Self Forcing: Bridging the Train-Test Gap in Autoregressive Video Diffusion

Xun Huang[1] **Zhengqi Li**[1] **Guande He**[2] **Mingyuan Zhou**[2] **Eli Shechtman**[1]
[1]Adobe Research [2]The University of Texas at Austin

https://self-forcing.github.io/

Abstract

We introduce Self Forcing, a novel training paradigm for autoregressive video diffusion models. It addresses the longstanding issue of exposure bias, where models trained on ground-truth context must generate sequences conditioned on their own imperfect outputs during inference. Unlike prior methods that denoise future frames based on ground-truth context frames, Self Forcing conditions each frame's generation on previously self-generated outputs by performing autoregressive rollout with key-value (KV) caching during training. This strategy enables supervision through a holistic loss at the video level that directly evaluates the quality of the entire generated sequence, rather than relying solely on traditional frame-wise objectives. To ensure training efficiency, we employ a few-step diffusion model along with a stochastic gradient truncation strategy, effectively balancing computational cost and performance. We further introduce a rolling KV cache mechanism that enables efficient autoregressive video extrapolation. Extensive experiments demonstrate that our approach achieves real-time streaming video generation with sub-second latency on a single GPU, while matching or even surpassing the generation quality of significantly slower and non-causal diffusion models.

1 Introduction

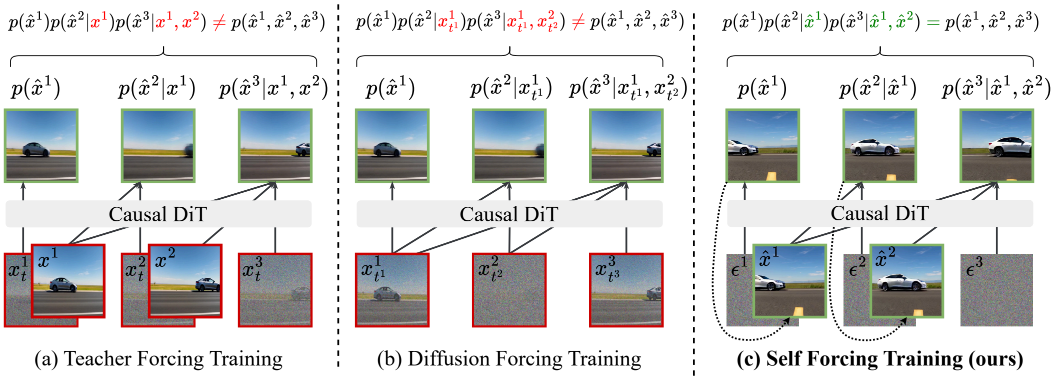
Recent years have witnessed tremendous progress in video synthesis, with state-of-the-art systems now capable of generating remarkably realistic content with complex temporal dynamics [6]. However, these results are typically achieved with diffusion transformers (DiT) [62, 83] that denoise all frames simultaneously using bidirectional attention. This design allows the future to affect the past and requires generating the entire video at once, fundamentally limiting their applicability to real-time streaming applications where future information is unknown when generating the current frame.

In contrast, autoregressive (AR) models [17, 27, 38, 94, 104] generate videos sequentially, a paradigm that naturally aligns with the causal structure of temporal media. This approach not only significantly reduces the viewing latency of generated videos but also unlocks numerous applications, including real-time interactive content creation [9, 46], game simulation [11, 61, 78, 102], and robotics learning [42, 96, 101]. However, AR models often struggle to match the visual fidelity achieved by state-of-the-art video diffusion models due to their reliance on lossy vector quantization techniques [79].

To combine the best of both worlds, two recent techniques have emerged to equip video diffusion models with AR generation capabilities: Teacher Forcing (TF) [16, 28, 33, 106] and Diffusion Forcing (DF) [8, 10, 20, 69, 73, 100]. Teacher Forcing, a well-established paradigm in sequence modeling, trains the model to predict the next token conditioned on ground-truth tokens. When applied to video diffusion, TF involves denoising each frame using clean, ground-truth context frames (Figure 1 (a)),

Correspondence: xuhuang@adobe.com

Preprint. Under review.


$p(\hat{x}^1)p(\hat{x}^2|x^1)p(\hat{x}^3|x^1, x^2) \neq p(\hat{x}^1, \hat{x}^2, \hat{x}^3)$
$p(\hat{x}^1)$
$p(\hat{x}^2|x^1)$
$p(\hat{x}^3|x^1, x^2)$
Causal DiT
x_t^1
x^1
x_t^2
x^2
x_t^3
$p(\hat{x}^1)p(\hat{x}^2|x^1_{t^1})p(\hat{x}^3|x^1_{t^1}, x^2_{t^2}) \neq p(\hat{x}^1, \hat{x}^2, \hat{x}^3)$
$p(\hat{x}^1)$
$p(\hat{x}^2|x^1_{t^1})$
$p(\hat{x}^3|x^1_{t^1}, x^2_{t^2})$
Causal DiT
$x^1_{t^1}$
$x^2_{t^2}$
$x^3_{t^3}$
$p(\hat{x}^1)p(\hat{x}^2|\hat{x}^1)p(\hat{x}^3|\hat{x}^1, \hat{x}^2) = p(\hat{x}^1, \hat{x}^2, \hat{x}^3)$
$p(\hat{x}^1)$
$p(\hat{x}^2|\hat{x}^1)$
$p(\hat{x}^3|\hat{x}^1, \hat{x}^2)$
Causal DiT
ϵ^1
$\hat{x}^1$
ϵ^2
$\hat{x}^2$
ϵ^3

(a) Teacher Forcing Training

(b) Diffusion Forcing Training

(c) Self Forcing Training (ours)

Figure 1: **Training paradigms for AR video diffusion models.** (a) In Teacher Forcing, the model is trained to denoise each frame conditioned on the preceding clean, ground-truth context frames. (b) In Diffusion Forcing, the model is trained to denoise each frame conditioned on the preceding context frames with varying noise levels. Both (a) and (b) generate outputs that do not belong to the distribution the model generates during inference. (c) Our Self Forcing approach performs autoregressive self-rollout *during training*, denoising the next frame based on previous context frames generated by itself. A distribution-matching loss (*e.g.*, SiD, DMD, GAN) is computed on the final output video to align the distribution of generated videos with that of real videos. Our training paradigm closely mirrors the inference process, thereby bridging the train-test distribution gap.

a strategy commonly referred to as next-frame prediction. In contrast, Diffusion Forcing trains the model on videos with noise levels independently sampled for each frame, denoising each frame based on noisy context frames (Figure 1 (b)). This ensures the autoregressive inference scenario, where context frames are clean and the current frame is noisy, is covered by the training distribution.

However, models trained with TF or DF often suffer from error accumulation during autoregressive generation, leading to degraded video quality over time [84, 100, 105]. This issue is more broadly known as *exposure bias* [60, 71], where a model is trained exclusively on ground-truth context but must rely on its own imperfect predictions at inference time, resulting in a distributional mismatch that compounds errors as generation progresses. While some approaches attempt to mitigate this issue in video diffusion models by incorporating noisy context frames during inference [8, 11, 105], such design sacrifices temporal consistency, complicates the KV-cache design, increases generation latency, and does not fundamentally resolve the exposure bias problem.

In this work, we propose *Self Forcing* (SF), a novel algorithm addressing exposure bias in autoregressive video generation. Inspired by early RNN-era sequence modeling techniques [40, 65, 103], our approach bridges the train-test distribution gap by explicitly unrolling autoregressive generation during training, generating each frame conditioned on previously self-generated frames rather than ground-truth ones. This enables supervision with holistic distribution-matching losses [18, 98, 99] applied to complete generated video sequences. By forcing the model to encounter and learn from its own prediction errors, Self Forcing effectively mitigates exposure bias and reduces error accumulation.

While Self Forcing may seem computationally prohibitive due to its sequential nature preventing parallel training, we demonstrate that it can be efficiently implemented as an algorithm in the post-training stage where the model does not require a large number of gradient updates to converge. By employing a few-step diffusion backbone and a carefully designed gradient truncation strategy, Self Forcing is surprisingly more efficient than alternative parallel strategies, achieving superior performance within the same wall-clock training time. Additionally, we introduce a rolling KV cache mechanism that enhances the efficiency of video extrapolation.

Extensive experiments demonstrate that our model enables real-time video generation at 17 FPS with sub-second latency on a single H100 GPU, while achieving competitive or superior generation quality compared to recent slow bidirectional and autoregressive video diffusion models. These advances open the door to genuinely interactive video generation use cases—live streaming, gaming, and world simulation—where latency budgets are measured in milliseconds rather than minutes.

2 Related Work

GANs for Video Generation. Early video generation approaches relied primarily on generative adversarial networks (GANs) [18], either using convolutional networks to generate entire videos in parallel [5, 68, 82] or employing recurrent architectures to produce frames sequentially [14, 44, 49, 77, 81]. Recently, GANs have also been applied to distill video diffusion models [47, 56, 91, 108]. Since the generator in GANs follows the same process during training and inference, it inherently avoids exposure bias. Our work draws inspiration from this fundamental GAN principle by directly optimizing the alignment between the generator's output distribution and the target distribution.

Autoregressive/Diffusion Models for Video Generation. Modern video generation models have largely shifted toward diffusion or autoregressive models due to their stronger scaling abilities. Video diffusion models typically adopt bidirectional attention mechanisms to simultaneously denoise all video frames [3, 4, 6, 13, 23–26, 39, 64, 80, 83, 97]. Autoregressive models, in contrast, are trained with next-token prediction objectives and generate spatiotemporal tokens sequentially at inference time [7, 38, 66, 86, 88, 94].

Autoregressive-Diffusion Hybrid Models. Very recently, hybrid models integrating autoregressive and diffusion frameworks have emerged as a promising direction in generative modeling of videos [8, 16, 20, 22, 28, 33, 45, 50, 52, 89, 100, 106, 107] as well as other sequence domains [1, 12, 43, 53, 59, 90, 110]. They typically rely on a long, iterative prediction chain (both temporally autoregressive and spatially iterative denoising), which could lead to significant error accumulation. Our work addresses this issue by training the model conditioned on its own predictions and teaching it to correct its own mistakes.

Rolling Diffusion and Variants. Another line of work [35, 67, 69, 76, 93, 105] trains video diffusion models with a progressive noise schedule, where the noise level gradually increases from earlier to later frames. While these methods support sequential long video generation with less accumulated errors and are sometimes also referred to as autoregressive, they do not strictly follow the autoregressive chain rule decomposition. Consequently, they would exhibit significant latency in interactive applications, as future frames are partially pre-generated before the current frame is presented to the user. This premature commitment restricts the impact of real-time user-injected controls, resulting in limited responsiveness in immediately subsequent frames.

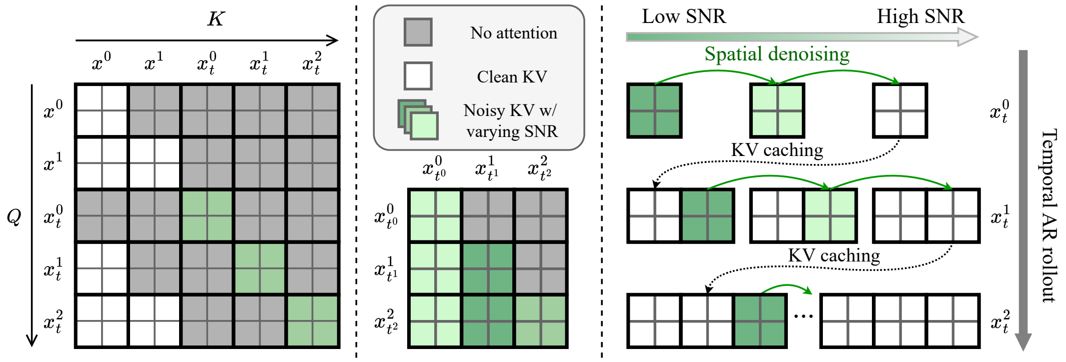
CausVid. Our work is most closely related to CausVid [100], which trains few-step autoregressive diffusion models using the DF scheme and distribution matching distillation (DMD). However, CausVid suffers from a critical flaw that its training outputs (generated via DF) do not come from the distribution the model produces at inference time, therefore the DMD loss is matching the wrong distribution. We pinpoint this issue and propose a solution that matches the true model distribution.

3 Self Forcing: Briding Train-Test Gap via Holistic Post-Training

We first provide a formal definition of autoregressive video diffusion models and describe standard training approaches in Section 3.1. In Section 3.2, we introduce the main part of our Self Forcing training algorithm and describe how it can be efficiently implemented with a few-step diffusion model. In Section 3.3, we describe various choices of holistic, video-level distribution-matching training objectives. Finally, we introduce a rolling key-value cache mechanism that enables efficient generation of arbitrarily long videos in Section 3.4.

3.1 Preliminaries: Autoregressive Video Diffusion Models

Autoregressive video diffusion model is a hybrid generative model that combines autoregressive chain-rule decomposition with denoising diffusion models for video generation. Specifically, given a sequence of N video frames $x^{1:N} = (x^1, x^2, \dots, x^N)$, it factorizes the joint distribution into product of conditionals using the chain rule $p(x^{1:N}) = \prod_{i=1}^{N} p(x^i|x^{<i})$. Each conditional distribution $p(x^i|x^{<i})$ is then modeled using a diffusion process, where a frame is generated by progressively denoising an initial Gaussian noise conditioned on previously generated frames. This formulation combines the strengths of both autoregressive models and diffusion models for capturing sequential dependencies while enabling high-quality generation of continuous-valued visual signal. In practice,



(a) Teacher Forcing (b) Diffusion Forcing (c) **Self Forcing**/AR Inference

Figure 2: **Attention mask configurations.** Both Teacher Forcing (a) and Diffusion Forcing (b) train the model on the entire video in parallel, enforcing causal dependencies with custom attention masks. In contrast, our Self-Forcing Training (c) mirrors the autoregressive (AR) inference process with KV caching and does not rely on special attention masks. For illustration purposes, we show a scenario where the video contains 3 frames, and each frame consists of 2 tokens.

we can also choose to generate one chunk of frames rather than a single frame at a time [69, 100]. For simplicity of notation, however, we continue to denote each chunk as a frame in this section.

Most existing autoregressive video diffusion models are trained using frame-wise denoising loss within the paradigm of Teacher Forcing (TF) or Diffusion Forcing (DF). Specifically, each frame x^i is corrupted by the forward process $q_{t^i|0}(x^i_{t^i}|x^i_0)$ such that $x^i_{t^i} = \Psi(x^i, \epsilon^i, t^i) = \alpha_{t^i} x^i + \sigma_{t^i}\epsilon^i$, where $\alpha_{t^i}, \sigma_{t^i}$ are pre-defined noise schedule within a finite time horizon $t^i \in [0, 1000]$ and $\epsilon^i \sim \mathcal{N}(0, I)$ is Gaussian noise. In TF, the timesteps t^i are usually shared across all frames, whereas in DF, they are sampled independently for each frame. A generative model is learned through the time-reversal of the forward process, where each denoising step can be achieved by predicting the noise ϵ^i added to each frame with a neural network $\hat{\epsilon}^i_\theta := G_\theta(x^i_{t^i}, t^i, c)$ conditioned on the context c. The context consists of clean ground-truth frames $x^{<i}$ in TF or noisy context frames $x^{j<i}_{t^j}$ in DF. The model is trained to minimize the frame-wise mean squared error (MSE) between the predicted noise and the true added noise: $\mathcal{L}^{\text{DM}}_\theta = \mathbb{E}_{x^i, t^i, \epsilon^i}\left[w_{t^i}\|\hat{\epsilon}^i_\theta - \epsilon^i\|^2_2\right]$, where w_{t^i} is a weighting function.

We focus on the transformer-based architecture [62] of diffusion models with text conditioning (omitted from equations for clarity) operating in a compressed latent space encoded by a causal 3D variational autoencoder (VAE) [37]. The autoregressive chain-rule decomposition is implemented via causal attention. Figures 2 (a) and (b) illustrate the attention mask configurations of Teacher Forcing and Diffusion Forcing approaches. For Teacher Forcing, we describe an efficient variant that processes all frames in parallel using block sparse attention masks, rather than denoising one frame at each training iteration [33]. Such design has been used in MAR-based [43] autoregressive video generation [111] and concurrently in other autoregressive video diffusion models [106, 107].

3.2 Autoregressive Diffusion Post-Training via Self-Rollout

The core idea of Self Forcing is to generate videos through autoregressive self-rollout during training following the inference-time recipe. Specifically, we sample a batch of videos $\{x^{1:N}_\theta\} \sim p_\theta(x^{1:N}) = \prod_{i=1}^N p_\theta(x^i|x^{<i})$ where each frame x^i is generated by performing iterative denoising conditioned on self-generated outputs, including both clean context frames in the past and noisy frames at the current time step. Unlike most previous autoregressive models that only utilize KV caching during inference, our Self Forcing method innovatively employs KV caching during training, as shown in Figure 2 (c).

Nevertheless, implementing Self Forcing with standard many-step diffusion models would be computationally prohibitive, as it requires unrolling and backpropagation through long denoising chains. Therefore, we choose to use a few-step diffusion model G_θ to approximate each conditional distribution $p_\theta(x^i|x^{<i})$ in the autoregressive factorization. Consider $\{t_0 = 0, t_1, \ldots, t_T = 1000\}$ a subsequence of timesteps $[0, ..., 1000]$, at each denoising step t_j and frame index i, the model denoises an intermediate noisy frame $x^i_{t_j}$ conditioned on previous clean frames $x^{<i}$. It then injects Gaussian noise with a lower noise level into the denoised frame through the forward process Ψ to

Algorithm 1 Self Forcing Training

```
Require: Denoise timesteps $\{t_1, \ldots, t_T\}$
Require: Number of video frames $N$
Require: AR diffusion model $G_\theta$ (returns KV embeddings via $G_\theta^{\text{KV}}$)
loop
    Initialize model output $\mathbf{X}_\theta \leftarrow []$
    Initialize KV cache $\mathbf{KV} \leftarrow []$
    Sample $s \sim \text{Uniform}(1, 2, \ldots, T)$
    for $i = 1, \ldots, N$ do
        Initialize $x_{t_T}^i \sim \mathcal{N}(0, I)$
        for $j = T, \ldots, s$ do
            if $j = s$ then
                Enable gradient computation
                Set $\hat{x}_0^i \leftarrow G_\theta(x_{t_j}^i; t_j, \mathbf{KV})$
                $\mathbf{X}_\theta$.append($\hat{x}_0^i$)
                Disable gradient computation
                Cache $\mathbf{kv}^i \leftarrow G_\theta^{\text{KV}}(\hat{x}_0^i; 0, \mathbf{KV})$
                $\mathbf{KV}$.append($\mathbf{kv}^i$)
            else
                Disable gradient computation
                Set $\hat{x}_0^i \leftarrow G_\theta(x_{t_j}^i; t_j, \mathbf{KV})$
                Sample $\epsilon \sim \mathcal{N}(0, I)$
                Set $x_{t_{j-1}}^i \leftarrow \Psi(\hat{x}_0^i, \epsilon, t_{j-1})$
            end if
        end for
    end for
    Update $\theta$ via distribution matching loss
end loop
```

Algorithm 2 Autoregressive Diffusion Inference with Rolling KV Cache

```
Require: KV cache of size $L$ frames
Require: Denoise timesteps $\{t_1, \ldots, t_T\}$
Require: Number of generated frames $M$
Require: AR diffusion model $G_\theta$ (returns KV embeddings via $G_\theta^{\text{KV}}$)
Initialize model output $\mathbf{X}_\theta \leftarrow []$
Initialize KV cache $\mathbf{KV} \leftarrow []$
for $i = 1, \ldots, M$ do
    Initialize $x_{t_T}^i \sim \mathcal{N}(0, I)$
    for $j = T, \ldots, 1$ do
        Set $\hat{x}_0^i \leftarrow G_\theta(x_{t_j}^i; t_j, \mathbf{KV})$
        if $j = 1$ then
            $\mathbf{X}_\theta$.append($\hat{x}_0^i$)
            Cache $\mathbf{kv}^i \leftarrow G_\theta^{\text{KV}}(\hat{x}_0^i; 0, \mathbf{KV})$
            if $|\mathbf{KV}| = L$ then
                $\mathbf{KV}$.pop(0)        ▷ Cache eviction
            end if
            $\mathbf{KV}$.append($\mathbf{kv}^i$)
        else
            Sample $\epsilon \sim \mathcal{N}(0, I)$
            Set $x_{t_{j-1}}^i \leftarrow \Psi(\hat{x}_0^i, \epsilon, t_{j-1})$
        end if
    end for
end for
return $\mathbf{X}_\theta$
```

obtain the noisy frame $x_{t_{j-1}}^i$ as the input to the next denoising step, following the standard practice in few-step diffusion models [74, 98]. The model distribution $p_\theta(x^i | x^{<i})$ is implicitly defined as $f_{\theta,t_1} \circ f_{\theta,t_2} \circ \ldots \circ f_{\theta,t_T}(x_{t_T}^i)$, where $f_{\theta,t_j}(x_{t_j}^i) = \Psi(G_\theta(x_{t_j}^i, t_j, x^{<i}), \epsilon_{t_{j-1}}, t_{j-1})$, and $x_{t_T}^i \sim \mathcal{N}(0, I)$.

Even with few-step models, however, naively backpropagating through the entire autoregressive diffusion process would still lead to excessive memory consumption. To address this challenge, we propose a gradient truncation strategy that limits the backpropagation to only the final denoising step of each frame. Moreover, instead of always using T denoising steps (as in inference time), we randomly sample a denoising step s from $[1, T]$ for each sample sequence at each training iteration, and use the denoised output of the s-th step as the final output. This stochastic sampling approach ensures all intermediate denoising steps receive supervision signals. We additionally detach the gradients of the previous frames from the current frame during training by restricting gradient flow into KV cache embeddings. For a complete description of the training process, see Algorithm 1.

3.3 Holistic Distribution Matching Loss

Autoregressive self-rollout generates samples directly from the inference-time model distribution, enabling us to apply holistic, video-level losses that align the distribution of generated videos $p_\theta(x^{1:N})$ with that of real videos $p_{\text{data}}(x^{1:N})$. To leverage pre-trained diffusion models and enhance training stability [32], we inject noise to both distributions and match $p_{\theta,t}(x_t^{1:N})$ and $p_{\text{data},t}(x_t^{1:N})$, where each represents the respective distribution after applying the forward diffusion process: $p_{\cdot,t}(x_t^{1:N}) = \int q_{t|0}(x_t^{1:N} | x^{1:N}) p_\cdot(x^{1:N}) \mathrm{d}x^{1:N}$. Our framework is generally applicable to various divergence measures and distribution matching frameworks, and we consider three approaches in this paper:

- **Distribution Matching Distillation (DMD)** [98, 99]: This approach minimizes the reverse Kullback-Leibler divergence $\mathbb{E}_t[D_{\text{KL}}(p_{\theta,t} \| p_{\text{data},t})]$ by leveraging the score difference between distributions to guide gradient updates.

- **Score Identity Distillation (SiD)** [112, 113]: This method performs distribution matching via Fisher divergence $\mathbb{E}_{t,p_{\theta,t}}[||\nabla \log p_{\theta,t} \; - \nabla \log p_{\text{data},t}||^2]$.
- **Generative Adversarial Networks (GANs)** [18]: It approximates the Jensen-Shannon divergence through a minimax game between the generator (our autoregressive diffusion model) and a discriminator that distinguishes between real and generated videos.

Importantly, our training objective matches the *holistic* distribution of the entire video sequence to the data distribution $D(p_{\text{data}}(x^{1:N})||p_\theta(x^{1:N}))$. In contrast, TF/DF can be understood as performing *frame-wise* distribution matching: $\mathbb{E}_{\{x^{<i}\}\sim p_{\text{data}}} D_{KL}(p_{\text{data}}(x^i|x^{<i})||p_\theta(x^i|x^{<i}))$[1], where DF additionally samples context frames from a noise-corrupted data distribution $\{x^{<i}\} \sim \tilde{p}_{\text{data}}$. Our formulation fundamentally transforms the training dynamics—context frames $\{x^{<i}\}$ are sampled from the model's own distribution p_θ rather than from the data distribution (clean or noisy). This alignment between training and inference distributions effectively addresses exposure bias and forces the model to learn from its own imperfections, thereby developing robustness to error accumulation.

While all three objectives have been used in the context of timestep distillation of diffusion models, our primary motivation differs fundamentally from distillation: we aim to enhance the *quality* of autoregressive video generation by addressing exposure bias via distribution matching, rather than merely accelerating sampling. This distinction makes other popular distillation methods [74] inapplicable to our framework as they only focus on timestep reduction without directly aligning the generator output distribution. Although CausVid [100] similarly employs DMD to match the distribution of generated videos, the distribution it optimizes during training (using Diffusion Forcing outputs) deviates from the actual inference-time distribution, significantly undermining its effectiveness.

3.4 Long Video Generation with Rolling KV Cache

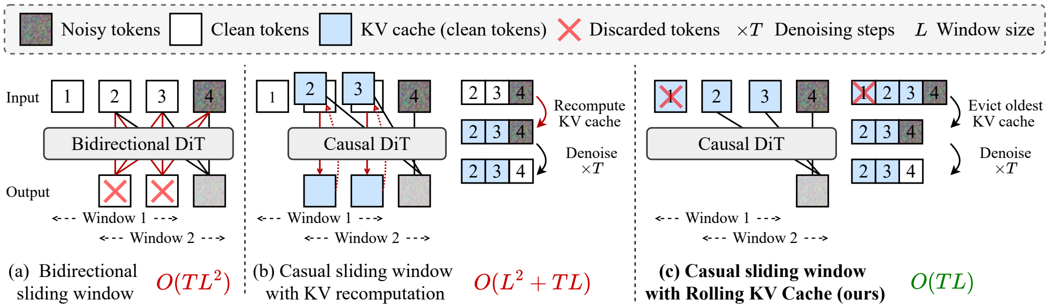
A key advantage of autoregressive models over standard video diffusion models is their extrapolative ability, in principle allowing the generation of infinitely long videos via sliding-window inference. While bidirectional attention models trained with Diffusion Forcing [10, 73] can also generate videos autoregressively, they do not support KV caching, requiring complete recomputation of attention matrices for each new frame. This leads to excessive computational complexity of $O(TL^2)$ (where T represents the number of denoising steps and L is thewindow size), as shown in Figure 3 (a).

Models with causal attention, on the other hand, can leverage KV caching to improve efficiency. However, existing implementations [69, 100] require recomputing KV cache for overlapping frames between consecutive sliding windows, as illustrated in Figure 3 (b). This leads to $O(L^2 + TL)$ complexity when employing dense sliding windows. As a result, prior implementations adopt larger strides with minimal overlap to reduce computational costs, which compromises temporal consistency since frame at the beginning of each window relies on a significantly reduced historical context.

Inspired by research in large language models [92], we propose a rolling KV cache mechanism for autoregressive diffusion models that allows infinitely long video generation without any need of recomputing the KV cache. As illustrated in Figure 3 (c), we maintain a fixed-size KV cache that stores the KV embeddings of tokens in the most recent L frames. When generating a new frame, we first check if the KV cache is full. If it is, we remove the oldest KV cache entry before adding the new one. This approach enables endless frame generation with a time complexity of $O(TL)$, while still maintaining a sufficient context length when generating each new frame. Algorithm 2 provides a detailed description of our autoregressive long video generation algorithm with rolling KV cache.

However, naive implementation of this mechanism leads to severe flickering artifacts due to distribution mismatch. Specifically, the first latent frame has different statistical properties than other frames: it only encodes the first image without performing temporal compression. The model, having always seen the first frame as the image latent during training, fails to generalize when the image latent is no longer visible in the rolling KV cache scenario. While we experimented with strategies similar to StreamingLLM [92], keeping the first frame fixed while rolling KV cache of other frames, these proved ineffective for video generation. Our solution is straightforward but effective: during training, we restrict the attention window so the model cannot attend to the first chunk when denoising the final chunk, thereby simulating the conditions encountered during long video generation.

[1] With a specific weighting per noise level [36, 75], denoising loss approximates the maximum likelihood objective, equivalent to minimizing the KL divergence between per-frame data and model distributions.



(a) Bidirectional sliding window

(b) Casual sliding window with KV recomputation

(c) Casual sliding window with Rolling KV Cache (ours)

Figure 3: **Efficiency comparisons for video extrapolation.** When performing video extrapolation through sliding window inference, (a) bidirectional diffusion models trained with TF/DF [10, 73] do not support KV cache. (b) Prior causal diffusion models [69, 100] require re-computing KV when shifting the window. (c) Our method does not recompute KV and enables more efficient extrapolation.

4 Experiments

Implementation. We implement Self Forcing with Wan2.1-T2V-1.3B [83], a Flow Matching [48] based model that generates 5s videos at 16 FPS with a resolution of 832 × 480. Following CausVid's initialization protocol [100], we first finetune the base model with causal attention masking on 16k ODE solution pairs sampled from the base model. For both ODE initialization and Self Forcing training, we sample text prompts from a filtered and LLM-extended version of VidProM [85]. We use 4-step diffusion and implement both frame-wise and chunk-wise autoregressive variants, with the latter generating a chunk of 3 latent frames at a time. We adopt the R3GAN [29] objective, which consists of relativistic pairing GAN loss [34] with R1 + R2 regularization [58]. We use the 14B base model to generate 70k videos as the dataset for training GANs [70] and fine-tuning many-step TF/DF AR diffusion baselines. Notably, DMD/SiD implementations of our algorithm remain data-free, capable of converting a pre-trained video diffusion model into an autoregressive model without any video training data. Additional implementation details are provided in Appendix A.

Evaluation metrics. We adopt VBench [31] and user preference study to evaluate both visual quality and semantic alignment. We also rigorously evaluate the efficiency of our method for real-time applications. While some recent works claim "real-time" video generation abilities [24, 109] based solely on *throughput*, we argue that true real-time performance requires both sufficient throughput (exceeding video playback rate) and lower *latency* than the perceptual threshold which could be application-dependent [41]. We therefore evaluate both throughput and first-frame latency to provide a comprehensive assessment of real-time capabilities, with all speed tests conducted on a single NVIDIA H100 GPU.

Comparison with existing baselines. We compare our model with relevant open-source video generation models of similar scale. Our comparisons include two diffusion models: Wan2.1-1.3B [83] (our initialization weights) and LTX-Video [24] (known for efficiency). We also compare with several autoregressive models including Pyramid Flow [33], NOVA [13], SkyReels-V2 [10], MAGI-1 [69], and CausVid [100] (also initialized from Wan-1.3B).

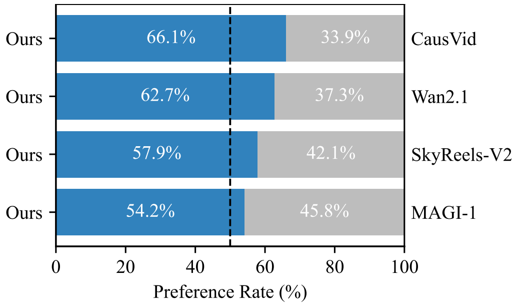


Figure 4: **User preference study.** Self Forcing outperforms all baselines in human preference.

As shown in Table 1, our chunk-wise autoregressive model achieves the highest VBench scores across all compared models while simultaneously delivering real-time throughput (17.0 FPS) with sub-second latency, low enough for certain real-time applications such as live video streaming [2]. Figure 4 shows the user study results comparing our chunk-wise Self Forcing model against several important baselines. Our approach is consistently preferred over all alternatives, including the many-step diffusion model Wan2.1 that our model is initialized from. Our frame-wise variant maintains strong generation quality while providing the lowest latency (0.45s), making it particularly suitable for latency-sensitive real-time applications. Results here are obtained using the DMD loss objective;

Table 1: **Comparison with relevant baselines.** We compare Self Forcing with representative open-source video generation models of similar parameter sizes and resolutions.

Model	#Params	Resolution	Throughput (FPS) ↑	Latency (s) ↓	Evaluation scores ↑		
					Total Score	Quality Score	Semantic Score
Diffusion models							
LTX-Video [24]	1.9B	768×512	8.98	13.5	80.00	82.30	70.79
Wan2.1 [83]	1.3B	832×480	0.78	103	84.26	**85.30**	80.09
Chunk-wise autoregressive models							
SkyReels-V2 [10]	1.3B	960×540	0.49	112	82.67	84.70	74.53
MAGI-1 [69]	4.5B	832×480	0.19	282	79.18	82.04	67.74
CausVid [100]*	1.3B	832×480	**17.0**	0.69	81.20	84.05	69.80
Self Forcing (Ours, chunk-wise)	1.3B	832×480	**17.0**	0.69	**84.31**	85.07	**81.28**
Autoregressive models†							
NOVA [13]	0.6B	768×480	0.88	4.1	80.12	80.39	79.05
Pyramid Flow [33]	2B	640×384	6.7	2.5	81.72	84.74	69.62
Self Forcing (Ours, frame-wise)	1.3B	832×480	8.9	**0.45**	84.26	85.25	80.30

* We compare with the official implementation of CausVid that uses the same base model (Wan-1.3B).
† The distinction of AR/non-AR applies to the temporal dimension.

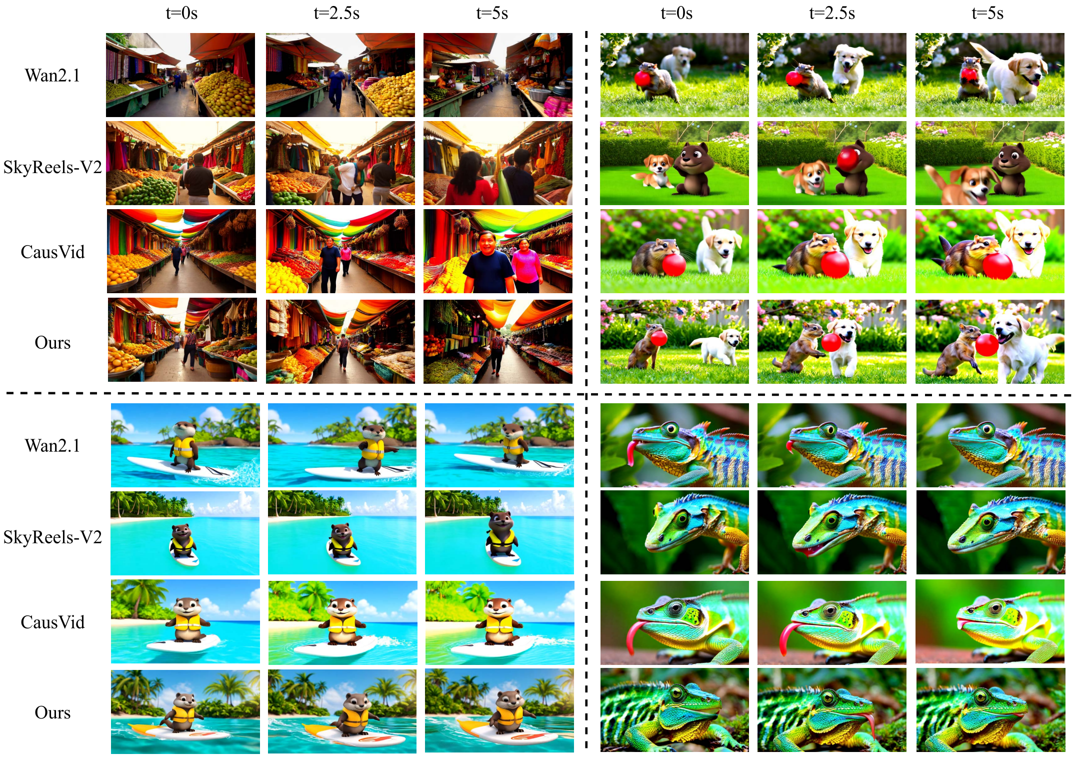


Figure 5: **Qualitative comparisons.** We visualize videos generated by Self Forcing (Ours) against those by Wan2.1 [83], SkyReels-V2 [10], and CausVid [100] at three time steps. All models share the same architecture with 1.3B parameters.

models trained with SiD and GAN objectives achieve comparable performance as detailed in our ablation studies. As shown in Figure 5, CausVid suffers from the error accumulation problem that causes the saturation to increase over time. Our approach obtains slightly better visual quality than Wan2.1/SkyReels-V2, while being around **150x** faster in latency. More example videos are provided in the project website (`https://self-forcing.github.io/`).

Table 2: **Ablation study.** We conduct controlled ablation studies comparing different training paradigms and distribution matching objectives under our training setup across chunk-wise (left) and frame-wise (right) AR models. Self Forcing works well with all different distribution matching objectives and consistently outperforms alternative training approaches.

Chunk-wise AR	Evaluation scores ↑		
	Total Score	Quality Score	Semantic Score
Many (50×2)-step models			
Diffusion Forcing (DF)	82.95	83.66	80.09
Teacher Forcing (TF)	83.58	84.34	80.52
Few (4)-step models			
DF + DMD	82.76	83.49	79.85
TF + DMD	82.32	82.73	80.67
Self Forcing (Ours, DMD)	**84.31**	85.07	**81.28**
Self Forcing (Ours, SiD)	84.07	**85.52**	78.24
Self Forcing (Ours, GAN)	83.88	85.06	79.16

Frame-wise AR	Evaluation scores ↑		
	Total Score	Quality Score	Semantic Score
Many (50×2)-step models			
Diffusion Forcing (DF)	77.24	79.72	67.33
Teacher Forcing (TF)	80.34	81.34	76.34
Few (4)-step models			
DF + DMD	80.56	81.02	78.71
TF + DMD	78.12	79.62	72.11
Self Forcing (Ours, DMD)	**84.26**	**85.25**	**80.30**
Self Forcing (Ours, SiD)	83.54	84.71	78.86
Self Forcing (Ours, GAN)	83.27	84.57	78.08

Ablation Studies. We perform controlled comparisons of Self Forcing with alternative autoregressive diffusion training approaches. We evaluate: (1) AR Diffusion models trained with denoising diffusion loss using either Teacher Forcing or Diffusion Forcing, and (2) few-step AR Diffusion models trained with TF/DF inputs but optimized with distribution matching objectives. The latter configuration with DF and DMD essentially replicates CausVid [100] within our implementation framework, allowing direct comparison under identical training conditions.

Table 2 demonstrates that Self Forcing performs robustly across various distribution matching objectives (DMD, SiD, and GAN), consistently outperforming all baselines. While baseline methods exhibit notable quality degradation when shifting from chunk-wise to frame-wise AR due to error accumulation associated with increased AR unrolling steps, usually manifesting as progressive over-saturation or over-sharpening (similar to CausVid in Appendix B Fig. 5), Self Forcing maintains consistent performance across both setups, highlighting its effectiveness at addressing exposure bias.

Rolling KV cache. We observe that recomputing KV cache when shifting sliding window (Fig. 3 (b)) results in significantly reduced throughput (only 4.6 FPS) when generating 10-second videos. While naive rolling KV cache maintains high throughput, it introduces severe visual artifacts, as illustrated in the examples in Appendix B. By training the model to generate frames without seeing the initial image latent, we effectively mitigate these artifacts while maintaining high throughput (16.1 FPS).

Training efficiency. One might expect Self Forcing training to be computationally prohibitive given its sequential nature that contradicts the parallelizable paradigm of transformers. Surprisingly, our experiments reveal that Self Forcing actually outperforms alternative strategies in training efficiency. As shown in Fig. 6 (left), Self Forcing achieves comparable per-iteration training time to Teacher Forcing and Diffusion Forcing. Furthermore, Fig. 6 (right) demonstrates that Self Forcing achieves superior quality given same wall-clock training budgets compared to both alternative approaches. Each Self Forcing experiment with DMD converges in approximately 1.5 hours on 64 H100 GPUs.

This counter-intuitive result stems from two key factors: First, while Self Forcing performs sequential rollout, it still processes all tokens within each individual frame/chunk in parallel, maintaining high GPU utilization during training. Second, TF and DF require specialized attention masking patterns to enforce causal dependencies, introducing additional computational overhead even with specialized implementations like FlexAttention [15]. On the other hand, Self Forcing always uses full attention during training and can leverage highly optimized attention kernels such as FlashAttention-3 [72].

5 Discussion

In this section, we examine the broader implications of our results, discuss additional perspectives, and outline potential directions for future research.

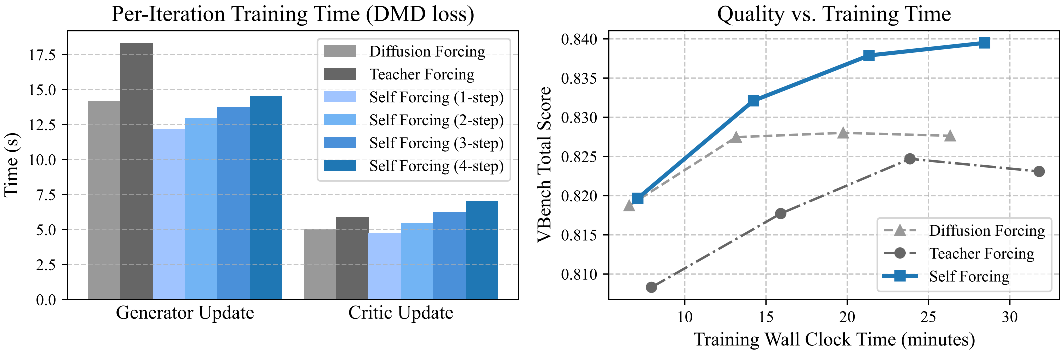


Figure 6: **Training efficiency comparison.** *Left:* Per-iteration time across different chunk-wise, few-step autoregressive video diffusion training algorithms (using DMD as the distribution matching objective). *Right:* Video quality (VBench score) vs. wall clock training time.

Fundamental limitation of the parallelizable training paradigm. Parallelizable training has been pivotal to transformers' success by enabling efficient scaling. However, this parallelism introduces fundamental limitations. Prior research [57] demonstrates that parallel architectures inherently limit expressiveness in sequential state-tracking problems. Our work highlights another critical limitation: parallelizable training paradigms creates misalignment between training and inference distributions, leading to the accumulation of errors over time. We advocate a new paradigm of *parallel pre-training* and *sequential post-training* that combines the best of both worlds. While this paradigm shift is gaining momentum in language modeling through reinforcement learning [21], our work represents the first step towards this direction for the video domain. We believe our framework is general and can be applied to other sequence domains, especially where the data is continuous.

Interplay between AR, Diffusion, and GANs. Autoregressive models, diffusion models, and GANs have traditionally been viewed as distinct paradigms in generative modeling. Our work highlights their complementary nature and demonstrates how they can be effectively integrated. Specifically, autoregressive and diffusion models provide complementary ways to factorize distributions (chain-rule vs. latent-variable), which can be composed in a nested manner. The core idea behind GANs—matching the distribution of an implicit generator to the target distribution by drawing samples from the implicit generator—can be employed to train a generator powered by autoregressive-diffusion factorization.

Limitation and future directions. While our method effectively mitigates error accumulation within the training context length, quality degradation remains observable when generating videos substantially longer than those seen during training. Additionally, our gradient truncation strategies—while necessary for memory efficiency—may limit the model's ability to learn long-range dependencies. Future work could explore both improved extrapolation techniques and inherently recurrent architectures like state-space models [19, 63] that better balance memory efficiency with long-context modeling.

Acknowledgments

We thank Tianwei Yin, Beidi Chen, Kaiwen Zheng, Kai Zhang, and Yi Gu for valuable discussions.

References

[1] Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *ICLR*, 2025.

[2] Abdelhak Bentaleb, May Lim, Mehmet N Akcay, Ali C Begen, Sarra Hammoudi, and Roger Zimmermann. Toward one-second latency: Evolution of live media streaming. *IEEE Communications Surveys & Tutorials*, 2025.

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

[4] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023.

[5] Tim Brooks, Janne Hellsten, Miika Aittala, Ting-Chun Wang, Timo Aila, Jaakko Lehtinen, Ming-Yu Liu, Alexei Efros, and Tero Karras. Generating long videos of dynamic scenes. *NeurIPS*, 2022.

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

[7] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *ICML*, 2024.

[8] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *NeurIPS*, 2024.

[9] Feng Chen, Zhen Yang, Bohan Zhuang, and Qi Wu. Streaming video diffusion: Online video editing with diffusion models. *arXiv preprint arXiv:2405.19726*, 2024.

[10] Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, Juncheng Zhu, Mingyuan Fan, Hao Zhang, Sheng Chen, Zheng Chen, Chengchen Ma, et al. Skyreels-v2: Infinite-length film generative model. *arXiv preprint arXiv:2504.13074*, 2025.

[11] Julian Decart, Quinn Quevedo, Spruce McIntyre, Xinlei Campbell, Robert Chen, and Wachen. Oasis: A universe in a transformer, 2024.

[12] Chaorui Deng, Deyao Zhu, Kunchang Li, Shi Guang, and Haoqi Fan. Causal diffusion transformers for generative modeling. *arXiv preprint arXiv:2412.12095*, 2024.

[13] Haoge Deng, Ting Pan, Haiwen Diao, Zhengxiong Luo, Yufeng Cui, Huchuan Lu, Shiguang Shan, Yonggang Qi, and Xinlong Wang. Autoregressive video generation without vector quantization. In *ICLR*, 2025.

[14] Emily L Denton et al. Unsupervised learning of disentangled representations from video. In *NeurIPS*, 2017.

[15] Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. *ArXiv*, abs/2412.05496, 2024.

[16] Kaifeng Gao, Jiaxin Shi, Hanwang Zhang, Chunping Wang, Jun Xiao, and Long Chen. Ca2-vdm: Efficient autoregressive video diffusion model with causal generation and cache sharing. *arXiv preprint arXiv:2411.16375*, 2024.

[17] Songwei Ge, Thomas Hayes, Harry Yang, Xi Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. Long video generation with time-agnostic vqgan and time-sensitive transformer. In *ECCV*, 2022.

[18] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[19] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *COLM*, 2024.

[20] Yuchao Gu, Weijia Mao, and Mike Zheng Shou. Long-context autoregressive video modeling with next-frame prediction. *arXiv preprint arXiv:2503.19325*, 2025.

[21] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[22] Yuwei Guo, Ceyuan Yang, Ziyan Yang, Zhibei Ma, Zhijie Lin, Zhenheng Yang, Dahua Lin, and Lu Jiang. Long context tuning for video generation. *arXiv preprint arXiv:2503.10589*, 2025.

[23] Agrim Gupta, Lijun Yu, Kihyuk Sohn, Xiuye Gu, Meera Hahn, Fei-Fei Li, Irfan Essa, Lu Jiang, and José Lezama. Photorealistic video generation with diffusion models. In *ECCV*, 2024.

[24] Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

[25] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey A. Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen video: High definition video generation with diffusion models. *ArXiv*, abs/2210.02303, 2022.

[26] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. In *NeurIPS*, 2022.

[27] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. In *ICLR*, 2023.

[28] Jinyi Hu, Shengding Hu, Yuxuan Song, Yufei Huang, Mingxuan Wang, Hao Zhou, Zhiyuan Liu, Wei-Ying Ma, and Maosong Sun. Acdit: Interpolating autoregressive conditional modeling and diffusion transformer. *arXiv preprint arXiv:2412.07720*, 2024.

[29] Nick Huang, Aaron Gokaslan, Volodymyr Kuleshov, and James Tompkin. The gan is dead; long live the gan! a modern gan baseline. In *NeurIPS*, 2024.

[30] Zemin Huang, Zhengyang Geng, Weijian Luo, and Guo-jun Qi. Flow generator matching. *arXiv preprint arXiv:2410.19310*, 2024.

[31] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *CVPR*, 2024.

[32] Simon Jenni and Paolo Favaro. On stabilizing generative adversarial training with noise. In *CVPR*, 2019.

[33] Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In *ICLR*, 2025.

[34] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. In *ICLR*, 2019.

[35] Jihwan Kim, Junoh Kang, Jinyoung Choi, and Bohyung Han. Fifo-diffusion: Generating infinite videos from text without training. In *NeurIPS*, 2024.

[36] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *NeurIPS*, 2021.

[37] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[38] Dan Kondratyuk, Lijun Yu, Xiuye Gu, Jose Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, et al. Videopoet: A large language model for zero-shot video generation. In *ICML*, 2024.

[39] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

[40] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *NeurIPS*, 2016.

[41] Qing Li, Xun Tang, Junkun Peng, Yuanzheng Tan, and Yong Jiang. Latency reducing in real-time internet video transport: A survey. *SSRN 4654242*, 2023.

[42] Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.

[43] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *NeurIPS*, 2024.

[44] Zhengqi Li, Qianqian Wang, Noah Snavely, and Angjoo Kanazawa. Infinitenature-zero: Learning perpetual view generation of natural scenes from single images. In *ECCV*, 2022.

[45] Zongyi Li, Shujie Hu, Shujie Liu, Long Zhou, Jeongsoo Choi, Lingwei Meng, Xun Guo, Jinyu Li, Hefei Ling, and Furu Wei. Arlon: Boosting diffusion transformers with autoregressive models for long video generation. In *ICLR*, 2025.

[46] Feng Liang, Akio Kodaira, Chenfeng Xu, Masayoshi Tomizuka, Kurt Keutzer, and Diana Marculescu. Looking backward: Streaming video-to-video translation with feature banks. In *ICLR*, 2025.

[47] Shanchuan Lin, Xin Xia, Yuxi Ren, Ceyuan Yang, Xuefeng Xiao, and Lu Jiang. Diffusion adversarial post-training for one-step video generation. *arXiv preprint arXiv:2501.08316*, 2025.

[48] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

[49] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *ICCV*, 2021.

[50] Haozhe Liu, Shikun Liu, Zijian Zhou, Mengmeng Xu, Yanping Xie, Xiao Han, Juan C Pérez, Ding Liu, Kumara Kahatapitiya, Menglin Jia, et al. Mardini: Masked autoregressive diffusion for video generation at scale. *arXiv preprint arXiv:2410.20280*, 2024.

[51] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

[52] Yaofang Liu, Yumeng Ren, Xiaodong Cun, Aitor Artola, Yang Liu, Tieyong Zeng, Raymond H Chan, and Jean-michel Morel. Redefining temporal modeling in video diffusion: The vectorized timestep approach. *arXiv preprint arXiv:2410.03160*, 2024.

[53] Zhijun Liu, Shuai Wang, Sho Inoue, Qibing Bai, and Haizhou Li. Autoregressive diffusion transformer for text-to-speech synthesis. *arXiv preprint arXiv:2406.05551*, 2024.

[54] Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. In *NeurIPS*, 2023.

[55] Weijian Luo, Zemin Huang, Zhengyang Geng, J Zico Kolter, and Guo-jun Qi. One-step diffusion distillation through score implicit matching. *NeurIPS*, 2024.

[56] Xiaofeng Mao, Zhengkai Jiang, Fu-Yun Wang, Jiangning Zhang, Hao Chen, Mingmin Chi, Yabiao Wang, and Wenhan Luo. Osv: One step is enough for high-quality image to video generation. In *CVPR*, 2025.

[57] William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *TACL*, 2023.

[58] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *ICML*, 2018.

[59] Sicheng Mo, Thao Nguyen, Xun Huang, Siddharth Srinivasan Iyer, Yijun Li, Yuchen Liu, Abhishek Tandon, Eli Shechtman, Krishna Kumar Singh, Yong Jae Lee, et al. X-fusion: Introducing new modality to frozen large language models. *arXiv preprint arXiv:2504.20996*, 2025.

[60] Mang Ning, Mingxiao Li, Jianlin Su, Albert Ali Salah, and Itir Onal Ertugrul. Elucidating the exposure bias in diffusion models. In *ICLR*, 2024.

[61] Jack Parker-Holder, Philip Ball, Jake Bruce, Vibhavari Dasagi, Kristian Holsheimer, Christos Kaplanis, Alexandre Moufarek, Guy Scully, Jeremy Shar, Jimmy Shi, Stephen Spencer, Jessica Yung, Michael Dennis, Sultan Kenjeyev, Shangbang Long, Vlad Mnih, Harris Chan, Maxime Gazeau, Bonnie Li, Fabio Pardo, Luyu Wang, Lei Zhang, Frederic Besse, Tim Harley, Anna Mitenkova, Jane Wang, Jeff Clune, Demis Hassabis, Raia Hadsell, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 2: A large-scale foundation world model, 2024.

[62] William S Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

[63] Ryan Po, Yotam Nitzan, Richard Zhang, Berlin Chen, Tri Dao, Eli Shechtman, Gordon Wetzstein, and Xun Huang. Long-context state-space video world models. *arXiv preprint arXiv:2505.20171*, 2025.

[64] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

[65] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *ICLR*, 2016.

[66] Shuhuai Ren, Shuming Ma, Xu Sun, and Furu Wei. Next block prediction: Video generation via semi-auto-regressive modeling. *arXiv preprint arXiv:2502.07737*, 2025.

[67] David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling diffusion models. In *ICML*, 2024.

[68] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping. In *ICCV*, 2017.

[69] Sand-AI. Magi-1: Autoregressive video generation at scale, 2025.

[70] Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024.

[71] Florian Schmidt. Generalization in generation: A closer look at exposure bias. *EMNLP-IJCNLP 2019*, page 157, 2019.

[72] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. In *NeurIPS*, 2024.

[73] Kiwhan Song, Boyuan Chen, Max Simchowitz, Yilun Du, Russ Tedrake, and Vincent Sitzmann. History-guided video diffusion. *arXiv preprint arXiv:2502.06764*, 2025.

[74] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *ICML*, 2023.

[75] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. In *NeurIPS*, 2021.

[76] Mingzhen Sun, Weining Wang, Gen Li, Jiawei Liu, Jiahui Sun, Wanquan Feng, Shanshan Lao, SiYu Zhou, Qian He, and Jing Liu. Ar-diffusion: Asynchronous video generation with auto-regressive diffusion. In *CVPR*, 2025.

[77] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *CVPR*, 2018.

[78] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *ICLR*, 2025.

[79] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *NeurIPS*, 2017.

[80] R Villegas, H Moraldo, S Castro, M Babaeizadeh, H Zhang, J Kunze, PJ Kindermans, MT Saffar, and D Erhan. Phenaki: Variable length video generation from open domain textual descriptions. In *ICLR*, 2023.

[81] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *ICLR*, 2017.

[82] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *NeurIPS*, 2016.

[83] Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[84] Jing Wang, Fengzhuo Zhang, Xiaoli Li, Vincent YF Tan, Tianyu Pang, Chao Du, Aixin Sun, and Zhuoran Yang. Error analyses of auto-regressive video diffusion models: A unified framework. *arXiv preprint arXiv:2503.10704*, 2025.

[85] Wenhao Wang and Yi Yang. Vidprom: A million-scale real prompt-gallery dataset for text-to-video diffusion models. In *NeurIPS*, 2024.

[86] Yuqing Wang, Tianwei Xiong, Daquan Zhou, Zhijie Lin, Yang Zhao, Bingyi Kang, Jiashi Feng, and Xihui Liu. Loong: Generating minute-level long videos with autoregressive language models. *arXiv preprint arXiv:2410.02757*, 2024.

[87] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *NeurIPS*, 2023.

[88] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. In *ICLR*, 2020.

[89] Wenming Weng, Ruoyu Feng, Yanhui Wang, Qi Dai, Chunyu Wang, Dacheng Yin, Zhiyuan Zhao, Kai Qiu, Jianmin Bao, Yuhui Yuan, et al. Art-v: Auto-regressive text-to-video generation with diffusion models. In *CVPR*, 2024.

[90] Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Jian Jiao, Juntao Li, Jian Guo, Nan Duan, Weizhu Chen, et al. Ar-diffusion: Auto-regressive diffusion model for text generation. In *NeurIPS*, 2023.

[91] Yushu Wu, Zhixing Zhang, Yanyu Li, Yanwu Xu, Anil Kag, Yang Sui, Huseyin Coskun, Ke Ma, Aleksei Lebedev, Ju Hu, et al. Snapgen-v: Generating a five-second video within five seconds on a mobile device. In *CVPR*, 2025.

[92] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024.

[93] Desai Xie, Zhan Xu, Yicong Hong, Hao Tan, Difan Liu, Feng Liu, Arie Kaufman, and Yang Zhou. Progressive autoregressive video diffusion models. *arXiv preprint arXiv:2410.08151*, 2024.

[94] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

[95] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[96] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. In *ICLR*, 2024.

[97] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *ICLR*, 2025.

[98] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and Bill Freeman. Improved distribution matching distillation for fast image synthesis. *NeurIPS*, 2024.

[99] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024.

[100] Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *CVPR*, 2025.

[101] Alan Yu, Ge Yang, Ran Choi, Yajvan Ravan, John Leonard, and Phillip Isola. Learning visual parkour from generated images. In *CoRL*, 2024.

[102] Jiwen Yu, Yiran Qin, Xintao Wang, Pengfei Wan, Di Zhang, and Xihui Liu. Gamefactory: Creating new games with generative interactive videos. *arXiv preprint arXiv:2501.08325*, 2025.

[103] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, 2017.

[104] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, et al. Language model beats diffusion–tokenizer is key to visual generation. In *ICLR*, 2024.

[105] Lvmin Zhang and Maneesh Agrawala. Packing input frame context in next-frame prediction models for video generation. *arXiv preprint arXiv:2504.12626*, 2025.

[106] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025.

[107] Yuan Zhang, Jiacheng Jiang, Guoqing Ma, Zhiying Lu, Haoyang Huang, Jianlong Yuan, and Nan Duan. Generative pre-trained autoregressive diffusion transformer. *arXiv preprint arXiv:2505.07344*, 2025.

[108] Zhixing Zhang, Yanyu Li, Yushu Wu, Anil Kag, Ivan Skorokhodov, Willi Menapace, Aliaksandr Siarohin, Junli Cao, Dimitris Metaxas, Sergey Tulyakov, et al. Sf-v: Single forward video generation model. In *NeurIPS*, 2024.

[109] Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. In *ICLR*, 2025.

[110] Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. In *ICLR*, 2025.

[111] Deyu Zhou, Quan Sun, Yuang Peng, Kun Yan, Runpei Dong, Duomin Wang, Zheng Ge, Nan Duan, Xiangyu Zhang, Lionel M Ni, et al. Taming teacher forcing for masked autoregressive video generation. In *CVPR*, 2025.

[112] Mingyuan Zhou, Huangjie Zheng, Yi Gu, Zhendong Wang, and Hai Huang. Adversarial score identity distillation: Rapidly surpassing the teacher in one step. In *ICLR*, 2025.

[113] Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *ICML*, 2024.

A Implementation Details

Our implementation is largely based on the open-source code of Wan2.1 [83] and CausVid [100]. The attention implementation of Diffusion Forcing and Teacher Forcing baselines is based on FlexAttention [15], while the attention in Self Forcing is based on FlashAttention-3 [72].

Noise schedule and model parameterization. Following the Wan2.1 series, we adopt the flow matching framework [48, 51], with time step shifting $t'(k,t) = (kt/1000)/(1+(k-1)(t/1000)) \cdot 1000$ and a shift factor $k = 5$. The forward process is specified as $x_t = \frac{t'}{1000}x + \frac{1-t'}{1000}\epsilon, \epsilon \sim \mathcal{N}(0, I)$ with $t \in [0, 1000]$.

The data prediction model is given by:

$$G_\theta(x, t, c) = c_{\text{skip}} \cdot \epsilon - c_{\text{out}} \cdot v_\theta(c_{\text{in}} \cdot x_t, c_{\text{noise}}(t'), c). \tag{1}$$

We keep the preconditioning coefficients the same as the base models' configuration, i.e., $c_{\text{skip}} = c_{\text{in}} = c_{\text{out}} = 1$ and $c_{\text{noise}}(t) = t$. Our few-step diffusion process employs a uniform 4-step schedule $[t_4, t_3, t_2, t_1] = [1000, 750, 500, 250]$.

Prompt preprocessing. We use the VidProS subset from VidProM [85], which contains around 1M semantically unique user-written text-to-video prompts. We filter out prompts that are too short (less than 20 characters), contain command line arguments (e.g., –ar 16:9), or have a NSFW probability greater than 0.01 for any annotated category (toxicity, obscenity, identity attack, insult, threat, and sexual explicitness). This results in a total of around 250k prompts. We then expand those prompts with `Qwen/Qwen2.5-7B-Instruct` [95], using the system prompt (English version) provided in the open-source implementation of Wan2.1 [83]. For VBench evaluation, we similarly rewrite the test prompts using `Qwen/Qwen2.5-7B-Instruct`. We note that the VBench results of the Wan2.1 base model are also obtained with prompt rewriting, and we report baseline results with prompt rewriting, provided that the model supports such enhancements.

Training details. Most of our training runs use 64 NVIDIA GPUs (80GB memory each) with a per-GPU batch size of 1. We implement gradient accumulation for configurations requiring a larger effective batch size than 64. Our DMD training runs take only approximately 1.5 hours to converge, while SiD/GAN training takes 2-3 hours on 64 H100 GPUs. We initialize the real score network and critic network using the pretrained weights of the base model. We list all other training configurations, as well as the choice of real score network and critic network for different distribution matching objectives, in Table 3. We describe detailed training configurations for each distribution matching objective below.

For DMD, the gradient of the reverse Kullback-Leibler divergence is given by [54, 87, 99]:

$$\nabla_\theta \mathbb{E}_t[D_{\text{KL}}(p_{\theta,t} \| p_{\text{data},t})] = -\mathbb{E}_{t,\hat{x}_t \sim q_{t|0}(\hat{x}_t|\hat{x}),\hat{x}\sim p_\theta(\hat{x})}\left[(s_{\text{real}}(\hat{x}_t, t) - s_{\text{fake}}(\hat{x}_t, t))\frac{\partial \hat{x}}{\partial \theta}\right], \tag{2}$$

where $s_{\text{real}}(\cdot, t)$ is the score function for $p_{\text{data},t}$, approximated by a pretrained diffusion model $f_\phi(\cdot, t)$, also referred to as the real score network, and $s_{\text{fake}}(\cdot, t)$ is the score function for $p_{\theta,t}$ and is learned

Table 3: Specification of training hyperparameters

Hyperparameters	DMD	SiD	GAN
Real score network	Wan2.1-T2V-14B	Wan2.1-T2V-1.3B	N/A
Real score CFG weight	3.0	3.0	N/A
Critic network initialization	Wan2.1-T2V-1.3B	Wan2.1-T2V-1.3B	Wan2.1-T2V-1.3B
Batch size	64	64	768
Optimizer (G_θ)	AdamW, $\beta_1 = 0$, $\beta_2 = 0.999$, ϵ = 1e-8, weight_decay= 0.01	Adam, $\beta_1 = 0$, $\beta_2 = 0.999$, ϵ = 1e-8, weight_decay= 0	AdamW, $\beta_1 = 0$, $\beta_2 = 0.999$, ϵ = 1e-8, weight_decay= 0.01
Optimizer (f_ψ)	AdamW, $\beta_1 = 0$, $\beta_2 = 0.999$, ϵ = 1e-8, weight_decay= 0.01	Adam, $\beta_1 = 0$, $\beta_2 = 0.999$, ϵ = 1e-8, weight_decay= 0	AdamW, $\beta_1 = 0$, $\beta_2 = 0.999$, ϵ = 1e-8, weight_decay= 0.01
Learning rate (G_θ)	2e-6	2e-6	2e-6
Learning rate (f_ψ)	4e-7	2e-6	2e-6
Generator/critic update ratio	5	5	1
EMA decay	0.99	0.99	0.99

through a critic network $f_\psi(\cdot, t)$ via the standard diffusion loss. The gradient in Eqn. (2) is equivalent to the following loss function:

$$\mathcal{L}_{\text{DMD}}(\theta) = \mathbb{E}_{t,\hat{x}_t,\hat{x}}\left[\frac{1}{2}\left\|\hat{x} - \text{sg}\left[\hat{x} - (f_\psi(\hat{x}_t,t) - f_\phi(\hat{x}_t,t))\right]\right\|^2\right], \tag{3}$$

where $\text{sg}[\cdot]$ denotes the stop gradient operator.

Similar to the pipeline of DMD, the SiD loss is given by [113]:

$$\mathcal{L}_{\text{SiD}}(\theta) = \mathbb{E}_{t,\hat{x}_t,\hat{x}}\left[(f_\phi(\hat{x}_t,t) - f_\psi(\hat{x}_t,t))^T(f_\psi(\hat{x}_t,t) - \hat{x}) + (1-\alpha)\|f_\phi(\hat{x}_t,t) - f_\psi(\hat{x}_t,t)\|^2\right], \tag{4}$$

which can be shown that the case of $\alpha = 0.5$ corresponds the gradient of the Fisher divergence $\mathbb{E}_{t,p_{\theta,t}}[\|\nabla \log p_{\theta,t} - \nabla \log p_{\text{data},t}\|^2]$ [30, 55]. Empirically, it is observed that the second term often leads to unstable training and thus $\alpha = 1$ is typically adopted for better performance [112, 113], which is also followed in this work.

For GAN training, we add additional cross-attention layers and classification heads to the initialized critic network. We employ relativistic loss [34] and approximate the regularization terms (R1 and R2) using finite difference following Seaweed-APT [47]. Specifically, we perturb the noisy real/fake data with additional small Gaussian noise and encourage the discriminator output to be similar to the original one. The final training objective is defined as:

$$\mathcal{L}_{\text{reg}} = \frac{1}{2}\mathbb{E}_{t,x_t,\hat{x}_t,\epsilon,\hat{\epsilon}}\left[\|f_\psi(x_t) - f_\psi(x_t + \sigma\cdot\epsilon)\|_2^2 + \|f_\psi(\hat{x}_t) - f_\psi(\hat{x}_t + \sigma\cdot\hat{\epsilon})\|_2^2\right] \tag{5}$$

$$\mathcal{L}_D(\psi) = -\mathbb{E}_{t,x_t,\hat{x}_t}\left[\log\left(\text{sigmoid}\left(f_\psi(x_t) - f_\psi(\hat{x}_t)\right)\right)\right] + \lambda\mathcal{L}_{\text{reg}} \tag{6}$$

$$\mathcal{L}_G(\theta) = -\mathbb{E}_{t,x_t,\hat{x}_t}\left[\log\left(\text{sigmoid}\left(f_\psi(\hat{x}_t) - f_\psi(x_t)\right)\right)\right] \tag{7}$$

where $x_t \sim p_{\text{data},t}$, $\hat{x}_t \sim p_{\theta,t}$ are the noisy real and fake data, respectively, ϵ and $\hat{\epsilon}$ are Gaussian noise sampled from $\mathcal{N}(0,1)$, and f_ψ is the critic network (discriminator) of GAN. We use $\lambda = 30$, $\sigma = 0.05$ for all experiments. For a video generated from the output of the s-th step (see Algorithm 1 for details), we find that only sampling t from $[t_{s-1}, t_s]$ helps stabilize the training. We also adopt a large batch size of 768 for training stability.

B Importance of local attention training in rolling KV cache

We qualitatively ablate two training settings for video extrapolation using the rolling KV cache technique. In the naive baseline, the model is trained such that every chunk always attends to the first chunk during denoising. In contrast, our proposed method restricts the attention window to prevent the model from attending to the first chunk when denoising the last chunk. As shown in Fig. 7, the naive baseline exhibits visual artifacts when extrapolating videos beyond the training context length, whereas our proposed solution mitigates this issue.

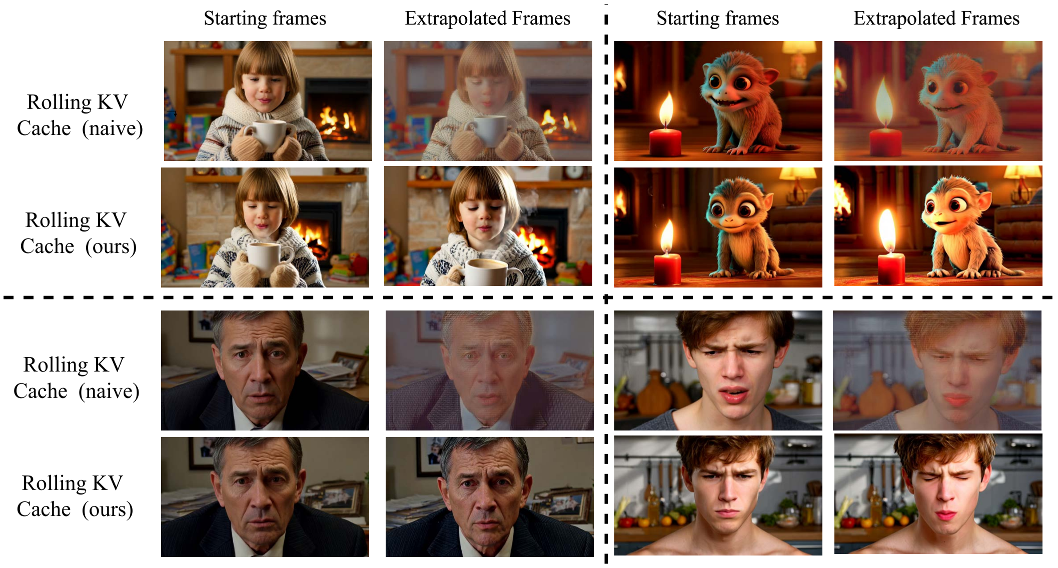


Figure 7: **Qualitative comparisons on video extrapolation.** We present a visual comparison between the naive baseline and our proposed technique for rolling KV cache-based video extrapolation. Compared to our method using local attention window training, extrapolated video frames from the naive baseline exhibit severe visual artifacts.

C VBench Scores Across All Dimensions

In Fig. 8, we evaluate Self Forcing (both chunk-wise and frame-wise AR versions) using all 16 VBench metrics against representative models. Self Forcing generally outperforms other models in terms of semantic alignment, evidenced by the high scores in scene, object class, multiple objects, and human action dimensions. Our methods also achieve good frame-wise quality, as indicated by the high scores in aesthetic quality and imaging quality. Our frame-wise AR variant exhibits more dynamic motion (high dynamic degree score) but worse temporal consistency (worse background consistency, motion smoothness, and larger temporal flickering) than the chunk-wise AR variant.

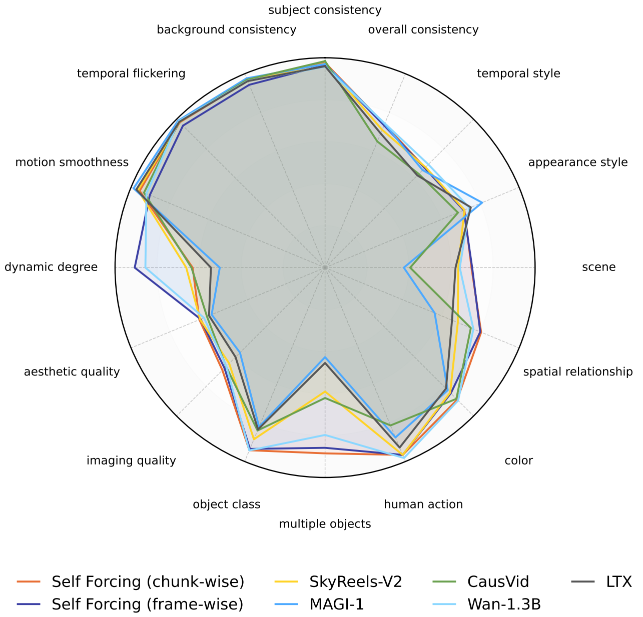


Figure 8: **VBench scores visualization.** We compare Self Forcing with SkyReels-V2 [10], Wan2.1-1.3B [83], MAGI-1 [69], and CausVid [100] using all 16 VBench metrics.

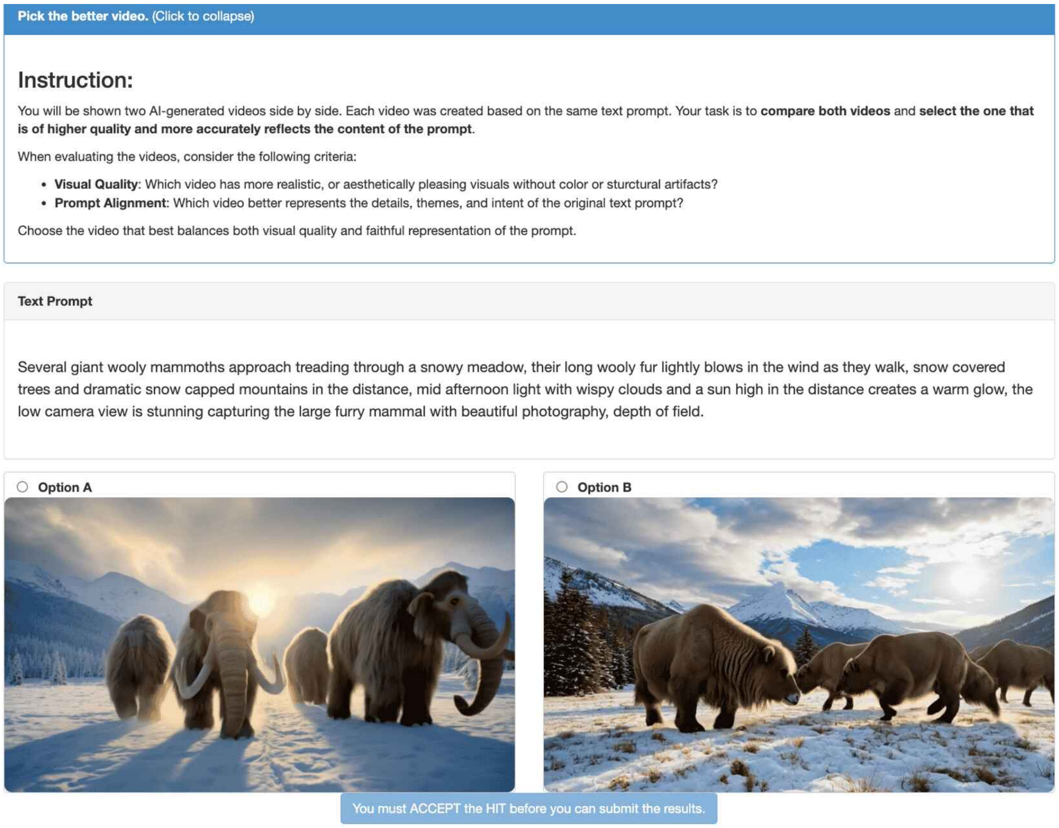
D Broader Societal Impact

Generative modeling—particularly for videos—carries significant potential for misuse. It can lead to serious societal consequences, most notably the spread of disinformation through deepfakes that become increasingly difficult to distinguish from authentic content. Additionally, these models can reinforce harmful stereotypes and amplify existing societal biases without careful governance and responsible deployment.

Our research on real-time video generation creates additional complexities, as it removes one of the practical barriers (computational cost) that currently limits widespread misuse. While our methods enable positive applications like creative content production and accessibility tools, we acknowledge the dual-use nature of this technology and encourage continued research into detection methods, watermarking techniques, and policy frameworks that can help mitigate potential harms.

E User Study Details

In the user preference study, we show users two videos side by side using the same text prompt. We ask the users to select the one that is overall better, considering both quality and prompt alignment. Detailed instructions are shown in Fig. 9. We use all 1003 prompts from MovieGenBench [64] and each prompt is evaluated by a single user.



Pick the better video. (Click to collapse)

Instruction:

You will be shown two AI-generated videos side by side. Each video was created based on the same text prompt. Your task is to **compare both videos** and **select the one that is of higher quality and more accurately reflects the content of the prompt**.

When evaluating the videos, consider the following criteria:

- **Visual Quality**: Which video has more realistic, or aesthetically pleasing visuals without color or sturctural artifacts?
- **Prompt Alignment**: Which video better represents the details, themes, and intent of the original text prompt?

Choose the video that best balances both visual quality and faithful representation of the prompt.

Text Prompt

Several giant wooly mammoths approach treading through a snowy meadow, their long wooly fur lightly blows in the wind as they walk, snow covered trees and dramatic snow capped mountains in the distance, mid afternoon light with wispy clouds and a sun high in the distance creates a warm glow, the low camera view is stunning capturing the large furry mammal with beautiful photography, depth of field.

○ **Option A**

○ **Option B**

Figure 9: **User study instruction screenshots.**