high-quality generation at extremely low step counts, achieving performance competitive with recent state-of-the-art open-source models.

2 Related Work

Efficient architectures and few-step generation. Diffusion Transformers (DiTs) [31] have demonstrated that transformer backbones are highly competitive, yet their computational cost scales quadratically with token count. To mitigate this per-step overhead, prior work either accelerates exact attention (*e.g.*, FlashAttention [29,30]) or replaces softmax attention with sub-quadratic variants like linear or gated-linear attention (*e.g.*, Sana, DiG, Sana-Video, and S^2DiT) [6–9]. While sparse attention methods [10,12,13,32] further reduce complexity, they often follow a "dense-to-sparse" paradigm, requiring a pre-trained dense model as a high-fidelity starting point. This reliance on pre-existing weights poses a significant barrier to truly efficient, from-scratch video training.

Orthogonally, few-step generation methods address the iterative sampling bottleneck by drastically reducing the number of function evaluations (NFEs). A prominent line of work achieves this by distilling pre-trained teachers via Consistency Models [21,23,28], Distribution Matching Distillation (DMD) [27], or Score Identity Distillation (SiD) [33]. While these methods enable one- or few-step generation, they introduce substantial engineering complexity—via multi-stage pipelines and a strict reliance on expensive pre-trained teachers. To overcome these constraints, recent advancements such as MeanFlow [24] and SoFlow [25] learn few-step generative dynamics directly from scratch. SoFlow, in particular, parameterizes a bi-time solution function trained through a combination of flow-matching and solution-consistency objectives. Our work bridges these efficient architecture and teacher-free few-step generation tracks: we adopt SoFlow to eliminate distillation overhead and introduce a novel token-droppable VideoDiT backbone alongside path-drop guided (PDG) training to make this paradigm highly scalable at video resolutions.

Token dropping for diffusion transformers. Reducing sequence length is the most direct path to lowering attention costs. Token merging (*e.g.*, ToMe [34]) has been widely adapted for diffusion inference to preserve visual quality while reducing FLOPs. Recent extensions to video (*e.g.*, VidToMe [14] and vid-TLDR [15]) merge temporally or spatially redundant tokens to ensure multi-frame consistency without heavy compute. Alternatively, importance-aware pruning (*e.g.*, ToME-SD [35]) selectively discards less influential tokens based on attention scores or classifier-free guidance magnitudes. While inference-time reduction is valuable, random token dropping during training is essential to lower end-to-end training costs. However, naive dropping destroys grid locality and induces a severe train–inference mismatch. Existing attempts to mitigate this mismatch rely on masked autoencoding auxiliary tasks (*e.g.*, MaskDiT [36], MDTv2 [37]) or sparse–dense residual fusion (*e.g.*, SPRINT [16]).

Crucially, while standard DiTs can tolerate token dropping, their quadratic complexity remains a concern. Conversely, recent efficient architectures replace softmax with linear attention but often rely on convolutions to recover local expressivity [6–8]. Because convolutions assume a dense, regular spatial grid, these models fundamentally degrade under aggressive random token removal. To resolve this conflict, our Gated Local–Global Attention (GLGA) replaces rigid convolutions with local sliding-window attention. Since window attention operates over available tokens rather than fixed grids, it preserves linear efficiency and local expressivity while remaining inherently robust to token dropping.

3 Preliminary

3.1 Solution Flow Matching

Flow models [38–42] specify a transport between π_0 and a reference π_1 using a velocity ODE, $\frac{\mathrm{d}x_t}{\mathrm{d}t} = v(x_t, t, c)$, where $t \in [0, 1]$ and c is condition such as text. Following [43], we use the linear Gaussian coupling $x_t \sim \mathcal{N}(\alpha_t x_0, \sigma_t^2 I)$ with $\alpha_t = 1 - t$ and $\sigma_t = t$. A network v_θ is trained by flow matching to regress the induced ground-truth conditional velocity, *i.e.*, $v(x_t, t, c|x_0) = x_1 - x_0$:

$$\min_\theta \ \mathbb{E}_{x_0, x_1, t}\Big[\|v_\theta(x_t, t, c) - v(x_t, t, c \,|\, x_0)\|_2^2\Big]. \tag{1}$$

SoFlow [25] generalizes this by directly modeling the *solution map* $f(\mathbf{x}_t, t, s, c)$ for any $0 \le s \le t \le 1$, which returns the ODE solution at time s given the state $\mathbf{x}_t$ at time t under conditioning c. Under regularity assumptions, the ground-truth solution satisfies the boundary condition $f(\mathbf{x}_t, t, t, c) = \mathbf{x}_t$ and the consistency relation $\partial_s f(\mathbf{x}_t, t, s, c) = v(f(\mathbf{x}_t, t, s, c), s, c)$. SoFlow shows it suffices to learn a parameterized f_θ that satisfies the boundary condition and an associated PDE. To enforce the boundary condition, we use an Euler-style parameterization

$$f_\theta(\mathbf{x}_t, t, s, c) = \mathbf{x}_t + (s - t)\, F_\theta(\mathbf{x}_t, t, s, c), \tag{2}$$

where F_θ is implemented by a DiT backbone.

Flow-matching objective. At boundary $s = t$, we obtain velocity $v(\mathbf{x}_t, t, c) = \partial_s f_\theta(\mathbf{x}_t, t, t, c)$. From Eq. 2, $\partial_s f_\theta(\mathbf{x}_t, t, t, c) = F_\theta(\mathbf{x}_t, t, t, c)$, giving the FM loss

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_1, t}\left[w_{\text{FM}}(t)\ \|F_\theta(\mathbf{x}_t, t, t, c) - v(\mathbf{x}_t, t, c \,|\, x_0)\|_2^2\right], \tag{3}$$

where $w_{\text{FM}}(t) = 1/(\text{MSE} + \epsilon)^p$; MSE indicates original mean squared error, ϵ is a factor for numerical stability, and p is a factor that determines the robustness.
Solution consistency matching. For $s < t$, SoFlow enforces a finite solution-consistency constraint over an intermediate time l with $s \le l \le t$. Using a first-order Taylor approximation to the intermediate state, $\mathbf{x}_l \approx \mathbf{x}_t + v(\mathbf{x}_t, t, c \,|\, x_0)(l-t)$, and a stop-gradient target f_{θ^-}, the solution-consistency matching (SCM) loss is

$$\mathcal{L}_{\text{SCM}} = \mathbb{E}\left[w_{\text{SCM}}\left\|f_\theta(\mathbf{x}_t, t, s, c) - f_{\theta^-}\big(\mathbf{x}_t + v(\mathbf{x}_t, t, c \,|\, x_0)(l-t),\, l,\, s\big)\right\|_2^2\right], \tag{4}$$

where $w_{\text{SCM}}(t) = \frac{1}{(t-l)(t-s)} \times \frac{1}{(\frac{\text{MSE}}{(t-l)^2} + \epsilon)^p}$ following [25]. The training loss is $\mathcal{L} = \lambda\,\mathcal{L}_{\text{FM}} + (1-\lambda)\,\mathcal{L}_{\text{SCM}}$, where λ controls the fraction of a data batch dedicated to $\mathcal{L}_{\text{FM}}$. In the following sections, we introduce architectural and training innovations to make this solution-flow paradigm scalable and robust for high-resolution video generation.

3.2 Training with Token-Dropping.

Token-dropping reduces the quadratic attention cost in DiTs by shortening the token sequence. For a drop ratio r, we remove $\lfloor rN \rfloor$ tokens and process only the remaining subset, yielding substantial compute savings. Recently, SPRINT [16] stabilized training with high-ratio dropping (*e.g.*, 75%) by restructuring the transformer into three stages—an encoder, a deep middle stack, and a decoder—and connecting encoder and decoder with a long dense residual path.

Concretely, SPRINT computes dense shallow features, applies token dropping before the middle blocks, and runs middle stack only on the surviving sparse tokens. To fuse sparse and dense features, it lifts the sparse output back to N by padding dropped positions with a fixed `[MASK]` token, and fuses it with the dense shallow features via long residual connection. Training follows a two-stage schedule: long high-drop pre-training, followed by short full-token fine-tuning where the middle blocks switch back to dense inputs to close the train–inference gap. In our method, we adopt this SPRINT's design, training schedule, and *apply token-dropping only to the middle blocks.*

4 EFlow: Training Few-Step Video Generator from Scratch

We propose a novel framework that addresses the two computational bottlenecks of video diffusion training: (i) the per-step spatiotemporal attention computation, and (ii) the number of denoising steps in iterative sampling. To overcome these,

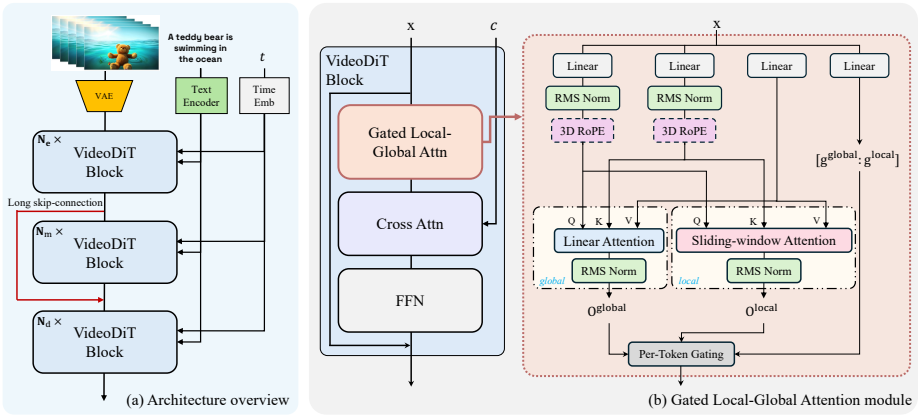


Fig. 3: Overview of our VideoDiT architecture. (a) The network processes video tokens through stacked DiT blocks, utilizing long skip connections to stabilize information flow when tokens are aggressively dropped. (b) Given a DiT block with Gated Local–Global Attention (GLGA) module, input tokens are projected into shared queries, keys, and values. These features are processed by two parallel mechanisms: a linear attention branch for efficient global context and a sliding-window attention branch for expressive local detail. Finally, an input-aware gating (g) adaptively fuses the global (O^{global}) and local (O^{local}) outputs on per-token basis.

our framework introduces: 1) An **efficient VideoDiT backbone** featuring *Gated Local–Global Attention (GLGA)*, which substantially reduces per-step compute while remaining robust to aggressive token dropping. 2) An **efficient few-step VideoDiT training recipe** that reduces the training overhead of solution-flow matching [25] via path-drop guided training, while incorporating a novel global long-jump consistency objective to minimize integration errors and ensure high-fidelity generation at extremely low step counts.

4.1 Efficient & Token Droppable VideoDiT Architecture

We address the primary bottleneck in VideoDiTs: the quadratic complexity of spatiotemporal attention. While replacing softmax with linear variants reduces this overhead, existing designs typically attempt to recover lost locality and expressivity by incorporating convolutional mixing (*e.g.*, the Mix-FFN in Sana-Video [7]). However, convolutions rely on a consistent spatial neighborhood layout. It makes them incompatible with aggressive random token dropping, as removing tokens destroys the structured grid locality upon which convolutions depend.

In contrast, we propose a **token-droppable hybrid attention** block that preserves linear efficiency and recovers local expressivity through a mechanism that remains well-defined even when tokens are dropped. Our **Gated Local–Global Attention (GLGA)** block combines: **(i)** *Global Linear Attention* for long-range spatiotemporal context, and **(ii)** *Local Sliding-Window Attention* for high-frequency details, and **(iii)** *Input-Aware Gating* to adaptively fuse the two

branches at a per-token level. Since sliding-window attention operates over the set of available tokens within a window rather than a dense grid, it is naturally compatible with token dropping. Overall architecture is illustrated in Fig. 3.

Global Linear Attention. Given an input sequence $X \in \mathbb{R}^{N\times d}$, we compute queries, keys, and values $Q = XW_Q$, $K = XW_K$, $V = XW_V$, apply Rotary Positional Embeddings (RoPE) [44] to Q and K. For clarity, we describe a single-head formulation; multi-head attention follows standard practice. Following [45, 46], we use a denominator-free linear attention variant:

$$O_i^{\text{global}} = Q_i\, S, \qquad S = \sum_{j=1}^{N} K_j^\top V_j \in \mathbb{R}^{d\times d}, \tag{5}$$

where S is a global key–value summary shared across all queries. This reduces the quadratic $QK^\top$ interaction to a single global aggregation plus per-token projection, yielding linear scaling in N. We omit the normalization denominator typical of kernelized linear attention, as it can induce numerical instability when combined with RoPE. Instead, we apply RMS normalization to each branch output, improving training stability and ensuring well-behaved fusion with the local attention branch.

Local Window Attention. Since linear attention can underfit local structures [7, 46, 47], we compute softmax attention over a sliding window based on rasterized token order. Let $\mathcal{W}(i)$ denote the index set of keys within the window of radius $|\mathcal{W}|$ centered at token i. We compute

$$O_i^{\text{local}} = \sum_{j\in\mathcal{W}(i)} \text{softmax}\left(\frac{Q_i K_j^\top}{\sqrt{d}}\right) V_j. \tag{6}$$

Note that removing tokens simply reduces the size of $\mathcal{W}(i)$ without altering the operation itself. This requires $\mathcal{O}(N|\mathcal{W}|)$ compute, where $|\mathcal{W}|$ is the maximum tokens per window. To minimize parameter and compute overhead, we reuse the same Q, K, V projections as the linear-attention branch.

Input-Aware Gating and Fusion. To dynamically balance global and local contexts, we predict per-token scalar gates $g_i^{\text{global}}, g_i^{\text{local}} \in [0, 1]$:

$$g = [g^{\text{global}}, g^{\text{local}}] = \sigma(XW_g), \tag{7}$$

where $W_g \in \mathbb{R}^{d\times 2}$ is a linear projection and σ is the sigmoid function. The final output is the gated sum:

$$O_i = g_i^{\text{global}} \odot O_i^{\text{global}} + g_i^{\text{local}} \odot O_i^{\text{local}}. \tag{8}$$

Sparse softmax interleaving & Token-dropping compatibility. To further boost expressivity, we sparsely interleave full softmax attention blocks (*e.g.*, every K blocks) following recent hybrid architectures [48, 49]. Because our GLGA backbone relies solely on attention and pointwise layers—avoiding spatial convolutions—it is inherently compatible with random token dropping.

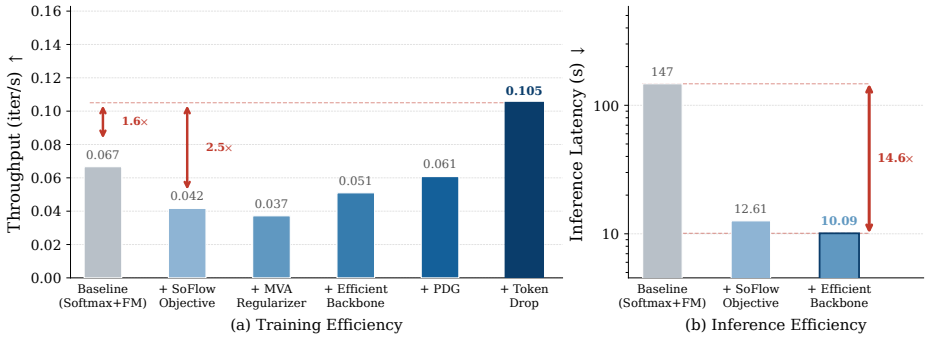


Fig. 4: End-to-end efficiency gains from our framework. (a) **Training efficiency** measured as throughput (iterations/sec). Relative to the standard flow-matching baseline, introducing the solution-flow (SoFlow) objective and the MVA regularizer increases per-iteration overhead and therefore reduces throughput. In contrast, our efficient backbone, PDG training, and token dropping provide consistent speedups, improving throughput by up to 1.6× and 2.5× over the baselines. (b) **Inference efficiency.** Our overall framework substantially reduces end-to-end inference time relative to the softmax+flow-matching baseline (147 → 10.09s).

4.2 Efficient Few-Step VideoDiT Training

We build on SoFlow [25] to train a bi-time solution function $f_\theta(x_t, t, s, c)$ for few-step generation. While this enables teacher-free few-step training from scratch, scaling guided solution-flow training to long, high-resolution videos is challenging due to its substantially higher training cost than standard flow matching. Pseudo-codes for training and inference are provided in the supplementary materials.

Computational overheads in video solution-flow training. Compared to standard flow matching, solution-flow training introduces additional forward-pass overhead from two sources. First, the *Solution-Consistency (SCM)* objective evaluates the model at multiple time pairs, requiring extra model evaluations beyond the single forward used in flow matching. Second, *Classifier-Free Guidance (CFG)* distillation necessitates constructing a guided velocity target during training:

$$v_g(x_t, t, c \,|\, x_0) \;=\; w\, v(x_t, t, c \,|\, x_0) + (1 - w)\, v(x_t, t, \emptyset \,|\, x_0), \tag{9}$$

In SoFlow, the conditional term is approximated by the ground-truth velocity $v(x_t, t, c|x_0)$, but the unconditional term requires an additional network evaluation $v_\theta(x_t, t, \emptyset)$. Combined with the heavy quadratic attention of video DiTs, these extra passes make naive solution-flow training prohibitively expensive to scale.

Scaling Solution-Flow via Multi-Level Efficiency. We propose complementary mechanisms to reduce the cost of each forward-pass and reduce guidance overhead. First, we use our *efficient GLGA backbone* (Sec. 4.1) to lower per-forward attention cost. Next, we apply *token dropping* during most of training to shorten the sequence length, reducing compute and memory for every forward

pass (including SCM). Finally, to slash guidance overhead, we propose *Path-Drop Guided (PDG) training*, which replaces the unconditional forward with a cheap *weak-path* forward that skips a large fraction of middle transformer blocks. Additionally, we introduce *Mean-Velocity Additivity (MVA)* to reduce error accumulation in solution-flow over large-jumps in few-step sampling. Together, these changes make guided solution-flow training practical at large-scale video.

(a) Path-Drop Guided (PDG) training. Given a model with L blocks $\{B_1, \ldots, B_L\}$, we define a weak path that skips intermediate blocks B_{i+1} through B_j as:

$$\mathrm{Weak}(X) = (B_L \circ \cdots \circ B_{j+1}) \circ \mathrm{Id} \circ (B_i \circ \cdots \circ B_1)(X), \qquad 1 \le i < j \le L, \tag{10}$$

where Id denotes the identity mapping. Conveniently, we align this skipped region (i, j) with the sparse middle stack of our architecture. We replace the expensive unconditional baseline in CFG with this weak-path prediction:

$$v_{\mathrm{PDG}}(x_t, t, c \,|\, x_0) = w\, v(x_t, t, c \,|\, x_0) + (1 - w)\, v_\theta^{\mathrm{weak}}(x_t, t, \emptyset). \tag{11}$$

We then plug v_{PDG} into SoFlow's FM and SCM objectives to train the full-path model. Inspired by early works [16, 50, 51], interpolating with a computationally "free" weak model provides a sufficiently aligned baseline to stabilize guided targets while eliminating the compute overhead of a full unconditional pass.

(b) Global long-jump consistency via Mean-Velocity Additivity. Local consistency objectives in Eq. 4 accumulate integration errors over the large jumps required for few-step sampling. To reduce this, we introduce a global regularizer derived from the exact additivity of the mean velocity. Let $\bar{v}_\theta(x_t, t, s, c) \triangleq \frac{x_t - f_\theta(x_t, t, s, c)}{t - s}$ denote the predicted mean velocity over $[s, t]$. For the ground-truth flow, displacements are additive across any intermediate time $s < \ell < t$, yielding the exact constraint:

$$(t - s)\, \bar{v}(x_t, t, s, c) = (t - \ell)\, \bar{v}(x_t, t, \ell, c) + (\ell - s)\, \bar{v}(x_\ell, \ell, s, c).$$

We enforce this constraint using a stop-gradient network, θ^-. By substituting the mean velocity definition into the L_2 residual of this constraint, the x_t terms perfectly cancel. The mean-velocity additivity (MVA) objective simplifies to a direct composition error, enforcing the semigroup property of the flow:

$$\mathcal{L}_{\mathrm{MVA}} = \mathbb{E}_{x_t, c, t, \ell, s}\left[\left\| f_\theta(x_t, t, s, c) - f_{\theta^-}(f_{\theta^-}(x_t, t, \ell, c), \ell, s, c) \right\|_2^2 \right]. \tag{12}$$

Our theoretical analysis (see Appendix) shows that minimizing MVA loss strictly controls the global integration error over large discretization steps. To directly align the training objective with the inference trajectory, we restrict t, s, and ℓ to a predefined discrete set of evaluation timesteps (*e.g.*, an 64-step schedule), and randomly sample long-jump intervals (t, s, l) from this set such that $s < \ell < t$. We optimize the total loss via a compute-efficient batch-splitting strategy: $\mathcal{L} = a\mathcal{L}_{\mathrm{FM}} + b\mathcal{L}_{\mathrm{SCM}} + (1 - a - b)\mathcal{L}_{\mathrm{MVA}}$. Specifically, we partition each batch into three disjoint groups with ratios a, b, and $(1 - a - b)$. Thus, the two additional EMA forward passes required for $\mathcal{L}_{\mathrm{MVA}}$ are only computed on a strict subset of

the batch. We further lower this cost via aggressive token dropping in our GLGA backbone. To ensure stable convergence, we employ a delayed-activation schedule: the MVA ratio $(1-a-b)$ is strictly zero during early training, allowing the model to learn the base flow dynamics before the long-jump consistency constraint is introduced.

Empirical Efficiency Gains. Fig. 4 summarizes our cumulative gains on 480p video. On the training side (throughput in iters/s), adopting SoFlow incurs 40% additional overhead compared to a Flow-Matching (0.067 → 0.042 it/s) due to extra forward passes from solution consistency and guidance. Integrating our global MVA regularizer adds a 7% further cost (0.042 → 0.037 it/s). Our efficient GLGA backbone begins to recover this compute cost (0.037 → 0.051 it/s), and PDG further minimizes guidance overhead (0.051 → 0.061 it/s). Finally, enabling aggressive token dropping yields the largest gain (0.061 → 0.105 it/s), resulting in an overall **1.6× and 2.5× training speedup** over standard FM and SFM, respectively. On the inference side (latency), few-step solution-flow sampling provides a massive reduction versus standard iterative diffusion (147 → 12.61 s), and our efficient backbone further reduces the per-step cost (12.61 → 10.09 s). This culminates in a **14.6× overall inference latency reduction**.

5 Experiment

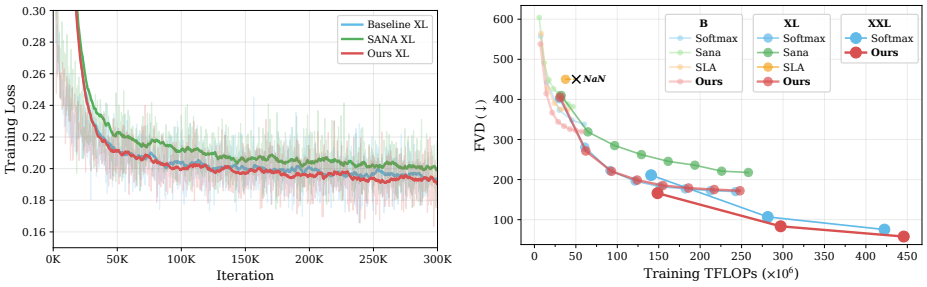
5.1 Implementation Details

Our architecture integrates the Wan [1] structure with SPRINT [16] sparse-training paradigm. Specifically, our transformer blocks follow an encoder–decoder structure connected by a long residual path (Fig. 3), with each block comprising self-attention, cross-attention, and a feed-forward network. We replace self-attention with our GLGA, sparsely interleaving full softmax attention every 8 layers. We conduct experiments on two datasets: (i) class-conditional Kinetics-700 [52], and (ii) a large-scale, proprietary text-to-video dataset. All models are trained from scratch with random initialization. We apply 75% token-dropping to the middle stack for the first 80% of training and fine-tune the remainder with full tokens. Detailed model hyperparameters and training configurations are provided in the Appendix.

5.2 Controlled Experiments on Kinetics-700

To isolate the expressiveness of our efficient self-attention block, we systematically compare GLGA against several baselines using an identical model structure (without token-dropping). These include standard softmax attention, linear attention with Mix-FFN used in Sana-Video [7], and Sparse Linear Attention (SLA) [10]. We set SLA to an 80% sparsity level (default=95%), as we empirically observed that higher ratios induced training instability.

Expressiveness. Fig. 5(a) reports the training loss over iterations. GLGA matches or outperforms the softmax baseline in early optimization and attains



(a) Training loss vs. iterations. (b) FVD vs. training FLOPs.

Fig. 5: Systemic comparison on Kinetics 700. (a) **Training loss vs. iterations** comparing the softmax-attention baseline, Sana-Video attention, and our GLGA module. (b) **FVD vs. training compute** across various model scales for softmax, Sana-Video attention, sparse-linear attention (SLA), and GLGA. GLGA consistently achieves lower loss and lower FVD at matched compute, and its advantage persists as model scale increases; SLA become unstable and diverge under the same training setup.

a consistently lower loss throughout the training. In contrast, Sana-Video's attention exhibits substantially higher loss across the full schedule, indicating slower convergence and a higher final error. This shows that GLGA's hybrid design—with its adaptive gating between global aggregation and local refinement—preserves optimization stability while retaining superior modeling capacity. Fig. 5(b) highlights scalability by plotting FVD [53] against training FLOPs across three model scales (B, XL, XXL). Across all compute budgets and parameter scales, GLGA consistently achieves a lower FVD at matched compute, effectively delivering better generation quality per FLOP. Notably, this advantage persists as we scale from B → XL → XXL, with GLGA scaling much more favorably than the sparse/linear baselines and remaining highly competitive with the heavy softmax baseline. Finally, we note that even at the lowered 80% sparsity level, SLA proved highly unstable and diverged to NaN across all scaled regimes, severely limiting training ability. In contrast, GLGA remains remarkably stable across all scales while delivering an optimal quality–efficiency trade-off.

Improvement against baselines. Tab. 2 disentangles our architectural and objective improvements against baselines. 1) **Base1→Base2**: Swapping softmax for Sana's attention block degrades FVD from 188.9 to 217.7, demonstrating limited capacity of Sana. 2) **Base1→Ours2**: GLGA backbone trained with flow-matching fully recovers the performance (189.7 FVD). 3) **Ours2→Ours4**: Upgrading to our PDG-SFM objective then yields a massive compound benefit: reducing inference cost by 10× (40 to 4 NFEs) while simultaneously improving generation quality to 146.7 FVD. Qualitative result in Fig. 6 further demonstrates this substantial improvement.

5.3 Comparison of Text-to-Video Models

We evaluate our 1.5B EFlow model on the challenging task of high-resolution (480p) text-to-video generation. To demonstrate its competitive performance and

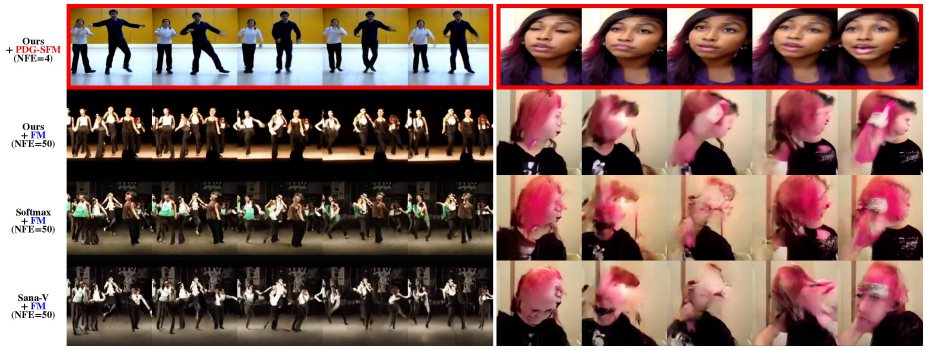


Fig. 6: Qualitative comparison on Kinetics-700 on 400K training iterations. Our complete framework (Ours+PDG-SFM) successfully generates highly detailed, temporally consistent videos in only 4 inference steps. In contrast, baselines trained with standard Flow Matching (FM)—including standard softmax, and Sana-video's attention—struggle to resolve vivid details and exhibit severe blurring or structural distortion despite using 50 steps.

efficiency, we compare against recent state-of-the-art video models, including Sana-Video [7], rCM [23], SLA [10], VSA [11], and Wan 2.1 [1].

Efficiency. Fig. 2(a) reports forward-pass latency against token count. Wan 2.1 with softmax attention scales poorly, with latency increasing sharply in the high-token regime. While SLA reduces cost relative to softmax, it remains noticeably slower as the token count grows for sparsity of 90%. Sana-Video is slightly faster than GLGA at 125K tokens, reflecting its streamlined linear-style design. However, the experiments (in Fig. 5 and Tab. 2) show that this speed advantage comes with a degradation in optimization and quality: Sana-Video attains higher training loss and worse FVD at matched training compute. In contrast, EFlow strikes a superior balance, outperforming softmax in speed while strictly preserving generation quality. Furthermore, EFlow is compatible with token dropping—applying a 75% token drop yields the absolute lowest latency and the most favorable scaling curve, effectively handling the high-resolution compute bottleneck. Beyond per-step scaling, Fig. 2(b) demonstrates the compounding benefits of our framework for end-to-end 480p video generation. By our efficient architecture with the 4-step PDG training recipe, EFlow (1.5B) achieves a remarkable 3.09s diffusion latency (excluding VAE). This is 1.4× to 1.8× faster than recent few-step distillation methods like VSA and rCM, and completely eclipses standard 50-step iterative models—running 23.3× faster than Sana (2.0B), 45.3× faster than Wan2.1-1.3B, and up to 234.6× faster than the heavy Wan2.1-14B.

Quantitative result. To provide a comprehensive assessment of generation quality across diverse semantic and temporal dimensions, detailed VBench evaluations comparing EFlow against the baselines are provided in the appendix.

Table 2: FVD comparison against baselines. All models are trained for 400K iterations.

	Attn.	Objective	MVA	Token drop	Params	NFE ↓	FVD ↓
Base1	Softmax	FM	✗	✗	457M	40	188.9
Base2	Sana	FM	✗	✗	607M	40	217.7
Base3	Sana	FM	✗	✓	607M	40	456.1
Ours1	GLGA	FM	✗	✗	529M	40	191.3
Ours2	GLGA	FM	✗	✓	529M	40	189.7
Ours3	GLGA	PDG-SFM	✗	✓	529M	**4**	185.2
Ours4	GLGA	PDG-SFM	✓	✓	529M	**4**	**146.7**

Table 3: Ablation study on our GLGA components. All trained for 300K iterations.

	Linear Attn.	Sliding Window Attn.	Input-aware Gating	FVD ↓
(a)	✓			420.8
(b)		✓		390.2
(c)	✓	✓		352.9
(d)	✓	✓	✓	**325.7**

Table 4: Ablation on training objectives. All trained for 300K iterations.

Method	NFE ↓	FVD ↓
FM	50	325.7
FM	4	513.8
SFM	4	347.1
PDG-SFM (Ours)	4	**322.2**

5.4 Analysis

Here, we present results of the analysis conducted on Kinetics-700 dataset.

Ablation on architecture. Tab. 3 isolates GLGA's internal components. Relying solely on global linear or local sliding-window attention yields poor quality (420.8 and 390.2 FVD, respectively). While element-wise summation improves FVD to 352.9, our input-aware gating achieves the best result (325.7 FVD), improving 22% from pure linear attention. This confirms that dynamically routing global context and local detail per-token is essential for high-fidelity generation.

Ablation on training objective. Tab. 4 evaluates objectives using a fixed architecture. Standard SFM reduces sampling to 4 NFEs, (347.1 FVD) but incurs massive multi-pass training overhead as described in Fig. 4(a). Our PDG-SFM completely recovers training throughput driving the 4-step FVD down to 322.2 via weak-path guidance.

Ablation on Mean-Velocity Additivity. In Tab. 2, **Ours3→Ours4** isolates the impact of MVA regularizer. While the PDG-SFM objective alone can reduce sampling to 4 NFEs (185.2 FVD), it still suffers from integration errors across large time jumps. Explicitly enforcing long-jump consistency via MVA corrects these errors, driving the FVD down to 146.7 at the exact same 4-step inference cost. This substantial 38.5-point improvement confirms that MVA is essential for preserving fidelity in few-step solution flows.

Robustness to token-dropping. Tab. 2 also validates GLGA's compatibility with token-drop training. **Base2→Base3** shows that dropping tokens in the Sana baseline causes a catastrophic collapse in FVD (456.1 from 217.7) because its convolutions assume a dense spatial grid. In contrast, **Ours1→Ours2** demonstrates

that GLGA's local branch relies on sliding-window attention—which operates naturally over unstructured sets—allowing it to survive severe dropping and unlock massive training speedups without degradation.

6 Conclusion

In this paper, we introduced EFlow, an efficient framework that tackles the dual bottlenecks of softmax-attention complexity and iterative sampling in video diffusion models. We proposed Gated Local–Global Attention, a token-droppable attention that maintains expressivity and efficiency under aggressive token dropping. We also developed Path-Drop Guided solution-flow matching, augmented by a Mean-Velocity Additivity regularizer to ensure efficient and high-fidelity generation. Ultimately, EFlow enables practical few-step video generator training from scratch, establishing a highly efficient paradigm for future video synthesis.

References

1. T. Wan, A. Wang, B. Ai, B. Wen, C. Mao, C.-W. Xie, D. Chen, F. Yu, H. Zhao, J. Yang, *et al.*, "Wan: Open and advanced large-scale video generative models," in *arXiv preprint arXiv:2503.20314*, 2025. 2, 11, 13
2. Z. Zheng, X. Peng, T. Yang, C. Shen, S. Li, H. Liu, Y. Zhou, T. Li, and Y. You, "Open-Sora: Democratizing efficient video production for all," 2024. 2
3. Z. Yang, J. Teng, W. Zheng, M. Ding, S. Huang, J. Xu, Y. Yang, W. Hong, X. Zhang, G. Feng, D. Yin, Y. Zhang, W. Wang, Y. Cheng, B. Xu, X. Gu, Y. Dong, and J. Tang, "CogVideoX: Text-to-video diffusion models with an expert transformer," in *ICLR*, 2025. 2
4. OpenAI, "Video generation models as world simulators." https://openai.com/index/video-generation-models-as-world-simulators/, 2023. 2
5. Kuaishou, "Kling." https://kling.kuaishou.com/en, 2024. 2
6. E. Xie, J. Chen, J. Chen, H. Cai, H. Tang, Y. Lin, Z. Zhang, M. Li, L. Zhu, Y. Lu, *et al.*, "Sana: Efficient high-resolution image synthesis with linear diffusion transformers," in *arXiv preprint arXiv:2410.10629*, 2024. 2, 4, 5
7. J. Chen, Y. Zhao, J. Yu, R. Chu, J. Chen, S. Yang, X. Wang, Y. Pan, D. Zhou, H. Ling, *et al.*, "Sana-video: Efficient video generation with block linear diffusion transformer," in *arXiv preprint arXiv:2509.24695*, 2025. 2, 3, 4, 5, 7, 8, 11, 13
8. L. Zhu, Z. Huang, B. Liao, J. H. Liew, H. Yan, J. Feng, and X. Wang, "Dig: Scalable and efficient diffusion models with gated linear attention," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025. 2, 4, 5
9. L. Zhao, Y. Wu, A. Lebedev, D. Lahiri, M. Dong, A. Sahni, M. Vasilkovsky, H. Chen, J. Hu, A. Siarohin, S. Tulyakov, Y. Wang, A. Kag, and Y. Li, "S2dit: Sandwich diffusion transformer for mobile streaming video generation," 2026. 2, 3, 4
10. J. Zhang, H. Wang, K. Jiang, S. Yang, K. Zheng, H. Xi, Z. Wang, H. Zhu, M. Zhao, I. Stoica, *et al.*, "Sla: Beyond sparsity in diffusion transformers via fine-tunable sparse-linear attention," in *arXiv preprint arXiv:2509.24006*, 2025. 2, 4, 11, 13
11. P. Zhang, Y. Chen, H. Huang, W. Lin, Z. Liu, I. Stoica, E. Xing, and H. Zhang, "Vsa: Faster video diffusion with trainable sparse attention," in *arXiv preprint arXiv:2505.13389*, 2025. 2, 13
12. P. Zhang, Y. Chen, R. Su, H. Ding, I. Stoica, Z. Liu, and H. Zhang, "Fast video generation with sliding tile attention," in *Forty-second International Conference on Machine Learning*, 2025. 2, 4
13. H. Xi, S. Yang, Y. Zhao, C. Xu, M. Li, X. Li, Y. Lin, H. Cai, J. Zhang, D. Li, J. Chen, I. Stoica, K. Keutzer, and S. Han, "Sparse video-gen: Accelerating video diffusion transformers with spatial-temporal sparsity," in *Forty-second International Conference on Machine Learning*, 2025. 2, 4
14. X. Li, C. Ma, X. Yang, and M.-H. Yang, "Vidtome: Video token merging for zero-shot video editing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2, 5
15. J. Choi, S. Lee, J. Chu, M. Choi, and H. J. Kim, "vid-tldr: Training free token merging for light-weight video transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2, 5
16. D. Park, M. Haji-Ali, Y. Li, W. Menapace, S. Tulyakov, H. J. Kim, A. Siarohin, and A. Kag, "Sprint: Sparse-dense residual fusion for efficient diffusion transformers," in *arXiv preprint arXiv:2510.21986*, 2025. 2, 5, 6, 10, 11

17. T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park, "One-step diffusion with distribution matching distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6613–6623, 2024. 2
18. T. Yin, M. Gharbi, T. Park, R. Zhang, E. Shechtman, F. Durand, and W. T. Freeman, "Improved distribution matching distillation for fast image synthesis," in *ArXiv preprint*, vol. abs/2405.14867, 2024. 2
19. Z. Zhang, Y. Li, Y. Wu, yanwu xu, A. Kag, I. Skorokhodov, W. Menapace, A. Siarohin, J. Cao, D. N. Metaxas, S. Tulyakov, and J. Ren, "SF-v: Single forward video generation model," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2
20. D. Park, T. Lee, M. Joo, and H. J. Kim, "Blockwise flow matching: Improving flow matching models for efficient high-quality generation," *arXiv preprint arXiv:2510.21167*, 2025. 2
21. Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *arXiv preprint arXiv:2303.01469*, 2023. 2, 4
22. F.-Y. Wang, Z. Huang, A. W. Bergman, D. Shen, P. Gao, M. Lingelbach, K. Sun, W. Bian, G. Song, Y. Liu, *et al.*, "Phased consistency model," in *arXiv preprint arXiv:2405.18407*, 2024. 2
23. K. Zheng, Y. Wang, Q. Ma, H. Chen, J. Zhang, Y. Balaji, J. Chen, M.-Y. Liu, J. Zhu, and Q. Zhang, "Large scale diffusion distillation via score-regularized continuous-time consistency," in *arXiv preprint arXiv:2510.08431*, 2025. 2, 3, 4, 13
24. Z. Geng, M. Deng, X. Bai, J. Z. Kolter, and K. He, "Mean flows for one-step generative modeling," in *arXiv preprint arXiv:2505.13447*, 2025. 2, 3, 4
25. T. Luo, H. Yuan, and Z. Liu, "Soflow: Solution flow models for one-step generative modeling," in *arXiv preprint arXiv:2512.15657*, 2025. 2, 4, 5, 6, 7, 9
26. D. Park, S. Lee, S. Kim, T. Lee, Y. Hong, and H. J. Kim, "Constant acceleration flow," *Advances in Neural Information Processing Systems*, 2024. 2
27. T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park, "One-step diffusion with distribution matching distillation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6613–6623, 2024. 3, 4
28. C. Lu and Y. Song, "Simplifying, stabilizing and scaling continuous-time consistency models," in *arXiv preprint arXiv:2410.11081*, 2024. 3, 4
29. T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," in *Advances in neural information processing systems*, 2022. 3, 4
30. T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," in *arXiv preprint arXiv:2307.08691*, 2023. 3, 4
31. W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023. 4
32. Y. Xia, S. Ling, F. Fu, Y. Wang, H. Li, X. Xiao, and B. Cui, "Training-free and adaptive sparse attention for efficient long video generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15982–15993, October 2025. 4
33. M. Zhou, H. Zheng, Z. Wang, M. Yin, and H. Huang, "Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation," in *Forty-first International Conference on Machine Learning*, 2024. 4
34. D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token merging: Your vit but faster," in *arXiv preprint arXiv:2210.09461*, 2022. 5

35. D. Bolya and J. Hoffman, "Token merging for fast stable diffusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4599–4603, 2023. 5
36. H. Zheng, W. Nie, A. Vahdat, and A. Anandkumar, "Fast training of diffusion models with masked transformers," in *arXiv preprint arXiv:2306.09305*, 2023. 5
37. S. Gao, P. Zhou, M.-M. Cheng, and S. Yan, "Mdtv2: Masked diffusion transformer is a strong image synthesizer," in *arXiv preprint arXiv:2303.14389*, 2023. 5
38. J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *NeurIPS*, 2020. 5
39. J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *arXiv preprint arXiv:2010.02502*, 2020. 5
40. D. Park, S. Kim, S. Lee, and H. J. Kim, "Ddmi: Domain-agnostic latent diffusion models for synthesizing high-quality implicit neural representations," *arXiv preprint arXiv:2401.12517*, 2024. 5
41. Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *arXiv preprint arXiv:2210.02747*, 2022. 5
42. X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate with rectified flow," in *ICLR*, 2023. 5
43. Y. Ma *et al.*, "Sit: Scaling up diffusion models with transformers," in *arXiv preprint arXiv:2401.00000*, 2024. 5
44. J. Su, Y. Lu, S. Pan, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," 2021. 8
45. Y. Zhang, Z. Lin, X. Yao, J. Hu, F. Meng, C. Liu, X. Men, S. Yang, Z. Li, W. Li, E. Lu, W. Liu, Y. Chen, W. Xu, L. Yu, Y. Wang, Y. Fan, L. Zhong, E. Yuan, D. Zhang, Y. Zhang, Y. T. Liu, H. Wang, S. Fang, W. He, S. Liu, Y. Li, J. Su, J. Qiu, B. Pang, J. Yan, Z. Jiang, W. Huang, B. Yin, J. You, C. Wei, Z. Wang, C. Hong, Y. Chen, G. Chen, Y. Wang, H. Zheng, F. Wang, Y. Liu, M. Dong, Z. Zhang, S. Pan, W. Wu, Y. Wu, L. Guan, J. Tao, G. Fu, X. Xu, Y. Wang, G. Lai, Y. Wu, X. Zhou, Z. Yang, and Y. Du, "Kimi linear: An expressive, efficient attention architecture," 2025. 8
46. S. Yang, J. Kautz, and A. Hatamizadeh, "Gated delta networks: Improving mamba2 with delta rule," in *arXiv preprint arXiv:2412.06464*, 2024. 8
47. Q. Fan, H. Huang, and R. He, "Breaking the low-rank dilemma of linear attention," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 25271–25280, 2025. 8
48. G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, L. Rouillard, T. Mesnard, G. Cideron, J. bastien Grill, S. Ramos, E. Yvinec, M. Casbon, E. Pot, I. Penchev, G. Liu, F. Visin, K. Kenealy, L. Beyer, X. Zhai, A. Tsitsulin, R. Busa-Fekete, A. Feng, N. Sachdeva, B. Coleman, Y. Gao, B. Mustafa, I. Barr, E. Parisotto, D. Tian, M. Eyal, C. Cherry, J.-T. Peter, D. Sinopalnikov, S. Bhupatiraju, R. Agarwal, M. Kazemi, D. Malkin, R. Kumar, D. Vilar, I. Brusilovsky, J. Luo, A. Steiner, A. Friesen, A. Sharma, A. Sharma, A. M. Gilady, A. Goedeckemeyer, A. Saade, A. Feng, A. Kolesnikov, A. Bendebury, A. Abdagic, A. Vadi, A. György, A. S. Pinto, A. Das, A. Bapna, A. Miech, A. Yang, A. Paterson, A. Shenoy, A. Chakrabarti, B. Piot, B. Wu, B. Shahriari, B. Petrini, C. Chen, C. L. Lan, C. A. Choquette-Choo, C. Carey, C. Brick, D. Deutsch, D. Eisenbud, D. Cattle, D. Cheng, D. Paparas, D. S. Sreepathihalli, D. Reid, D. Tran, D. Zelle, E. Noland, E. Huizenga, E. Kharitonov, F. Liu, G. Amirkhanyan, G. Cameron, H. Hashemi, H. Klimczak-Plucińska, H. Singh, H. Mehta, H. T. Lehri, H. Hazimeh, I. Ballantyne, I. Szpektor, I. Nardini, J. Pouget-Abadie, J. Chan, J. Stanton, J. Wieting, J. Lai, J. Orbay, J. Fernandez, J. Newlan,

J. yeong Ji, J. Singh, K. Black, K. Yu, K. Hui, K. Vodrahalli, K. Greff, L. Qiu, M. Valentine, M. Coelho, M. Ritter, M. Hoffman, M. Watson, M. Chaturvedi, M. Moynihan, M. Ma, N. Babar, N. Noy, N. Byrd, N. Roy, N. Momchev, N. Chauhan, N. Sachdeva, O. Bunyan, P. Botarda, P. Caron, P. K. Rubenstein, P. Culliton, P. Schmid, P. G. Sessa, P. Xu, P. Stanczyk, P. Tafti, R. Shivanna, R. Wu, R. Pan, R. Rokni, R. Willoughby, R. Vallu, R. Mullins, S. Jerome, S. Smoot, S. Girgin, S. Iqbal, S. Reddy, S. Sheth, S. Põder, S. Bhatnagar, S. R. Panyam, S. Eiger, S. Zhang, T. Liu, T. Yacovone, T. Liechty, U. Kalra, U. Evci, V. Misra, V. Roseberry, V. Feinberg, V. Kolesnikov, W. Han, W. Kwon, X. Chen, Y. Chow, Y. Zhu, Z. Wei, Z. Egyed, V. Cotruta, M. Giang, P. Kirk, A. Rao, K. Black, N. Babar, J. Lo, E. Moreira, L. G. Martins, O. Sanseviero, L. Gonzalez, Z. Gleicher, T. Warkentin, V. Mirrokni, E. Senter, E. Collins, J. Barral, Z. Ghahramani, R. Hadsell, Y. Matias, D. Sculley, S. Petrov, N. Fiedel, N. Shazeer, O. Vinyals, J. Dean, D. Hassabis, K. Kavukcuoglu, C. Farabet, E. Buchatskaya, J.-B. Alayrac, R. Anil, Dmitry, Lepikhin, S. Borgeaud, O. Bachem, A. Joulin, A. Andreev, C. Hardin, R. Dadashi, and L. Hussenot, "Gemma 3 technical report," 2025. 8

49. I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," in *arXiv:2004.05150*, 2020. 8
50. T. Karras, M. Aittala, T. Kynkäänniemi, J. Lehtinen, T. Aila, and S. Laine, "Guiding a diffusion model with a bad version of itself," in *Advances in Neural Information Processing Systems*, 2024. 10
51. M. Haji-Ali, W. Menapace, I. Skorokhodov, D. Park, A. Kag, M. Vasilkovsky, S. Tulyakov, V. Ordonez, and A. Siarohin, "One model, many budgets: Elastic latent interfaces for diffusion transformers," *arXiv preprint arXiv:2603.12245*, 2026. 10
52. J. Carreira, E. Noland, C. Hillier, and A. Zisserman, "A short note on the kinetics-700 human action dataset," 2022. 11
53. T. Unterthiner, S. van Steenkiste, K. Kurach, R. Marinier, M. Michalski, and S. Gelly, "Towards accurate generative models of video: A new metric & challenges," 2019. 12