

Live Avatar: Streaming Real-time Audio-Driven Avatar Generation with Infinite Length

Yubo Huang^{1,2} Hailong Guo^{2,3} Fangtai Wu^{2,4} Weiqiang Wang⁵
 Shifeng Zhang² Shijie Huang² Qijun Gan⁴ Lin Liu¹
 Sirui Zhao^{1,*} Enhong Chen^{1,*} Jiaming Liu^{2,‡} Steven Hoi²

¹ University of Science and Technology of China ² Alibaba Group
³ Beijing University of Posts and Telecommunications ⁴ Zhejiang University
⁵ Monash University

snake1124@mail.ustc.edu.cn liujiaming.ljl@alibaba-inc.com

* Corresponding authors. ‡ Project Leader.

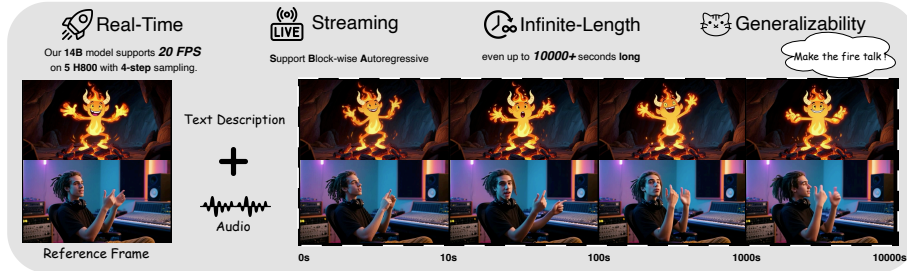


Fig. 1: We propose Live Avatar, a powerful real-time streaming model capable of infinitely long audio-driven avatar generation, producing lifelike avatars that talk, react, and persist seamlessly over hours.

Abstract. Audio-driven avatar interaction demands real-time, streaming, and infinite-length generation—capabilities fundamentally at odds with the sequential denoising and long-horizon drift of current diffusion models. We present Live Avatar, an algorithm-system co-designed framework that addresses both challenges for a 14-billion-parameter diffusion model. On the algorithm side, a two-stage pipeline distills a pre-trained bidirectional model into a causal, few-step streaming one, while a set of complementary long-horizon strategies eliminate identity drift and visual artifacts, enabling stable autoregressive generation exceeding 10,000 seconds. On the system side, Timestep-forcing Pipeline Parallelism (TPP) assigns each GPU a fixed denoising timestep, converting the sequential diffusion chain into an asynchronous spatial pipeline that simultaneously boosts throughput and improves temporal consistency. Live Avatar achieves 45 FPS with a TTFF of 1.21s on 5 H800 GPUs, and to our knowledge is the first to enable practical real-time streaming of a 14B diffusion model for infinite-length avatar generation. We further introduce GenBench, a standardized long-form benchmark, to facilitate reproducible evaluation. Our project page is at <https://liveavatar.github.io/>.

Table 1: Comparison of state-of-the-art audio-driven avatar generation methods. Live Avatar simultaneously achieves streaming, real-time, and infinite-length generation with a large-scale (14B) diffusion model.

Method	streaming	real-time	infinite-length	scale
Hallo3 [6]	✗	✗	✗	5B
StableAvatar [41]	✗	✗	✓	1.3B
Wan-S2V [13]	✗	✗	✗	14B
Ditto [23]	✓	✓	✓	0.2B
InfiniteTalk [49]	✗	✗	✓	14B
OmniAvatar [12]	✗	✗	✗	14B
Live Avatar (Ours)	✓	✓	✓	14B

1 Introduction

Audio-driven avatar generation, the synthesis of photorealistic human face video whose motion is driven by an input audio stream, is a foundational technology for interactive digital communication. Its applications are expansive, ranging from virtual reality and live streaming to digital assistants. The demand for systems capable of producing high-fidelity, expressive, and real-time avatars has driven significant recent advancements, particularly with the rise of diffusion models for high-fidelity video synthesis [2, 16, 43]. Despite their success in setting new benchmarks for visual quality, deploying these powerful generative models in real-time (≥ 24 FPS), streaming environments faces fundamental and conflicting challenges.

The first challenge is the real-time–fidelity trade-off. Large-scale diffusion models [13] yield superior visual quality but are inherently slow due to sequential multi-step denoising, while existing real-time approaches [23] sidestep this sequential bottleneck entirely by adopting non-iterative generation methods or very small models, at the cost of fidelity. Reconciling these two desiderata at scale remains an open problem.

The second challenge is long-horizon consistency. Existing methods suffer from compounding errors that cause identity drift and color artifacts within minutes [41]. Recent progress such as Self-Forcing [18] mitigates the train–test gap but still degrades rapidly under minute-level rollout; LongLive [50] sustains longer generation via native long-duration training but is designed for text-to-video and too costly to scale to 14B models.

To address these critical challenges, we propose **Live Avatar**, an algorithm–system co-designed framework that enables large diffusion models (up to 14 billion parameters) for real-time, streaming, and infinite-length audio-driven avatar generation without compromising visual fidelity.

For real-time streaming, we distill the model into a few-step causal one via Self-Forcing and propose Timestep-forcing Pipeline Parallelism (TPP), which pipelines denoising steps across GPUs to break the sequential sampling bottle-

neck, together with system-level optimizations to achieve high throughput and low latency at 14B scale.

For infinite-length stability, we first leverage the static-scene prior of avatar interaction to anchor identity in a persistent sink frame and store only *noisy* representations in the KV cache—the noise acts as a low-pass filter that suppresses accumulated artifacts while preserving motion dynamics, and keeps the latent distribution compact to prevent out-of-distribution drift. We further introduce Adaptive Attention Sink and Rolling RoPE to eliminate residual distribution and conditioning drift, together enabling stable infinite-length generation.

The core contributions of Live Avatar are as follows:

- **Causal, Streamable Adaptation Framework.** We propose a two-stage framework that adapts a pretrained bidirectional diffusion model into a causal, few-step streaming model, with a novel motion-frame-as-scaffold mechanism that bridges Stage 1 and Stage 2 by providing functionally analogous training signals, yielding a $5\times$ distillation convergence speedup.
- **Long-Horizon Stability.** We introduce three complementary strategies—History Corrupt, Adaptive Attention Sink, and Rolling RoPE—that jointly address error accumulation, distribution drift, and test-time conditioning drift in long autoregressive generation, enabling high-fidelity streaming that remains stable beyond 10,000 seconds.
- **Timestep-forcing Pipeline Parallelism (TPP).** We propose TPP, which assigns each GPU a fixed denoising timestep, breaking the sequential sampling bottleneck while naturally aligning each KV cache with its corresponding noise level to reduce flickering. Combined with system-level optimizations, TPP achieves 45 FPS with a TTFF of 1.21s on $5\times$ H800 GPUs, to our knowledge the first practical real-time streaming of a 14B diffusion model.
- **Standardized Benchmark.** We introduce GenBench (Short/Long), including long-video test cases exceeding five minutes, and will release the data and evaluation scripts in the camera-ready version.

2 Related Work

Streaming and Long Video Generation. Streaming and long video generation require efficient management of both computation and memory resources [7, 20, 25, 53]. Diffusion Forcing [3] introduces varied noise levels to sequential targets, enabling flexible-length streaming generation. Self-Forcing [18] addresses train-inference mismatch by conditioning on previously generated frames, yet still suffers from exposure bias and fails to produce minute-level long videos. LongLive [50] applies sliding window distillation for streaming long video generation, but is natively designed for text-to-video and unsuitable for image-conditioned tasks, and its training inefficiency prevents scaling to large-scale models such as 14B-parameter architectures. While these approaches improve video quality and temporal consistency, none achieve real-time, streaming long video generation with large-scale diffusion models.

Audio-driven Avatar Video Generation. Audio-driven avatar video generation requires subject consistency and effective motion control. Early works [34,55] leverage GANs and 3D motion models for lip-syncing and facial animation. With the success of diffusion models, several studies [40,54] adapt diffusion frameworks and ReferenceNet architectures for avatar generation. DiT-based video generation models [2,16,21,43] have demonstrated remarkable visual quality, inspiring a surge of DiT-based avatar methods [8,15,23,26,31,45,56,58]. Meanwhile, autoregressive approaches [1,4,22,48] offer an alternative by combining autoregressive and diffusion strategies. However, none achieve real-time, streaming, infinite-length avatar generation with large-scale diffusion models.

Diffusion Distillation. Diffusion model distillation accelerates video generation through various paradigms, including trajectory distillation [11], Consistency Models [27,37,57], and Distribution Matching Distillation (DMD) [29,30,52]. Among these, DMD has proven particularly effective for streaming video generation: CausVid [53], Self-Forcing [7,18], and LongLive [50] all employ DMD-based few-step distillation on temporally segmented long videos, achieving both significant sampling speedup and improved long-video quality. Recent analysis further suggests that DMD functions analogously to reinforcement learning, with the pretrained diffusion model serving as a reward signal [28], potentially explaining its effectiveness in streaming settings.

3 Preliminaries

3.1 Video Diffusion Models

Video diffusion models generate high-fidelity video sequences by progressively denoising from a Gaussian prior $x_T \sim N(0, I)$, following the reverse process of a forward diffusion. In this work, we adopt the flow matching [24], where noisy latents at time t are constructed as

$$x_t = (1 - s_t) \cdot x_0 + s_t \cdot x_T, \quad (1)$$

where $s_t \in [0, 1]$ is a scheduling function that controls the interpolation between clean and noise. The model is trained to predict the target velocity

$$v = x_T - x_0 \quad (2)$$

leading to the standard mean squared error objective:

$$\mathcal{L} = \mathbb{E}_{x_0, x_T, t} [\|v_\theta(x_t, t, c) - (x_T - x_0)\|_2^2] \quad (3)$$

where c denotes conditional inputs such as text embeddings.

To reduce computational cost, most modern video diffusion models operate in a compressed latent space. We use a causal 3D VAE to encode input videos into temporal-latent representations, ensuring that future frames do not leak during training. Text conditioning is achieved through a pre-trained language model that produces contextual embeddings fed into the diffusion backbone.

3.2 Distribution Matching Distillation

Distribution Matching Distillation (DMD) [52] aims to distill a pre-trained teacher diffusion model into a student model that operates with fewer sampling steps. Let $p_{\theta,t}(\mathbf{x}_t)$ denote the distribution induced by the few-step student model $\mathbf{x} = G_{\theta}(\mathbf{z})$, and let $p_{\text{data},t}(\mathbf{x}_t)$ represent the corresponding ground-truth distribution produced by the teacher diffusion model at time step t . The primary objective of DMD is to minimize the distribution, i.e., reverse Kullback–Leibler (KL) divergence, between these two distributions at each time step t : $\mathbb{E}_t [D_{\text{KL}}(p_{\theta,t} \parallel p_{\text{data},t})]$. The gradient of DMD loss is given by:

$$\nabla_{\theta} \mathcal{L}_{\text{DMD}} = -\mathbb{E}_{t,\mathbf{z}} \left[(s_{\text{real}}(\mathbf{x}_t, t) - s_{\text{fake},\phi}(\mathbf{x}_t, t))^{\top} \frac{\partial G_{\theta}(\mathbf{z})}{\partial \theta} \right] \quad (4)$$

where $\mathbf{x}_t = \Psi(\hat{\mathbf{x}}, t)$ is the noise scheduler, $\hat{\mathbf{x}} = G_{\theta}(\mathbf{z})$ is the data prediction of the distilled model, s_{real} and $s_{\text{fake},\phi}$ denote the score functions corresponding to the pre-trained teacher diffusion model and the student generator, respectively.

s_{real} , $s_{\text{fake},\phi}$, and $G_{\theta}(\mathbf{z})$ are all initialized from the pre-trained teacher model induced by v_{θ} . The training proceeds by alternately updating $s_{\text{fake},\phi}$ and $G_{\theta}(\mathbf{z})$, in which $s_{\text{fake},\phi}$ is trained on samples generated by the current student generator $G_{\theta}(\mathbf{z})$, and $G_{\theta}(\mathbf{z})$ is trained using the DMD loss defined above. For multi-step distillation, DMD first performs a multi-step sampling trajectory using the student generator: $\mathbf{z} \xrightarrow{G_{\theta}} \hat{\mathbf{x}}_{t_1} \xrightarrow{\Psi} \mathbf{x}_{t_1} \xrightarrow{G_{\theta}} \hat{\mathbf{x}}_{t_2} \xrightarrow{\Psi} \mathbf{x}_{t_2} \rightarrow \dots \rightarrow \mathbf{x}_{t_N}$. Then, at each training iteration, a random intermediate state $\mathbf{x}_{t_i}$ from this trajectory is selected and used in place of pure noise $\mathbf{z}$ as the starting point for the DMD training procedure [51].

4 The Live Avatar Framework

In this section, we present the technical details of Live Avatar. We first detail the model architecture in Sec. 4.1, followed by the overall training framework in Sec. 4.2 and Fig. 2. We then investigate long video generation in Sec. 4.3, proposing three strategies to address visual quality degradation, identity drift, and color artifacts in long-term generation. Finally, the inference framework and Timestep-forcing Pipeline Parallelism are demonstrated in Sec. 4.4.

4.1 Model Architecture

In order to enable streaming video generation, the Live Avatar adopts autoregressive generation by factorizing the joint distribution

$$B_{t-1}^i = D_{\theta}(B_t^i, \underbrace{B_t^{(i-w):(i-1)}}_{\text{kv cache}}, I, a^i, t^i) \quad (5)$$

combining diffusion-based frame synthesis with causal dependencies across chunks. B in Eq. 5 are blocks of consecutive noisy frame latents. In our work, we set the

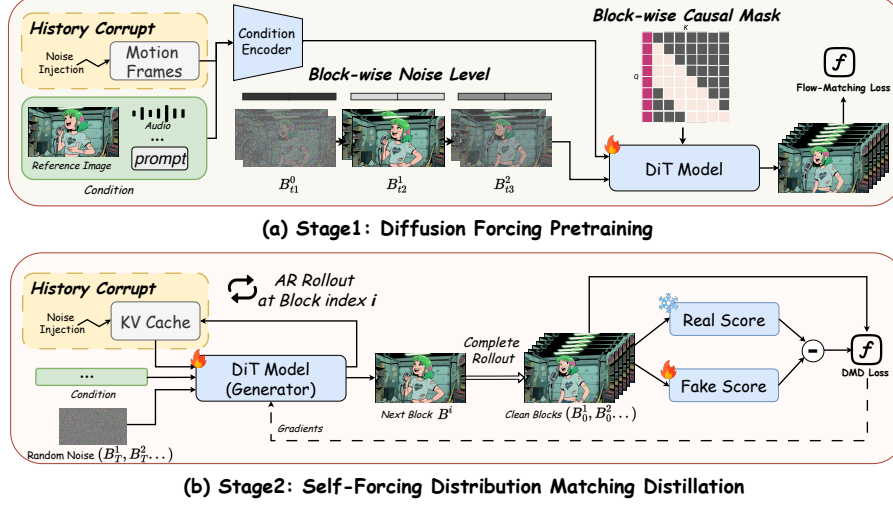


Fig. 2: The Live Avatar Training Framework. (a) Stage 1 Diffusion Forcing Pretraining with motion-frame scaffolding: noisy motion frames provide auxiliary temporal context alongside block-wise independent noise and causal attention masks. (b) Stage 2 Self-Forcing Distillation with History Corrupt: the motion frames are removed and replaced by the rolling KV cache, where noise level consistency between the KV cache and the noisy latents is enforced.

number of frame latents to 3. w is the rolling KV-cache window size. I denotes the sink frame, which provides the appearance; a^i and t^i are the audio embedding and prompt embedding for the i -th block respectively. The underscore t denotes the denoising step, and the superscript i denotes the block index. Note that in the model, the kv cache $B_{t^{(i-w):(i-1)}}^i$ and the noisy block B_t^i share the same noise level, this is a crucial design that improves generation quality and maximize the inference speed, which will be illustrated in Sec. 4.4.

4.2 Model Training

Our overall training framework is illustrated in Figure 2, which consists of two stages: 1) Stage 1, **Diffusion Forcing Pretraining** with motion-frame scaffolding, and 2) Stage 2, **Self-Forcing Distribution Matching Distillation** with History Corrupt.

Stage 1: Diffusion Forcing Pretraining. Following prior practice [3, 53], we apply block-wise independent noise scheduling and causal attention masks—frames within a block share full attention while inter-block attention is strictly causal—and train the model with the standard flow-matching loss. We additionally introduce motion frames [6] as auxiliary context, whose role is detailed below.

Motion-Frame as Scaffold. Employing a naive two-stage training converges slowly on long-horizon capabilities, yet each self-forcing rollout step is far more

expensive than supervised training. Inspired by [14], we repurpose the motion frames—context frames preceding the current clip, originally for clip continuation—as scaffolding in Stage 1 by injecting noise into them. Since both noisy motion frames and the Stage 2 noisy KV cache serve a functionally analogous role as temporal context for continuing generation, this cheaply teaches dynamics–identity decoupling without costly self-forcing rollouts. The motion frames are then entirely replaced by the KV cache in Stage 2, yielding a $5\times$ convergence speedup.

Stage 2: Self-Forcing Distillation. In Stage 2 we distill the bidirectional teacher into a causal, few-step streaming model following Self-Forcing [18]. The causal student denoises one block at a time while conditioning on a rolling KV cache of previously generated blocks. Crucially, we omit the extra clean-cache refresh forward pass used in prior work [18], so that the KV cache always contains *noisy* representations—a strategy we call **History Corrupt**, whose motivation is detailed in Sec. 4.3. During the denoising of each block, the model attends to the KV cache from previous blocks at the *same* timestep; the noise-level implications of this design are also discussed in Sec. 4.3.

4.3 Long Video Generation

Existing talking-avatar systems exhibit pronounced degradation over long, autoregressive generation—manifesting as identity drift, color shifts, and temporal instability [41]. In practice, we perform inference in a rolling KV cache fashion [18], which extends the temporal horizon but, on its own, does not prevent collapse. We attribute these long-horizon failures to three internal phenomena:

- (i) **Test-time conditioning drift.** The conditioning pattern at test time (e.g., the RoPE-relative positioning between the sink frame and current target blocks) gradually diverges from the training-time setup, weakening identity cues.
- (ii) **Distribution drift.** The distribution of generated frames progressively deviates from realistic video distributions, likely driven by persistent factors (e.g., a real-data sink frame whose distribution subtly differs from the model’s generation manifold) that continuously push the rolling generation toward unrealistic outputs.
- (iii) **Error accumulation.** Subtle imperfections in each generated block are inherited and compounded frame-by-frame through the clean KV cache, as the model attends to fine-grained details—including artifacts—from previous blocks. This compounding causes rapid quality deterioration over time.

To address these challenges, we propose three complementary strategies—History Corrupt, Adaptive Attention Sink, and Rolling RoPE—that together enable stable infinite-length generation.

History Corrupt. For avatar interaction, the subject typically resides in a relatively static scene: apart from facial expressions, body gestures, and mild background dynamics, the visual content does not undergo rapid change. This prior allows a simplifying design assumption—the sink frame can provide sufficiently useful appearance and identity information for *every* generated frame throughout the entire sequence. Under this assumption, the role of the rolling

KV cache reduces to conveying *motion dynamics* alone, while fine-grained identity and appearance details should be sourced exclusively from the persistent sink frame.

This motivates a simple yet effective design: we store only *noisy* representations in the KV cache, rather than the conventional clean cache. Intuitively, Gaussian noise acts as a low-pass filter [36] on the cached representations, suppressing high-frequency details (including accumulated artifacts) while preserving low-frequency motion structure. This forces the model to extract dynamic cues from the noisy history while relying on the clean sink frame for identity and appearance, achieving an effective *dynamics-identity decoupling*. As a result, generation errors in previous blocks are no longer faithfully propagated, directly addressing error accumulation (iii).

Furthermore, noisy KV caches also mitigate distribution drift (ii). At higher noise levels, the marginal distribution $p_t(\mathbf{x}_t)$ converges toward the Gaussian prior and occupies a progressively more compact region of the latent space [38]. Consequently, the noisy cache is far less likely to drift into out-of-distribution regions compared to a clean cache, which must remain on the narrow data manifold where small perturbations can compound into distributional shift.

We further observe empirically that enforcing the noisy block and its attended KV cache to share the *same* noise level—a strategy we call **timestep-forcing**—significantly reduces inter-frame flickering, consistent with observations in TalkingMachines [26]. This timestep-forcing constraint also naturally lends itself to efficient multi-GPU inference, as discussed in Sec. 4.4.

Adaptive Attention Sink (AAS)¹. By default, the user-provided reference image (i.e., the image-to-video conditioning input) serves as the sink frame. However, this real-data image resides on a slightly different distribution from the model’s own generation manifold, introducing a persistent bias that accumulates into color, exposure, or style deviations over long runs. To counteract this form of distribution drift (ii), AAS replaces the sink frame with the model’s own first generated latent immediately after the first block is produced, and uses it as the persistent sink for all subsequent conditioning. By keeping the sink frame within the model’s learned generation manifold, AAS eliminates the distributional mismatch between conditioning and generated content.

Rolling RoPE¹. To mitigate test-time conditioning drift (i), we introduce a dynamic position-alignment mechanism for the sink frame. The sink frame is permanently cached in KV memory and its temporal offset is adjusted via a controllable RoPE shift so that its relative position to the current noisy states remains consistent with training. This dynamic RoPE alignment lets the model continuously reference identity features from the sink frame without rigidly constraining local motion, thereby stabilizing long-range identity and structural fidelity.

¹ Details of AAS and Rolling RoPE are provided in the supplementary materials.

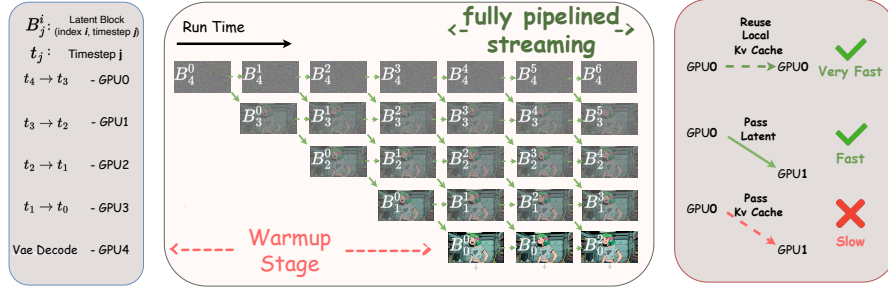


Fig. 3: A visual illustration of Timestep-forcing Pipeline Parallelism (TPP). After warm-up fills the pipeline, all GPUs denoise **simultaneously** in the fully pipelined stage, turning the sequential diffusion chain into an asynchronous spatial pipeline. For example, GPU2 always performs the $t_2 \rightarrow t_1$ step: it reuses its local KV cache (**very fast**) and sends only the latent to GPU3 (**fast**).

4.4 Timestep-forcing Pipeline Parallelism

Deploying large video generation models in real-time settings remains challenging due to the inherent sequential structure of diffusion-based sampling. Our distilled 14B model, for example, reaches only 5 FPS on a single GPU and 6 FPS under conventional 4-GPU sequential parallelization. In contrast, existing real-time streaming methods, such as CausVid [53] and LongLive [50], achieve higher frame rates by substantially reducing model capacity—often relying on lightweight 1.3B models and aggressive quantization—at the cost of generation fidelity. This establishes a long-standing tradeoff between model quality and real-time performance that has yet to be resolved.

To overcome the sequential bottleneck of diffusion sampling, we introduce **Timestep-forcing Pipeline Parallelism (TPP)** (illustrated in Figure 3), which assigns each GPU a fixed timestep t_i and partitions the T denoising steps across T devices. Each GPU repeatedly performs its designated transformation $t_i \rightarrow t_{i-1}$, converting the sequential diffusion chain into an asynchronous spatial pipeline. Through this reparameterization, throughput is determined by a single denoise forward rather than the sum of all diffusion steps, yielding an ideal speedup proportional to the total number of denoising steps. Importantly, TPP is both *model-agnostic* and *hardware-agnostic*: it applies to any causal diffusion model—not only distilled ones—and only requires that each device can execute a single denoising step with minimal inter-device bandwidth, since TPP communicates only the compact latent between stages.

TPP operates in two stages. During warm-up, the first block is propagated through all timesteps to fill the pipeline, which completes quickly given the small number of sampling steps. Once filled, the system enters the fully pipelined streaming phase: each GPU repeatedly performs its assigned denoising step, passes the latent to the next device, and immediately processes the next block, achieving maximal parallel throughput. Since TPP assigns one timestep per GPU, the timestep-forcing strategy (Sec. 4.3) maps directly onto the pipeline:

each GPU_i maintains its own local rolling KV cache, requiring no inter-GPU communication and thus remaining extremely fast and efficient. After finishing its step, GPU_i passes only the latent to GPU_{i+1} , keeping the communication cost negligible. To prevent pipeline bottlenecks, the VAE decoding stage is offloaded to an additional dedicated GPU, which consumes the clean latent and outputs synchronized video chunks.

5 Experiments

5.1 Experimental Settings

Implementation Details. The overall model architecture is borrowed from WanS2V [13]. In Stage 1, we initialize from its weights and implement motion frames with the frozen FramePack encoder from WanS2V; each motion frame is independently noised via the flow-matching forward process [24] with $t \sim \mathcal{U}[0, 1000]$. In Stage 2, the motion-frame encoder is removed, the teacher score and fake score branches are initialized from WanS2V, and the student is initialized from Stage 1. All training and inference are performed at a fixed resolution of 720×400 and 84 frames. Experiments are conducted on 128 NVIDIA H800 GPUs, with 25 K steps for Stage 1 and 500 steps for Stage 2. The per-GPU batch size is 1. To handle the high memory demand of Self-Forcing training, we adopt FSDP with gradient accumulation to reduce memory consumption. The learning rate is $1e-5$ for the student and $2e-6$ for the fake score. We group every 3 latents into a block, set the KV-cache window size $w=4$ blocks (Eq. 5), and use a single sink frame. We train models with a LoRA, whose rank and alpha are set to 128 and 64, respectively. At inference time, we apply a series of kernel-level optimizations (collectively referred to as Kernel Opt. in Table 3): FP8 quantization, FlashAttention-3, cuDNN fused kernels, torch.compile, LoRA weight merging, and VAE feature caching which caches intermediate features to enable streaming decoding. All reported metrics reflect this optimized configuration.

Datasets. We train on AVSpeech [10], a large-scale audio-visual dataset of talking-head clips. We adopt the preprocessing of OmniAvatar [12] and keep only clips longer than 10s, yielding 400K training samples. To evaluate our model’s out-of-domain (OOD) generalization, we created a synthetic benchmark named GenBench². This test set was generated using Gemini-2.5 Pro, Qwen-Image [46], and CosyVoice [9]. It is composed of two subsets: GenBench-ShortVideo, comprising 100 test samples with an approximate duration of 10 seconds, and GenBench-LongVideo, which contains 15 test videos, each exceeding 5 minutes in duration. The benchmark is designed to be challenging, featuring a wide diversity of character styles (photorealistic humans, animated characters, and anthropomorphic non-humans) and visual compositions, including frontal and profile views, as well as half-body and full-body shots. This variety allows for a robust assessment of the model’s performance on unseen data.

² We will release GenBench and the full evaluation scripts in the camera-ready version to facilitate standardized and reproducible long-form benchmarking.

Table 2: Quantitative comparisons of our methods with state-of-the-art methods.

Dataset	Model	Metrics					
		ASE $\uparrow$	IQA $\uparrow$	Sync-C $\uparrow$	Sync-D $\downarrow$	Dino-S $\uparrow$	FPS $\uparrow$
GenBench-ShortVideo	Ditto [23]	3.31	4.24	4.09	10.76	0.99	21.80
	Echomimic-V2 [32]	2.82	3.61	5.57	9.13	0.79	0.53
	Hallo3 [6]	3.12	3.97	4.74	10.19	0.94	0.26
	StableAvatar [41]	3.52	4.47	3.42	11.33	0.93	0.64
	OmniAvatar [12]	3.53	4.49	6.77	8.22	0.95	0.16
	WanS2V [13]	3.36	4.29	5.89	9.08	0.95	0.25
	Ours	3.44	4.51	7.03	8.30	0.96	45.2
GenBench-LongVideo	Ditto [23]	2.90	4.48	3.98	10.57	0.97	21.80
	Hallo3 [6]	2.65	4.04	6.18	9.29	0.83	0.26
	StableAvatar [41]	3.00	4.66	1.97	13.57	0.94	0.64
	OmniAvatar [12]	2.36	2.86	8.00	7.59	0.66	0.16
	WanS2V [13]	2.63	3.99	6.04	9.12	0.80	0.25
	Ours	3.42	4.76	7.16	8.31	0.97	45.2

Evaluation Metrics. We employ Q-Align [47] to evaluate perceptual quality (IQA) and aesthetic appeal (ASE). Audio-visual synchronization is measured via Sync-C and Sync-D [5]. Identity consistency is assessed by DINOv2 [33] cosine similarity (Dino-S) between generated frames and the reference image. For the in-domain AVSpeech evaluation (reported in the supplementary), we additionally include FID [17] and FVD [42] since ground-truth videos are available.

5.2 Comparison with Existing Methods

We compare Live Avatar against current state-of-the-art open-sourced audio-driven avatar generation approaches, including Ditto [23], Echomimic-V2 [32], Hallo3 [6], Stable-Avatar [41], OmniAvatar [12], and WanS2V [13]. All methods are benchmarked at 720×400 on a single H800 node, with sequence-parallel methods using their maximum supported parallelism and all other settings kept at official defaults. Quantitative results on GenBench are presented in Table 2.

GenBench-ShortVideo. On short-form evaluation, our method achieves the best IQA and competitive ASE with OmniAvatar and Stable-Avatar, outperforming the rest. Although we build upon WanS2V and use step distillation, we surpass the teacher on visual quality; this aligns with the tendency of DMD-distilled models to concentrate the output distribution and attain slightly higher perceptual scores [39], and the DMD loss has been interpreted as a specialized reinforcement learning objective that effectively optimizes both aesthetic appeal and foundational visual fidelity [28, 30]. On Dino-S we obtain 0.96; Ditto’s 0.99 is expected as it is a face-editing model that retains the reference face more than a full video-generation pipeline. Audio-visual synchronization is comparable to OmniAvatar (Sync-C and Sync-D are both strong for both methods). The most pronounced gap is inference speed: we reach 45.2 FPS. Ditto is the only other

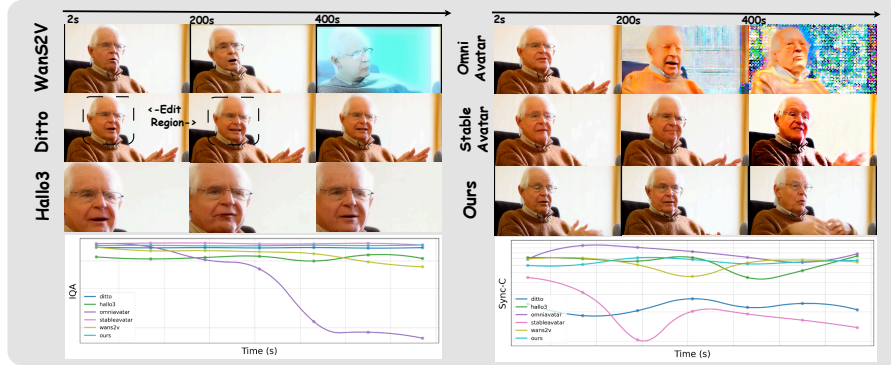


Fig. 4: Qualitative comparisons with state-of-the-art methods.

method with near real-time throughput (21.8 FPS); we use roughly $70\times$ its parameters yet reach 45.2 FPS, which underscores the effectiveness of DMD, TPP, and our kernel-level optimizations.

GenBench-LongVideo. Long-duration generation reveals a clear separation among methods. Echomimic-V2 is excluded as it does not support long-video inference. Comparing Short to Long results in Table 2, most generative baselines degrade noticeably: OmniAvatar drops from 3.53/4.49 to 2.36/2.86 (ASE/IQA), WanS2V from 3.36/4.29 to 2.63/3.99, and Hallo3 from 3.12/3.97 to 2.65/4.04. Ditto, as a face-editing model, maintains relatively stable quality but lags in audio-visual synchronization (Sync-C 3.98). Our method, by contrast, preserves near-identical quality across durations (ASE $3.44 \rightarrow 3.42$, IQA $4.51 \rightarrow 4.76$, Dino-S $0.96 \rightarrow 0.97$). OmniAvatar achieves the highest Sync-C/Sync-D on Long, but its severe visual collapse (Dino-S 0.66) suggests that loss of identity constraints artificially inflates sync scores. We achieve competitive synchronization (Sync-C 7.16) while maintaining the best overall visual and identity quality, yielding a more balanced profile across all metrics. Figure 4 corroborates this: most baselines, especially OmniAvatar and WanS2V, exhibit visible quality degradation over time, whereas ours maintains stable fidelity throughout.

Table 3: Ablation on Inference Efficiency. **Table 4:** Ablation on Long Video Generation (GenBench-LongVideo). SP_4 : sequence parallelism across 4 GPUs.

DMD	SP_4	TPP	VAE	Para.	Kern.	Opt.	#GPU	NFE	FPS $\uparrow$	TTFF $\downarrow$	Methods	Metrics			
												ASE $\uparrow$	IQA $\uparrow$	Sync-C $\uparrow$	Dino-S $\uparrow$
$\times$	$\times$	$\times$	$\times$	$\times$	$\times$		1	80	0.29	45.50	w/o AAS	3.13	4.68	7.25	0.96
$\checkmark$	$\times$	$\times$	$\times$	$\times$	$\times$		1	5	3.66	4.56	w/o Rolling RoPE	3.38	4.82	7.29	0.86
$\checkmark$	$\checkmark$	$\times$	$\times$	$\times$	$\times$		4	5	4.50	3.94	w/o History Corrupt	2.90	3.88	7.14	0.81
$\checkmark$	$\times$	$\checkmark$	$\times$	$\times$	$\times$		4	4	10.16	4.73	Ours	3.42	4.76	7.16	0.97
$\checkmark$	$\times$	$\checkmark$	$\checkmark$	$\times$	$\times$		5	4	20.88	2.89					
$\checkmark$	$\times$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$		5	4	45.2	1.21					

5.3 Ablation Study

Study of inference efficiency. Table 3 incrementally enables each component. DMD removes CFG and cuts NFE from 80 to 5 (4 denoising steps + 1 clean-cache pass needed by Self-Forcing [53]). SP_4 provides only marginal speedup since per-block sequences are short (3 latents). TPP more than doubles throughput by pipelining steps across GPUs and eliminating the clean-cache pass (NFE 5 $\rightarrow$ 4). VAE parallelization and kernel-level optimizations yield further gains, bringing the full system to 45.2 FPS / 1.21 s TTFF on 5 H800 GPUs.

Study on long-horizon stability strategies. As shown in Table 4 and Figure 5, each component targets a distinct failure mode. Removing History Corrupt causes the most severe degradation (ASE/IQA drop to 2.90/3.88, Dino-S to 0.81). Without AAS, progressive color desaturation emerges, with noticeably grayed-out frames (ASE drops to 3.13; Figure 5). Removing Rolling RoPE triggers identity drift (Dino-S 0.97 $\rightarrow$ 0.86), with visible changes in hair details and facial features (Figure 5).

Study on denoising step count. As shown in Table 5, increasing steps from 2 to 4 brings modest visual gains but markedly improves audio-visual synchronization (Sync-C 6.41 $\rightarrow$ 7.03), since fewer steps leave insufficient budget for early, motion-critical denoising stages. Thanks to TPP, FPS stays at 45.2 regardless of step count; only TTFF grows (0.68 s $\rightarrow$ 1.21 s), which remains practical for interactive streaming where end-to-end latency is dominated by network transport. We therefore default to 4 steps.

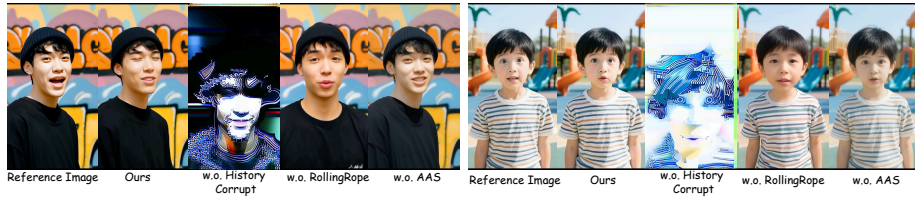
Table 5: Effect of denoising step count on GenBench-ShortVideo. TPP keeps FPS constant regardless of step count; TTFF scales with the number of steps.

Method	ASE $\uparrow$	IQA $\uparrow$	Sync-C $\uparrow$	Sync-D $\downarrow$	Dino-S $\uparrow$	TTFF (s) $\downarrow$	FPS $\uparrow$
Ours (2 steps)	3.37	4.25	6.41	9.02	0.94	0.68	45.2
Ours (3 steps)	3.41	4.36	6.58	8.85	0.95	0.95	45.2
Ours (4 steps)	3.44	4.51	7.03	8.30	0.96	1.21	45.2

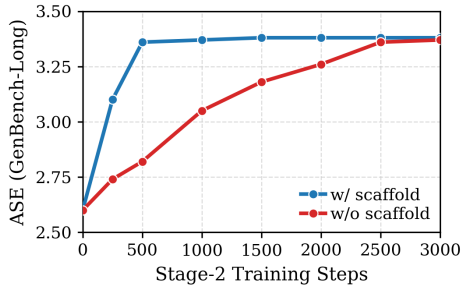
Study on noise level of KV cache. Table 6 compares three KV-cache noise schedules on GenBench-Long: clean-KV-cache, fix-noisy-KV-cache (noise fixed at $t=557$), and our timestep-forcing. Both baselines follow standard Self-Forcing inference [53]. We additionally report T.Flicker [19], which measures temporal consistency. Clean-KV-cache degrades rapidly over long horizons due to error accumulation. Fixed noise improves quality by decoupling identity and motion dynamics in the cache [36], but both baselines suffer from severe flickering (T.Flicker 0.876/0.891). Only timestep-forcing resolves this (T.Flicker 0.971) by maintaining step-matched caches. Furthermore, timestep-forcing is the only schedule compatible with TPP.

Table 6: Effect of KV-cache noise-level scheduling on GenBench-Long.

Setting	Metrics					
	ASE $\uparrow$	IQA $\uparrow$	Sync-C $\uparrow$	Sync-D $\downarrow$	Dino-S $\uparrow$	T.Flicker $\uparrow$
clean-kv-cache	3.05	4.44	6.11	9.10	0.90	0.876
fix-noisy-kv-cache	3.34	4.63	7.00	8.38	0.93	0.891
timestep-forcing	3.42	4.76	7.16	8.31	0.97	0.971

**Fig. 5:** Visual ablation of long-horizon stability components on long video generation.

Study on motion-frame scaffolding. Starting from the same 25 K-step Stage 1 with all other hyperparameters aligned, we compare Stage 2 convergence with and without motion-frame scaffolding (Figure 6). The scaffolded variant saturates at ~ 500 steps, while the baseline requires ~ 2500 steps—a $5\times$ speedup with no loss in final quality.

**Fig. 6:** Study of motion-frame scaffolding.

6 Conclusion

We present **Live Avatar**, an algorithm-system co-designed framework that, to our knowledge, is the first to enable practical real-time streaming of a 14-billion-parameter diffusion model for infinite-length audio-driven avatar generation. On the algorithm side, a two-stage training pipeline adapts a pretrained bidirectional model into a causal, few-step streaming one. Three complementary long-horizon strategies—History Corrupt, Adaptive Attention Sink, and Rolling RoPE—enable stable generation beyond 10,000 seconds³. On the system side, Timestep-forcing Pipeline Parallelism (TPP) assigns each GPU a fixed denoising timestep, converting the sequential diffusion chain into an asynchronous spatial pipeline that achieves 45 FPS with a TTFF of 1.21 s on 5 H800 GPUs. We also

³ We provide experimental results on 10,000-second generation in the supplementary material.

introduce GenBench, a long-form benchmark exceeding five minutes, to support reproducible evaluation.

Limitation. Our long-horizon stability strategies are grounded in the static-scene prior inherent to avatar interaction, where the subject and background remain largely unchanged; as native long-duration training becomes feasible, such priors may become unnecessary. Additionally, the current TTFF of 1.21 s, combined with network transport overhead, yields an end-to-end latency of approximately 3 s, which does not yet meet the stringent requirements of bidirectional real-time interaction.

References

1. Ao, T.: Body of her: A preliminary study on end-to-end humanoid agent. arXiv preprint arXiv:2408.02879 (2024)
2. Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., Ramesh, A.: Video generation models as world simulators (2024), <https://openai.com/research/video-generation-models-as-world-simulators>
3. Chen, B., Martí Monsó, D., Du, Y., Simchowitz, M., Tedrake, R., Sitzmann, V.: Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems* **37**, 24081–24125 (2024)
4. Chen, M., Cui, L., Zhang, W., Zhang, H., Zhou, Y., Li, X., Tang, S., Liu, J., Liao, B., Chen, H., et al.: Midas: Multimodal interactive digital-human synthesis via real-time autoregressive video generation. arXiv preprint arXiv:2508.19320 (2025)
5. Chung, J.S., Zisserman, A.: Out of time: automated lip sync in the wild. In: *Computer Vision–ACCV 2016 Workshops: ACCV 2016 International Workshops, Taipei, Taiwan, November 20–24, 2016, Revised Selected Papers, Part II* 13. pp. 251–263. Springer (2017)
6. Cui, J., Li, H., Zhan, Y., Shang, H., Cheng, K., Ma, Y., Mu, S., Zhou, H., Wang, J., Zhu, S.: Hallo3: Highly dynamic and realistic portrait image animation with video diffusion transformer. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. pp. 21086–21095 (2025)
7. Cui, J., Wu, J., Li, M., Yang, T., Li, X., Wang, R., Bai, A., Ban, Y., Hsieh, C.J.: Self-forcing++: Towards minute-scale high-quality video generation. arXiv preprint arXiv:2510.02283 (2025)
8. Du, F., Li, T., Zhang, Z., Qiao, Q., Yu, T., Zhen, D., Jia, X., Yang, Y., Yin, S., Liu, S.: Rap: Real-time audio-driven portrait animation with video diffusion transformer. arXiv preprint arXiv:2508.05115 (2025)
9. Du, Z., Wang, Y., Chen, Q., Shi, X., Lv, X., Zhao, T., Gao, Z., Yang, Y., Gao, C., Wang, H., Yu, F., Liu, H., Sheng, Z., Gu, Y., Deng, C., Wang, W., Zhang, S., Yan, Z., Zhou, J.: Cosyvoice 2: Scalable streaming speech synthesis with large language models (2024), <https://arxiv.org/abs/2412.10117>
10. Ephrat, A., Mosseri, I., Lang, O., Dekel, T., Wilson, K., Hassidim, A., Freeman, W.T., Rubinstein, M.: Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation. arXiv preprint arXiv:1804.03619 (2018)
11. Frans, K., Hafner, D., Levine, S., Abbeel, P.: One step diffusion via shortcut models. arXiv preprint arXiv:2410.12557 (2024)

12. Gan, Q., Yang, R., Zhu, J., Xue, S., Hoi, S.: Omniaavatar: Efficient audio-driven avatar video generation with adaptive body animation. arXiv preprint arXiv:2506.18866 (2025)
13. Gao, X., Hu, L., Hu, S., Huang, M., Ji, C., Meng, D., Qi, J., Qiao, P., Shen, Z., Song, Y., Sun, K., Tian, L., Wang, G., Wang, Q., Wang, Z., Xiao, J., Xu, S., Zhang, B., Zhang, P., Zhang, X., Zhang, Z., Zhou, J., Zhuo, L.: Wan-s2v: Audio-driven cinematic video generation (2025), <https://arxiv.org/abs/2508.18621>
14. Gelberg, Y., Eguchi, K., Akiba, T., Cetin, E.: Extending the context of pretrained llms by dropping their positional embeddings (2025), <https://arxiv.org/abs/2512.12167>
15. Guo, Y., Liu, X., Zhen, C., Yan, P., Wei, X.: Arig: Autoregressive interactive head generation for real-time conversations. arXiv preprint arXiv:2507.00472 (2025)
16. HaCohen, Y., Chiprut, N., Brazowski, B., Shalem, D., Moshe, D., Richardson, E., Levin, E., Shiran, G., Zabari, N., Gordon, O., et al.: Ltx-video: Realtime video latent diffusion. arXiv preprint arXiv:2501.00103 (2024)
17. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* **30** (2017)
18. Huang, X., Li, Z., He, G., Zhou, M., Shechtman, E.: Self forcing: Bridging the train-test gap in autoregressive video diffusion. arXiv preprint arXiv:2506.08009 (2025)
19. Huang, Z., He, Y., Yu, J., Zhang, F., Si, C., Jiang, Y., Zhang, Y., Wu, T., Jin, Q., Chanpaisit, N., Wang, Y., Chen, X., Wang, L., Lin, D., Qiao, Y., Liu, Z.: Vbench: Comprehensive benchmark suite for video generative models (2023), <https://arxiv.org/abs/2311.17982>
20. Kodaira, A., Hou, T., Hou, J., Tomizuka, M., Zhao, Y.: Streamdit: Real-time streaming text-to-video generation. arXiv preprint arXiv:2507.03745 (2025)
21. Kong, W., Tian, Q., Zhang, Z., Min, R., Dai, Z., Zhou, J., Xiong, J., Li, X., Wu, B., Zhang, J., et al.: Hunyuanvideo: A systematic framework for large video generative models. arXiv preprint arXiv:2412.03603 (2024)
22. Li, T., Tian, Y., Li, H., Deng, M., He, K.: Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems* **37**, 56424–56445 (2024)
23. Li, T., Zheng, R., Yang, M., Chen, J., Yang, M.: Ditto: Motion-space diffusion for controllable realtime talking head synthesis. arXiv preprint arXiv:2411.19509 (2024)
24. Lipman, Y., Chen, R.T., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling. arXiv preprint arXiv:2210.02747 (2022)
25. Liu, K., Hu, W., Xu, J., Shan, Y., Lu, S.: Rolling forcing: Autoregressive long video diffusion in real time. arXiv preprint arXiv:2509.25161 (2025)
26. Low, C., Wang, W.: Talkingmachines: Real-time audio-driven facetime-style video via autoregressive diffusion models. arXiv preprint arXiv:2506.03099 (2025)
27. Lu, C., Song, Y.: Simplifying, stabilizing and scaling continuous-time consistency models. arXiv preprint arXiv:2410.11081 (2024)
28. Luo, W.: Diff-instruct++: Training one-step text-to-image generator model to align with human preferences. arXiv preprint arXiv:2410.18881 (2024)
29. Luo, W., Hu, T., Zhang, S., Sun, J., Li, Z., Zhang, Z.: Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *Advances in Neural Information Processing Systems* **36**, 76525–76546 (2023)
30. Luo, Y., Hu, T., Sun, J., Cai, Y., Tang, J.: Learning few-step diffusion models by trajectory distribution matching. arXiv preprint arXiv:2503.06674 (2025)

31. Meng, D., Xiao, S., Zhang, X., Wang, G., Zhang, P., Wang, Q., Zhang, B., Bo, L.: Mirrorme: Towards realtime and high fidelity audio-driven halfbody animation. arXiv preprint arXiv:2506.22065 (2025)
32. Meng, R., Zhang, X., Li, Y., Ma, C.: Echomimicv2: Towards striking, simplified, and semi-body human animation (2025), <https://arxiv.org/abs/2411.10061>
33. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.Y., Li, S.W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jégou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: Dinov2: Learning robust visual features without supervision. Transactions on Machine Learning Research (2024), <https://arxiv.org/abs/2304.07193>
34. Prajwal, K., Mukhopadhyay, R., Namboodiri, V.P., Jawahar, C.: A lip sync expert is all you need for speech to lip generation in the wild. In: Proceedings of the 28th ACM international conference on multimedia. pp. 484–492 (2020)
35. Quignon, N., Chopin, B., Wang, Y., Dantcheva, A.: Theval. evaluation framework for talking head video generation (2025), <https://arxiv.org/abs/2511.04520>
36. Song, K., Chen, B., Simchowitz, M., Du, Y., Tedrake, R., Sitzmann, V.: History-guided video diffusion (2025), <https://arxiv.org/abs/2502.06764>
37. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models (2023)
38. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. CoRR **abs/1907.05600** (2019), <http://arxiv.org/abs/1907.05600>
39. Team, I., Cai, H., Cao, S., Du, R., Gao, P., Hoi, S., Hou, Z., Huang, S., Jiang, D., Jin, X., Li, L., Li, Z., Li, Z.Y., Liu, D., Liu, D., Shi, J., Wu, Q., Yu, F., Zhang, C., Zhang, S., Zhou, S.: Z-image: An efficient image generation foundation model with single-stream diffusion transformer (2025), <https://arxiv.org/abs/2511.22699>
40. Tian, L., Wang, Q., Zhang, B., Bo, L.: Emo: Emote portrait alive generating expressive portrait videos with audio2video diffusion model under weak conditions. In: European Conference on Computer Vision. pp. 244–260. Springer (2024)
41. Tu, S., Pan, Y., Huang, Y., Han, X., Xing, Z., Dai, Q., Luo, C., Wu, Z., Jiang, Y.G.: Stableavatar: Infinite-length audio-driven avatar video generation. arXiv preprint arXiv:2508.08248 (2025)
42. Unterthiner, T., Van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., Gelly, S.: Towards accurate generative models of video: A new metric & challenges. arXiv preprint arXiv:1812.01717 (2018)
43. Wan, T., Wang, A., Ai, B., Wen, B., Mao, C., Xie, C.W., Chen, D., Yu, F., Zhao, H., Yang, J., et al.: Wan: Open and advanced large-scale video generative models. arXiv preprint arXiv:2503.20314 (2025)
44. Wang, M., Wang, Q., Jiang, F., Fan, Y., Zhang, Y., Qi, Y., Zhao, K., Xu, M.: Fantasytalking: Realistic talking portrait generation via coherent motion synthesis. In: Proceedings of the 33rd ACM International Conference on Multimedia. pp. 9891–9900 (2025)
45. Wang, Z., Zhang, P., Qi, J., Xu, G.W.S., Zhang, B., Bo, L.: Omnitalker: Real-time text-driven talking head generation with in-context audio-visual style replication. arXiv e-prints pp. arXiv–2504 (2025)
46. Wu, C., Li, J., Zhou, J., Lin, J., Gao, K., Yan, K., ming Yin, S., Bai, S., Xu, X., Chen, Y., Chen, Y., Tang, Z., Zhang, Z., Wang, Z., Yang, A., Yu, B., Cheng, C., Liu, D., Li, D., Zhang, H., Meng, H., Wei, H., Ni, J., Chen, K., Cao, K., Peng, L., Qu, L., Wu, M., Wang, P., Yu, S., Wen, T., Feng, W., Xu, X., Wang, Y., Zhang, Y., Zhu, Y., Wu, Y., Cai, Y., Liu, Z.: Qwen-image technical report (2025), <https://arxiv.org/abs/2508.02324>

47. Wu, H., Zhang, Z., Zhang, W., Chen, C., Liao, L., Li, C., Gao, Y., Wang, A., Zhang, E., Sun, W., et al.: Q-align: Teaching lmms for visual scoring via discrete text-defined levels. arXiv preprint arXiv:2312.17090 (2023)
48. Xie, Y., Gu, T., Li, Z., Zhang, C., Song, G., Zhao, X., Liang, C., Jiang, J., Xu, H., Luo, L.: X-streamer: Unified human world modeling with audiovisual interaction. arXiv preprint arXiv:2509.21574 (2025)
49. Yang, S., Kong, Z., Gao, F., Cheng, M., Liu, X., Zhang, Y., Kang, Z., Luo, W., Cai, X., He, R., et al.: Infinitetalk: Audio-driven video generation for sparse-frame video dubbing. arXiv preprint arXiv:2508.14033 (2025)
50. Yang, S., Huang, W., Chu, R., Xiao, Y., Zhao, Y., Wang, X., Li, M., Xie, E., Chen, Y., Lu, Y., et al.: Longlive: Real-time interactive long video generation. arXiv preprint arXiv:2509.22622 (2025)
51. Yin, T., Gharbi, M., Park, T., Zhang, R., Shechtman, E., Durand, F., Freeman, B.: Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems* **37**, 47455–47487 (2024)
52. Yin, T., Gharbi, M., Zhang, R., Shechtman, E., Durand, F., Freeman, W.T., Park, T.: One-step diffusion with distribution matching distillation. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 6613–6623 (2024)
53. Yin, T., Zhang, Q., Zhang, R., Freeman, W.T., Durand, F., Shechtman, E., Huang, X.: From slow bidirectional to fast autoregressive video diffusion models. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. pp. 22963–22974 (2025)
54. Yu, H., Wang, Z., Pan, Y., Cheng, M., Yang, H., Wang, C., Xie, T., Xu, X., Wei, X., Cai, X.: Llia-enabling low-latency interactive avatars: Real-time audio-driven portrait video generation with diffusion models. arXiv preprint arXiv:2506.05806 (2025)
55. Zhang, W., Cun, X., Wang, X., Zhang, Y., Shen, X., Guo, Y., Shan, Y., Wang, F.: Sadtalker: Learning realistic 3d motion coefficients for stylized audio-driven single image talking face animation. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 8652–8661 (2023)
56. Zhen, D., Yin, S., Qin, S., Yi, H., Zhang, Z., Liu, S., Qi, G., Tao, M.: Teller: Real-time streaming audio-driven portrait animation with autoregressive motion generation. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. pp. 21075–21085 (2025)
57. Zheng, K., Wang, Y., Ma, Q., Chen, H., Zhang, J., Balaji, Y., Chen, J., Liu, M.Y., Zhu, J., Zhang, Q.: Large scale diffusion distillation via score-regularized continuous-time consistency. arXiv preprint arXiv:2510.08431 (2025)
58. Zhu, Y., Zhang, L., Rong, Z., Hu, T., Liang, S., Ge, Z.: Infp: Audio-driven interactive head generation in dyadic conversations. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. pp. 10667–10677 (2025)

A Overview of Supplementary Material

This supplementary document provides comprehensive details, additional experiments, and implementation specifics to support the findings in the main paper. The content is organized as follows:

- **Section C: Additional Experimental Results.** We evaluate our model’s long-horizon autoregressive extrapolation up to *10,000 s* (Table 7, Figure 7), provide additional comparison against state-of-the-art on the AV-Speech test set (Table 8), present a user study (Table 9, Figure 8), and ablate the effect of block size (Table 10).
- **Section D: Additional Evaluation Details.** We detail the hardware and inference configurations and runtime metric definitions used in our benchmarking.
- **Section E: Kernel-Level Optimizations.** We describe the kernel-level optimizations (referred to as *Kernel Opt.* in the efficiency ablation of the main paper) that enable real-time streaming inference, including FP8 quantization, FlashAttention-3, cuDNN fused attention, torch.compile, LoRA weight merging, streaming VAE decoding, and model offloading.
- **Section F: Algorithm Details.** We provide complete pseudocode (Algorithm 1, 2, 3, and 4) for our methods, including the *History Corrupt* and *Block-wise Gradient Accumulation* training strategies. We also provide detailed implementation descriptions and inference procedures for *AAS* and *Rolling RoPE*, as well as *TPP*. We further include visualizations (Figures 9, 10, and 11) to illustrate the mechanisms of different inference settings.
- **Section G: Additional Visual Results.** We showcase further qualitative examples to demonstrate the temporal consistency and visual fidelity of our method.
- **Section H: Ethical Consideration.** We discuss the potential societal impacts, privacy concerns, and responsible usage guidelines for our audio-driven avatar generation framework.

B LLM Usage Statement

We use LLMs (e.g., Gemini-2.5 and GPT-5) to polish our paragraphs.

C Additional Experimental Results

C.1 Extending Autoregressive Generation to 10,000 Seconds

To rigorously evaluate the long-horizon autoregressive capability of our model, we construct a stress test far exceeding the temporal range seen during training. Although the model is trained exclusively on 5-second clips—and its RoPE positions during training are randomly shifted only within a short-range window of a few minutes—we extend inference to an extreme 10,000-second horizon.

Each audio input in GenBench-LongVideo (≈ 7 minutes per sample) is looped to fill the full 10,000-second duration, ensuring continuous and valid audio conditioning throughout the sequence while avoiding silence gaps. The model then performs fully autoregressive, block-wise generation following Self-Forcing [18], relying entirely on accumulated KV caches and our long-horizon stability strategies (History Corrupt, AAS, and Rolling RoPE) throughout the 10,000-second rollout.

This setup intentionally exposes the model to RoPE indices over two orders of magnitude larger than those encountered in training (10k seconds corresponds to RoPE positions around 40k), a regime where existing methods typically suffer severe attention degradation, ID drift, or visual collapse. Self-Forcing++ [7] demonstrates video generation of roughly 4 minutes, representing the longest horizon reported in prior work. In contrast, our model shows no observable quality decay or identity instability over the entire 10,000-second sequence. As shown in Table 7, perceptual quality (ASE, IQA), audio-visual synchronization (Sync-C), and semantic consistency (Dino-S) remain nearly unchanged across segments sampled at 0–10s, 100–110s, 1000–1010s, and 10000–10010s. Figure 7 provides a representative case, demonstrating consistent identity and visual fidelity even at the 10k-second horizon.

Together, these results indicate that our long-video generation strategies—History Corrupt, Adaptive Attention Sink (AAS), and Rolling RoPE—allow the model to stably extrapolate far beyond its training regime, achieving an unprecedented 10,000-second autoregressive rollout without quality degradation.

Table 7: Evaluation of long-horizon temporal extrapolation at different time segments.

Methods	Metrics			
	ASE $\uparrow$	IQA $\uparrow$	Sync-C $\uparrow$	Dino-S $\uparrow$
0-10s	3.41	4.77	7.10	0.97
100-110s	3.43	4.75	7.22	0.96
1000-1010s	3.40	4.73	6.98	0.96
10000-10010s	3.42	4.76	7.14	0.96

C.2 Additional Comparison with Existing Methods

Although we have already provided comprehensive comparisons on GenBench in the main paper, we further evaluate the robustness of our method within its training domain, AV-Speech. Specifically, we hold out 5% of the original training videos and randomly sample 50 clips (5–10 seconds each) from this subset as an unseen test set. We report the same metrics used in the main evaluation; additionally, since ground-truth videos are available for this test set, we include FID and FVD to assess distribution alignment more thoroughly. The results are presented in Table 8.

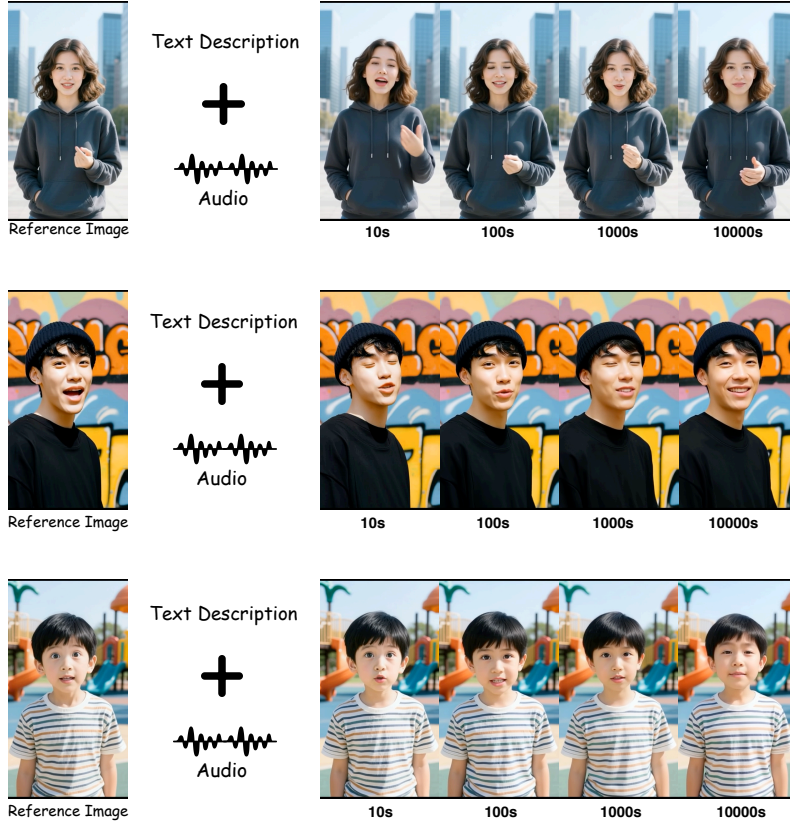


Fig. 7: Visualization of the generated video at 10 s, 100 s, 1000 s, and 10000 s, demonstrating the model’s strong capability in long-horizon temporal extrapolation.

The results in Table 8 show that our method achieves competitive or superior performance across most metrics on the in-domain AV-Speech evaluation. Notably, OmniAvatar is also trained on AV-Speech and therefore serves as a strong in-domain baseline, yet our method remains competitive under this setting. Together with the results reported on multiple public benchmarks in the main paper, this additional experiment verifies that our approach performs reliably within its training domain and alleviates potential concerns about relying solely on benchmark-specific evaluations.

C.3 User Study

Prior work such as THEval [35] has shown that popular metrics for talking-avatar evaluation (e.g., Sync-C) often diverge from human perception, as models can exploit them by exaggerating lip motion. To bridge this gap, we conduct a double-blind user study with 20 participants, who rate generated videos

Table 8: Quantitative comparisons on the in-domain AV-Speech test set.

Dataset	Model	Metrics							
		FID↓	FVD↓	ASE↑	IQA↑	Sync-C↑	Sync-D↓	Dino-S↑	FPS↑
AV-Speech	Ditto [23]	46.27	660.01	2.21	3.75	4.84	9.05	0.99	21.80
	Echomimic-V2 [32]	176.74	2059.81	1.88	3.29	4.07	9.38	0.78	0.53
	Hallo3 [6]	138.40	1412.93	1.87	3.35	4.50	9.99	0.93	0.26
	StableAvatar [41]	98.32	730.12	2.14	3.55	5.72	9.01	0.93	0.64
	OmniAvatar [12]	50.42	570.32	2.16	3.68	6.04	8.37	0.95	0.16
	WanS2V [13]	73.68	642.48	2.20	3.71	4.90	9.02	0.95	0.25
	Ours	48.91	502.37	2.30	3.88	6.21	8.31	0.95	45.2

from all methods on three perceptual dimensions: Naturalness, Synchronization, and Consistency. Here, Synchronization refers to the holistic audiovisual coherence [44]—including facial expressions, gestures, and posture transitions—rather than the frame-level lip alignment measured by Sync-C. The final scores are averaged across participants and normalized to a 0–100 scale for comparison. As summarized in Table 9, our method attains the highest Synchronization and Consistency scores. WanS2V achieves the best Naturalness, likely because its non-distilled diffusion backbone preserves smoother motion dynamics; we observe that our distilled model introduces slightly accelerated motion tempo, which marginally affects perceived naturalness but does not compromise synchronization or identity consistency. Figure 8 provides representative qualitative examples. OmniAvatar, despite achieving strong objective metrics, produces exaggerated motions that compromise identity preservation over time, leading to lower Naturalness ratings from human evaluators. EchoMimic V2, which relies on fixed hand-landmark templates, tends to ignore the body pose present in the reference image and instead generates a fixed default posture, causing severe visual distortion when the input depicts non-standard poses.

Table 9: User study results on perceptual evaluation (higher is better). Each score denotes the mean normalized user rating.

Model	Naturalness↑	Synchronization↑	Consistency↑
Ditto [23]	78.2	40.5	90.2
EchoMimic-V2 [32]	60.3	71.1	38.7
Hallo3 [6]	78.5	69.2	89.3
StableAvatar [41]	68.7	70.8	88.9
OmniAvatar [12]	71.1	78.5	90.8
WanS2V [13]	84.3	85.2	92.0
Ours	80.1	86.0	93.2

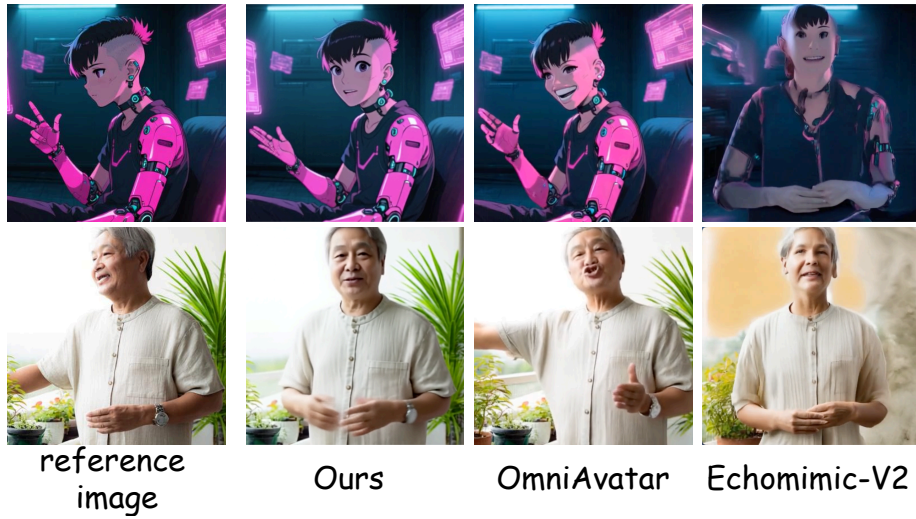


Fig. 8: Qualitative examples from the user study. OmniAvatar exhibits exaggerated motion with degraded identity fidelity, while EchoMimic V2 ignores the reference pose and produces a fixed body template, leading to visible distortion.

C.4 Effect of Block Size

Table 10: Effect of block size (number of latent frames per block) on GenBench-ShortVideo. All variants are trained end-to-end (Stage 1 + Stage 2) with identical hyperparameters.

Block Size	ASE $\uparrow$	IQA $\uparrow$	Sync-C $\uparrow$	Sync-D $\downarrow$	Dino-S $\uparrow$	TTFF (s) $\downarrow$
1 latent	3.30	4.35	4.12	10.41	0.96	0.40
2 latents	3.36	4.41	5.38	9.52	0.96	0.67
3 latents	3.44	4.49	7.03	8.30	0.96	0.94
4 latents	3.43	4.51	6.98	8.26	0.96	1.21

We ablate the number of latent frames per block while keeping all other training and inference hyperparameters fixed. As shown in Table 10, reducing the block size to 1 or 2 latents leads to a pronounced drop in audio-visual synchronization (Sync-C drops from 7.03 to 4.12 and 5.38, respectively), while visual quality metrics (ASE, IQA) degrade more mildly and identity consistency (Dino-S) remains essentially unchanged. Qualitatively, we observe that smaller block sizes produce videos with severely diminished motion dynamics—the generated avatar exhibits near-static facial expressions and minimal gesture variation—consistent with findings in CausVid [53], where single-frame autoregressive generation suffers from limited temporal expressiveness. Increasing the block size to

4 yields performance on par with 3—block size 4 slightly edges ahead on IQA and Sync-D, while block size 3 is marginally better on ASE and Sync-C—but incurs a noticeably higher TTFF (1.21 s vs. 0.94 s). We therefore adopt a block size of 3 as the default, which provides the best trade-off between generation quality and interactive latency.

D Additional Evaluation Details

Inference Configuration. All methods are evaluated on a single H800 node under identical hardware conditions. To ensure fair comparison, we utilize multi-GPU parallel inference for all methods where the official open-source code provides appropriate scripts. For methods that lack support for parallel acceleration, specifically EchoMimic V2 and Ditto, inference is conducted on a single H800 GPU. Regarding resolution, we use a fixed resolution of 720×400 for all models except Hallo3, which does not support arbitrary aspect ratios or resolutions. For Hallo3, we crop both the input and the ground-truth frames to 512×512 . Furthermore, EchoMimic V2 is excluded from the long-video comparative experiments, as its reliance on per-frame skeleton templates prevents it from effectively performing long-duration inference.

Runtime Metrics. For runtime evaluation, we report two key efficiency metrics. **FPS (Frames Per Second)** is measured from the moment the inference pipeline is initialized and includes the full end-to-end cost: the diffusion model’s denoising time, VAE decoding time, and any additional CPU-side processing. **Time-to-First-Frame (TTFF)** accounts for the total latency from audio signal arrival to visual output, calculated as the sum of (i) the random arrival latency—the waiting time between the arrival of an audio interaction signal and the next frame boundary—followed by (ii) the full denoising latency of the first frame and (iii) its VAE decoding cost. Note that random arrival latency depends on the output frame rate, and thus TTFF is inherently coupled with the FPS of the method. All FPS measurements for all methods are evaluated on the GenBench-LongVideo benchmark, where long-sequence testing provides more stable and accurate runtime estimation.

E Kernel-Level Optimizations

This section details the kernel-level optimizations referred to as *Kernel Opt.* in the efficiency ablation study of the main paper (Sec. 4.2). To enable real-time streaming inference with a 14B-parameter diffusion transformer, we apply a set of optimizations spanning quantization, attention kernels, graph compilation, weight merging, and VAE decoding.

FP8 Quantization. All linear projections in the DiT are quantized to FP8 (E4M3) with per-tensor dynamic scaling. Embedding layers, the output head, and the audio encoder’s final projection are kept in FP16 to preserve numerical stability. This reduces per-GPU VRAM from ~ 80 GB to under 48 GB, enabling deployment on single 48 GB GPUs. To verify that FP8 quantization does not

degrade generation quality, we compare the quantized model against its FP16 counterpart on both GenBench subsets (Table 11). Disabling FP8 yields only marginal quality differences—IQA improves slightly while other metrics remain virtually unchanged—yet throughput drops from 45.2 to 36.0 FPS. We therefore adopt FP8 as the default, and all metrics reported in the main paper use this configuration.

Table 11: Effect of FP8 quantization on generation quality and throughput.

Dataset	Setting	ASE↑	IQA↑	Sync-C↑	Sync-D↓	Dino-S↑	FPS↑
GenBench-Short	w/o FP8 (FP16)	3.45	4.55	7.02	8.28	0.96	36.0
	Ours (FP8)	3.44	4.51	7.03	8.30	0.96	45.2
GenBench-Long	w/o FP8 (FP16)	3.42	4.79	7.16	8.30	0.97	36.0
	Ours (FP8)	3.42	4.76	7.16	8.31	0.97	45.2

FlashAttention-3 and cuDNN Fused Attention. We adopt a tiered attention dispatch strategy. On Hopper-class GPUs, FlashAttention-3 with variable-length sequence support serves as the primary kernel, efficiently handling the concatenation of cached KV entries and current tokens in streaming inference. FlashAttention-2 is used as a fallback on older architectures. Where applicable, we further dispatch to a fused cuDNN scaled dot-product kernel that computes the entire QKV attention in a single pass, avoiding materialization of the full attention matrix.

Graph Compilation. We apply PyTorch’s ahead-of-time graph compilation to the DiT forward pass, RoPE computation, and streaming VAE decoding. This enables automatic operator fusion (e.g., fusing RoPE element-wise operations, layer-norm with subsequent linear projections) and kernel auto-tuning across the denoising loop.

LoRA Weight Merging. LoRA adapters trained during distillation are permanently merged into the base model weights before inference, eliminating the runtime overhead of maintaining and computing through separate low-rank branches.

Streaming VAE Feature Caching. We implement a causal VAE decoder that processes latents frame-by-frame while maintaining a temporal feature cache: each causal convolution layer retains the activations of the most recent 2 frames, which are prepended as causal context when decoding the next frame, enabling incremental decoding without re-computation. In the multi-GPU TPP configuration, a dedicated GPU runs the streaming VAE in parallel with the DiT—decoded blocks are sent via point-to-point communication as soon as they are produced, effectively hiding VAE latency behind DiT compute.

Model Offloading. For single-GPU deployment, the text encoder, audio encoder, and VAE are offloaded to CPU after their respective encoding phases,

freeing VRAM for the DiT denoising loop. KV caches can optionally be offloaded to CPU between clips to further reduce peak memory.

Together, these optimizations yield $\sim 2.5\times$ peak and $\sim 3\times$ average FPS improvement over the unoptimized baseline, achieving stable 45+ FPS real-time streaming on $5\times H800$ with 4-step sampling. On a more cost-effective $5\times H20$ configuration, the system still delivers 18 FPS end-to-end with a TTFF of 3.12s, demonstrating practical deployability beyond high-end datacenter hardware.

F Algorithm Details

We provide detailed pseudocode for completeness and reproducibility.

Training. Our two-stage training framework is described as follows. In Stage 1 (Diffusion Forcing Pretraining), we apply block-wise independent noise scheduling and causal attention, with motion frames serving as scaffolding to bootstrap long-horizon temporal modeling (see main text Sec. 3.2). In Stage 2 (Self-Forcing Distillation), the motion frames are removed and replaced by the rolling KV cache. As shown in Algorithm 1, the self-forcing DMD training follows [18] but removes the additional forward pass used to refresh the KV cache after denoising. This ensures that the model is consistently trained with noisy KV states, aligning the training process with the actual autoregressive inference and improving robustness to error accumulation—a strategy we call **History Corrupt**. Due to the substantial memory footprint of DMD training, we additionally implement a lightweight memory-reduction strategy using **block-wise gradient accumulation** (Algorithm 2), which partitions the backward graph by blocks and accumulates the resulting gradients across multiple steps, preserving training behavior while significantly reducing peak memory usage and enabling even single-node H800 training.

Inference. The single-GPU inference procedure is provided in Algorithm 3. It builds upon the rolling-KV-cache inference algorithm from [18] with the addition of AAS and Rolling RoPE. Below we detail the concrete implementation of these two mechanisms.

Adaptive Attention Sink (AAS). As described in the main text, the user-provided reference image initially serves as the sink frame. To eliminate the distributional mismatch between real conditioning and model-generated content, AAS replaces the sink frame with the model’s own first generated *latent* (i.e., the denoised output of the first block) immediately after the first block is produced. Crucially, this replacement operates entirely in latent space—no additional VAE encoding or decoding is required—keeping the sink frame on the model’s generation manifold with negligible overhead. The updated sink frame then persists as the sole identity anchor for all subsequent blocks (see Algorithm 3, line 16–17; Algorithm 4, line 13–14 and 18–19).

Rolling RoPE. During standard autoregressive rollout, the sink frame’s RoPE index remains fixed at position 0 while the generated blocks’ indices grow monotonically (e.g., 1, 2, 3, ...). As generation progresses, the relative positional distance between the sink frame and the current block becomes arbitrar-

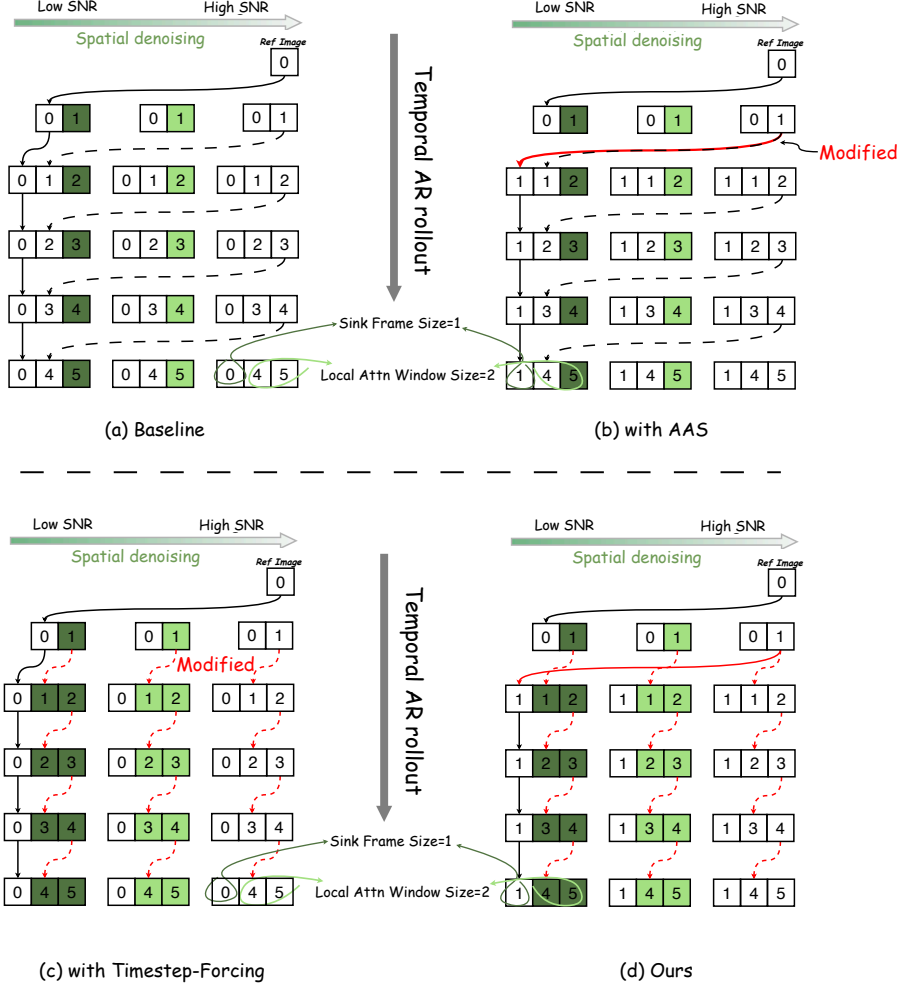


Fig. 9: Illustration of different inference settings. Horizontally, each row follows the spatial denoising order from low to high SNR; vertically, each column shows the autoregressive rollout over frames. Each small **rectangle** denotes the latent of a block, and the **number** inside represents its block index. **Solid arrows** indicate direct sink frame passing, whereas **dashed arrows** indicate KV-cache passing. **Red marks** indicate the components modified relative to the baseline. (a) Baseline with a fixed sink frame and standard rolling-kv-cache. (b) **AAS** with the sink replaced by the first generated latent. (c) **Timestep-forcing** with each noisy latent attending only to KV caches of the same timestep. (d) **Ours** with both AAS and timestep-forcing.

ily large—far exceeding positions seen during training—causing attention to the sink frame to degrade. Rolling RoPE addresses this by dynamically reassigning the sink frame’s RoPE index at every block step: specifically, the sink frame’s

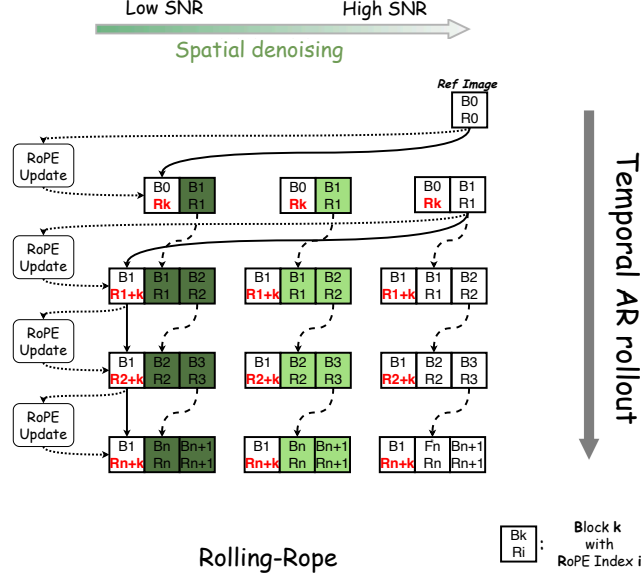


Fig. 10: Visualization of the proposed Rolling-RoPE mechanism. Horizontally, each row follows the spatial denoising order from low to high SNR; vertically, each column shows the autoregressive rollout over frames. Each small **rectangle** denotes the latent of a block, and the **number** inside represents its block index and its RoPE index, respectively. **Red** marks indicate the components modified relative to the baseline. **Solid arrows** denote sink-frame passing, **sparse dashed arrows** denote standard KV-cache passing, and **dense dashed arrows** indicate RoPE updates, where each block is re-assigned updated positional embeddings. Rolling-RoPE dynamically **increases the RoPE index** of the sink frame throughout AR rollout, keeping the sink frame’s RoPE index slightly larger than that of the current noisy block, ensuring a stable and appropriate relative positional distance throughout AR rollout.

index is set to be slightly ahead of the current noisy block’s index (within the training-time offset range), ensuring that the relative positional distance remains bounded and consistent with training throughout the entire rollout. This update is applied to the sink frame’s cached KV entries by recomputing RoPE embeddings in place, denoted as $\Phi(\mathbf{Sink})$ in Algorithms 3 and 4. Figure 10 illustrates the RoPE update performed during autoregressive generation, where the sink frame is continuously re-assigned with updated positional embeddings.

The multi-GPU TPP inference is detailed in Algorithm 4, which outlines the parallel execution procedure with minimal computation overlap and communication overhead. Figure 11 further visualizes the computation and waiting time of each GPU. After the initial warmup (including a second warmup required for AAS), the majority of each GPU’s time is devoted to DiT computation (red), demonstrating high utilization and stable frame rates.

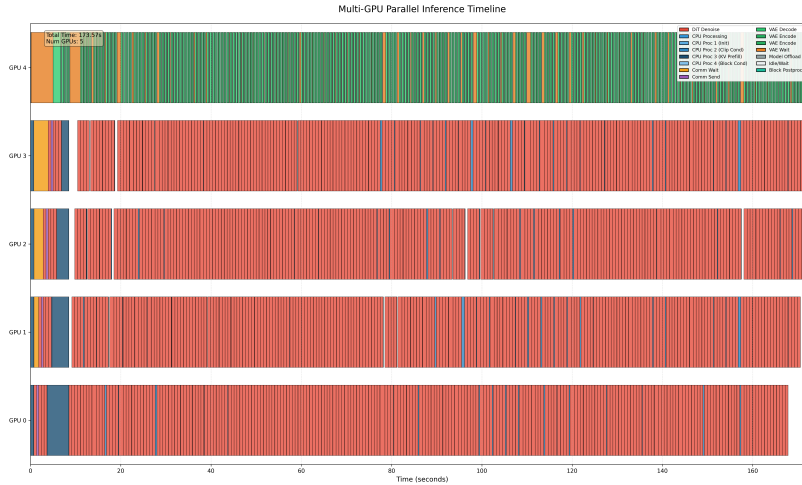


Fig. 11: Multi-GPU Parallel Inference Timeline. This chart visualizes the computation and waiting periods for each GPU. The two distinct **white** gaps on the left represent the initial warmup phase (including the secondary warmup for AAS). Following these, the majority of the processing time is dedicated to DiT computation (shown in **red**), reflecting high utilization and stable frame rates. Sporadic **white** gaps (idle time) appearing thereafter are present due to rate fluctuations, but their rarity ensures a negligible impact on performance.

G Additional Visual Results

We provide additional qualitative examples to further illustrate the model’s long-horizon generation capability. As shown in Figure 7, our method maintains stable identity, consistent lip movements, and coherent visual appearance when extrapolating videos far beyond the training horizon. For three different subjects, the generated frames at 10 s, 100 s, 1000 s, and 10000 s remain visually aligned with the reference image and follow the audio-driven motion patterns without exhibiting temporal drift or degradation. These results highlight the robustness of our approach in producing coherent long-duration talking-face videos.

H Ethical Consideration

Our work focuses on enabling real-time and long-horizon audio-driven avatar generation, which naturally raises concerns related to privacy, consent, and potential misuse. All identity data used in training and evaluation is collected with permission, and our method does not store or reconstruct unauthorized identities. While high-fidelity avatars may be susceptible to impersonation risks, our system is intended solely for legitimate telepresence and interactive applications. We encourage responsible deployment practices such as access control and watermarking to prevent malicious use.

Algorithm 1 Self-Forcing DMD with History Corrupt

Require: Timesteps $\{t_1, \dots, t_T\}$
Require: Number of video frames N
Require: Conditions of N frames $C_{1:N}$ (including audio, text, ref image)
Require: Generator G_θ (returns KV embeddings via G_θ^{KV})

```

1: loop
2:   Initialize model output  $\mathbf{X}_\theta \leftarrow []$ 
3:   Initialize KV cache  $\mathbf{KV} \leftarrow []$ 
4:   Sample  $s \sim \text{Uniform}(1, 2, \dots, T)$ 
5:   for  $i = 1, \dots, N$  do
6:     Initialize  $x_{t_T}^i \sim \mathcal{N}(0, I)$ 
7:     for  $j = T, \dots, s$  do
8:       if  $j = s$  then
9:         Enable gradient computation
10:        Set  $\mathbf{kv}^i, \hat{x}_0^i \leftarrow G_\theta^{\text{KV}}(x_{t_j}^i; t_j, \mathbf{KV}, C_i)$ 
11:         $\mathbf{X}_\theta.\text{append}(\hat{x}_0^i)$ 
12:        Detach  $\mathbf{kv}^i$  from gradient graph
13:         $\mathbf{KV}.\text{append}(\mathbf{kv}^i)$  ▷ Noisy KV Cache
14:      else
15:        Disable gradient computation
16:        Set  $\hat{x}_0^i \leftarrow G_\theta(x_{t_j}^i; t_j, \mathbf{KV}, C_i)$ 
17:        Sample  $\epsilon \sim \mathcal{N}(0, I)$ 
18:        Set  $x_{t_{j-1}}^i \leftarrow \Psi(\hat{x}_0^i, \epsilon, t_{j-1})$ 
19:      end if
20:    end for
21:  end for
22:  Update  $\theta$  via distribution matching loss
23: end loop
  
```

Algorithm 2 Self-Forcing DMD with History Corrupt and Block-wise Gradient Accumulation

Require: Timesteps $\{t_1, \dots, t_T\}$
Require: Number of video frames N
Require: Conditions of N frames $C_{1:N}$ (including audio, text, ref image)
Require: Generator G_θ (extra returns KV embeddings via G_θ^{KV})

```

1: loop
2:   Initialize model output  $\mathbf{X}_\theta \leftarrow []$ 
3:   Initialize noisy latent cache  $\mathbf{X}_{\text{cache}} \leftarrow []$ 
4:   Initialize KV cache  $\mathbf{KV} \leftarrow []$ 
5:   Sample  $s \sim \text{Uniform}(1, 2, \dots, T)$ 
6:   Disable gradient computation
7:   for  $i = 1, \dots, N$  do
8:     Initialize  $x_{t_T}^i \sim \mathcal{N}(0, I)$ 
9:     for  $j = T, \dots, s$  do
10:      if  $j = s$  then
11:        Set  $\mathbf{kv}^i, \hat{x}_0^i \leftarrow G_\theta^{\text{KV}}(x_{t_j}^i; t_j, \mathbf{KV}, C_i)$ 
12:        Detach  $x_{t_j}^i, \hat{x}_0^i, \mathbf{kv}^i$  from gradient graph
13:         $\mathbf{X}_{\text{cache}}.append(x_{t_j}^i)$ 
14:         $\mathbf{X}_\theta.append(\hat{x}_0^i)$ 
15:         $\mathbf{KV}.append(\mathbf{kv}^i)$ 
16:      else
17:        Set  $\hat{x}_0^i \leftarrow G_\theta(x_{t_j}^i; t_j, \mathbf{KV}, C_i)$ 
18:        Sample  $\epsilon \sim \mathcal{N}(0, I)$ 
19:        Set  $x_{t_{j-1}}^i \leftarrow \Psi(\hat{x}_0^i, \epsilon, t_{j-1})$ 
20:      end if
21:    end for
22:  end for
23:  for  $i = N, \dots, 1$  do
24:    Initialize temporary model output  $\mathbf{X}_i \leftarrow []$ 
25:    for  $j = 1, \dots, N$  do
26:      if  $i = j$  then
27:        Set  $x_{t_s}^i \leftarrow \mathbf{X}_{\text{cache}}[j]$ 
28:        Enable gradient computation
29:        Set  $\hat{x}_0^i \leftarrow G_\theta(x_{t_s}^i; t_s, \mathbf{KV}, C_i)$ 
30:         $\mathbf{X}_i.append(\hat{x}_0^i)$  ▷ Handle Partial Gradient
31:        Disable gradient computation
32:      else
33:         $\mathbf{X}_i.append(\mathbf{X}_\theta[j])$ 
34:      end if
35:    end for
36:    Accumulate gradient of  $\theta$  via DMD loss
37:     $\mathbf{KV}.pop(i)$  ▷ Free Memory
38:  end for
39:  Update  $\theta$ 
40: end loop
  
```

Algorithm 3 Single-GPU AR Inference with AAS and Rolling RoPE

Require: Per-timestep KV caches, each with size L
Require: Timesteps $\{t_1, \dots, t_T\}$
Require: Number of generated frames M
Require: Conditions of N frames $C_{1:N}$ (including audio, text)
Require: Ref image R
Require: Flow-Matching Model v_θ^{KV} (extra returns KV embeddings)
Require: Rolling RoPE transform $\Phi(\cdot)$
Require: VAE Decoder $\text{VAE}(\cdot)$

- 1: Initialize model output $\mathbf{X}_\theta \leftarrow []$
- 2: Initialize KV caches $\{\mathbf{KV}_1, \dots, \mathbf{KV}_T\} \leftarrow []$
- 3: Initialize Rolling Sink Frame $\text{Sink} \leftarrow R$
- 4: Initialize $dt \leftarrow -1/T$
- 5: **for** $i = 1, \dots, M$ **do**
- 6: Initialize $x^i \sim \mathcal{N}(0, I)$
- 7: **for** $j = T, \dots, 1$ **do**
- 8: Set $\hat{v}_j^i, \mathbf{kv}_j^i \leftarrow v_\theta^{\text{KV}}(x^i; t_j, \mathbf{KV}_j, C_i, \Phi(\text{Sink}))$ ▷ RoPE Update
- 9: Set $x^i \leftarrow x^i + \hat{v}_j^i dt$
- 10: **if** $|\mathbf{KV}_j| = L$ **then**
- 11: $\mathbf{KV}_j.\text{pop}(0)$
- 12: **end if**
- 13: $\mathbf{KV}_j.\text{append}(\mathbf{kv}_j^i)$
- 14: **end for**
- 15: $\mathbf{X}_\theta.\text{append}(\text{VAE}(x^i))$
- 16: **if** $i = 1$ **then**
- 17: $\text{Sink} \leftarrow x^i$ ▷ AAS Update
- 18: **end if**
- 19: **end for**
- 20: **return** $\mathbf{X}_\theta$

Algorithm 4 TPP with AAS and Rolling RoPE

Require: GPU Index k
Require: Per-timestep KV caches, each with size L
Require: Timesteps $\{t_1, \dots, t_T\}$
Require: Number of generated frames M
Require: Conditions of N frames $C_{1:N}$ (including audio, text)
Require: Ref image R
Require: Flow-Matching Model v_θ^{KV} (extra returns KV embeddings)
Require: Rolling RoPE transform $\Phi(\cdot)$
Require: VAE Decoder $\text{VAE}(\cdot)$

- 1: Initialize model output $\mathbf{X}_\theta \leftarrow []$
- 2: Initialize KV cache $\mathbf{KV} \leftarrow []$
- 3: Initialize Rolling Sink Frame $\mathbf{Sink} \leftarrow R$
- 4: Initialize $dt \leftarrow -1/T$
- 5: **for** $i = 1, \dots, M$ **do**
- 6: **if** $k=1$ **then**
- 7: Initialize $x^i \sim \mathcal{N}(0, I)$
- 8: **else**
- 9: $x^i \leftarrow \text{dist.recv}()$
- 10: **end if**
- 11: **if** $k=T+1$ **then** ▷ VAE Device
- 12: $\mathbf{X}_\theta.\text{append}(\text{VAE}(x^i))$
- 13: **if** $i = 1$ **then**
- 14: $\text{dist.broadcast}(x^i)$ ▷ Broadcast AAS Latent
- 15: **end if**
- 16: **continue**
- 17: **else** ▷ DiT Device
- 18: **if** $i=2$ **then**
- 19: $\mathbf{Sink} \leftarrow \text{dist.recv}()$ ▷ Update AAS
- 20: **end if**
- 21: Set $\hat{v}_k^i, \mathbf{kv}^i \leftarrow v_\theta^{\text{KV}}(x^i; t_{T-k+1}, \mathbf{KV}, C_i, \Phi(\mathbf{Sink}))$
- 22: **if** $|\mathbf{KV}| = L$ **then**
- 23: $\mathbf{KV}.\text{pop}(0)$
- 24: **end if**
- 25: $\mathbf{KV}.\text{append}(\mathbf{kv}^i)$
- 26: Set $x^i \leftarrow x^i + \hat{v}_k^i dt$
- 27: $\text{dist.send}(x^i)$
- 28: **end if**
- 29: **end for**
- 30: **return** $\mathbf{X}_\theta$
