Packing Input Frame Context in Next-Frame Prediction Models for Video Generation

Lvmin Zhang and Maneesh Agrawala
Stanford University

We present a neural network structure, FramePack, to train next-frame (or next-frame-section) prediction models for video generation. The FramePack compresses input frames to make the transformer context length a fixed number regardless of the video length. As a result, we are able to process a large number of frames using video diffusion with computation bottleneck similar to image diffusion. This also makes the training video batch sizes significantly higher (batch sizes become comparable to image diffusion training). We also propose an anti-drifting sampling method that generates frames in inverted temporal order with early-established endpoints to avoid exposure bias (error accumulation over iterations). Finally, we show that existing video diffusion models can be finetuned with FramePack, and their visual quality may be improved because the next-frame prediction supports more balanced diffusion schedulers with less extreme flow shift timesteps.

1 Introduction

Forgetting and drifting are the two most critical problems in next-frame (or next-frame-section) prediction models for video generation. Herein, "forgetting" refers to the fading of memory as the model struggles to remember earlier content and maintain consistent temporal dependencies, whereas "drifting" refers to the iterative degradation of visual quality due to error accumulation over time (also called exposure bias).

A fundamental dilemma emerges when attempting to simultaneously address both forgetting and drifting: any method that mitigates forgetting by enhancing memory may also make error accumulation/propagation faster, thereby exacerbating drifting; any method that reduces drifting by interrupting error propagation and weakening the temporal dependencies (*e.g.*, masking or re-noising the history) may also worsen the forgetting. This essential trade-off hinders the scalability of next-frame prediction models.

The forgetting problem leads to a naive solution to encode more frames, but this quickly becomes computationally intractable due to the quadratic attention complexity of transformers (or sub-quadratic optimizations like FlashAttn, *etc*.). Moreover, video frames contain significant temporal redundancy, rendering naive full-context approaches less efficient. The substantial duplication of visual features across consecutive frames reveals the potential to design effective compression systems to facilitate memorization.

The drifting problem is influenced by memorizing mechanisms from multiple aspects. The source of drifting lies in initial errors that occur in individual frames, while the effect is the propagation and accumulation of these errors across subsequent frames, leading to degraded visual quality. On one hand, a stronger memorizing mechanism leads to better temporal consistency and reduces the occurrence of initial errors, thereby mitigating drifting. On the other hand, a stronger memorizing mechanism also memorizes more errors and thus accelerates error propagation when errors do occur, exacerbating drifting. This paradoxical relationship between memory mechanisms and drifting necessitates carefully designed training/sampling methods to facilitate error correction or interrupt error propagation.

In this paper, we propose FramePack as an anti-forgetting memory structure along with anti-drifting sampling methods. The FramePack structure addresses the forgetting problem by compressing input frames based on their relative importance, ensuring the total transformer context length converges to a fixed upper bound regardless of video duration. This enables the model to encode significantly more frames without increasing the computational bottleneck, thereby facilitating anti-forgetting. Moreover, we propose anti-drifting sampling methods that break the causal prediction chain and incorporate bi-directional context. These methods include generating endpoint frames before filling intermediate content, and an inverted temporal sampling approach where frames are generated in reverse order, each attempting to approach a known high-quality frame (*e.g.*, an input frame in image-to-video tasks). We show that these methods effectively reduce the occurrence of errors and prevent their propagation.

We demonstrate that existing pretrained video diffusion models (*e.g.*, HunyuanVideo, Wan, *etc.*) can be finetuned with FramePack. Our experiments reveal several findings: because next-frame prediction generates smaller tensor sizes per step compared to full-video generation, it enables more balanced diffusion schedulers with less extreme flow shift timesteps. We observe that the resulting less aggressive schedulers may lead to improved visual quality, beyond the direct goals of addressing the forgetting and drifting problems.

2 Related Work

2.1 Anti-forgetting and Anti-drifting

Noise scheduling and augmentation in history frames modify noise levels at specific timesteps, video times, or image frequencies to create causal computation or anti-drifting effects. These methods generally reduce dependency on past frames. DiffusionForcing [4] and RollingDiffusion [31] are typical examples. Our ablation studies investigate the influence of adding noise to history frames.

Classifier-Free Guidance (CFG) over history frames applies different masks or noise levels to opposite sides of guidance to amplify the forgetting-drifting trade-off. HistoryGuidance [33] demonstrates this approach. Our ablation studies include guidance-based noise scheduling.

Anchor frames can be used as planning elements for video generation. StreamingT2V [15] and ART-V [41] use reference images as anchors. Video planning approaches [25, 57, 16, 46, 1, 47] use image or video anchors for content planning.

2.2 Long Video Generation

Extending video generation beyond short clips remains an open problem. LVDM [14] generates long videos using latent diffusion, while Phenaki [37] creates variable-length videos from sequences of text prompts. Gen-L-Video [38] applies temporal co-denoising for multi-text conditioned videos, and FreeNoise [30] extends pretrained models without additional training via noise rescheduling. NUWA-XL [48] implements a Diffusion-over-Diffusion architecture with coarse-to-fine processing, while Video-Infinity [35] overcomes computational constraints through distributed generation. StreamingT2V [15] produces consistent, dynamic, and extendable videos without hard cuts, and CausVid [51] transforms bidirectional models into fast autoregressive ones through distillation. Recent advances include GPT-like architecture (ViD-GPT [11]), multi-event generation (MEVG [29]), attention control for multi-prompt generation (DiTCtrl [3]), precise temporal control (MinT [42]), history-based guidance (HistoryGuidance [33]), unified next-token and full-sequence diffusion (DiffusionForcing [4]), SpectralBlend temporal attention (FreeLong [26]), video autoregressive modeling (FAR [12]), and test-time training (TTT [6]).

2.3 Efficient Architectures for Video Generation

We discuss typical methods to improve video model efficiency. Linear attention [2, 45, 39, 5, 52, 20] reduces the attention complexity by reformulating linear operations. Sparse attention [43, 55, 56, 44] computes attention only on important token pairs, skipping negligible attention values. Low-bit computation [23, 58, 22] quantizes model weights and activations to lower precision for faster computation. Low-bit attention [54, 53] specifically optimizes attention computation through quantization techniques. Hidden state caching [28, 24] reuses intermediate computations across diffusion timesteps to avoid redundant calculations. Distillation [34, 27, 49, 50] transfers knowledge from larger models to smaller ones or reduces sampling steps while preserving quality.

3 Method

We consider a video generation model that predicts next frames repeatedly to form a video. For simplicity, we consider diffusion-based next-frame-section prediction models using Diffusion Transformers (DiTs) that generate a section of S unknown frames $\boldsymbol{X} \in \mathbb{R}^{S\times h\times w\times c}$ conditioned on T input frames $\boldsymbol{F} \in \mathbb{R}^{T\times h\times w\times c}$. All definitions of frames and pixels refer to latent representations, as most modern models operate in latent space.

For next-frame (or next-frame-section) prediction, S is typically 1 (or a small number). We focus on the challenging case where $T \gg S$. With per-frame context length L_f (typically $L_f \approx 1560$ for each 480p frame in Hunyuan/Wan/Flux), the vanilla DiT yields total context length $L = L_f(T+S)$. This causes context length explosion when T is large.

3.1 FramePack

We observe that the input frames have different importance when predicting the next frame, and we can prioritize the input frames according to their importance. Without loss of generality, we consider a simple case where the temporal proximity reflects importance: frames temporally closer to the prediction target can be more relevant. We enumerate all frames with $\boldsymbol{F}_0$ being the most important (*e.g.*, the most recent) and $\boldsymbol{F}_{T-1}$ being the least (*e.g.*, the oldest).

We define a length function $\phi(\boldsymbol{F}_i)$ that determines each frame's context length after VAE encoding and transformer patchifying, applying progressive compression to less important frames as

$$\phi(\boldsymbol{F}_i) = \frac{L_f}{\lambda^i}\,, \tag{1}$$

where $\lambda > 1$ is a compression parameter. The frame-wise compression is achieved by manipulating the transformer's patchify kernel size in the input layer (*e.g.*, $\lambda = 2, i = 5$ means a kernel size where the product of all dims equals $2^5 = 32$ like the 3D kernel $2 \times 4 \times 4$, or $8 \times 2 \times 2$, *etc.*). The total context length then follows a geometric progression

$$L = S \cdot L_f + L_f \cdot \sum_{i=0}^{T-1} \frac{1}{\lambda^i} = S \cdot L_f + L_f \cdot \frac{1 - 1/\lambda^T}{1 - 1/\lambda}\,, \tag{2}$$

and when $T \to \infty$, the total context length converges to

$$\lim_{T\to\infty} L = S \cdot L_f + \frac{1}{1-1/\lambda} \cdot L_f = \left(S + \frac{\lambda}{\lambda - 1}\right) \cdot L_f\,, \tag{3}$$

and this bounded context length makes FramePack's computation bottleneck invariant to the input frame number T.

Since most hardware supports efficient matrix processing by powers of 2, we mainly discuss the case of $\lambda = 2$ in this paper. Note that we can represent arbitrary compression rates by duplicating (or dropping) several specific terms in the power-of-2 sequence: considering the accumulation $\sum_{i=0}^{+\infty} \frac{1}{2^i} = \frac{2}{2-1} = 2$, if we want to loosen it a bit, for example to 2.625, we can duplicate the terms $\frac{1}{2}$ and $\frac{1}{8}$ so that $\frac{1}{1} + \frac{1}{2} + (\frac{1}{2}) + \frac{1}{4} + \frac{1}{8} + (\frac{1}{8}) + \frac{1}{16} + ... = \frac{1}{2} + \frac{1}{8} + \sum_{i=0}^{+\infty} \frac{1}{2^i} = 2.625$. Following this, one can cover arbitrary rates by converting the rate value to binary bits and then translate every bit.

The patchifying operations in most DiTs are 3D, and we denote the 3D kernel as (p_f, p_h, p_w) representing the steps in frame number, height, and width. A same compression rate can be achieved by multiple possible kernel sizes, *e.g.*, the compression rate of 64 can be achieved by $(1, 8, 8)$, $(4, 4, 4)$, $(16, 2, 2)$, $(64, 1, 1)$, *etc.* These would lead to different FramePack compression schedules.

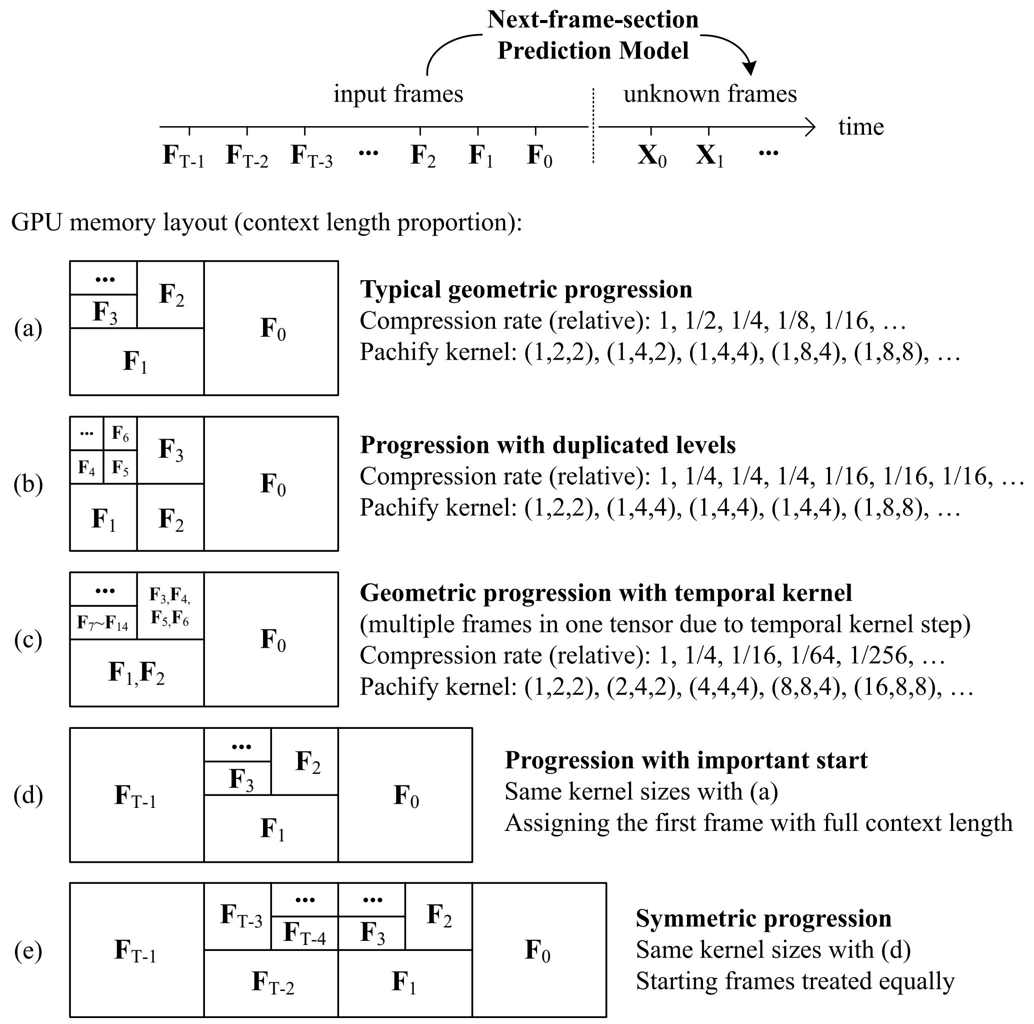
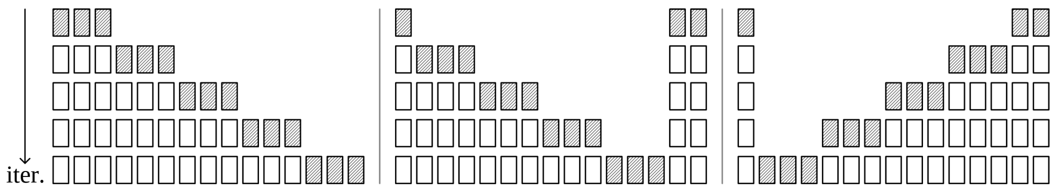

Next-frame-section Prediction Model
input frames
unknown frames
time
F_{T-1} F_{T-2} F_{T-3} ··· F_2 F_1 F_0
X_0 X_1 ···
GPU memory layout (context length proportion):
(a)
Typical geometric progression
Compression rate (relative): 1, 1/2, 1/4, 1/8, 1/16, …
Pachify kernel: (1,2,2), (1,4,2), (1,4,4), (1,8,4), (1,8,8), …
(b)
Progression with duplicated levels
Compression rate (relative): 1, 1/4, 1/4, 1/4, 1/16, 1/16, 1/16, …
Pachify kernel: (1,2,2), (1,4,4), (1,4,4), (1,4,4), (1,8,8), …
(c)
F_7~F_{14}
F_3,F_4, F_5,F_6
F_1,F_2
Geometric progression with temporal kernel
(multiple frames in one tensor due to temporal kernel step)
Compression rate (relative): 1, 1/4, 1/16, 1/64, 1/256, …
Pachify kernel: (1,2,2), (2,4,2), (4,4,4), (8,8,4), (16,8,8), …
(d)
Progression with important start
Same kernel sizes with (a)
Assigning the first frame with full context length
(e)
Symmetric progression
Same kernel sizes with (d)
Starting frames treated equally

Figure 1: **Ablation Variants of FramePack.** We present several typical kernel structures of FramePack with commonly used kernel sizes and compression rates. This list does not necessarily cover all popular variants, and more structures can be developed in a similar way.

Independent patchifying parameters We observe that features from deep neural networks at different compression rates exhibit notable differences. Empirical evidence shows that using independent parameters for the different input projections at multiple compression rates facilitates stabilized learning. We assign the most commonly used input compression kernels as independent neural network layers: $(2, 4, 4)$, $(4, 8, 8)$, and $(8, 16, 16)$. For higher compressions (*e.g.*, at $(16, 32, 32)$), we first downsample (*e.g.*, with $2 \times 2 \times 2$) and then use the largest kernel $(8, 16, 16)$. This allows us to handle all compression rates. When training these new input projection layers, we initialize their weights by interpolating from the pretrained patchifying projection (*e.g.*, the $(2, 4, 4)$ projection of HunyuanVideo/Wan).

Tail options While in theory FramePack can process videos of arbitrary length with a fixed, invariant context length, practical considerations arise when the input frame length becomes extremely large. In the tail area, frames may fall below a minimum unit size (*e.g.*, a single latent pixel). We discuss 3 options to handle the tail: (1) simply delete the tail; (2) allow each tail frame to increase the context length by a single latent pixel; (3) apply global average pooling to all tail frames and process them with the largest kernel. In our tests, the visual differences between these options are relatively negligible. We note that the tail refers to the least important frames, not always the oldest frames (in some cases, we can assign old frames with higher importance).



(a) Vanilla (b) Anti-drifting (c) Inverted anti-drifting

Figure 2: **Visualization of Sampling Approaches.** We present sampling approaches to generate frames in different temporal orders. The shadowed squares are the generated frames in each iteration, whereas the white squares are the iteration inputs. Note that only the vanilla approach relies on causal inputs, and both the anti-drifting and inverted anti-drifting approaches receive bi-directional inputs.

RoPE alignment When encoding inputs with different compression kernels, the different context lengths require RoPE (Rotary Position Embedding) alignment. RoPE generates complex numbers with real and imaginary parts for each token position across all channels, which we refer to as "phase". RoPE typically multiplies the phase to neural network features channel-wise. To match the RoPE encoding after compression, we directly downsample (using average pooling) the RoPE phases to match the compression kernels.

3.2 FramePack Variants

The above FramePack structure is discussed with the frame-wise importance defined by simple temporal proximity, and the compression schedule with simple geometric progression. We consider more variants for practical applications and optimal quality.

We visualize typical structures in Fig. 1. First, compression levels can be duplicated and combined with higher compression rates. In Fig. 1-(b), a power-of-4 sequence with each level repeated 3 times allows for same kernel sizes in frame width and height, making the compression more compact. Compression can also be applied in the temporal dimension (*e.g.*, using a power-of-two sequence), as in Fig. 1-(c). This method encodes multiple frames in the same tensor, which is naturally aligned with DiT architectures.

We discuss alternative frame-wise importance modeling beyond simple temporal proximity in Fig. 1-(d,e). For instance, we can assign full-length context to the oldest frame, or treat both beginning and ending frames as equally important while applying higher compression to middle frames. These structures are particularly effective for applications like image-to-video generation, where user-provided initial frames carry higher importance.

3.3 Anti-drifting Sampling

Drifting is a common problem in next-frame prediction models where visual quality degrades as video length increases. While the underlying cause remains an open research problem, we observe that drifting only happens in causal sampling (*i.e.*, when models only access past frames). We show that providing access to future frames (even a single future frame) will get rid of drifting. We point out that bi-directional context, rather than strictly causal dependencies, might be fundamental for maintaining video quality.

The vanilla sampling method shown in Fig. 2-(a), *i.e.*, iteratively predicting future frames, can be modified into Fig. 2-(b), where the first iteration simultaneously generates both beginning and ending sections, while subsequent iterations fill the gaps between these anchors. This bi-directional approach prevents drifting since the ending frames are established in the first iteration, and all future generations attempt to approximate them.

We discuss an important variant by inverting the sampling order in Fig. 2-(b) into Fig. 2-(c). This approach is effective for image-to-video generation because it can treat the user input as a high-quality first frame, and continuously refines generations to approximate the user frame (which is unlike Fig. 2-(b) that does not approximate the first frame), leading to overall high-quality videos.

All three methods in Fig. 2-(a,b,c) can generate videos of arbitrary length. Method (a) achieves this through direct iterative generation, while in methods (b) and (c), we can dynamically move the ending sections (or generated frames) to greater distances as our generated frames approach them. Alternatively, in practice, setting a sufficiently large time range (*e.g.*, 1 minute) in the first iteration typically satisfies practical requirements.

RoPE with random access These sampling methods require modifications to RoPE to support non-consecutive phases (time indices of frames). This is achieved by skipping the non-queried phases (indices) in the time dimension.

4 Experiments

4.1 Ablative Naming

To simplify the presentation of the experiments, we use a common naming convention for all ablative structures. A FramePack name is represented as a string such as `td_f16k4f4k2f1k1_g9_x_f1k1`. We explain the meaning of this notation:

Kernel: A kernel name is like `k1h2w2`. The `k` stands for "kernel", and `k1h2w2` indicates a patchify kernel with shape $(1, 2, 2)$, where the temporal size is 1, the height is 2, and the width is 2.

Kernel (simplified): For simplicity, since kernels that are multiples of $(1, 2, 2)$ are commonly used, we use abbreviated notation such as `k1` that only denotes the temporal dimension. Specifically, `k1` represents `k1h2w2` (the kernel $(1, 2, 2)$), `k2` represents `k2h4w4` (the kernel $(2, 4, 4)$), `k4` represents `k4h8w8` (the kernel $(4, 8, 8)$), *etc*.

Encoding frames: The notation `f16k4` indicates that 16 frames are encoded by the kernel `k4` (the simplified `k4h8w8`) with kernel size $(4, 8, 8)$.

Packing: The notation `f16k4f4k2f1k1` shows a way to encode 19 contiguous frames with three compression levels: the first 16 frames are encoded by the kernel `k4` (the kernel $(4, 8, 8)$), the next 4 frames are encoded by the kernel `k2` (the kernel $(2, 4, 4)$), and the last 1 frame is encoded by the kernel `k1` (the kernel $(1, 2, 2)$).

Tail: We append the notation with `td`, `ta`, or `tc` to indicate the tail frames before or after packing, such as `td_f16k4f4k2f1k1`. The three options are as discussed in Section 3.1. Herein, the "delete" option `td` deletes the tail. The "append" option `ta` compresses each tail frame by performing a 3D pooling of $(1, 32, 32)$ and then encodes with the nearest kernel, and the "compress" option `tc` uses global average pooling for all tail frames and compresses them with the nearest kernel.

Skipping: The notation `x` skips an arbitrary number of frames (including 0 frames).

Generating: The notation `g9` means generating 9 frames.

With the above naming convention, we can represent all ablative structures in a compact form. Note that this naming also implies the sampling approach as discussed in Section 3.3. Consider the three similar structures and their sampling:

`td_f16k4f4k2f1k1_g9`: The vanilla sampling that generates frames in temporal order.

`td_f16k4f4k2f1k1_g9_x_f1k1`: The anti-drifting sampling with an endpoint frame.

`f1k1_x_g9_f1k1f4k2f16k4_td`: The inverted anti-drifting sampling in inverted temporal order.

4.2 Base Model and Implementation Details

We implement FramePack with Wan and HunyuanVideo. We implement both the text-to-video and image-to-video structures, though both are naturally supported by next-frame-section prediction models and do not need architecture modifications.

The Wan base model is the official Wan2.1 model. The HunyuanVideo model is an improved version of official HunyuanVideo models to match the capability of Wan2.1, with the modifications: (1) adding the SigLip-Vision model `google/siglip-so400m-patch14-384` as a vision encoder, (2) removing the reliance on Tencent's internal MLLM, (3) freezing `LLama3.1` as a pure text model and ignoring its multi-modality, and (4) continued training on high-quality data.

Note that the numerical results presented in this paper should not be interpreted as direct comparisons between Wan and HunyuanVideo, as both were trained with similar computational budgets while Wan's larger model size means fewer relative resources (*e.g.*, batch size). Both models demonstrate comparable quality after sufficient training. We recommend HunyuanVideo as the default configuration for more efficient training and faster inference.

Dataset We follow the guidelines of LTXVideo [13]'s dataset collection pipeline to gather data at multiple resolutions and quality levels. All data are filtered using quality measurements and motion scores to ensure a high-quality diverse distribution. We use aspect ratio bucketing for multi-resolution training with a minimum unit size of 32 pixels. For example, the buckets at 480p resolution include: $(416, 960)$, $(448, 864)$, $(480, 832)$, $(512, 768)$, $(544, 704)$, $(576, 672)$, $(608, 640)$, $(640, 608)$, $(672, 576)$, $(704, 544)$, $(768, 512)$, $(832, 480)$, $(864, 448)$, and $(960, 416)$.

Small-scale Training FramePack achieves a batch size of 64 on a single 8×A100-80G node with the 13B HunyuanVideo model at 480p resolution (without using any image-based workaround training that many community LoRAs are built on top of). This batch size is comparable even to image diffusion models like the 12B Flux, making FramePack suitable for personal or laboratory-scale training and experimentation.

Training We conduct all experiments using H100 GPU clusters. Each FramePack variant in the ablation studies consumes approximately 48 hours of training, while the final models are trained for one week. We use the Adafactor optimizer with a fixed learning rate of 1e-5 without learning rate scheduling. The gradient norm clip is set to 0.5.

Lower flow shift Since the next-frame-section prediction methods generate smaller 3D tensors at each inference compared to full-video diffusion, the models can be trained with much lower flow shift values. We use Flux's dynamic flow shift to train all models. We find that this leads to sharp and clean results that are closer to real videos.

4.3 Evaluation Metrics

We discuss the metrics for evaluating ablative architectures. The tested inputs consist of 512 real user prompts for text-to-video and 512 image-prompt pairs for image-to-video tasks. All test samples were curated from real users to ensure diversity and real-world applicability. For quantitative tests, we by default use 30 seconds for long videos and 5 seconds for short videos.

4.3.1 Multi-dimension Metrics

Multiple metrics for video evaluations are consistent with common benchmarks, *e.g.*, VBench [18], VBench2 [59], *etc*. All scores are normalized using the same numerical system with VBench.

Clarity: The MUSIQ [21] image quality predictor trained on the SPAQ [10] dataset. This metric measures artifacts such as noise and blurring in the generated frames.

Aesthetic: The LAION aesthetic predictor [32]. This metric measures the aesthetic values perceived with a CLIP-based estimator.

Motion: The video frame interpolation model [17] modified by VBench to measure the smoothness of motion.

Dynamic: The RAFT [36] modified by VBench to estimate the degree of dynamics. Note that the "dynamic" metric and "motion" metric represent a trade-off, *e.g.*, a still image may rank high on motion smoothness but will be penalized by low dynamic degrees.

Semantic: The video-text score computed by ViCLIP [40]. This metric measures the overall semantic consistency between the generated video and the input text.

Anatomy: The ViT [9] pretrained by VBench for identifying the per-frame presence of hands, faces, bodies, *etc*.

Identity: The facial feature similarity using ArcFace [7] with face detection by RetinaFace [8] to measure the identity consistency.

Table 1: **Ablation study.** We evaluate different FramePack configurations across multiple global metrics, drifting metrics, and human assessments. The table is divided into three groups based on sampling approach: vanilla sampling (top), anti-drifting sampling (middle), and inverted anti-drifting sampling (bottom). The tests are conducted with HunyuanVideo as base. Bests in bold. ELO differences within ± 16 are considered ties.

Variant	Global Metrics ↑							Drifting Metrics ↓				Human	
	Clarity	Aesthetic	Motion	Dynamic	Semantic	Anatomy	Identity	$\Delta^{\text{Clarity}}_{\text{drift}}$	$\Delta^{\text{Motion}}_{\text{drift}}$	$\Delta^{\text{Semantic}}_{\text{drift}}$	$\Delta^{\text{Anatomy}}_{\text{drift}}$	ELO ↑	Rank ↓
`td_f8k8f4k4f2k2f1k1_g9`	67.33%	65.94%	94.53%	92.91%	20.06%	66.97%	71.72%	3.25%	3.45%	7.45%	16.56%	1084	8
`td_f64k8f16k4f4k2f1k1_g9`	67.71%	66.71%	95.09%	93.91%	21.39%	66.56%	71.50%	3.22%	3.38%	7.25%	16.43%	1068	9
`td_f512k8f64k4f8k2f1k1_g9`	67.74%	66.44%	94.27%	92.91%	21.90%	67.73%	71.35%	3.18%	3.32%	7.05%	15.95%	1088	8
`td_f512k16f64k8f8k2f1k1_g9`	66.13%	64.52%	94.77%	94.18%	19.21%	65.72%	71.51%	3.25%	3.45%	7.85%	17.62%	1072	9
`td_f16k4f4k2f1k1_g9`	67.37%	65.05%	95.97%	91.66%	19.15%	65.38%	71.73%	3.22%	3.48%	6.65%	17.36%	1076	9
`td_f16k4f2k2f1k1_g1`	65.81%	64.39%	94.91%	**94.92%**	19.20%	64.16%	69.70%	3.43%	3.77%	9.81%	20.89%	1036	11
`td_f16k4f2k2f1k1_g4`	66.57%	64.99%	94.00%	82.85%	19.40%	65.97%	69.06%	3.36%	3.68%	8.55%	19.09%	1052	10
`td_f16k4f2k2f1k1_g9`	67.15%	65.29%	94.08%	92.97%	20.73%	66.46%	71.08%	3.18%	3.42%	7.45%	18.05%	1080	9
`tc_f16k4f2k2f1k1_g9`	67.62%	65.07%	94.02%	90.33%	21.71%	71.70%	71.01%	3.15%	3.25%	7.25%	17.21%	1092	8
`ta_f16k4f2k2f1k1_g9`	67.02%	66.44%	94.11%	91.09%	21.48%	71.92%	72.18%	3.12%	3.18%	7.05%	17.66%	1096	8
`td_f8k8f4k4f2k2f1k1_g9_x_f1k1`	68.46%	66.95%	96.46%	75.55%	22.88%	85.10%	75.84%	2.95%	2.85%	5.95%	15.87%	1132	4
`td_f64k8f16k4f4k2f1k1_g9_x_f1k1`	68.21%	67.75%	95.74%	85.97%	22.79%	81.09%	76.29%	2.92%	2.98%	5.95%	15.99%	1136	4
`td_f512k8f64k4f8k2f1k1_g9_x_f1k1`	68.62%	67.72%	95.05%	76.50%	22.44%	82.01%	76.65%	2.88%	2.92%	5.75%	15.94%	1116	5
`td_f512k16f64k8f8k2f1k1_g9_x_f1k1`	68.35%	67.89%	97.73%	76.48%	22.75%	78.40%	76.16%	2.85%	2.85%	5.55%	14.04%	1120	5
`td_f16k4f4k2f1k1_g9_x_f1k1`	68.42%	67.59%	96.74%	74.37%	23.69%	79.14%	77.37%	2.82%	2.78%	5.35%	14.90%	1140	4
`td_f16k4f2k2f1k1_g1_x_f1k1`	65.32%	67.97%	95.26%	81.69%	19.97%	74.57%	77.93%	2.78%	2.72%	4.95%	14.80%	1084	7
`td_f16k4f2k2f1k1_g4_x_f1k1`	69.92%	67.49%	97.84%	71.90%	21.12%	74.84%	77.53%	2.75%	2.65%	4.95%	14.98%	1100	6
`td_f16k4f2k2f1k1_g9_x_f1k1`	69.51%	**69.15%**	96.97%	77.41%	23.03%	83.10%	69.25%	2.72%	2.58%	4.75%	13.73%	1144	4
`tc_f16k4f2k2f1k1_g9_x_f1k1`	69.62%	68.42%	96.45%	82.27%	23.08%	81.68%	69.08%	2.68%	2.52%	4.55%	13.54%	1148	4
`ta_f16k4f2k2f1k1_g9_x_f1k1`	69.21%	68.84%	97.87%	76.22%	22.77%	81.70%	75.23%	2.65%	2.45%	4.35%	13.76%	1152	4
`f1k1_x_g9_f1k1f2k2f4k4f8k8_td`	69.62%	67.87%	97.93%	88.79%	23.73%	86.99%	78.63%	2.55%	2.35%	4.15%	12.71%	**1205**	**1**
`f1k1_x_g9_f1k1f4k2f16k4f64k8_td`	69.56%	67.48%	97.42%	86.68%	24.48%	86.35%	78.06%	2.45%	2.25%	3.95%	12.52%	**1209**	**1**
`f1k1_x_g9_f1k1f8k2f64k4f512k8_td`	69.35%	67.89%	97.85%	88.21%	24.66%	76.99%	79.56%	2.35%	2.15%	3.75%	12.13%	**1213**	**1**
`f1k1_x_g9_f1k1f8k2f64k8f512k16_td`	69.20%	68.76%	**99.11%**	89.18%	24.35%	77.62%	79.88%	2.35%	2.05%	3.55%	11.10%	**1239**	**1**
`f1k1_x_g9_f1k1f4k2f16k4_td`	69.25%	67.40%	98.59%	89.01%	24.24%	77.31%	79.16%	2.45%	1.95%	3.35%	**9.22%**	**1221**	**1**
`f1k1_x_g1_f1k1f2k2f16k4_td`	69.74%	67.04%	98.09%	79.85%	24.89%	77.27%	80.86%	2.55%	2.15%	3.75%	11.37%	1148	3
`f1k1_x_g4_f1k1f2k2f16k4_td`	70.28%	68.11%	98.30%	79.45%	24.87%	77.95%	80.23%	2.45%	2.05%	3.75%	11.12%	1164	2
`f1k1_x_g9_f1k1f2k2f16k4_td`	70.73%	67.97%	98.11%	88.76%	25.79%	78.45%	**84.01%**	2.35%	1.95%	3.65%	11.98%	**1225**	**1**
`f1k1_x_g9_f1k1f2k2f16k4_tc`	70.48%	68.74%	98.16%	89.18%	**27.01%**	**87.21%**	81.04%	**2.18%**	1.85%	**2.54%**	11.71%	**1229**	**1**
`f1k1_x_g9_f1k1f2k2f16k4_ta`	**70.81%**	67.31%	98.15%	80.72%	25.60%	85.60%	82.76%	2.25%	**1.77%**	2.95%	9.85%	**1233**	**1**

4.3.2 Drifting Measurements

We discuss the methods to measure drifting. Several metrics from VBench-Long [19] already integrate long-range consistency measurements and may indicate drifting. We propose a more direct metric, namely start-end contrast, to measure drifting from multiple aspects.

We observe that when drifting occurs, a significant difference emerges between the beginning and ending portions of a video across various quality metrics. We define the start-end contrast $\Delta^{M}_{\text{drift}}$ for an arbitrary quality metric M as:

$$\Delta^{M}_{\text{drift}}(V) = |M(V_{\text{start}}) - M(V_{\text{end}})| \ , \tag{4}$$

where V is the tested video, V_{start} represents the first 15% of frames, and V_{end} represents the last 15% of frames. This start-end contrast can be applied to different metrics M (*e.g.*, motion score, image quality, *etc.*). The magnitude of $\Delta^{M}_{\text{drift}}(V)$ directly indicates the severity of drifting. Since video models may generate frames in different temporal orders (either forward or backward), we use the absolute difference to ensure our metric remains direction-agnostic.

4.3.3 Human Assessments

We collect human preferences from A/B tests. Each ablative architecture yields 100 results. The A/B tests are randomly distributed among ablations, and we ensure that each ablation covers at least 100 assessments. We report ELO-K32 score and the relative ranking.

4.4 Ablative Results

As shown in Table 1, we note several discoveries: (1) The inverted anti-drifting sampling method achieves the best results in 5 out of 7 metrics, while other sampling methods achieve at most a single best metric. (2) The inverted anti-drifting sampling achieves the best performance in all drifting metrics. (3) Human evaluations indicate that generating 9 frames per section yields better perception than generating 1 or 4 frames, as evidenced by the higher ELO scores for configurations with `g9` compared to their `g1` or `g4` counterparts. (4) While vanilla sampling achieved the highest dynamic score, this is likely attributable to drifting effects rather than genuine quality. (5) We also observe that

Table 2: **Comparison with relevant architectures.** We compare with other relevant methods across the same global metrics, drifting metrics, and human assessments used in the ablation study. The tests are conducted with HunyuanVideo as base. Bests in bold. ELO differences within ±16 are considered ties.

Method	Global Metrics ↑							Drifting Metrics ↓				Human	
	Clarity	Aesthetic	Motion	Dynamic	Semantic	Anatomy	Identity	$\Delta^{\text{Clarity}}_{\text{drift}}$	$\Delta^{\text{Motion}}_{\text{drift}}$	$\Delta^{\text{Semantic}}_{\text{drift}}$	$\Delta^{\text{Anatomy}}_{\text{drift}}$	ELO ↑	Rank ↓
Repeating image-to-video	56.73%	56.15%	94.34%	65.56%	17.74%	69.41%	73.06%	9.51%	3.92%	9.95%	19.88%	1016	5
Anchor frames (resembling StreamingT2V [15])	69.58%	67.35%	**99.96%**	74.97%	25.76%	85.01%	79.52%	2.85%	2.15%	3.45%	9.25%	1172	2
Causal attention (resembling CausVid [51])	62.88%	59.41%	96.98%	67.56%	19.15%	72.74%	75.32%	7.45%	3.15%	6.75%	15.96%	1087	4
Noisy history (resembling DiffusionForcing [4])	66.75%	63.78%	97.20%	**91.45%**	22.98%	77.60%	77.92%	4.35%	2.05%	3.15%	9.85%	1146	3
History guidance (resembling HistoryGuidance [33])	68.05%	**68.74%**	97.01%	73.39%	24.88%	81.84%	**83.42%**	7.35%	2.21%	5.25%	12.78%	1151	3
Proposed (`f1k1_x_g9_f1k1f2k2f16k4`)	**71.15%**	68.71%	99.45%	89.29%	**28.15%**	**86.53%**	82.11%	**2.25%**	**1.85%**	**2.68%**	**8.58%**	**1221**	**1**

differences between specific configuration options within the same sampling approach are relatively small and random, suggesting that the sampling category contributes more to the overall performance difference.

4.5 Comparison to Alternative Architectures

We discuss several relevant alternatives to generate videos in various ways. The involved methods either enable longer video generation, reduce computational bottlenecks, or both. To be specific, we implement these variants on top of HunyuanVideo default architecture (33 latent frames) using a simple naive sliding window with half context length for history inputs.

Repeating image-to-video: Directly repeat the image-to-video inference to make longer videos.

Anchor frames: Use an image as the anchor frame to avoid drifting. We implement a structure that resembles StreamingT2V [15].

Causal attention: Finetune full attention into causal attention for easier KV cache and faster inference. We implement a structure that resembles CausVid [51].

Noisy history: Delay the denoising timestep on history latents so that history latents are noisy (but less noisy than the current generating latents). Reducing the reliance on the history is beneficial for interrupting error accumulation, thus mitigates drifting, but at the cost of aggravating forgetting. We implement a structure that resembles DiffusionForcing [4].

History guidance: Delay the denoising timestep on history latents but also put the completely noised history on the unconditional side of CFG guidance. This will speed up error accumulation thus aggravating drifting, but also enhance memory to mitigate forgetting. We implement a structure that resembles HistoryGuidance [33].

As shown in Table 2, we observe several findings: (1) The proposed method achieves the best results in 3 global metrics, while other methods excel in at most one or two metrics. (2) The proposed method achieves the best results across all drifting metrics. (3) This evaluation aligns with human perceptions as evidenced by the ELO score.

5 Conclusion

In this paper, we presented FramePack, a neural network structure that aims to address the forgetting-drifting dilemma in next-frame prediction models for video generation. FramePack applies progressive compression to input frames based on their importance, ensuring the total context length converges to a fixed upper bound regardless of video duration. This is achieved through manipulating transformer patchify kernel sizes for different compression rates. Combined with the anti-drifting sampling methods that incorporate bi-directional context through early-established endpoints or inverted temporal ordering, our approach enables longer video generation while maintaining unchanged computational bottlenecks. Experiments suggest that FramePack can improve model responsiveness and visual quality in inference, while allowing for higher batch sizes in training. The approach is compatible with existing video diffusion models and supports various compression variants that can be optimized for wider applications.

References

[1] H. Bansal, Y. Bitton, M. Yarom, I. Szpektor, A. Grover, and K.-W. Chang. Talc: Time-aligned captions for multi-scene text-to-video generation. *arXiv preprint arXiv:2405.04682*, 2024.

[2] H. Cai, J. Li, M. Hu, C. Gan, and S. Han. Efficientvit: Lightweight multi-scale attention for high-resolution dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17302–17313, 2023.

[3] M. Cai, X. Cun, X. Li, W. Liu, Z. Zhang, Y. Zhang, Y. Shan, and X. Yue. Ditctrl: Exploring attention control in multi-modal diffusion transformer for tuning-free multi-prompt longer video generation. *arXiv preprint arXiv:2412.18597*, 2024.

[4] B. Chen, D. Martí Monsó, Y. Du, M. Simchowitz, R. Tedrake, and V. Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2025.

[5] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[6] K. Dalal, D. Koceja, G. Hussein, J. Xu, Y. Zhao, Y. Song, S. Han, K. C. Cheung, J. Kautz, C. Guestrin, T. Hashimoto, S. Koyejo, Y. Choi, Y. Sun, and X. Wang. One-minute video generation with test-time training, 2025. URL `https://arxiv.org/abs/2504.05298`.

[7] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.

[8] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kotsia, and S. Zafeiriou. Retinaface: Single-stage dense face localisation in the wild, 2019. URL `https://arxiv.org/abs/1905.00641`.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[10] Y. Fang, H. Zhu, Y. Zeng, K. Ma, and Z. Wang. Perceptual quality assessment of smartphone photography. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3674–3683, 2020. doi: 10.1109/CVPR42600.2020.00373.

[11] K. Gao, J. Shi, H. Zhang, C. Wang, and J. Xiao. Vid-gpt: Introducing gpt-style autoregressive generation in video diffusion models, 2024. URL `https://arxiv.org/abs/2406.10981`.

[12] Y. Gu, W. Mao, and M. Z. Shou. Long-context autoregressive video modeling with next-frame prediction, 2025. URL `https://arxiv.org/abs/2503.19325`.

[13] Y. HaCohen, N. Chiprut, B. Brazowski, D. Shalem, D. Moshe, E. Richardson, E. Levin, G. Shiran, N. Zabari, O. Gordon, et al. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

[14] Y. He, T. Yang, Y. Zhang, Y. Shan, and Q. Chen. Latent video diffusion models for high-fidelity long video generation. *arXiv preprint arXiv:2211.13221*, 2022.

[15] R. Henschel, L. Khachatryan, D. Hayrapetyan, H. Poghosyan, V. Tadevosyan, Z. Wang, S. Navasardyan, and H. Shi. Streamingt2v: Consistent, dynamic, and extendable long video generation from text. *arXiv preprint arXiv:2403.14773*, 2024.

[16] P. Hu, J. Jiang, J. Chen, M. Han, S. Liao, X. Chang, and X. Liang. Storyagent: Customized storytelling video generation via multi-agent collaboration. *arXiv preprint arXiv:2411.04925*, 2024.

[17] Z. Huang, T. Zhang, W. Heng, B. Shi, and S. Zhou. Real-time intermediate flow estimation for video frame interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[18] Z. Huang, Y. He, J. Yu, F. Zhang, C. Si, Y. Jiang, Y. Zhang, T. Wu, Q. Jin, N. Chanpaisit, Y. Wang, X. Chen, L. Wang, D. Lin, Y. Qiao, and Z. Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[19] Z. Huang, F. Zhang, X. Xu, Y. He, J. Yu, Z. Dong, Q. Ma, N. Chanpaisit, C. Si, Y. Jiang, Y. Wang, X. Chen, Y.-C. Chen, L. Wang, D. Lin, Y. Qiao, and Z. Liu. VBench++: Comprehensive and versatile benchmark suite for video generative models. *arXiv preprint arXiv:2411.13503*, 2024.

[20] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[21] J. Ke, Q. Wang, Y. Wang, P. Milanfar, and F. Yang. Musiq: Multi-scale image quality transformer, 2021. URL `https://arxiv.org/abs/2108.05997`.

[22] M. Li*, Y. Lin*, Z. Zhang*, T. Cai, X. Li, J. Guo, E. Xie, C. Meng, J.-Y. Zhu, and S. Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[23] X. Li, Y. Liu, L. Lian, H. Yang, Z. Dong, D. Kang, S. Zhang, and K. Keutzer. Q-diffusion: Quantizing diffusion models. In *ICCV*, 2023.

[24] F. Liu, S. Zhang, X. Wang, Y. Wei, H. Qiu, Y. Zhao, Y. Zhang, Q. Ye, and F. Wan. Timestep embedding tells: It's time to cache for video diffusion model. *arXiv preprint arXiv:2411.19108*, 2024.

[25] F. Long, Z. Qiu, T. Yao, and T. Mei. Videostudio: Generating consistent-content and multi-scene videos, 2024. URL `https://arxiv.org/abs/2401.01256`.

[26] Y. Lu, Y. Liang, L. Zhu, and Y. Yang. Freelong: Training-free long video generation with spectralblend temporal attention. *Advances in Neural Information Processing Systems*, 37: 131434–131455, 2025.

[27] S. Luo, Y. Tan, L. Huang, J. Li, and H. Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv: 2310.04378*, 2023.

[28] Z. Lv, C. Si, J. Song, Z. Yang, Y. Qiao, Z. Liu, and K.-Y. K. Wong. Fastercache: Training-free video diffusion model acceleration with high quality. 2024.

[29] G. Oh, J. Jeong, S. Kim, W. Byeon, J. Kim, S. Kim, and S. Kim. Mevg: Multi-event video generation with text-to-video models. In *European Conference on Computer Vision*, pages 401–418. Springer, 2024.

[30] H. Qiu, M. Xia, Y. Zhang, Y. He, X. Wang, Y. Shan, and Z. Liu. Freenoise: Tuning-free longer video diffusion via noise rescheduling. *arXiv preprint arXiv:2310.15169*, 2023.

[31] D. Ruhe, J. Heek, T. Salimans, and E. Hoogeboom. Rolling diffusion models, 2024.

[32] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35: 25278–25294, 2022.

[33] K. Song, B. Chen, M. Simchowitz, Y. Du, R. Tedrake, and V. Sitzmann. History-guided video diffusion. *arXiv preprint arXiv:2502.06764*, 2025.

[34] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. In *ICML*, 2023.

[35] Z. Tan, X. Yang, S. Liu, and X. Wang. Video-infinity: Distributed long video generation. *arXiv preprint arXiv:2406.16260*, 2024.

[36] Z. Teed and J. Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020. URL `https://arxiv.org/abs/2003.12039`.

[37] R. Villegas, M. Babaeizadeh, P.-J. Kindermans, H. Moraldo, H. Zhang, M. T. Saffar, S. Castro, J. Kunze, and D. Erhan. Phenaki: Variable length video generation from open domain textual description. *arXiv preprint arXiv:2210.02399*, 2022.

[38] F.-Y. Wang, W. Chen, G. Song, H.-J. Ye, Y. Liu, and H. Li. Gen-l-video: Multi-text to long video generation via temporal co-denoising. *arXiv preprint arXiv:2305.18264*, 2023.

[39] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[40] Y. Wang, Y. He, Y. Li, K. Li, J. Yu, X. Ma, X. Li, G. Chen, X. Chen, Y. Wang, et al. Internvid: A large-scale video-text dataset for multimodal understanding and generation. In *The Twelfth International Conference on Learning Representations*, 2023.

[41] W. Weng, R. Feng, Y. Wang, Q. Dai, C. Wang, D. Yin, Z. Zhao, K. Qiu, J. Bao, Y. Yuan, C. Luo, Y. Zhang, and Z. Xiong. Art•v: Auto-regressive text-to-video generation with diffusion models. *arXiv preprint arXiv:2311.18834*, 2023.

[42] Z. Wu, A. Siarohin, W. Menapace, I. Skorokhodov, Y. Fang, V. Chordia, I. Gilitschenski, and S. Tulyakov. Mind the time: Temporally-controlled multi-event video generation. *arXiv preprint arXiv:2412.05263*, 2024.

[43] H. Xi, S. Yang, Y. Zhao, C. Xu, M. Li, X. Li, Y. Lin, H. Cai, J. Zhang, D. Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

[44] Y. Xia, S. Ling, F. Fu, Y. Wang, H. Li, X. Xiao, and B. Cui. Training-free and adaptive sparse attention for efficient long video generation, 2025.

[45] E. Xie, J. Chen, J. Chen, H. Cai, H. Tang, Y. Lin, Z. Zhang, M. Li, L. Zhu, Y. Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint arXiv:2410.10629*, 2024.

[46] Z. Xie, D. Tang, D. Tan, J. Klein, T. F. Bissyand, and S. Ezzini. Dreamfactory: Pioneering multi-scene long video generation with a multi-agent framework. *arXiv preprint arXiv:2408.11788*, 2024.

[47] D. Yang, C. Zhan, Z. Wang, B. Wang, T. Ge, B. Zheng, and Q. Jin. Synchronized video storytelling: Generating video narrations with structured storyline. *arXiv preprint arXiv:2405.14040*, 2024.

[48] S. Yin, C. Wu, H. Yang, J. Wang, X. Wang, M. Ni, Z. Yang, L. Li, S. Liu, F. Yang, et al. Nuwa-xl: Diffusion over diffusion for extremely long video generation. *arXiv preprint arXiv:2303.12346*, 2023.

[49] T. Yin, M. Gharbi, T. Park, R. Zhang, E. Shechtman, F. Durand, and W. T. Freeman. Improved distribution matching distillation for fast image synthesis. *arXiv preprint arXiv:2405.14867*, 2024.

[50] T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024.

[51] T. Yin, Q. Zhang, R. Zhang, W. T. Freeman, F. Durand, E. Shechtman, and X. Huang. From slow bidirectional to fast causal video generators. *arXiv preprint arXiv:2412.07772*, 2024.

[52] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10819–10829, 2022.

[53] J. Zhang, H. Huang, P. Zhang, J. Wei, J. Zhu, and J. Chen. Sageattention2: Efficient attention with thorough outlier smoothing and per-thread int4 quantization, 2024. URL `https://arxiv.org/abs/2411.10958`.

[54] J. Zhang, J. Wei, P. Zhang, J. Zhu, and J. Chen. Sageattention: Accurate 8-bit attention for plug-and-play inference acceleration. In *International Conference on Learning Representations (ICLR)*, 2025.

[55] J. Zhang, C. Xiang, H. Huang, J. Wei, H. Xi, J. Zhu, and J. Chen. Spargeattn: Accurate sparse attention accelerating any model inference, 2025. URL `https://arxiv.org/abs/2502.18137`.

[56] P. Zhang, Y. Chen, R. Su, H. Ding, I. Stoica, Z. Liu, and H. Zhang. Fast video generation with sliding tile attention, 2025.

[57] C. Zhao, M. Liu, W. Wang, W. Chen, F. Wang, H. Chen, B. Zhang, and C. Shen. Moviedreamer: Hierarchical generation for coherent long visual sequence. *arXiv preprint arXiv:2407.16655*, 2024.

[58] T. Zhao, T. Fang, E. Liu, W. Rui, W. Soedarmadji, S. Li, Z. Lin, G. Dai, S. Yan, H. Yang, et al. Vidit-q: Efficient and accurate quantization of diffusion transformers for image and video generation. *arXiv preprint arXiv:2406.02540*, 2024.

[59] D. Zheng, Z. Huang, H. Liu, K. Zou, Y. He, F. Zhang, Y. Zhang, J. He, W.-S. Zheng, Y. Qiao, and Z. Liu. VBench-2.0: Advancing video generation benchmark suite for intrinsic faithfulness. *arXiv preprint arXiv:2503.21755*, 2025.