

AnyFlow: Any-Step Video Diffusion Model with On-Policy Flow Map Distillation

Yuchao Gu^{1,2}, Guian Fang², Yuxin Jiang², Weijia Mao², Song Han^{1,3},
Han Cai^{1,*}, and Mike Zheng Shou^{2,*}

¹ NVIDIA

² Show Lab, National University of Singapore

³ MIT

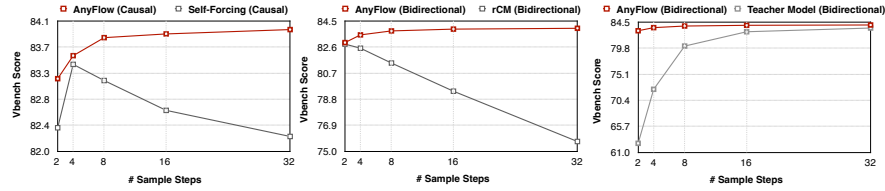
Abstract. Few-step video generation has been significantly advanced by consistency distillation. However, the performance of consistency-distilled models often degrades as more sampling steps are allocated at test time, limiting their effectiveness for any-step video diffusion. We argue that this limitation arises because consistency distillation replaces the original probability-flow ODE trajectory with a consistency-sampling trajectory, weakening the desirable test-time scaling behavior of ODE sampling. To address this limitation, we introduce **AnyFlow**, the first any-step video diffusion distillation framework based on flow maps. Instead of distilling a model for only a few fixed sampling steps, AnyFlow optimizes the full ODE sampling trajectory. To this end, we shift the distillation target from endpoint consistency mapping ($\mathbf{z}_t \rightarrow \mathbf{z}_0$) to flow-map transition learning ($\mathbf{z}_t \rightarrow \mathbf{z}_r$) over arbitrary time intervals. We further propose **Flow Map Backward Simulation**, which decomposes a full Euler rollout into shortcut flow-map transitions, enabling efficient on-policy distillation that reduces test-time errors (*i.e.*, discretization error in few-step sampling and exposure bias in causal generation). Extensive experiments across both bidirectional and causal architectures, at scales ranging from 1.3B to 14B parameters, demonstrate that AnyFlow achieves performance comparable to or better than consistency-based counterparts in the few-step regime, while supporting flexible and scalable sampling under varying step budgets. Code is released at <https://github.com/NVlabs/AnyFlow>.

Keywords: Flow Map Distillation · Video Diffusion Distillation · On-Policy Distillation

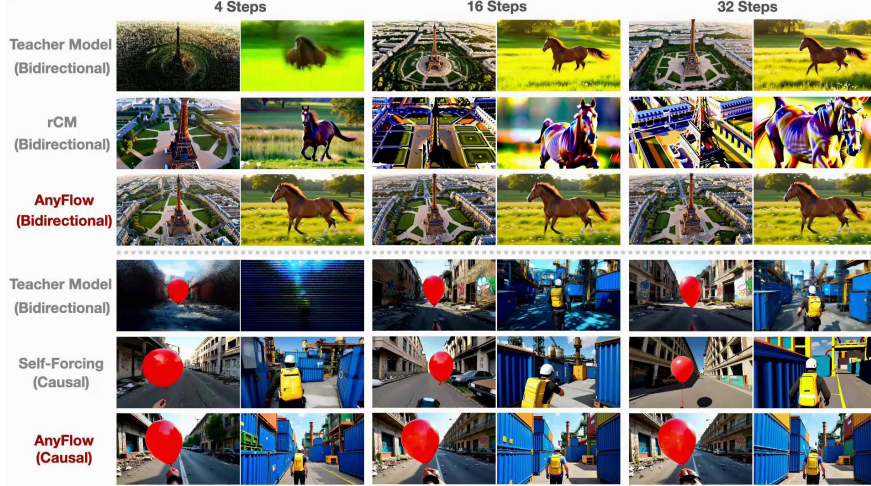
1 Introduction

Video diffusion models have recently achieved remarkable generation quality at scale [1, 9, 17, 37, 41], but most pipelines remain tied to fixed inference-step budgets. In practice, users require flexible generation: rapid previews for quick

* Corresponding author.



(a) Quantitative comparison of AnyFlow test-time scaling against consistency-based methods and the teacher model.



(b) Qualitative comparison of AnyFlow test-time scaling against consistency-based methods and the teacher model.

Fig. 1: Test-Time Scaling of AnyFlow. Compared to consistency-based methods (*e.g.*, rCM [48] for bidirectional video diffusion and Self-Forcing [12] for causal video diffusion), AnyFlow achieves strong performance in the few-step regime and scales effectively with increased sample steps. Compared to the teacher model, AnyFlow preserves test-time scaling ability while significantly improving the full trajectory. Readers can [click and play](#) the video clips in this figure using Adobe Acrobat.

iteration and higher-fidelity outputs for final delivery. This motivates the need for *any-step* models that can trade latency for quality at test time without retraining.

Existing few-step video distillation methods are predominantly built on consistency models [12, 24, 33, 34, 48]. While effective under very small sampling budgets, they do not provide robust any-step behavior. As illustrated in Fig. 1, representative methods (*e.g.*, rCM [48] for bidirectional models and Self-Forcing [12] for causal models) often degrade as the number of sampling steps increases, instead of improving with additional computation. The limitation is structural: consistency formulations emphasize fixed-point mappings to $\mathbf{z}_0$. During multi-step sampling, repeatedly re-noising intermediate states introduces cumulative bias, causing trajectories to drift away from the target PF-ODE path [31] rather than being progressively refined, as shown in Fig. 2(a).

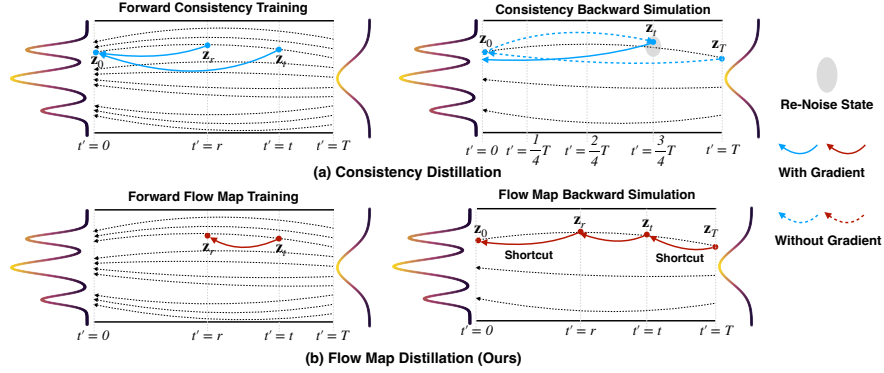


Fig. 2: Comparison of Distillation Paradigms. (a) Consistency distillation learns a fixed-point mapping from z_t to z_0 . Its backward simulation requires re-noising intermediate states, which causes trajectory drift under multi-step sampling. (b) Flow map distillation learns transitions from z_t to z_r . Its backward simulation decomposes Euler sampling trajectories into shortcut segments and enables efficient simulation of transitions across different step sizes.

To address this gap, we propose **AnyFlow**, the first any-step video diffusion distillation framework based on a two-time flow map formulation [2, 5, 31]. Flow maps generalize endpoint consistency by learning transitions between arbitrary time pairs, *i.e.*, $z_t \rightarrow z_r$ rather than only $z_t \rightarrow z_0$ (as shown in Fig. 2(b)), which naturally supports variable step sizes and inference budgets.

Specifically, AnyFlow contains two complementary stages. In the first stage, we develop an improved forward flow map training recipe to convert pretrained video diffusion models into flow map models, providing a strong initialization for any-step generation. However, forward flow map training alone cannot fully remove test-time errors: discretization error remains pronounced under low-NFE sampling, and exposure bias is still severe in causal generation. In the second stage, we introduce on-policy flow map distillation to optimize reverse divergence on model rollouts to mitigate test-time errors. The core design is flow map backward simulation, which replaces expensive full-trajectory simulation with shortcut decomposition, enabling efficient training of intermediate transitions over different time ranges. By combining forward flow map training with reverse-divergence supervision during on-policy distillation, AnyFlow achieves strong few-step quality while continuing to improve with more sampling steps. Moreover, because AnyFlow preserves the fine-grained flow field, the distilled model can be further adapted to downstream datasets via continued fine-tuning.

We validate AnyFlow on both bidirectional and causal video diffusion models, ranging from 1.3B to 14B parameters. Across these settings, **AnyFlow matches or surpasses consistency-based counterparts even with few steps and continues to improve as the number of sampling steps increases**. Our contributions are summarized as follows:

- We introduce AnyFlow, the first any-step video diffusion distillation framework built on flow maps, enabling a single model to support arbitrary inference budgets.
- We propose flow map backward simulation for on-policy flow map distillation to mitigate test-time errors (*i.e.*, discretization error and exposure bias) through efficient trajectory decomposition.
- We validate AnyFlow across architectures and model scales, establishing it as a scalable solution for high-quality, any-step video generation.

2 Related Work

2.1 Consistency Models

Consistency Models (CMs) [34] accelerate diffusion sampling by directly learning a mapping from $\mathbf{z}_t$ to $\mathbf{z}_0$ along the probability-flow ODE (PF-ODE) defined by a teacher model. For multi-step generation, CMs repeatedly inject noise into intermediate states and perform iterative denoising. Subsequent studies improve CM training stability through annealed time schedules [6, 33] and segmented consistency objectives [18, 30, 38]. sCM [24] further simplifies these design choices and provides a strong practical baseline. Building on sCM, rCM [48] introduces score distillation as a regularizer, achieving strong performance in both image and video diffusion distillation. In contrast to prior CM-based approaches, we explore flow map distillation to enable any-step video generation.

2.2 Flow Map Models

Flow Map Models [2, 16, 31] offer a unified perspective on diffusion modeling and diffusion distillation by learning a generalized transition operator $\mathbf{f}_\theta(\mathbf{z}_t, t, r)$ between arbitrary points along PF-ODE trajectories. From this perspective, the formulation recovers consistency modeling [34] when $r = 0$ and reduces to standard flow matching [21, 22] when $t = r$. However, training flow map models is challenging because it requires accurate transition learning across arbitrary time pairs. MeanFlow [5] is a representative approach that connects instantaneous and average velocities, but it relies on Jacobian-vector products (JVPs), which are difficult to scale under Fully Sharded Data Parallel (FSDP). To overcome this limitation, subsequent studies either approximate JVP terms numerically [39] or derive JVP-free algebraic formulations [7, 26]. Building on these advances, AnyFlow leverages flow map distillation for scalable any-step video diffusion, enabling efficient and flexible inference for high-fidelity generation.

2.3 Video Diffusion Distillation

As shown in *supplementary material*, recent video diffusion distillation methods typically follow a two-stage pipeline: (1) forward training for initialization and (2) on-policy distillation for refinement. The first stage improves optimization

stability, while the second reduces rollout drift during few-step inference. For example, rCM [48] initializes from sCM [24] and then applies consistency backward simulation [43] with distribution matching [44]. In causal video generation, Self-Forcing [12] adopts data-free consistency ODE initialization [45] before performing a similar on-policy refinement stage. Our method follows the same high-level pipeline but replaces consistency modeling with a flow map formulation, enabling stronger any-step performance.

Causal Video Diffusion. A growing line of work studies causal video diffusion models [15, 32, 40, 46]. These models often suffer from exposure bias, which leads to error accumulation during autoregressive generation. Self-Forcing [12] mitigates this issue through on-policy distillation, but its consistency-based formulation is mainly tailored to few-step settings for rollout efficiency. Another line of work [8] focuses more on multi-step models by explicitly modeling test-time errors during pretraining, but it is not directly designed for efficient few-step sampling. By contrast, our method starts from a flow map formulation and uses flow map backward simulation to support both few-step and multi-step causal video generation within a unified framework.

2.4 On-Policy Distillation

On-policy distillation mitigates test-time errors by training the student on its own generated trajectories and supervising them with a stronger teacher. This approach has demonstrated benefits in both large language models [25, 42, 47] and diffusion models [12, 43, 44].

For diffusion models, on-policy distillation is typically implemented via Distribution Matching Distillation (DMD) [43, 44] or adversarial distillation [19, 20]. Self-Forcing [12] combines self-rollouts with distribution matching against a bidirectional teacher, while APT2 [20] evaluates one-step video rollouts using a discriminator. Our method follows the same on-policy distillation paradigm but introduces flow map backward simulation, specifically designed for the flow map formulation, enabling any-step video generation.

3 Preliminary

3.1 Flow Map Formulation

We first introduce the basic flow map formalism used in AnyFlow. Let $\mathbf{z}_t$ denote the latent state at continuous time $t \in [0, 1]$ under the probability-flow ODE (PF-ODE):

$$\frac{d\mathbf{z}_t}{dt} = \mathbf{v}(\mathbf{z}_t, t), \quad (1)$$

where $\mathbf{v}$ is the velocity field.

Define the exact flow map $\Phi_{r \leftarrow t}$ of this ODE as the operator that transports states from time t to time r , *i.e.*, $\Phi_{r \leftarrow t}(\mathbf{z}_t) = \mathbf{z}_r$ for $1 \geq t \geq r \geq 0$. It satisfies two

Method	NFEs	Bidirectional Video Diffusion			Causal Video Diffusion		
		Quality	Semantic	Overall	Quality	Semantic	Overall
Forward Training							
Flow Matching Training	4×2	77.82	61.91	74.64	79.05	67.79	76.80
	32×2	85.48	77.24	83.83	85.21	76.65	83.50
Consistency ODE-Init [12,45]	4	83.01	70.13	80.44	76.99	61.88	73.97
	32	84.78	75.15	82.86	79.73	68.81	77.55
Flow Map Training	4	84.39	71.20	81.75	82.80	71.16	80.48
	32	85.35	75.63	83.40	85.23	74.71	83.13
Forward Training + On-Policy Distillation							
Consistency ODE-Init + Consistency Backward Simulation [12]	4	84.77	75.71	82.96	84.37	74.97	82.49
	32	83.70	64.18	79.80	82.43	68.48	79.64
Flow Map Training + Consistency Backward Simulation	4	85.47	75.88	83.55	84.99	74.97	82.99
	32	85.15	74.19	82.96	85.67	74.78	83.49
Flow Map Training + Flow Map Backward Simulation (Ours)	4	85.24	76.41	83.48	85.60	75.30	83.54
	32	85.70	76.99	83.96	85.92	76.12	83.96

Table 1: Quantitative ablation of key designs in AnyFlow. Flow map training provides a stronger initialization for few-step sampling than flow matching training and consistency ODE-Init. However, forward training alone still suffers from test-time errors, which are further mitigated by the on-policy distillation stage.

standard properties: identity $\Phi_{t \leftarrow t}(\mathbf{z}) = \mathbf{z}$ and composition $\Phi_{q \leftarrow r} \circ \Phi_{r \leftarrow t} = \Phi_{q \leftarrow t}$ for $t \geq r \geq q$. In practice, a neural flow map model learns an approximation

$$\mathbf{f}_\theta(\mathbf{z}_t, t, r) \approx \mathbf{z}_r, \quad 1 \geq t > r \geq 0, \quad (2)$$

with boundary condition $\mathbf{f}_\theta(\mathbf{z}_t, t, t) = \mathbf{z}_t$. Compared with endpoint-only mappings, this parameterization models transitions between arbitrary time pairs, which naturally supports variable step sizes and any-step inference.

MeanFlow Objective. MeanFlow [5] is a representative algorithm for training flow map models. It parameterizes the averaged transport velocity on $[r, t]$ and approximates the transition as $\mathbf{f}_\theta(\mathbf{z}_t, t, r) = \mathbf{z}_t - (t - r) \mathbf{u}_\theta(\mathbf{z}_t, r, t)$. Following this formulation, we optimize $\mathbf{u}_\theta$ by:

$$\mathcal{L}(\theta) = \mathbb{E} \left[\left\| \mathbf{u}_\theta(\mathbf{z}_t, r, t) - \text{sg}(\mathbf{u}_{\text{tgt}}) \right\|_2^2 \right], \quad (3)$$

where $\mathbf{u}_{\text{tgt}} = \mathbf{v}(\mathbf{z}_t, t) - (t - r) \frac{d\mathbf{u}_\theta(\mathbf{z}_t, r, t)}{dt}$. Here, $\text{sg}(\cdot)$ denotes stop-gradient.

Differential Derivation Equation. Computing the Jacobian-vector product (JVP) for the derivative term $\frac{d\mathbf{u}_\theta(\mathbf{z}_t, r, t)}{dt}$ is expensive and not fully compatible with Fully Sharded Data Parallel (FSDP). Transition Model [39] proposes to approximate this derivative using the finite difference method:

$$\frac{d}{dt} \mathbf{u}(\mathbf{z}_t, r, t) \approx \frac{\mathbf{u}(\mathbf{z}_{t+\Delta t}, r, t + \Delta t) - \mathbf{u}(\mathbf{z}_{t-\Delta t}, r, t - \Delta t)}{2\Delta t}, \quad (4)$$

This approximation requires only two forward passes per training step and is compatible with FSDP.



Fig. 3: Qualitative Ablation of On-Policy Distillation. (a) Forward flow map training alone still suffers from test-time errors, including discretization error and exposure bias. (b) On-policy flow map distillation optimizes reverse divergence on model self-rollouts to mitigate these test-time errors. Readers can [click](#) and [play](#) the video clips in this figure using Adobe Acrobat.

3.2 On-Policy Distillation

On-policy distillation is designed to reduce train-test mismatch by training the student on its own rollout states while being guided by a strong teacher. In diffusion models, this paradigm is mainly instantiated as Distribution Matching Distillation (DMD). It typically consists of two key components: backward simulation and a distribution-matching objective.

Backward Simulation. Given noise $\mathbf{z}$, the student first produces a sample through its inference trajectory, $\hat{\mathbf{z}}_0 = \mathbf{f}_\theta(\mathbf{z})$. This self-generated trajectory is then used to compute the distillation loss, which helps mitigate exposure bias and discretization error at test time.

Distribution Matching Objective. We follow DMD-style reverse-KL training [43, 44]. First, DMD re-noises the self-rollout sample $\hat{\mathbf{z}}_0$ at $t \in [0, T]$ as $\mathbf{z}_t = (1 - t)\hat{\mathbf{z}}_0 + t\epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. It then estimates the DMD gradient as follows:

$$\nabla_\theta \mathcal{L}_{\text{DMD}} = -\mathbb{E}_{t, \mathbf{z}} \left[\left(s_{\text{real}}(\mathbf{z}_t, t) - s_{\text{fake}}(\mathbf{z}_t, t) \right) \frac{\partial \mathbf{f}_\theta(\mathbf{z})}{\partial \theta} \right], \quad (5)$$

where s_{real} and s_{fake} are the real and fake score functions, respectively.

4 Method

4.1 AnyFlow Motivation

As summarized in *supplementary material*, most recent few-step video distillation methods are built on consistency models. Although effective under very small sampling budgets, their performance often degrades as the number of sampling steps increases (Fig. 1). The underlying reason is that consistency-style

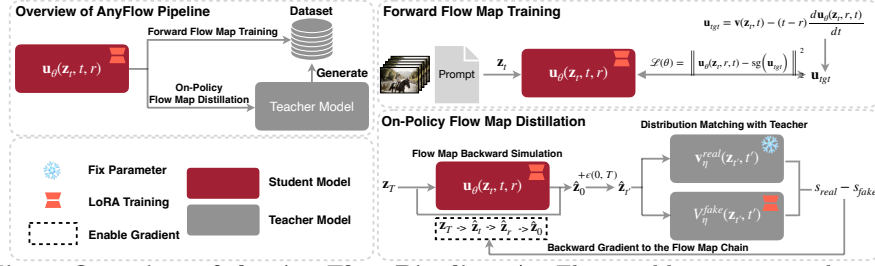


Fig. 4: Overview of the AnyFlow Pipeline. AnyFlow enables any-step video generation by jointly learning forward flow map training from synthetic data and on-policy distillation with flow map backward simulation under teacher guidance.

sampling repeatedly performs endpoint projection and re-noising, which gradually drives the trajectory away from the target PF-ODE path during multi-step inference.

Motivation. Our goal is to preserve the strong few-step efficiency of distilled video diffusion models while enabling quality to improve, rather than deteriorate, as more sampling steps are used. To this end, AnyFlow shifts the distillation target from fixed-point endpoint mapping to flow map transition learning, *i.e.*, $\mathbf{f}_\theta : (\mathbf{z}_t, t, r) \mapsto \mathbf{z}_r$. Instead of only predicting $\mathbf{z}_0$, the model learns transitions between arbitrary time pairs. This formulation naturally supports both small time gaps for stable local refinement and large time gaps for efficient long-range jumps, making it well suited for any-step generation.

Challenge. Forward flow map training alone (*e.g.*, MeanFlow [5]) is insufficient for strong test-time performance in video generation. As shown in Fig. 3 and Tab. 1, it still suffers from test-time errors, especially discretization error in few-step sampling and exposure bias in causal generation. This motivates an additional on-policy distillation stage to correct rollout mismatch. However, designing such a stage for flow map models is non-trivial. Unlike consistency models, which can naturally reach $\mathbf{z}_0$ at intermediate steps and therefore allow KL gradients to be computed directly at endpoint states, flow map models require a new simulation strategy that can accommodate transitions between arbitrary time pairs while remaining efficient for training.

4.2 AnyFlow Pipeline

Overview. As illustrated in Fig. 4, AnyFlow is trained in two complementary stages: forward flow map training (Sec. 4.2.1) and on-policy distillation (Sec. 4.2.2). In Stage 1, we fine-tune a pretrained video diffusion model on teacher-synthesized data to learn a stable transition operator $\mathbf{f}_\theta : (\mathbf{z}_t, t, r) \mapsto \mathbf{z}_r$, which provides a strong initialization for any-step generation. In Stage 2, starting from the converged Stage-1 checkpoint, we jointly optimize forward flow map training and DMD-based on-policy distillation. This stage uses flow map backward simulation to generate student rollouts and apply reverse-divergence supervision from a strong teacher, thereby reducing discretization error and exposure bias while preserving any-step capability.

Algorithm 1: AnyFlow: Forward Flow Map Training

```

# fn(z, c, t, r): model to predict u
# x: training batch
# c: text embedding
# g: CFG scale

t, r = sample_t_r() # t > r
e = randn_like(x)

z = (1 - t) * x + t * e
v = e - x

# Guidance-Fused Training
u_c = fn(z, c, t, r)
u_∅ = fn(z, ∅, t, r)
u =  $\frac{1}{g}$  (u_c - (1 - g) * stopgrad(u_∅))

# Differential Derivation Equation
z+ = z + εv
z- = z - εv
 $\frac{du}{dt} = \frac{fn(z^+, c, t + \epsilon, r) - fn(z^-, c, t - \epsilon, r)}{2\epsilon g}$ 
utgt = v - (t - r)  $\frac{du}{dt}$ 

loss = metric(u - stopgrad(utgt))

```

Algorithm 2: AnyFlow: On-Policy Flow Map Distillation

```

# fn(z, c, t, r): model to predict u
# x: training batch
# c: text embedding

while true do
    # Sample Inference Step
    s ~ Uniform(1, 2, ..., T)

    # Sample Gradient Step
    t ~ Uniform(1, 2, ..., s)
    r = t - T/s

    # Sample Initial Noise
    z_T = randn_like(x)

    # Flow Map Backward Simulation
    z_t = fn(z_T, c, T, t) # With Grad
    z_r = fn(z_t, c, t, r) # With Grad
    z_0 = fn(z_r, c, r, 0) # With Grad

    Update θ via distribution matching
    loss using z_0 and renoise at [0, T]
end

```

4.2.1 Flow Map Training We employ the MeanFlow objective (eq. (3)) alongside the differential derivation equation (eq. (4)) as our primary training framework. Because the original MeanFlow architecture is optimized for pre-training, directly adapting it to post-training for video diffusion models requires several additional design choices for strong performance. To this end, we introduce several modifications: interpolated timestep conditioning, time sampling, guidance-fused training, and adaptive loss reweighting.

Interpolated Timestep Conditioning. Converting a pretrained diffusion model into a flow map model requires introducing an additional timestep, r . A natural choice is to follow MeanFlow [5] and use $\text{emb}(t) + \text{emb}'(t - r)$. Prior work such as TMD [28] further applies a zero-initialized output projection to emb' for a smooth start in post-training. However, we find this design unstable in our setting: it tends to produce timestep embeddings with much larger norms than the pretrained ones, leading to over-saturated generation results, as shown in *supplementary material*.

To address this issue, we adopt an interpolated timestep conditioning scheme:

$$g \cdot \text{emb}(t) + (1 - g) \cdot \text{emb}'(r),$$

where emb' is initialized from the pretrained emb and g is fixed to 0.25 in all experiments. At the beginning of training, this formulation reduces to the pretrained timestep embedding in the boundary case $t = r$. As verified in *supple-*

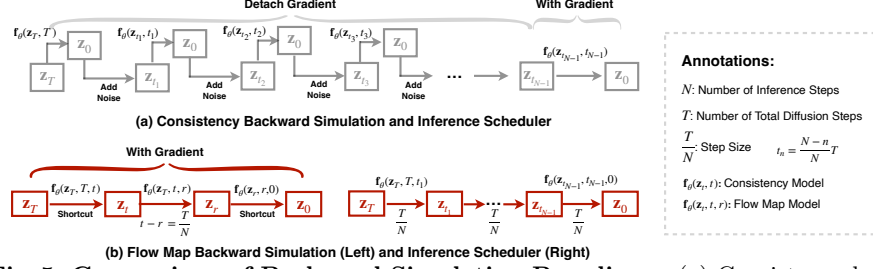


Fig. 5: Comparison of Backward Simulation Paradigms. (a) Consistency backward simulation and sampling both follow consistency trajectories, where truncated gradients are used to reduce training cost of on-policy distillation. (b) Flow map backward simulation enables efficient simulation of Euler sampling trajectories. It decomposes the full rollout trajectory into shortcut segments to reduce the training cost of on-policy distillation.

mentary material, interpolated timestep conditioning keeps the embedding norm well aligned with the pretrained model and suppresses over-saturated results observed in zero-init timestep conditioning.

Time Sampler. Following MeanFlow [5], we first sample t and r from a uniform distribution and then reorder them as $t = \max(t, r)$ and $r = \min(t, r)$. We further introduce a reweighting function $w(t)$ based on the sampled timestep t to balance the training objective across different noise levels. Different choices of $w(t)$ are discussed in the *supplementary material*.

Guidance-Fused Training. We also incorporate classifier-free guidance (CFG) into flow map training following MeanFlow [5], which allows CFG to be omitted at test time for faster inference. Unlike MeanFlow, which fuses the CFG objective into the target velocity field $\mathbf{u}_{\text{tgt}}$, we fuse it into the prediction to better align with the guidance scale defined by the pretrained diffusion model:

$$\mathbf{u} = \frac{1}{g} (\mathbf{u}_c - (1 - g) \text{sg}(\mathbf{u}_\emptyset)), \quad (6)$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operation and $\mathbf{u}_\emptyset$ is the model prediction with null conditioning.

Adaptive Loss Reweighting. To stabilize flow map training, we introduce an adaptive loss reweighting scheme. Unlike MeanFlow [5], which uses a weighting function $w = 1/(\|\Delta\|_2^2 + c)$ for pre-training, where Δ denotes the regression loss and c is a small constant, our approach is designed for post-training, where a reliable instantaneous velocity field has already been established at $t = r$. We use the loss at $t = r$ as a baseline to dynamically scale the loss for other time steps $t \neq r$. In each iteration, we sample 50% of the batch from the boundary cases ($t = r$) and define the adaptive weight $w_{t,r}$ as:

$$w_{t,r} = \begin{cases} 1, & t = r \\ \frac{\mu_{t=r}}{\|\Delta\|_2^2 + c}, & t \neq r \end{cases} \quad (7)$$



Fig. 6: Qualitative Ablation of Backward Simulation. (a) Consistency backward simulation exhibits over-smoothed textures and degraded motion as more sampling steps are allocated. (b) Flow map backward simulation produces a straighter trajectory, enabling test-time scaling while maintaining strong few-step quality. Readers can [click](#) and [play](#) the video clips in this figure using Adobe Acrobat.

where $\mu_{t=r} = \mathbb{E}_{t=r}[\|\Delta\|_2^2]$ is the average regression loss over the boundary samples. This formulation aligns the loss magnitude at $t \neq r$ with the well-optimized field at $t = r$, thereby preserving the learned instantaneous velocity and preventing gradient instability.

Discussion. The full algorithm of our improved forward flow map training is summarized in Alg. 1. As shown in Tab. 1, compared with standard flow matching, flow map training achieves comparable multi-step performance while delivering significantly better few-step performance. Compared with Consistency ODE-Init, it achieves stronger multi-step performance while maintaining comparable few-step quality. Overall, flow map training provides a strong initialization for the subsequent on-policy distillation stage.

4.2.2 On-Policy Flow Map Distillation Although forward flow map training provides a strong initialization for any-step generation, it still suffers from test-time errors, especially discretization error at low NFEs and exposure bias in causal generation, as shown in Fig. 3. To correct rollout drift, we further introduce on-policy distillation with teacher guidance. Following prior work [12, 48], we instantiate on-policy diffusion distillation with distribution matching distillation (DMD), which requires the student to perform a self-rollout (*i.e.*, backward simulation) to $\mathbf{z}_0$ before re-noising in order to compute the Kullback–Leibler (KL) gradient. Below, we describe our design of backward simulation for flow map models.

Limitations of Consistency Backward Simulation. Existing backward simulation methods for DMD [12, 48] largely follow the logic of consistency models, as illustrated in Fig. 5(a). This design is convenient for on-policy distillation because the consistency scheduler can reach the $\mathbf{z}_0$ state at intermediate steps, making endpoint-based supervision straightforward. In practice, gradients are often truncated before the target gradient step to avoid the memory cost and

instability of backpropagating through a long rollout chain. In inference time, the same consistency sampler is then reused.

However, unlike ODE trajectories learned through flow-matching pretraining, consistency samplers require additional re-noising to obtain intermediate states during multi-step sampling. These re-noised states deviate from the base PF-ODE trajectory, which limits generalization beyond the simulation step used during simulation. Moreover, simulating arbitrary step counts with consistency backward simulation is computationally expensive because it requires rolling out the full trajectory.

Flow Map Backward Simulation. Unlike consistency backward simulation, our design directly exploits the learned transition capability of flow map models. Since a flow map model is able to predict transitions between arbitrary time pairs, it can naturally shortcut long rollout trajectories instead of explicitly simulating every intermediate step, based on the composition property of flow maps:

$$\mathbf{f}_\theta(\mathbf{z}_t, t, q) \approx \mathbf{f}_\theta(\mathbf{f}_\theta(\mathbf{z}_t, t, r), r, q), \quad t > r > q. \quad (8)$$

Concretely, for a target sampling budget of N steps, we first choose an intermediate timestep t along the sampled trajectory and then set the next timestep r such that $t - r = \frac{T}{N}$. As illustrated in Fig. 5, this decomposes a rollout trajectory from T to 0 into three segments: $T \rightarrow t$, $t \rightarrow r$, and $r \rightarrow 0$. The first and last segments, $T \rightarrow t$ and $r \rightarrow 0$, are handled by shortcut transitions of the learned flow map, while $t \rightarrow r$ is the target transition along the sampled trajectory. Once the KL gradient is computed at z_0 , it is backpropagated through the whole chain. By varying N during training, we can efficiently simulate different inference step budgets with the same computation cost.

After on-policy distillation, we can simply use a standard Euler scheduler for sampling. Because flow map backward simulation trains transitions over different time intervals, the same model naturally supports different inference budgets at test time.

Discussion. The full algorithm of our on-policy distillation stage is shown in Alg. 2. As shown in Tab. 1, Consistency ODE-Init combined with consistency backward simulation exhibits the weakest test-time scaling, especially at 32 NFEs. Replacing the forward training stage with flow map training narrows this gap, but noticeable degradation still remains under multi-step sampling. By further adopting our flow map backward simulation, both few-step and multi-step performance are improved through on-policy distillation. As shown in Fig. 6, flow map backward simulation produces a straighter trajectory, which leads to more consistent improvement as the sampling budget increases, whereas consistency-based simulation degrades at 32 NFEs.

5 Experiments

5.1 Implementation Details

We implement AnyFlow on top of Wan2.1 [37] in the Diffusers framework [29]. We train the model on a synthetic dataset of 256K prompt-video pairs gen-

Model	#Params	Resolution	NFEs	Evaluation Scores $\uparrow$		
				Quality	Semantic	Total
<i>Bidirectional Video Diffusion Models</i>						
LTX-Video [9]	1.9B	512×768	40×2	82.30	70.79	80.00
CogVideoX-2B [41]	2B	480×720	50×2	82.48	77.81	81.55
HunyuanVideo [17]	13B	720×1280	50×2	85.09	76.88	83.24
$\dagger$ Wan2.1-T2V-1.3B [37]	1.3B	480×832	50×2	84.99	76.23	83.24
$\dagger$ rCM-Wan2.1-T2V-1.3B [48]	1.3B	480×832	4	84.71	73.74	82.51
$\dagger$ AnyFlow-Wan2.1-T2V-1.3B	1.3B	480×832	4	85.24	76.41	83.48
$\dagger$ AnyFlow-Wan2.1-T2V-1.3B	1.3B	480×832	32	85.70	76.99	83.96
$\dagger$ Wan2.1-T2V-14B [37]	14B	480×832	50×2	85.77	75.58	83.74
$\dagger$ rCM-Wan2.1-T2V-14B [48]	14B	480×832	4	85.47	76.72	83.73
$\dagger$ AnyFlow-Wan2.1-T2V-14B	14B	480×832	4	85.70	77.38	84.04
$\dagger$ AnyFlow-Wan2.1-T2V-14B	14B	480×832	32	85.76	77.44	84.10
<i>Causal Video Diffusion Models</i>						
MAGI-1 [36]	4.5B	720×720	8	80.98	65.68	77.92
SkyReels-V2 [3]	1.3B	540×960	50×2	84.70	74.53	82.67
$\dagger$ Self-Forcing [12]	1.3B	480×832	4	85.23	76.01	83.39
$\dagger$ AnyFlow-FAR-Wan2.1-1.3B	1.3B	480×832	4	85.60	75.30	83.54
$\dagger$ AnyFlow-FAR-Wan2.1-1.3B	1.3B	480×832	32	85.92	76.12	83.96
$\dagger$ LightX2V-Wan2.1-14B-CausVid [4]	14B	480×832	9	85.29	75.96	83.42
$\dagger$ FastVideo-CausalWan2.2-A14B-Preview [35]	14B	480×832	8	84.28	78.49	83.12
$\dagger$ Krea-Realtime-Wan2.1-14B [27]	14B	480×832	4	84.80	77.07	83.25
$\dagger$ AnyFlow-FAR-Wan2.1-14B	14B	480×832	4	85.82	76.97	84.05
$\dagger$ AnyFlow-FAR-Wan2.1-14B	14B	480×832	32	86.12	77.55	84.41

Table 2: Text-to-Video Evaluation on VBench. $\dagger$ indicates that we re-evaluate model performance using the official VBench augmented prompts under the same setting. Results for all other models are taken directly from their respective papers.

Model	#Params	Resolution	NFEs	Evaluation Scores $\uparrow$		
				Quality	I2V	Total
CogVideoX-5B-I2V [41]	5B	480×720	50×2	78.61	94.79	86.70
HunyuanVideo-I2V [17]	13B	720×1280	50×2	78.54	95.10	86.82
Step-Video-TI2V [11]	30B	540×960	50×2	81.22	95.50	88.36
MAGI-1 [36]	24B	720×1280	32×2	82.44	96.12	89.28
$\dagger$ Wan2.1-I2V-14B [37]	14B	480×832	50×2	80.30	95.12	87.71
$\dagger$ FastVideo-CausalWan2.2-A14B-Preview [35]	14B	480×832	8	78.82	94.81	86.82
$\dagger$ AnyFlow-FAR-Wan2.1-14B	14B	480×832	4	80.39	95.35	87.87

Table 3: Image-to-Video Evaluation on VBench-I2V. $\dagger$ indicates that we re-evaluate model performance under the same setting.

erated from Wan2.1-T2V-14B, where each sample contains up to 81 frames at a resolution of 480×832 . Training is performed in two stages. In Stage 1, we optimize the forward flow map objective using AdamW [23] with a learning rate of 5×10^{-5} and a global batch size of 32 for the 1.3B model and 16 for the 14B model, for 6,000 and 4,000 iterations, respectively. In Stage 2, we perform on-policy distillation by jointly optimizing the forward flow map and on-policy objectives with AdamW at a learning rate of 2×10^{-6} for 800 iterations. In both stages, we use parameter-efficient fine-tuning with LoRA [10] of rank 256.

5.2 Evaluation Setting

For text-to-video (T2V) evaluation, we adopt VBench [13], which evaluates generation quality across 16 fine-grained dimensions grouped into two high-level

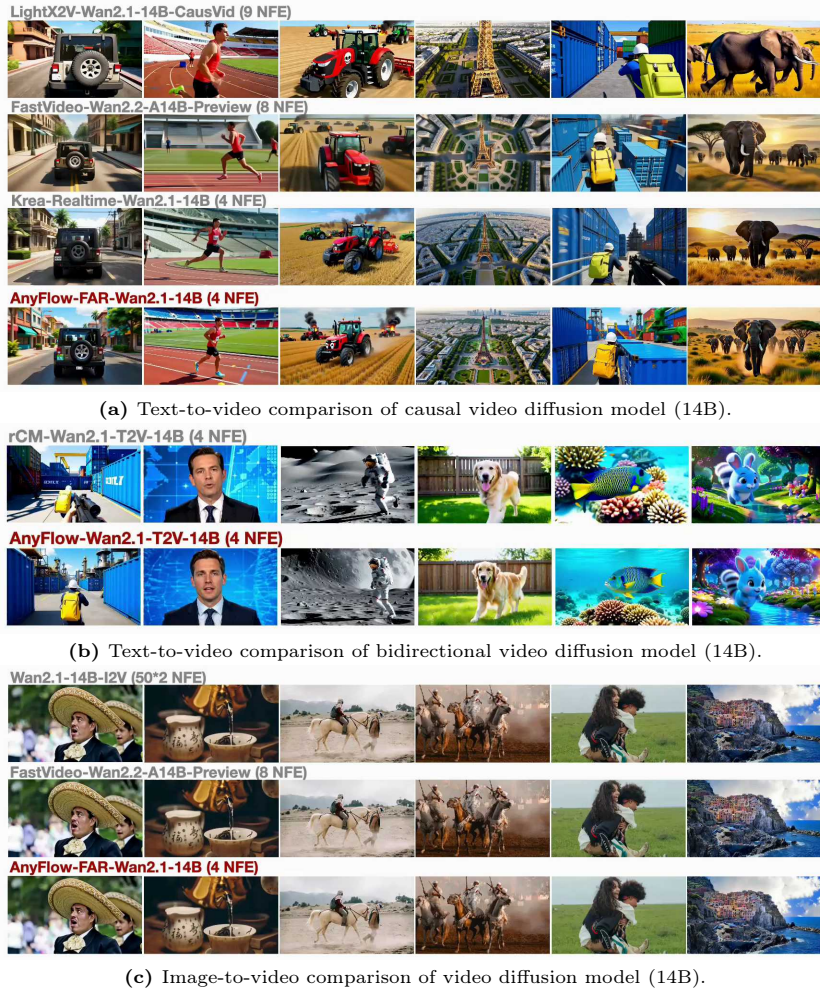


Fig. 7: Qualitative Comparison of AnyFlow and Baselines at the 14B Model Scale. Readers can [click](#) and [play](#) the video clips in this figure using Adobe Acrobat. categories: Quality score and Semantic score. For image-to-video (I2V) evaluation, we use VBench-I2V [14] and report both the Quality score and the I2V score. To ensure a fair comparison, we re-evaluate key counterparts under a unified protocol using the official VBench augmented prompts. These counterparts include the Wan2.1 base model [37], Self-Forcing [12], and rCM [48], as well as community-trained 14B causal models including FastVideo [35], Krea-Realtime-14B [27], and LightX2V [4]. Results for all other methods are taken directly from their original papers.

5.3 Main Results

Quantitative Comparison. As shown in Tab. 2, AnyFlow achieves strong few-step performance and continues to improve as the sampling budget increases.

On the 14B bidirectional backbone, AnyFlow-Wan2.1-T2V-14B obtains 84.04 at 4 NFEs, outperforming rCM-Wan2.1-T2V-14B (83.73 at 4 NFEs). On the 14B causal backbone, AnyFlow-FAR-Wan2.1-14B reaches 84.05 at 4 NFEs and further improves to 84.41 at 32 NFEs, outperforming strong community counterparts.

For image-to-video results in Tab. 3, AnyFlow-FAR-Wan2.1-14B achieves a VBench-I2V score of 87.87 using only 4 NFEs. This result is comparable to Wan2.1-I2V-14B with 50×2 NFEs (87.71) and surpasses FastVideo-CausalWan2.2-A14B-Preview (86.82), showing that AnyFlow improves both visual quality and I2V faithfulness while retaining strong sampling efficiency.

Qualitative Comparison. As shown in Fig. 7, we provide qualitative comparisons at the 14B scale for both T2V and I2V generation. In causal T2V examples, FastVideo-Wan2.2-A14B-Preview exhibits blurry details, LightX2V-Wan2.1-14B-CausVid shows flickering artifacts, and Krea-Realtime-14B produces unrealistic motion in some clips. By comparison, AnyFlow-FAR-Wan2.1-14B delivers the best visual quality and motion with 4 NFEs. In bidirectional T2V examples, AnyFlow-Wan2.1-T2V-14B shows slightly better motion than rCM-Wan2.1-T2V-14B.

For I2V generation, AnyFlow reuses the causal generator through our non-uniform chunk partition and maintains good first-frame faithfulness with smooth motion transitions. Compared with Wan2.1-I2V-14B, AnyFlow-FAR-Wan2.1-14B shows comparable temporal stability and motion quality, while FastVideo-Wan2.2-A14B-Preview exhibits visible flickering and inconsistencies with the input image. Overall, these qualitative results are consistent with the quantitative trends and further support the effectiveness of AnyFlow in few-step sampling.

6 Conclusion

We present AnyFlow, the first any-step video diffusion distillation framework based on a two-time flow map formulation. By learning transitions between arbitrary time pairs, AnyFlow supports a wide range of sampling budgets within a single model. Built on large pretrained video diffusion backbones, it combines an improved forward flow map training recipe with flow map backward simulation for on-policy distillation, thereby reducing discretization error and exposure bias during sampling. Across both bidirectional and causal architectures, and across model scales from 1.3B to 14B, AnyFlow consistently matches or outperforms strong consistency-based counterparts in the few-step regime while continuing to improve as the sampling budget increases. These results highlight a practical and scalable path toward high-quality any-step video generation.

Acknowledgements

This project is supported by the Ministry of Education, Singapore, under the Academic Research Fund Tier 1 (FY2023), and by the College of Design and Engineering at the National University of Singapore.

References

1. Agarwal, N., Ali, A., Bala, M., Balaji, Y., Barker, E., Cai, T., Chattopadhyay, P., Chen, Y., Cui, Y., Ding, Y., et al.: Cosmos world foundation model platform for physical ai. arXiv preprint arXiv:2501.03575 (2025)
2. Boffi, N.M., Albergo, M.S., Vanden-Eijnden, E.: Flow map matching with stochastic interpolants: A mathematical framework for consistency models. arXiv preprint arXiv:2406.07507 (2024)
3. Chen, G., Lin, D., Yang, J., Lin, C., Zhu, J., Fan, M., Zhang, H., Chen, S., Chen, Z., Ma, C., et al.: Skyreels-v2: Infinite-length film generative model. arXiv preprint arXiv:2504.13074 (2025)
4. Contributors, L.: Lightx2v: Light video generation inference framework. <https://github.com/ModelTC/lightx2v> (2025)
5. Geng, Z., Deng, M., Bai, X., Kolter, J.Z., He, K.: Mean flows for one-step generative modeling. arXiv preprint arXiv:2505.13447 (2025)
6. Geng, Z., Pokle, A., Luo, W., Lin, J., Kolter, J.Z.: Consistency models made easy. arXiv preprint arXiv:2406.14548 (2024)
7. Guo, Y., Wang, W., Yuan, Z., Cao, R., Chen, K., Chen, Z., Huo, Y., Zhang, Y., Wang, Y., Liu, S., et al.: Splitmeanflow: Interval splitting consistency in few-step generative modeling. arXiv preprint arXiv:2507.16884 (2025)
8. Guo, Y., Yang, C., He, H., Zhao, Y., Wei, M., Yang, Z., Huang, W., Lin, D.: End-to-end training for autoregressive video diffusion via self-resampling. arXiv preprint arXiv:2512.15702 (2025)
9. HaCohen, Y., Chiprut, N., Brazowski, B., Shalem, D., Moshe, D., Richardson, E., Levin, E., Shiran, G., Zabari, N., Gordon, O., Panet, P., Weissbuch, S., Kulikov, V., Bitterman, Y., Melumian, Z., Bibi, O.: LTX-Video: Realtime video latent diffusion. arXiv preprint arXiv:2501.00103 (2024)
10. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al.: Lora: Low-rank adaptation of large language models. *Iclr* **1**(2), 3 (2022)
11. Huang, H., Ma, G., Duan, N., Chen, X., Wan, C., Ming, R., Wang, T., Wang, B., Lu, Z., Li, A., Zeng, X., Zhang, X., Yu, G., Yin, Y., Wu, Q., Sun, W., An, K., Han, X., Sun, D., Ji, W., Huang, B., Li, B., Wu, C., Huang, G., Xiong, H., He, J., Wu, J., Yuan, J., Wu, J., Liu, J., Guo, J., Tan, K., Chen, L., Chen, Q., Sun, R., Yuan, S., Yin, S., Liu, S., Chen, W., Dai, Y., Luo, Y., Ge, Z., Guan, Z., Song, X., Zhou, Y., Jiao, B., Chen, J., Li, J., Zhou, S., Zhang, X., Xiu, Y., Zhu, Y., Shum, H.Y., Jiang, D.: Step-video-ti2v technical report: A state-of-the-art text-driven image-to-video generation model (2025), <https://arxiv.org/abs/2503.11251>
12. Huang, X., Li, Z., He, G., Zhou, M., Shechtman, E.: Self forcing: Bridging the train-test gap in autoregressive video diffusion. In: *NeurIPS* (2025)
13. Huang, Z., He, Y., Yu, J., Zhang, F., Si, C., Jiang, Y., Zhang, Y., Wu, T., Jin, Q., Chanpaisit, N., Wang, Y., Chen, X., Wang, L., Lin, D., Qiao, Y., Liu, Z.: VBench: Comprehensive benchmark suite for video generative models. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024)
14. Huang, Z., Zhang, F., Xu, X., He, Y., Yu, J., Dong, Z., Ma, Q., Chanpaisit, N., Si, C., Jiang, Y., et al.: Vbench++: Comprehensive and versatile benchmark suite for video generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2025)
15. Jin, Y., Sun, Z., Li, N., Xu, K., Jiang, H., Zhuang, N., Huang, Q., Song, Y., MU, Y., Lin, Z.: Pyramidal flow matching for efficient video generative modeling. In: *ICLR* (2025)

16. Kim, D., Lai, C.H., Liao, W.H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., Ermon, S.: Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In: ICLR (2024)
17. Kong, W., Tian, Q., Zhang, Z., Min, R., Dai, Z., Zhou, J., Xiong, J., Li, X., Wu, B., Zhang, J., et al.: Hunyuanvideo: A systematic framework for large video generative models. arXiv preprint arXiv:2412.03603 (2024)
18. Lee, S., Xu, Y., Geffner, T., Fanti, G., Kreis, K., Vahdat, A., Nie, W.: Truncated consistency models. arXiv preprint arXiv:2410.14895 (2024)
19. Lin, S., Xia, X., Ren, Y., Yang, C., Xiao, X., Jiang, L.: Diffusion adversarial post-training for one-step video generation. In: ICML (2025)
20. Lin, S., Yang, C., He, H., Jiang, J., Ren, Y., Xia, X., Zhao, Y., Xiao, X., Jiang, L.: Autoregressive adversarial post-training for real-time interactive video generation. In: NeurIPS (2025)
21. Lipman, Y., Chen, R.T., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling. arXiv preprint arXiv:2210.02747 (2022)
22. Liu, X., Gong, C., Liu, Q.: Flow straight and fast: Learning to generate and transfer data with rectified flow. arXiv preprint arXiv:2209.03003 (2022)
23. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
24. Lu, C., Song, Y.: Simplifying, stabilizing and scaling continuous-time consistency models. arXiv preprint arXiv:2410.11081 (2024)
25. Lu, K., Lab, T.M.: On-policy distillation. Thinking Machines Lab: Connectionism (2025). <https://doi.org/10.64434/tml.20251026>, <https://thinkingmachines.ai/blog/on-policy-distillation>
26. Luo, T., Yuan, H., Liu, Z.: Soflow: Solution flow models for one-step generative modeling. arXiv preprint arXiv:2512.15657 (2025)
27. Millon, E.: Krea realtime 14b: Real-time video generation (2025), <https://github.com/krea-ai/realtime-video>
28. Nie, W., Berner, J., Ma, N., Liu, C., Xie, S., Vahdat, A.: Transition matching distillation for fast video generation. arXiv preprint arXiv:2601.09881 (2026)
29. von Platen, P., Patil, S., Lozhkov, A., Cuenca, P., Lambert, N., Rasul, K., Davaadorj, M., Nair, D., Paul, S., Berman, W., Xu, Y., Liu, S., Wolf, T.: Diffusers: State-of-the-art diffusion models. <https://github.com/huggingface/diffusers> (2022)
30. Ren, Y., Xia, X., Lu, Y., Zhang, J., Wu, J., Xie, P., Wang, X., Xiao, X.: HyperSD: Trajectory segmented consistency model for efficient image synthesis. NeurIPS (2024)
31. Sabour, A., Fidler, S., Kreis, K.: Align your flow: Scaling continuous-time flow map distillation. arXiv preprint arXiv:2506.14603 (2025)
32. Song, K., Chen, B., Simchowitz, M., Du, Y., Tedrake, R., Sitzmann, V.: History-guided video diffusion. In: ICML (2024)
33. Song, Y., Dhariwal, P.: Improved techniques for training consistency models. In: ICLR (2024)
34. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models (2023)
35. Team, F.: Causalwan2.2-i2v-a14b-preview-diffusers. <https://huggingface.co/FastVideo/CausalWan2.2-I2V-A14B-Preview-Diffusers> (2025)
36. Teng, H., Jia, H., Sun, L., Li, L., Li, M., Tang, M., Han, S., Zhang, T., Zhang, W., Luo, W., et al.: MAGI-1: Autoregressive video generation at scale. arXiv preprint arXiv:2505.13211 (2025)

37. Wan, T., Wang, A., Ai, B., Wen, B., Mao, C., Xie, C.W., Chen, D., Yu, F., Zhao, H., Yang, J., et al.: Wan: Open and advanced large-scale video generative models. arXiv preprint arXiv:2503.20314 (2025)
38. Wang, F.Y., Huang, Z., Bergman, A., Shen, D., Gao, P., Lingelbach, M., Sun, K., Bian, W., Song, G., Liu, Y., et al.: Phased consistency models. *Advances in neural information processing systems* **37**, 83951–84009 (2024)
39. Wang, Z., Zhang, Y., Yue, X., Yue, X., Li, Y., Ouyang, W., Bai, L.: Transition models: Rethinking the generative learning objective. arXiv preprint arXiv:2509.04394 (2025)
40. Yang, S., Huang, W., Chu, R., Xiao, Y., Zhao, Y., Wang, X., Li, M., Xie, E., Chen, Y., Lu, Y., et al.: Longlive: Real-time interactive long video generation. arXiv preprint arXiv:2509.22622 (2025)
41. Yang, Z., Teng, J., Zheng, W., Ding, M., Huang, S., Xu, J., Yang, Y., Hong, W., Zhang, X., Feng, G., et al.: Cogvideox: Text-to-video diffusion models with an expert transformer. arXiv preprint arXiv:2408.06072 (2024)
42. Ye, T., Dong, L., Wu, X., Huang, S., Wei, F.: On-policy context distillation for language models. arXiv preprint arXiv:2602.12275 (2026)
43. Yin, T., Gharbi, M., Park, T., Zhang, R., Shechtman, E., Durand, F., Freeman, B.: Improved distribution matching distillation for fast image synthesis. *NeurIPS* (2024)
44. Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W.T., Park, T.: One-step diffusion with distribution matching distillation. In: *CVPR* (2024)
45. Yin, T., Zhang, Q., Zhang, R., Freeman, W.T., Durand, F., Shechtman, E., Huang, X.: From slow bidirectional to fast autoregressive video diffusion models. In: *CVPR* (2025)
46. Zhang, L., Cai, S., Li, M., Wetzstein, G., Agrawala, M.: Frame context packing and drift prevention in next-frame-prediction video diffusion models. In: *NeurIPS* (2025)
47. Zhao, S., Xie, Z., Liu, M., Huang, J., Pang, G., Chen, F., Grover, A.: Self-distilled reasoner: On-policy self-distillation for large language models. arXiv preprint arXiv:2601.18734 (2026)
48. Zheng, K., Wang, Y., Ma, Q., Chen, H., Zhang, J., Balaji, Y., Chen, J., Liu, M.Y., Zhu, J., Zhang, Q.: Large scale diffusion distillation via score-regularized continuous-time consistency. arXiv preprint arXiv:2510.08431 (2025)