BLOCKSTEP: ACCELERATING AUTOREGRESSIVE VIDEO DIFFUSION VIA ONLINE MULTI-MODEL ROUTING ACROSS BLOCKS AND STEPS

Anonymous authors
Paper under double-blind review



Figure 1: **Visual comparison of target vs Blockstep vs draft on a VBench prompt.** Here the target model and BlockStep properly follow the prompt and the draft model fails, rendering knitting instead of tying a tie. BlockStep matches the target model's generation quality (indicated by VBench total score) while achieving 2.37x speedup

ABSTRACT

Autoregressive video diffusion has become a compelling paradigm for streaming video synthesis, with step distillation as its standard path to fast inference. Yet step distillation reduces only the number of steps, not the size of the model run at each one: every step still invokes the full-size network – tens of billions of parameters – and replacing it with a smaller model can yield significant quality degradation. We introduce BlockStep, an online multi-model router: for each prompt, it routes every block and denoising step to a model of the appropriate size – the full-size base model or a smaller model distilled to the same few-step regime – invoking the base model only where it is needed to maintain quality while maximizing speedup. Unlike the leading autoregressive router, which scores each block with an external image-reward model – a costly extra network and an imperfect proxy for video quality – BlockStep does not rely on an extra reward model and is training-free. Specifically, BlockStep consists of three components: (1) a fixed block-0 anchor runs the base model on the first block, recovering most of the quality gap to the smaller model; (2) an online, block-level router upgrades selected blocks to the base model, based on the smaller model's attention entropy; and (3) within each upgraded block, a step-level schedule is selected based on block position. On Krea Realtime 14B with Wan2.1-1.3B Self-Forcing as the smaller model, BlockStep matches the base model's VBench total score at 2.37x wall-clock speedup.

1 INTRODUCTION

Autoregressive video diffusion has enabled real-time video generation (Huang et al., 2026; Chen et al., 2026c; Yin et al., 2025), with potential real-world applications including interactive live-streaming, world modeling, and robot learning (Huang et al., 2026; Feng et al., 2026; Liu et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

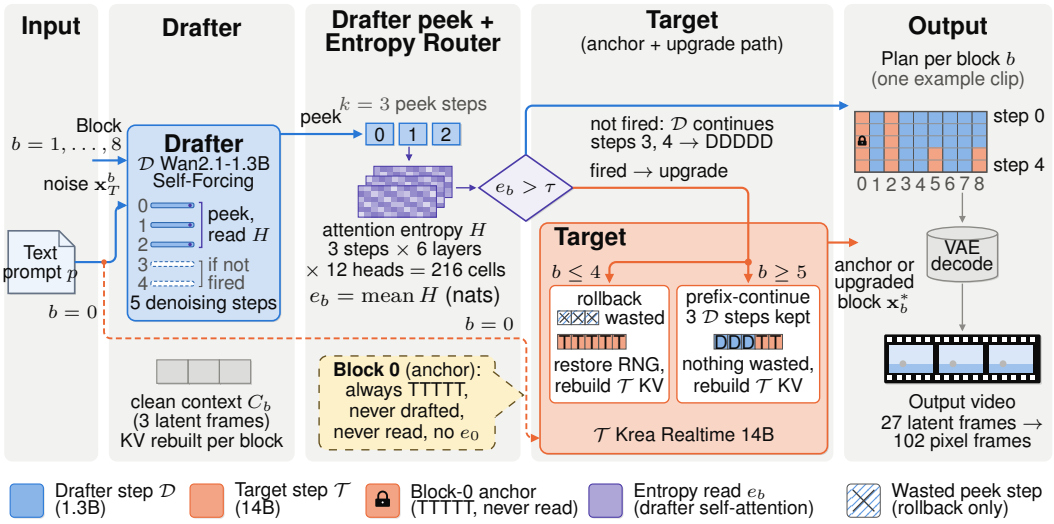


Figure 2: **BlockStep inference pipeline.** A clip is 9 blocks of 5 denoising steps; a block's plan is a 5-character string over the drafter $\mathcal{D}$ (Wan2.1-1.3B Self-Forcing, blue) and the target $\mathcal{T}$ (Krea Realtime 14B, orange).

2026b; Valevski et al., 2024; Yin et al., 2025). However, models exhibit a strict speed-quality trade-off based on their capacity (Bandyopadhyay et al., 2026). Larger models such as Krea-Realtime-14B (Millon, 2025) generate high quality videos at 11fps on a B200 with reduced inference steps, below the 16fps frame rate of the videos it generates, whereas smaller models like Wan2.1-T2V-1.3B Self-Forcing (Huang et al., 2026) achieve $3.00\times$ faster generation at the cost of an 1.56 drop (1.88%) in VBench total score (Huang et al., 2024). Conveniently, such video families are released at several sizes that share a latent space—for instance Wan2.1 at 14B and 1.3B—so a smaller same-family drafter is available off the shelf, requiring no extra training (Hu & Zhang, 2026). An important question is whether larger models can utilize the speedup of smaller models while maintaining their quality.

Block-causal autoregressive video diffusion has two levels of sequential computation: the denoising steps within a block to generate a clean latent, and the autoregressive block generation which conditions each block on preceding blocks (Yin et al., 2025; Huang et al., 2026; Millon, 2025; Chen et al., 2026c). Standard diffusion methods enforce uniform model capacity at both levels, each diffusion step of each block is done by a single model (Pan et al., 2025; Liu et al., 2023; Song et al., 2025). Lossy speculative decoding relaxes this uniform model capacity for autoregressive video diffusion using a larger target model and smaller draft model to achieve speedup at the cost of minor quality degradation (Hu & Zhang, 2026). Multi-model samplers relax step uniformity for image and non-autoregressive video diffusion similarly for speedup at the cost of minor quality degradation (Pan et al., 2025; Liu et al., 2023; Song et al., 2025).

To the best of our knowledge, no prior method relaxes both levels of uniformity through multi-model sampling to achieve speedup in autoregressive video diffusion, using the target model selectively across blocks and steps to maintain quality and generating the rest of the blocks and steps with a draft model for speedup. Such a block- and step-wise multi-model sampling algorithm requires three components: a step schedule for the pivotal first block, a method to select which blocks require the target model, and a step schedule for those upgraded blocks. Block-level routing has one prior instance (Hu & Zhang, 2026); step-level multi-model scheduling and the combination are, to our knowledge, untested for autoregressive video diffusion.

We propose BlockStep, to our knowledge the first block- and step-wise multi-model algorithm for autoregressive video diffusion. BlockStep has three primary components:

- **Block-0 anchor.** The first block is generated without context and conditions every later block, and it carries a disproportionate share of the target's value, so BlockStep always generates the first block using the target model.
- **Block-level routing.** For all subsequent blocks, an online block-level upgrade rule determines whether a block needs to be generated using the target model. BlockStep partially drafts every block after the anchor and upgrades a block when the drafter's own attention entropy, read part-way through its denoising steps, exceeds a fixed threshold.
- **Step-level schedules.** Upgraded blocks either continue denoising from the partially drafted blocks with the target model or are fully regenerated by the target model based on block position. Earlier blocks condition subsequent blocks and are critical for the dynamic degree of the entire clip, so they are given a target heavier upgrade schedule than later blocks.

Ported into the official Krea codebase (krea-ai/realtime-video) (Millon, 2025), BlockStep with Krea Realtime 14B as target and Wan2.1-T2V-1.3B Self-Forcing (Wan et al., 2025; Huang et al., 2026) as drafter matches the all-14B sampler's VBench total score while achieving $2.37\times$ wall-clock speedup.

2 Related Work

Video Diffusion. The current SOTA video generation models are bidirectional video diffusion transformers which denoise all their latent frames jointly (Wan et al., 2025; Kong et al., 2024; Yang et al., 2025; Polyak et al., 2024). These bidirectional models produce high quality videos, but must fully denoise all the latent frames before showing the first frame, limiting their applicability to long-range generation and video streaming (Huang et al., 2026; Yin et al., 2025).

To address this, other work makes these models autoregressive, generating block-by-block with a persistent KV cache (Chen et al., 2024a; Yin et al., 2025; Chen et al., 2025; Teng et al., 2025) and typically distilling a bidirectional base into a causal few-step generator—via distribution-matching (Yin et al., 2023; 2024; Zhou et al., 2024), progressive or consistency (Salimans & Ho, 2022; Song et al., 2023), or adversarial (Lin et al., 2025; Li et al., 2026) distillation. Self-Forcing (Huang et al., 2026), the drafter we build on, conditions each frame on its own previously generated outputs through key-value caching during training. A separate line extends such models to longer horizons (Chen et al., 2026c; Liu et al., 2025; Cui et al., 2025), but targets temporal drift rather than latency—a different objective from ours.

Block-causal autoregressive video diffusion thus has two sequential axes of computation: the denoising steps within each block that turn noise into a clean latent, and the autoregressive generation across blocks that conditions each block on the clean latents of preceding blocks. Our method operates on both axes, balancing quality and speedup along each.

Reducing Compute in Diffusion Denoising. A large body of work reduces the compute a model spends across denoising steps, but each line either assumes a long denoising schedule that our few-step setting does not have, or requires substantial retraining. Feature-caching methods reuse cached intermediate computations from earlier steps when adjacent steps change little, on full-sequence (bidirectional) diffusion (Liu et al., 2024; Zhao et al., 2024; Selvaraju et al., 2024; Chen et al., 2024b; Ma et al., 2023; Kahatapitiya et al., 2024; Ma et al., 2024); RhymeFlow (Dai et al., 2026) skips denoising steps for non-key frames; and SCOPE (Cui et al., 2026) brings such reuse to causal video with a per-step cache/predict/recompute scheduler. All of these assume a denoising-step budget large enough to cut—typically 50 or more steps (for example, 64 for MAGI-1)—in which neighbouring steps are redundant. Separately, DSA (Le et al., 2026), HiStream+ (Qiu et al., 2026), and Diagonal Distillation (Liu et al., 2026a) learn asymmetric per-block step schedules, and AdaDiff (Tang et al., 2023) an early-exit head for image diffusion; these add a dedicated training stage. Closest to our setting, Surprise Forcing (Shi et al., 2026) instead varies the per-block step count training-free, from adjacent-frame latent change after one step.

None of these lines, however, is a suitable baseline in our setting. The cross-step redundancy the caching and step-count methods exploit is largely gone in a distilled few-step causal model—Self-Forcing and Krea Realtime 14B (Millon, 2025) run only about four to five denoising steps per block, leaving no long schedule to cache or shorten—and the training-based methods above require

a learned component trained together with the generator, whereas our method is training-free. Step-count reducers such as Surprise Forcing (Shi et al., 2026), for instance, change how many denoising steps run per block on a single model, with no routing, whereas our BlockStep fixes the total number of steps and changes only which model runs them—orthogonal to step reduction and often faster, since one target step already costs more than the drafter's full five-step pass.

Finally, methods that cut compute along other axes—attention sparsification (Xi et al., 2025; Zhang et al., 2025; Li et al., 2025), KV-cache compression (Xiao et al., 2023; Ranganath et al., 2026), and multi-GPU block parallelism (Bandyopadhyay et al., 2026)—are orthogonal to our method.

Multi-Model Sampling. Multi-model sampling runs models of different capacities within a single diffusion process, offloading part of a large target's computation onto a smaller same-family variant. The idea began in image diffusion (Yang et al., 2024; Pan et al., 2025; Liu et al., 2023) and carries over to full-sequence (bidirectional) video. Most skip verification and partition by stage, noise level, or frame: SRDiffusion (Cheng et al., 2025) gives high-noise steps to the large model and low-noise steps to the small model (the same Wan2.1-14B/1.3B pair we use); FlowBlending (Song et al., 2025) uses an offline-calibrated large–small–large schedule; Hybrid SD (Yan et al., 2024) and EC-Diff (Xie et al., 2025) split early and late stages across an edge-cloud boundary (image, then CogVideoX); and EcoVideo (Chen et al., 2026b) routes high-entropy frames to the large model and interpolates the rest. ASDSV (Zhou et al., 2026) instead verifies, drafting steps with the small model and having the large model check them—speculative decoding (Leviathan et al., 2022) for video—though with continuous latents such checks are only approximate, never lossless.

All of these are full-sequence with a long denoising schedule: they optimize along either the denoising-step axis or the frame axis, both of which are already compressed on the few-step, block-wise autoregressive DiT our method is built on, leaving little to split. Naively reusing their per-block schedules—large model at each block's earliest and latest stages—degraded quality in our tests; we find instead that block-0 must be all-target, after which the small model can cover far more of each block than in image or non-autoregressive video, with the target refining only last two steps.

The closest method is SDVG (Hu & Zhang, 2026), which routes causally block-by-block: a 1.3B drafter proposes each block and a 14B target accepts it or fully regenerates it, deciding in pixel space by VAE-decoding the block and scoring it with an external reward model (ImageReward)—a signal its own analysis reports barely beats random routing. BlockStep differs in three ways. First, it routes along both the block and within-block step axes, not blocks alone. Second, it decides in latent space, with no VAE decode or reward model. Third, it needs no verification—no per-block decode, scoring, or regeneration—so the target generates rather than corrects, at no cost to quality. SDVG has not released code, so we cannot compare directly, but on their preferred metric (VBench total score) we retain more quality at higher speedup.

3 METHODOLOGY

We introduce BlockStep, an online multi-model scheduler that accelerates generation while preserving the target model's visual quality. Each clip follows a three-stage schedule. First, block-0 is generated entirely by the target model as an anchor (Section 3.2). For every subsequent block, an entropy-based router partially drafts the block and determines whether it must be upgraded to the target (Section 3.3). When a block is upgraded, its position dictates whether the drafted prefix is reused or the block is regenerated from scratch by the target (Section 3.4). Figure 2 illustrates the pipeline, and Algorithm 1 details the procedure.

3.1 PRELIMINARIES

We first establish notation for autoregressive (block-causal) video diffusion, the setting on which BlockStep operates. A video is generated as a sequence of B clean latent blocks $x^{0:B-1} = (x^0, \ldots, x^{B-1})$, where each block x^b is a fixed group of latent frames. The blocks are produced left to right through the chain-rule factorization

$$p(x^{0:B-1}) = \prod_{b=0}^{B-1} p(x^b \mid x^{<b}),$$

Algorithm 1 BlockStep Inference with Entropy-Routed, Position-Dependent Upgrades

Require: Target G_{T} and drafter G_{D} (return KV embeddings via $G_{\mathrm{T}}^{\mathrm{KV}}$, $G_{\mathrm{D}}^{\mathrm{KV}}$); entropy read $H_{\mathrm{D}}(x;\, t, \mathrm{KV})$ of the forward $G_{\mathrm{D}}(x;\, t, \mathrm{KV})$
Require: Denoise timesteps $\{t_1, \ldots, t_T\}$ with $T = 5$; number of blocks B; context operator $\mathcal{C}$ (clean latent frames from the emitted output)
Require: Peek depth $k = 3$; entropy threshold τ; position boundary $b^* = 4$; schedule $S = S_T \cdots S_1$, $S_j \in \{\mathrm{T}, \mathrm{D}\}$; r the first step run under S

Initialize model output $X \leftarrow []$
for $b = 0, \ldots, B-1$ **do**
  Initialize $x_{t_T}^b \sim \mathcal{N}(0, I)$
  if $b = 0$ **then**
    Set $S \leftarrow \mathrm{TTTTT}$, $r \leftarrow T$; Initialize $\mathrm{KV}_{\mathrm{T}} \leftarrow []$ ▷ Anchor: no context
  else
    Set $S \leftarrow \mathrm{DDDDD}$, $r \leftarrow T - k$
    Set $C_b \leftarrow \mathcal{C}(X)$; Cache $\mathrm{KV}_{\mathrm{D}} \leftarrow G_{\mathrm{D}}^{\mathrm{KV}}(C_b)$ ▷ Drafter context
    for $j = T, \ldots, T-k+1$ **do** ▷ Draft k steps
      Set $\hat{x}_0^b \leftarrow G_{\mathrm{D}}(x_{t_j}^b;\, t_j, \mathrm{KV}_{\mathrm{D}})$; Sample $\epsilon \sim \mathcal{N}(0, I)$; Set $x_{t_{j-1}}^b \leftarrow \Psi(\hat{x}_0^b, \epsilon, t_{j-1})$
    end for
    Set $e_b \leftarrow \frac{1}{k} \sum_{j=T-k+1}^{T} H_{\mathrm{D}}(x_{t_j}^b;\, t_j, \mathrm{KV}_{\mathrm{D}})$ ▷ Read on the drafter forwards above
    if $e_b > \tau$ **then** Cache $\mathrm{KV}_{\mathrm{T}} \leftarrow G_{\mathrm{T}}^{\mathrm{KV}}(C_b)$ ▷ Upgrade: target context
      if $b \leq b^*$ **then** Set $S \leftarrow \mathrm{TTTTT}$, $r \leftarrow T$ ▷ Rollback: restart from $x_{t_T}^b$
      else Set $S \leftarrow \mathrm{DDDTT}$, $r \leftarrow T - k$ ▷ Prefix-continue: keep the k drafter steps
      end if
    end if
  end if
  for $j = r, \ldots, 2$ **do** ▷ Remaining steps under schedule S
    Set $\hat{x}_0^b \leftarrow G_{S_j}(x_{t_j}^b;\, t_j, \mathrm{KV}_{S_j})$; Sample $\epsilon \sim \mathcal{N}(0, I)$; Set $x_{t_{j-1}}^b \leftarrow \Psi(\hat{x}_0^b, \epsilon, t_{j-1})$
  end for
  Set $\hat{x}_0^b \leftarrow G_{S_1}(x_{t_1}^b;\, t_1, \mathrm{KV}_{S_1})$; X.append($\hat{x}_0^b$)
end for
return X

so every block is conditioned on the clean latents of all preceding blocks. This conditioning is implemented with a key–value (KV) cache: before denoising block b, the model encodes the preceding clean latents $x^{<b}$ into KV_b, which the block's attention layers read from. The cache is rebuilt from $x^{<b}$ at each block and does not persist across blocks (Huang et al., 2026; Yin et al., 2025).

Each conditional $p(x^b \mid x^{<b})$ is sampled by an iterative denoising process of T steps over the timestep schedule $t_1, \ldots, t_T$, ordered from the noisiest t_T to the clean t_1. Starting from $x_{t_T}^b \sim \mathcal{N}(0, I)$, step j applies a diffusion model G_θ that predicts the clean latent $\hat{x}_0^b = G_\theta(x_{t_j}^b, t_j, \mathrm{KV}_b)$. For $j > 1$, a sampler Ψ reintroduces noise to form the next, less-noisy latent:

$$x_{t_{j-1}}^b = \Psi\big(G_\theta(x_{t_j}^b, t_j, \mathrm{KV}_b), \epsilon_{t_{j-1}}, t_{j-1}\big), \qquad \epsilon_{t_{j-1}} \sim \mathcal{N}(0, I).$$

After the final step the clean latent block $x^b := \hat{x}_0^b$ is appended to the sequence; to render frames it is mapped to pixels by the VAE decoder $\mathcal{D}$, producing $\mathcal{D}(x^b)$.

In our method, G_θ is not necessarily the same model for the entire denoising process. Let G_T denote the base target model and G_D the faster draft model from the same model family. If these models can operate on a common latent representation, each denoising step can be run either using G_T or G_D. Thus different T step schedule variations can be formed. We write these schedules in sampling order. $DDDTT$, for instance, indicates using G_D for the first three denoising steps, then G_T for the last two denoising steps.

3.2 Protect Shared Context: The Block-0 Anchor

The first block establishes scene composition and subject identity without any preceding video context, and every later block conditions on it through the autoregressive KV cache. Approximation

error introduced here can therefore propagate downstream through the clip. Following prior work that allocates extra computation to the first block (Hu & Zhang, 2026; Qiu et al., 2026; Liu et al., 2026a), we generate block 0 entirely with the target, $S_0 = \mathrm{TTTTT}$. This eliminates draft-induced error at the initial block and provides a target-generated context for subsequent blocks. Empirically, adding this anchor to draft-only generation raises VBench total from 81.12 to 82.60, recovering 94.8% of the target–draft gap. Appendix A.2 formalizes how local approximation errors can propagate under a simplified context-stability model.

3.3 Estimate Local Difficulty: Entropy-Based Routing

A sweep over per-block schedules with the anchor fixed shows that, once target budget is placed at the right positions, most prompts need only a small fraction of their blocks generated by the target to match the target's visual quality. Committing to a per-prompt schedule from such a sweep is far more expensive than simply running the target, so BlockStep instead decides online, per block, whether a block needs additional target computation.

For each $b > 0$ it first runs the $k = 3$ noisiest denoising steps $(t_T, \ldots, t_{T-k+1})$ with the drafter and reads their self-attention entropy—an inexpensive signal motivated by entropy-based analyses of visual generation (Chen et al., 2026a; Ma et al., 2026; Chen et al., 2026b). High entropy means that the drafter's attention is spread across more alternatives. We use it as a proxy for uncertainty in the drafter's approximation of the target. Attention entropy is an observable of the drafter, not a direct measurement of its discrepancy from the target.

Let $p_{ai}^{b,j,\ell,\eta}$ denote the drafter's self-attention probability from query a to key i at denoising step t_j, layer ℓ, and head η. The per-step entropy $h_{b,j}$ and routing score e_b are

$$h_{b,j} = -\frac{1}{|\mathcal{L}|\, n_{\mathrm{h}}\, N} \sum_{\ell \in \mathcal{L}} \sum_{\eta=1}^{n_{\mathrm{h}}} \sum_{a=1}^{N} \sum_{i} p_{ai}^{b,j,\ell,\eta} \log p_{ai}^{b,j,\ell,\eta}, \quad \text{and} \quad e_b = \frac{1}{k} \sum_{j=T-k+1}^{T} h_{b,j}. \tag{1}$$

Here $\mathcal{L}$ is the set of inspected attention layers and ℓ indexes it; $\eta \in \{1, \ldots, n_{\mathrm{h}}\}$ indexes the n_{h} heads; $a \in \{1, \ldots, N\}$ indexes the N query tokens of block b; and i ranges over the keys available to each query—the block's own tokens together with the cached context. The probabilities $p_{ai}^{b,j,\ell,\eta}$ are the usual scaled dot-product softmax over those keys, so $h_{b,j}$ averages the per-query attention entropy over all heads, layers, and queries at step t_j, and e_b averages $h_{b,j}$ over the k peeked steps.

If $e_b \le \tau$, the drafter completes the block, yielding $S_b = \mathrm{DDDDD}$; if $e_b > \tau$, the block is upgraded with the schedule of Section 3.4. The fixed threshold τ controls the quality–compute tradeoff and imposes no fixed number of upgrades per clip.

This decision needs no routing-side VAE decode or reward network. Building the draft KV cache, the three draft forwards, and the entropy computation together take ≈ 297 ms, versus ≈ 351 ms for target KV construction alone; blocks that stay with the drafter avoid both target denoising and target context construction.

Why entropy can guide routing. Attention entropy serves as a proxy for the local draft-error variance ν_r^2 at update r. Under conditionally mean-zero errors, these variances contribute to the expected squared deviation from a target rollout with shared noise (Appendix A.4). High entropy suggests that further drafting may add substantial error, motivating target updates; low entropy permits more drafting at lower cost. Proposition 3 establishes this relation for the leading error in a local attention model. For the full model, the averaged score must predict the relevant variances and upgrade benefit; it does not measure the current distance to the target. Appendix A gives the assumptions and proofs.

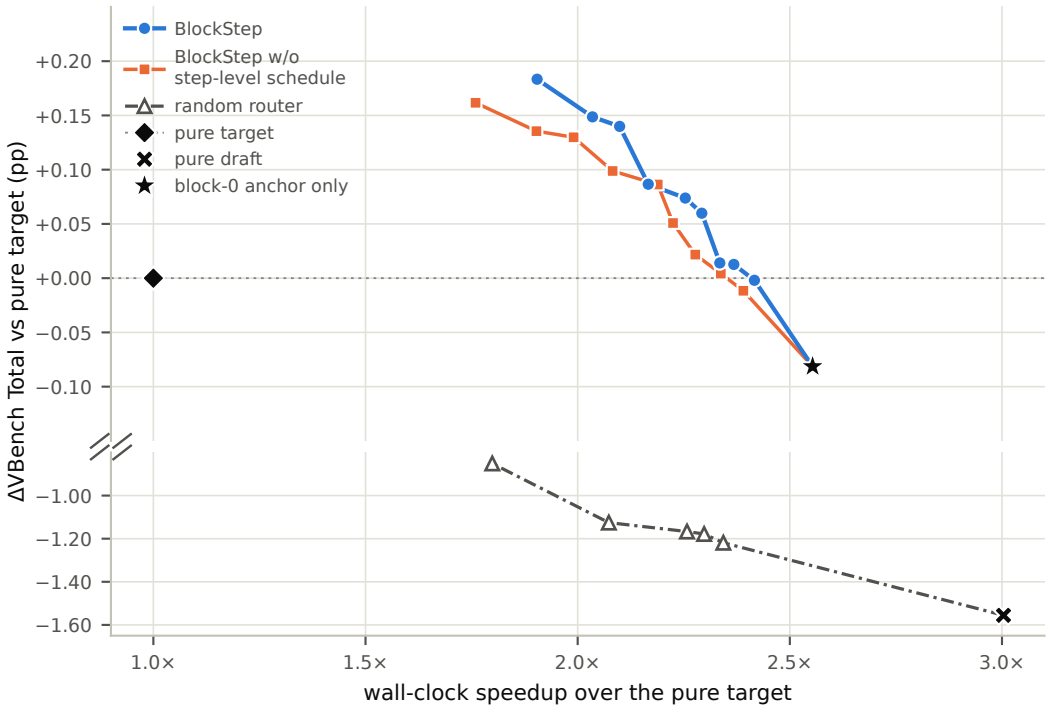


Figure 3: **Quality-speed comparison of BlockStep vs BlockStep w/o vs random routing.** BlockStep has significantly higher VBench total score than the random router at any speedup, while usually maintaining a slight edge over BlockStep without a step-level schedule.

3.4 Preserving Useful Work: Position-Dependent Step Schedules

An upgrade can either restart a block or continue its partially denoised latent. We use block position to choose:

$$S_b = \begin{cases} \text{TTTTT}, & b = 0, \\ \text{DDDDD}, & b > 0,\ e_b \le \tau, \\ \text{TTTTT (restart)}, & 1 \le b \le b^*,\ e_b > \tau, \\ \text{DDDTT (continue)}, & b > b^*,\ e_b > \tau, \end{cases} \qquad b^* = 4.$$

Earlier blocks influence more of the subsequent generation, motivating a full target restart from the saved initial noise. Proposition 2 formalizes this effect when context errors propagate. A restart discards the three draft steps used for routing and incurs eight forwards in total (three drafter peeks plus five target steps). For later blocks we retain the draft prefix and run only the two remaining steps with the target. The DDDTT schedule therefore avoids repeating completed work while preventing further draft updates within that block. It can reduce the existing draft-induced discrepancy when the target suffix is contractive (Proposition 1); denoising alone does not imply contraction. The peek depth $k = 3$ and boundary $b^* = 4$ are empirical design choices.

4 Experiments

We evaluate BlockStep on VBench and speedup compared to the target, draft, a random router, and SDVG (Section 4.1) and in ablation studies evaluate the individual importance of the block-0 anchor and the step-wise schedules (Section 4.2).

Models, hardware, generation settings. The target is Krea Realtime 14B, the drafter Wan2.1-T2V-1.3B Self-Forcing DMD, and all experiments are run on a single H100. A clip is 9 blocks of 5 forwards, and thus an all target schedule is bit-identical to the released Krea sampler at its five-step default.

Table 1: Quality-speed comparison. Target blocks is the mean number of blocks (out of 9) run by the target per clip; throughput and wall-clock time are measured end to end per clip; VBench is the VBench total score; LPIPS, PSNR, and SSIM are all based on the target model outputs.

Method	Target blocks	Speed		Visual Quality			
		Throughput (FPS)	Wall-clock time (s)	VBench ↑	LPIPS ↓	PSNR (dB) ↑	SSIM ↑
Target	9	4.50 (1.00×)	22.65	82.68	–	–	–
BlockStep	1.35	10.66 (2.37×)	9.57	82.69	0.3710	16.13	0.5270
BlockStep w/o step-level schedule	1.34	10.52 (2.34×)	9.69	82.68	0.3687	16.88	0.5295
SDVG	3.37	4.92 (1.09×)	21.36	81.57	0.4577	34.09	0.5541
Random routing	1.37	10.35 (2.30×)	9.86	81.50	0.6474	10.37	0.3605
Draft	0	13.52 (3.00×)	7.54	81.12	0.7022	9.24	0.3258

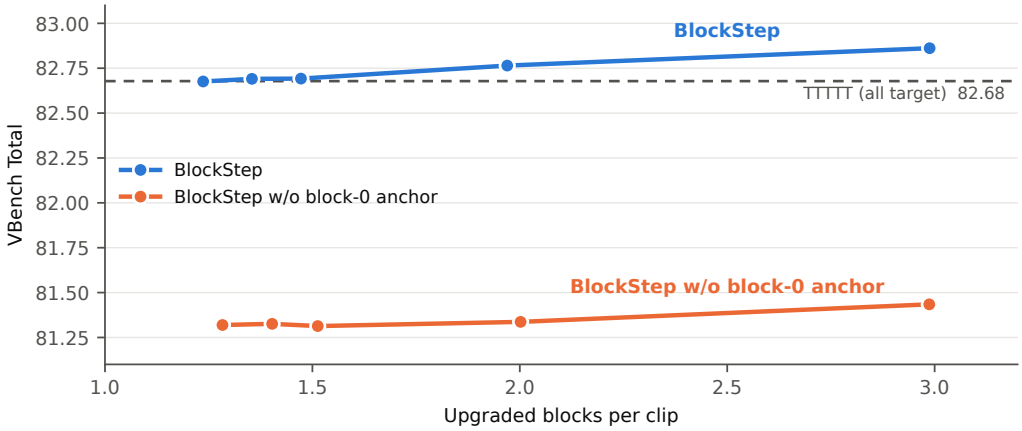


Figure 4: **Visual comparison of Blockstep vs anchorless BlockStep.** Without the block-0 anchor, BlockStep's VBench total decreases substantially.

Dataset and Metrics. Visual quality is assessed using VBench total score, LPIPS, PSNR, and SSIM. We use the VBench prompt set to generate our videos. To measure speed, we measure the throughput, wall-clock time, and speedup compared to the target model.

4.1 Effectiveness

Figure 3 compares the VBench total score and speedup of the methods, with routing functions being plotted at different thresholds. As seen in the figure, BlockStep consistently has higher VBench total score compared to the random router at the same speedup. When it crosses the target VBench total score, BlockStep achieves 2.41x speedup while a random router never reaches the target quality even when its speedup goes below 2.0x. Compared to BlockStep without a step-level schedule, BlockStep maintains a slight edge throughout most ranges of speedup.

Table 1 compares the quality and speedup of Blockstep and other methods, with routing functions other than SDVG fixed to have their number of target blocks be approximately 1.375 per prompt. As seen in the table comparison, BlockStep is the fastest method in terms of throughput and wall-clock time that is able to match the target's VBench total score. The random router is both slower and has a lower VBench total score. Blockstep has a lower LPIPS score, higher PSNR score, and higher SSIM score than both random routing and the draft as well, which indicates it generates videos more aligned with the target model. BlockStep without a step-level schedule and SDVG may have a better LPIPS, PSNR, or SSIM scores, but they pay the cost in terms of increased wall-clock time.

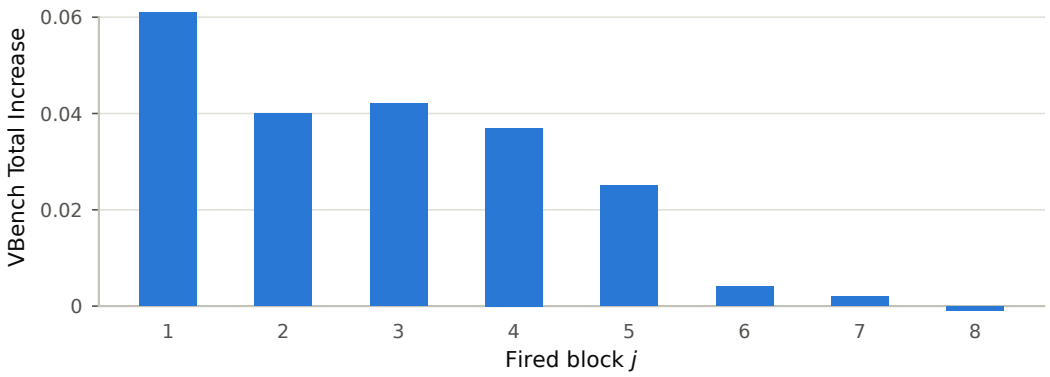


Figure 5: **Value of a single full target upgrade plan.** Blocks 1–5 carry the value ($+0.061$, $+0.040$, $+0.042$, $+0.037$, $+0.025$ pp), blocks 6–8 none ($+0.004$, $+0.002$, -0.001 pp); no single block is resolvable from another at five seeds (0 of 28 pairwise contrasts clear the 0.35 pp planning MDE; the largest, block 1 against block 8, is $+0.062$ pp with standard error 0.038).

4.2 Ablations

Block-0 anchor. To demonstrate the importance of the block-0 anchor, we recreate our BlockStep algorithm with one change: setting block-0 to always be drafted (DDDDD) instead. For a fair comparison, we gave this anchorless BlockStep an average of 1 more draft block in subsequent blocks by adjusting its threshold. Comparing across different numbers of upgraded blocks, as seen in Figure 4, anchorless BlockStep consistently has a VBench total score at least a point lower than its BlockStep counterpart.

Step-wise schedule. We isolated the importance of upgrade each block position to a full target block by modifying our BlockStep algorithm's upgrade plan so for a set j, a fired block b runs TTTTT if $b = j$ (rollback and re-run on the target) and DDDTT with prefix-continue otherwise (drafter steps 0–2 kept, steps 3–4 on the target, no rollback). We ran this for $j = 1, ..., B$ and measured how much upgrading block j improved upon the VBench score of BlockStep with a fixed upgrade plan of DDDTT, as shown in Figure 5. As seen in the figure, the first blocks lead to much greater improvemetns to VBench score than subsequent blocks, motivating us to upgrade these blocks to full target schedules and leaving subsequent blocks to have partial target schedules.

5 Limitations

We note that like other multi-model methods, our method can not be directly applied to any pair of models. Specifically, the models must be able to share a latent state, and one model must be faster than the other. This leaves us with a limited number of pairs to use for our testing, though we note that training smaller models could allow for additional pairs.

6 Conclusion

In conclusion, BlockStep accelerates inference in autoregressive video diffusion with minimal quality loss. Extensive experiments demonstrate that BlockStep maintains quality while enabling $2.37\times$ speedup for Krea without additional training. This enables greater flexibility in hardware and supports any application requiring real-time video generation.

AI USE STATEMENT

In this work, we used generative AI tools for setting up experiments, text editing and refinement, as well as testing our code and checking mathematical proofs. We have not used generative AI tools for research ideas, algorithm design, or experiment validation. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Hmrishav Bandyopadhyay, Nikhil Pinnaparaju, Rahim Entezari, Jim Scott, Yi-Zhe Song, and Varun Jampani. Block cascading: Training free acceleration of block-causal video models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4078–4088, 2026.

Eliot Beyler and Francis Bach. Convergence of deterministic and stochastic diffusion-model samplers: A simple analysis in wasserstein distance, 2025.

Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion, 2024a.

Guibin Chen, Dixuan Lin, Jiangping Yang, Chunze Lin, Junchen Zhu, Mingyuan Fan, Hao Zhang, Sheng Chen, Zheng Chen, Chengcheng Ma, Weiming Xiong, Wei Wang, Nuo Pang, Kang Kang, Zhiheng Xu, Yuzhe Jin, Yupeng Liang, Yubing Song, Peng Zhao, Boyuan Xu, Di Qiu, Debang Li, Zhengcong Fei, Yang Li, and Yahui Zhou. SkyReels-V2: Infinite-length Film Generative Model, 2025.

Jiayu Chen, Ruoyu Lin, Zihao Zheng, Jingxin Li, Maoliang Li, Guojie Luo, and Xiang Chen. Toprovar: Efficient visual autoregressive modeling via tri-dimensional entropy-aware semantic analysis and sparsity optimization. *arXiv preprint arXiv:2602.22948*, 2026a.

Jiayu Chen, Hengyi Zhang, Maoliang Li, Minyu Li, Zihao Zheng, Xuanzhe Liu, Guojie Luo, and Xiang Chen. EcoVideo: Entropy-Orchestrated Video Generation Paradigm in Cloud-Edge Dynamics, 2026b.

Pengtao Chen, Mingzhu Shen, Peng Ye, Jianjian Cao, Chongjun Tu, Christos-Savvas Bouganis, Yiren Zhao, and Tao Chen. Δ-DiT: A Training-Free Acceleration Method Tailored for Diffusion Transformers, 2024b.

Yukang Chen, Luozhou Wang, Wei Huang, Shuai Yang, Bohan Zhang, Yicheng Xiao, Ruihang Chu, Weian Mao, Qixin Hu, Shaoteng Liu, et al. Longlive-2.0: An nvfp4 parallel infrastructure for long video generation. *arXiv preprint arXiv:2605.18739*, 2026c.

Shenggan Cheng, Yuanxin Wei, Lansong Diao, Yong Liu, Bujiao Chen, Lianghua Huang, Yu Liu, Wenyuan Yu, Jiangsu Du, Wei Lin, and Yang You. SRDiffusion: Accelerate Video Diffusion Inference via Sketching-Rendering Cooperation, 2025.

Hanshuai Cui, Zhiqing Tang, Zhi Yao, Fanshuai Meng, Weijia Jia, and Wei Zhao. Not All Frames Deserve Full Computation: Accelerating Autoregressive Video Generation via Selective Computation and Predictive Extrapolation, 2026.

Justin Cui, Jie Wu, Ming Li, Tao Yang, Xiaojie Li, Rui Wang, Andrew Bai, Yuanhao Ban, and Cho-Jui Hsieh. Self-Forcing++: Towards Minute-Scale High-Quality Video Generation, 2025.

Chensheng Dai, Shengjun Zhang, Yifan Li, Zhang Zhang, Zheng Zhu, and Yueqi Duan. Rhymeflow: Training-free acceleration for video generation with asynchronous denoising flow scheduling. In Paolo Favaro, Zuzana Kukelova, Atsuto Maki, Anna Rohrbach, Konrad Schindler, and Federico Tombari (eds.), *Computer Vision – ECCV 2026*, pp. 313–330, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-37369-4.

Tianrui Feng, Zhi Li, Shuo Yang, Haocheng Xi, Muyang Li, Xiuyu Li, Lvmin Zhang, Keting Yang, Kelly Peng, Song Han, Maneesh Agrawala, Kurt Keutzer, Akio Kodaira, and Chenfeng Xu. Streamdiffusionv2: A streaming system for dynamic and interactive video generation. *Proceedings of Machine Learning and Systems*, 8:1656–1669, 2026.

Yuezhou Hu and Jintao Zhang. Speculative decoding for autoregressive video generation. *arXiv preprint arXiv:2604.17397*, 2026.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *Advances in Neural Information Processing Systems*, 38:167283–167308, 2026.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024.

Kumara Kahatapitiya, Haozhe Liu, Sen He, Ding Liu, Menglin Jia, Chenyang Zhang, Michael S. Ryoo, and Tian Xie. Adaptive Caching for Faster Video Generation with Diffusion Transformers, 2024.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The Lipschitz constant of self-attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5562–5571, 2021. URL `https://proceedings.mlr.press/v139/kim21i.html`.

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

Thanh-Tung Le, Yunhan Zhao, Menglei Chai, Zhengyang Shen, Zhe Cao, Danhang Tang, Xiaohui Xie, and Deying Kong. DSA: Dynamic Step Allocation for Fast Autoregressive Video Generation, 2026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast Inference from Transformers via Speculative Decoding, 2022.

Xingyang Li, Muyang Li, Tianle Cai, Haocheng Xi, Shuo Yang, Yujun Lin, Lvmin Zhang, Songlin Yang, Jinbo Hu, Kelly Peng, Maneesh Agrawala, Ion Stoica, Kurt Keutzer, and Song Han. Radial Attention: $O(n \log n)$ Sparse Attention with Energy Decay for Long Video Generation, 2025.

Zian Li, Litong Gong, Borui Liao, Pengfei Liu, Xinyu Wang, Xinyuan Wei, Yifan Gao, Tiezheng Ge, and Muhan Zhang. DUET: A Diversity-Quality Duet of Distillation Experts for Two-Step Video Generation, 2026.

Shanchuan Lin, Xin Xia, Yuxi Ren, Ceyuan Yang, Xuefeng Xiao, and Lu Jiang. Diffusion Adversarial Post-Training for One-Step Video Generation, 2025.

Enshu Liu, Xuefei Ning, Zinan Lin, Huazhong Yang, and Yu Wang. Oms-dpm: Optimizing the model schedule for diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 21915–21936. PMLR, 2023.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep Embedding Tells: It's Time to Cache for Video Diffusion Model, 2024.

Jinxiu Liu, Xuanming Liu, Kangfu Mei, Yandong Wen, Ming-Hsuan Yang, and Weiyang Liu. Streaming Autoregressive Video Generation via Diagonal Distillation, 2026a.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling Forcing: Autoregressive Long Video Diffusion in Real Time, 2025.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. In *International Conference on Learning Representations*, volume 2026, pp. 91177–91196, 2026b.

Xiaoxiao Ma, Feng Zhao, Pengyang Ling, Haibo Qiu, Zhixiang Wei, Hu Yu, Jie Huang, Zhixiong Zeng, and Lin Ma. Towards better & faster autoregressive image generation: From the perspective of entropy. *Advances in Neural Information Processing Systems*, 38:31466–31497, 2026.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating Diffusion Models for Free, 2023.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-Cache: Accelerating Diffusion Transformer via Layer Caching, 2024.

Erwann Millon. Krea realtime 14b: Real-time video generation, 2025. URL https://github.com/krea-ai/realtime-video.

Zizheng Pan, Bohan Zhuang, De-An Huang, Weili Nie, Zhiding Yu, Chaowei Xiao, Jianfei Cai, et al. T-stitch: Accelerating sampling in pre-trained diffusion models with trajectory stitching. In *International Conference on Learning Representations*, volume 2025, pp. 6103–6137, 2025.

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

Haonan Qiu, Shikun Liu, Zijian Zhou, Zhaochong An, Weiming Ren, Zhiheng Liu, Jonas Schult, Sen He, Shoufa Chen, Yuren Cong, et al. Histream: Efficient high-resolution video generation via redundancy eliminated streaming. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4603–4613, 2026.

Suraj Ranganath, Vaishak Menon, and Anish Patnaik. KV Cache Quantization for Self-Forcing Video Generation: A 33-Method Empirical Study, 2026.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. FORA: Fast-Forward Caching in Diffusion Transformer Acceleration, 2024.

Shuwei Shi, Zhen Li, Muyao Niu, Chuanhao Li, Bo Zheng, Kaipeng Zhang, and Yinqiang Zheng. Surprise forcing: What to remember, when to skip in long video generation. *arXiv preprint arXiv:2607.18436*, 2026.

Jibin Song, Mingi Kwon, Jaeseok Jeong, and Youngjung Uh. Flowblending: Stage-aware multi-model sampling for fast and high-fidelity video generation. *arXiv preprint arXiv:2512.24724*, 2025.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Shengkun Tang, Yaqing Wang, Caiwen Ding, Yi Liang, Yao Li, and Dongkuan Xu. AdaDiff: Accelerating Diffusion Models through Step-Wise Adaptive Computation, 2023.

Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, WQ Zhang, Weifeng Luo, et al. Magi-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines, 2024. URL https://arxiv.org/abs/2408.14837.

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

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse VideoGen: Accelerating Video Diffusion Transformers with Spatial-Temporal Sparsity, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks, 2023.

Jiajian Xie, Shengyu Zhang, Zhou Zhao, Fan Wu, and Fei Wu. EC-Diff: Fast and High-Quality Edge-Cloud Collaborative Inference for Diffusion Models, 2025.

Chenqian Yan, Songwei Liu, Hongjian Liu, Xurui Peng, Xiaojian Wang, Fangmin Chen, Lean Fu, and Xing Mei. Hybrid SD: Edge-Cloud Collaborative Inference for Stable Diffusion Models, 2024.

Shuai Yang, Yukang Chen, Luozhou Wang, Shu Liu, and Yingcong Chen. Denoising Diffusion Step-aware Models. In *International Conference on Learning Representations (ICLR)*, 2024.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, volume 2025, pp. 83048–83077, 2025.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T. Freeman, and Taesung Park. One-step Diffusion with Distribution Matching Distillation, 2023.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T. Freeman. Improved Distribution Matching Distillation for Fast Image Synthesis, 2024.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22963–22974. IEEE, 2025.

Peiyuan Zhang, Yongqi Chen, Runlong Su, Hangliang Ding, Ion Stoica, Zhengzhong Liu, and Hao Zhang. Fast Video Generation with Sliding Tile Attention, 2025.

Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-Time Video Generation with Pyramid Attention Broadcast, 2024.

Kaijun Zhou, Xingyu Yan, Xingda Wei, Xijun Li, and Jinyu Gu. Asdsv: Multimodal generation made efficient with approximate speculative diffusion and speculative verification. *Advances in Neural Information Processing Systems*, 38:91337–91363, 2026.

Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity Distillation: Exponentially Fast Distillation of Pretrained Diffusion Models for One-Step Generation, 2024.

A Error propagation and conditional routing principles

This appendix studies how errors from draft updates can accumulate and how target updates can control them. We first analyze errors within a block and their propagation through autoregressive context. We then give assumptions under which attention entropy indicates the size of local approximation errors and derive a bound on their accumulation. All norms are Euclidean. The results concern latent distance to a target reference; they do not establish VBench parity or optimality of the chosen schedules.

A.1 Shared-noise comparison within a block

Fix the prompt and preceding context. Index the $K = 5$ retained updates of a block in sampling order, $r = 1, \dots, K$, excluding any discarded routing prefix. Here $K = T$ in Section 3.1, and update r corresponds to timestep index $j = T - r + 1$. Let $m_r \in \{\mathrm{D}, \mathrm{T}\}$ be the model used at update r. Assume the update has the form

$$z_r = F_{m_r,r}(z_{r-1}) + \sigma_r \xi_r, \qquad z_r^{\mathrm{T}} = F_{\mathrm{T},r}(z_{r-1}^{\mathrm{T}}) + \sigma_r \xi_r, \qquad z_0 = z_0^{\mathrm{T}}, \tag{2}$$

where both trajectories use the same noise scales and realizations, with $\sigma_K = 0$ for the final clean prediction. The deterministic maps $F_{m,r}$ include the model prediction and the deterministic part of the sampling update. This shared-noise construction follows the coupling approach used in diffusion-sampler analysis (Beyler & Bach, 2025). The comparison holds for each realized schedule and retained noise sequence. If routing affects which noise sequence is retained, the resulting reference need not have the unconditional target sampling distribution.

Assume $F_{\mathrm{T},r}$ is L_r-Lipschitz on a region containing both trajectories. Define the trajectory error $\Delta_r = \|z_r - z_r^{\mathrm{T}}\|$ and the same-input discrepancy $\delta_r = \|F_{\mathrm{D},r}(z_{r-1}) - F_{\mathrm{T},r}(z_{r-1})\|$.

Proposition 1 (Within-block error)**.** *Under these assumptions,*

$$\Delta_K \le \sum_{r=1}^{K} \mathbf{1}\{m_r = \mathrm{D}\}\, \delta_r \prod_{i=r+1}^{K} L_i, \tag{3}$$

with an empty product equal to one.

Proof. The shared additive noise cancels. Adding and subtracting $F_{\mathrm{T},r}(z_{r-1})$ gives

$$\begin{aligned} \Delta_r &\le \|F_{m_r,r}(z_{r-1}) - F_{\mathrm{T},r}(z_{r-1})\| + \|F_{\mathrm{T},r}(z_{r-1}) - F_{\mathrm{T},r}(z_{r-1}^{\mathrm{T}})\| \\ &\le \mathbf{1}\{m_r = \mathrm{D}\}\delta_r + L_r \Delta_{r-1}. \end{aligned} \tag{4}$$

Unrolling this recurrence from $\Delta_0 = 0$ proves the claim. The argument holds for each realized schedule, including schedules chosen from observations of the draft prefix. □

Implications for upgrades. For DDDTT, the recurrence yields $\Delta_5 \le L_5 L_4 \Delta_3$. The two target updates add no new substitution error and reduce the inherited error when $L_5 L_4 < 1$. Denoising alone does not imply this contraction condition. A full target restart removes local substitution error relative to a target reference with the same context, initial noise, and restart noise. It does not remove errors in the preceding context. Paired evaluation must align the noise used at each update. Using the same initial seed is insufficient if a restart consumes additional random draws.

A.2 Error propagation across blocks: Why Block 0 is crucial?

Fix the prompt and a realized sequence of noise records $\omega_0, \dots, \omega_{B-1}$, with $B \ge 1$. In this subsection, assume that each block conditions only on the immediately preceding clean block. Write $\mathcal{T}_b(c; \omega_b)$ for the complete all-target generation of block b from context c using noise record ω_b. For $1 \le b < B$, assume

$$\|\mathcal{T}_b(c; \omega_b) - \mathcal{T}_b(c'; \omega_b)\| \le \rho \|c - c'\|, \qquad \rho \ge 0, \tag{5}$$

for all admissible contexts and noise records, with both rollouts' contexts in this domain.

Let $x^0, \dots, x^{B-1}$ be the mixed rollout and define the target reference recursively by

$$y^0 = \mathcal{T}_0(\varnothing; \omega_0), \qquad y^b = \mathcal{T}_b(y^{b-1}; \omega_b) \quad (1 \le b < B), \tag{6}$$

where $\varnothing$ denotes the common empty video context at block 0. Define $E_b = \|x^b - y^b\|$ for every block and the local substitution errors by

$$u_0 = \|x^0 - y^0\|, \qquad u_b = \|x^b - \mathcal{T}_b(x^{b-1}; \omega_b)\| \quad (1 \le b < B). \tag{7}$$

Thus u_b measures the error from using draft updates at the *same mixed context*, while E_b also includes the effect of different preceding contexts. At the first block, $E_0 = u_0$.

Proposition 2 (Position-weighted error)**.** *Under these assumptions, the following bounds hold for each realized pair of rollouts:*

$$E_b \le \sum_{a=0}^{b} \rho^{b-a} u_a \qquad (0 \le b < B), \tag{8}$$

and consequently

$$\sum_{b=0}^{B-1} E_b \le \sum_{b=0}^{B-1} \left(\sum_{q=0}^{B-1-b} \rho^q \right) u_b. \tag{9}$$

The zero-th power is interpreted as one, including when $\rho = 0$.

Proof. For $b \ge 1$, introduce the intermediate target block $\widetilde{y}^b = \mathcal{T}_b(x^{b-1}; \omega_b)$. The triangle inequality and Equation 5 give

$$\begin{aligned} E_b &\le \|x^b - \widetilde{y}^b\| + \|\widetilde{y}^b - y^b\| \\ &= u_b + \|\mathcal{T}_b(x^{b-1}; \omega_b) - \mathcal{T}_b(y^{b-1}; \omega_b)\| \\ &\le u_b + \rho E_{b-1}. \end{aligned} \tag{10}$$

We now prove Equation 8 by induction. For $b = 0$, it is the equality $E_0 = u_0$. If it holds at $b - 1$, Equation 10 and $\rho \ge 0$ imply

$$E_b \le u_b + \rho \sum_{a=0}^{b-1} \rho^{b-1-a} u_a = \sum_{a=0}^{b} \rho^{b-a} u_a, \tag{11}$$

which completes the induction. Finally, sum the blockwise bound and change the order of the finite sums:

$$\begin{aligned} \sum_{b=0}^{B-1} E_b &\le \sum_{b=0}^{B-1} \sum_{a=0}^{b} \rho^{b-a} u_a \\ &= \sum_{a=0}^{B-1} u_a \sum_{b=a}^{B-1} \rho^{b-a} = \sum_{a=0}^{B-1} u_a \sum_{q=0}^{B-1-a} \rho^q. \end{aligned} \tag{12}$$

This proves Equation 9. □

Early errors are more damaging. For $b < B - 1$, $w_b - w_{b+1} = \rho^{B-1-b}$. Hence earlier local errors have strictly larger coefficients when $\rho > 0$; when $\rho = 0$, all weights equal one and context errors do not propagate. An all-target anchor gives $x^0 = y^0$, and thus $u_0 = E_0 = 0$, under the shared-noise comparison. The weights describe how a local error can contribute to the total error over the rollout. They do not determine the sizes of the local errors or which upgrades provide the greatest benefit. For a longer context window, the analysis would need a sensitivity bound for every retained block. The geometric weights above apply to the stated one-block context model.

A.3 Attention entropy and approximation sensitivity

We use draft attention entropy as a cheap indicator of how reliably the drafter approximates the target's prediction. EcoVideo provides related empirical evidence: it uses early attention entropy to select frames for large-model generation, and its ablation favors entropy-based over uniform keyframe selection (Chen et al., 2026b, Sections 3.1 and 4.3). That evidence concerns frame selection. Here we study a local mechanism: when attention is divided between distinct alternatives, its output can be more sensitive to small errors in their relative scores.

A local model with two alternatives. Fix an input and consider shared value vectors $v_1, v_2 \in \mathbb{R}^d$. Holding these vectors fixed isolates errors in attention scores. Let $g(u) = (1 + \exp(-u))^{-1}$ and let ℓ be the target's difference between the two logits. The target and draft attention outputs are

$$p = g(\ell), \quad p_{\mathrm{D}} = g(\ell + \eta), \qquad A_{\mathrm{T}} = pv_1 + (1-p)v_2, \quad A_{\mathrm{D}} = p_{\mathrm{D}}v_1 + (1-p_{\mathrm{D}})v_2, \tag{13}$$

where η models a small draft error in the logit difference. Expectations are over η, with ℓ, v_1, v_2 fixed. The model describes errors in the scores, not random selection of attention tokens.

Proposition 3 (Sensitivity of binary attention)**.** *Suppose $a \geq 0$, $|\eta| \leq a$ almost surely, and $\mathbb{E}\eta^2 = s^2$. Writing $q = p(1-p)$ and $d_v = \|v_1 - v_2\|$, we have*

$$e^{-2a}s^2q^2d_v^2 \leq \mathbb{E}\|A_{\mathrm{D}} - A_{\mathrm{T}}\|^2 \leq e^{2a}s^2q^2d_v^2. \tag{14}$$

For fixed s and d_v, the leading factor $s^2q^2d_v^2$ is nondecreasing in the binary entropy $H(p) = -p\log p - (1-p)\log(1-p)$.

Proof. Since $g'(u) = g(u)(1-g(u)) > 0$, $\frac{d}{du}\log g'(u) = 1 - 2g(u)$ has absolute value at most one. Thus $e^{-a}q \leq g'(\ell + t) \leq e^a q$ whenever $|t| \leq a$. Integrating along the segment from ℓ to $\ell + \eta$ gives

$$e^{-a}q|\eta| \leq |g(\ell + \eta) - g(\ell)| \leq e^a q|\eta|. \tag{15}$$

Multiply by d_v, square, and take expectations to obtain Equation 14. Finally, $H(p)$ and q are symmetric about $p = 1/2$ and strictly increasing on $0 < p < 1/2$, since $H'(p) = \log((1-p)/p) > 0$ and $q'(p) = 1 - 2p > 0$ there. This establishes the claimed ordering of the leading factor. □

This bound holds for any perturbation satisfying $|\eta| \leq a$; it does not require η to have mean zero. For small perturbations, a Taylor expansion gives the first-order error $\varepsilon_{\mathrm{lin}} = q\eta(v_1 - v_2)$. If $\mathbb{E}\eta = 0$, this error has mean zero and expected squared norm $s^2q^2d_v^2$. At fixed s and d_v, this leading term increases with attention entropy. The proposition does not claim that entropy orders the exact errors for all finite perturbations. Also, a nonlinear attention error can have nonzero mean even when the logit error has mean zero.

Using the draft's entropy. The same derivative bound gives $e^{-a}q \leq p_{\mathrm{D}}(1-p_{\mathrm{D}}) \leq e^a q$. For a distribution over two alternatives, entropy determines $p(1-p)$. The draft entropy therefore estimates the target's sensitivity factor up to a factor of e^a. This explains the role of similar attention patterns in the argument. Matching entropy alone is weaker: two models can have the same entropy while attending to different tokens or using different value vectors.

Scope of the mechanism. For attention over M keys, let $p = \mathrm{softmax}(\ell)$, let V have rows $v_i^\top$, and write $J(p) = \mathrm{diag}(p) - pp^\top$ for the softmax Jacobian (Kim et al., 2021). With p and V fixed, let $\eta \in \mathbb{R}^M$ have mean zero and covariance Σ. The first-order output error is $V^\top J(p)\eta$, so

$$\mathbb{E}\|\varepsilon_{\mathrm{lin}}\|^2 = \mathrm{tr}\big(V^\top J(p)\Sigma J(p)V\big)\,. \tag{16}$$

The error depends on the value vectors and the distribution of score errors, as well as the attention weights. For general attention distributions, larger Shannon entropy need not imply larger error. For example, identical value vectors give zero error regardless of the attention entropy. Proposition 3 therefore establishes the relation in the stated model with two alternatives. It concerns sensitivity to score errors, not the input Lipschitz constant of the full target update. If the two models differ only through this attention output and apply the same C-Lipschitz function afterward, their update mean squared error is at most C^2 times the upper bound in Equation 14. Extending this conclusion to the full models requires bounds on their other differences and on how errors from different heads combine.

A.4 Mean squared rollout error: A random-walk analysis.

Proposition 1 allows draft errors to accumulate in the same direction. A smaller bound is possible if each error has mean zero given the history of the process. We study this assumption in a stochastic

model of approximation errors. It does not follow from the noise used by the diffusion sampler. Let $I_r = \mathbf{1}\{m_r = \mathrm{D}\}$ and compare

$$\begin{aligned} z_r &= F_{\mathrm{T},r}(z_{r-1}) + I_r\varepsilon_r + \sigma_r\xi_r, \\ z_r^{\mathrm{T}} &= F_{\mathrm{T},r}(z_{r-1}^{\mathrm{T}}) + \sigma_r\xi_r, \qquad z_0 = z_0^{\mathrm{T}}. \end{aligned} \tag{17}$$

Here ε_r represents the error introduced by a draft update, while $\sigma_r\xi_r$ is the shared sampler noise from Section A.1. Let $\{\mathcal{F}_r\}_{r=0}^K$ be a filtration, where $\mathcal{F}_r$ records the history after update r. Before update r, both states and the decision I_r are known from $\mathcal{F}_{r-1}$; the new error ε_r is realized during that update. Assume finite, deterministic Lipschitz bounds $L_r \geq 0$ for the target maps on the states reached, and

$$\mathbb{E}[\varepsilon_r \mid \mathcal{F}_{r-1}] = 0, \qquad \mathbb{E}[\|\varepsilon_r\|^2 \mid \mathcal{F}_{r-1}] \leq \nu_r^2, \tag{18}$$

where $\nu_r \geq 0$ is known from $\mathcal{F}_{r-1}$ and $\mathbb{E}\nu_r^2 < \infty$. Thus ν_r^2 may depend on the current states and routing observations. Because the conditional mean is zero, it bounds the conditional error variance, defined here as the expected squared norm. It may also be taken to equal that variance. This is a property of the model approximation error; σ_r instead specifies the sampler noise shared by both trajectories.

Proposition 4 (Mean squared trajectory error)**.** *Under the preceding assumptions,*

$$\mathbb{E}\|z_K - z_K^{\mathrm{T}}\|^2 \leq \sum_{r=1}^{K} \mathbb{E}[I_r\nu_r^2] \prod_{i=r+1}^{K} L_i^2. \tag{19}$$

In particular, if $L_r \leq 1$ and $\nu_r \leq \nu$ almost surely for a finite deterministic $\nu \geq 0$, then

$$\left(\mathbb{E}\|z_K - z_K^{\mathrm{T}}\|^2\right)^{1/2} \leq \nu\sqrt{\mathbb{E}\left[\sum_{r=1}^{K} I_r\right]} \leq \nu\sqrt{K}. \tag{20}$$

Proof. Set $e_r = z_r - z_r^{\mathrm{T}}$ and $d_r = F_{\mathrm{T},r}(z_{r-1}) - F_{\mathrm{T},r}(z_{r-1}^{\mathrm{T}})$. Shared sampler noise cancels, leaving $e_r = d_r + I_r\varepsilon_r$. Since d_r and I_r are $\mathcal{F}_{r-1}$-measurable and $I_r^2 = I_r$, conditional centering eliminates the cross term:

$$\begin{aligned} \mathbb{E}[\|e_r\|^2 \mid \mathcal{F}_{r-1}] &= \|d_r\|^2 + 2I_r\,\langle d_r, \mathbb{E}[\varepsilon_r \mid \mathcal{F}_{r-1}]\rangle + I_r\mathbb{E}[\|\varepsilon_r\|^2 \mid \mathcal{F}_{r-1}] \\ &\leq L_r^2\|e_{r-1}\|^2 + I_r\nu_r^2. \end{aligned} \tag{21}$$

Taking expectations and iterating from $e_0 = 0$ gives Equation 19. Bounding each product by one and each ν_r by ν, then taking square roots, gives Equation 20. □

For a fixed schedule with n_{D} draft updates, the root mean squared error is at most $\nu\sqrt{n_{\mathrm{D}}}$ when $L_r \leq 1$ and $\nu_r \leq \nu$ almost surely. For a fixed DDDTT schedule, Equation 19 gives

$$\mathbb{E}\|e_5\|^2 \leq L_5^2 L_4^2 \sum_{r=1}^{3} \mathbb{E}[\nu_r^2] \prod_{i=r+1}^{3} L_i^2. \tag{22}$$

The target suffix adds no new draft error and reduces the inherited error when $L_5L_4 < 1$. The bound describes the root mean squared latent error over the stochastic model; it need not hold for each individual trajectory.

Assumptions needed for the random-walk interpretation. The mean-zero assumption must hold given the past states and routing observations; a zero average error over a dataset is insufficient. Independence across steps is not required, but errors must have zero conditional mean as stated in Equation 18. For fixed trained networks, the update difference at a fixed input is deterministic. Proposition 4 therefore describes an idealized error model, such as the first-order model with conditionally mean-zero logit perturbations; Proposition 1 remains the bound for arbitrary errors of the fixed models. Systematic bias or persistent error directions can invalidate square-root accumulation. The shared sampler noise cancels in the comparison, so a large sampling variance alone does not imply a large draft–target discrepancy.

Intuition behind entropy routing. We use draft attention entropy as a proxy for the local error variance ν_r^2. Proposition 3 gives a setting in which higher entropy corresponds to a larger leading error term, while Equation 21 shows how local variance contributes to the rollout error. A high entropy score suggests that further draft updates may add substantial error. The router then invokes the target to limit further deviation and, when the target updates are contractive, reduce the error already introduced by the draft prefix. A low score suggests a smaller additional error, so the method accepts some further deviation in exchange for the lower cost of draft updates.

The upgrade can restart the block or continue from the draft prefix. A restart removes local substitution error under the shared-context, shared-noise comparison. Continuing with the target avoids further draft errors and preserves the work already done; reducing the existing error requires the contraction condition above. When context errors propagate ($\rho > 0$), the larger weights on early-block errors in Proposition 2 motivate using full restarts earlier and retaining draft prefixes later. The target-only first block prevents draft error from entering the initial context.

This interpretation treats entropy as an indicator of the risk of additional error, rather than a direct measurement of the current distance to the target. For BlockStep, it requires the average entropy over heads, layers, and the three draft steps to predict the relevant error variances. That relation, and its ability to predict the benefit of an upgrade, require empirical validation beyond the local model in Proposition 3.