StrideAR: Strided Target Blocks for Fast Autoregressive Video Generation

Hudeifa Hassan
Statite

Abstract

Long-horizon autoregressive video generation repeatedly invokes large models, incurring substantial computational cost while allowing generation errors to compound across successive blocks. StrideAR formulates target-block allocation as an offline scheduling problem and introduces a training-free mixed-rollout decoder that periodically interleaves target and drafter computation across the generation horizon, eliminating online routing and its runtime scoring overhead. Empirically, varying model allocation across successive halves of the rollout reveals a non-additive interaction in which late-stage quality depends critically on early model assignments. To capture this coupled dynamic, we formulate a two-mode trajectory-drift model that establishes sufficient conditions under which interleaved rollouts achieve lower surrogate loss than either standalone generator, motivating a compute-constrained offline budget selection rule. On Krea Realtime 14B, VisionReward increases monotonically across the evaluated mixed-rollout target budgets, with intermediate allocations surpassing full target-only quality at substantially lower latency. On 53.6-second (72-block) rollouts, StrideAR achieves a $1.89\times$ *speedup over target-only generation while improving whole-video VisionReward, with gains concentrated across the middle and late segments of the rollout. Across Krea Realtime and Helios, StrideAR delivers up to* $10.54\times$ *speedup over target-only generation and* $1.84\times$ *over SDVG. These results establish target-block allocation as a scheduling problem governing both inference efficiency and long-horizon trajectory dynamics.*

1. Introduction

Autoregressive video models [7] generate video sequentially in blocks, conditioning each new block on the preceding generation. Producing long rollouts requires repeated execution of a large target model, making inference computationally expensive. Mixed-model video generation reduces this cost by delegating most blocks to a smaller drafter while reserving target-model computation for selected positions along the rollout. This reduces the frequency of expensive target-model evaluations, but leaves the allocation of a limited target-generation budget across the horizon as a central design problem.

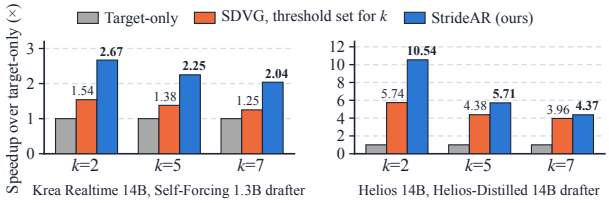


Figure 1. Speedup over target-only generation for StrideAR at each target budget k and for SDVG at the corresponding operating point (threshold selected using held-out prompts to approximate StrideAR's draft fraction), on two target–drafter pairs; each panel has its own axis. StrideAR is 2.04–$2.67\times$ faster than the target on Krea Realtime and 4.37–$10.54\times$ on Helios, and faster than SDVG at every budget. Full results in Table 1.

Existing speculative video methods, such as SDVG [6], use online image-quality scores to decide when to invoke the target model. This routing incurs substantial scoring overhead yet yields no statistically significant quality gain over random block selection at a matched target budget. The underlying limitation is that quality degradation arises from cumulative trajectory drift across consecutive draft blocks, which individual block scores fail to capture. Online routing therefore permits long uninterrupted draft runs in which errors compound, making the spacing of target interventions essential to controlling drift.

Generation quality depends on both the number and placement of target blocks, as well as how early model assignments condition later stages of the rollout. At a fixed target budget, quality degrades as the longest uninterrupted draft run increases across placement strategies, linking generation quality to the spacing of target interventions (§3.1). Moreover, crossing first-half and second-half model allocations from matched 36-block prefixes at $H = 72$ reveals a statistically significant non-additive interaction (§3.2): executing a strided mixed schedule in the first half elevates the quality of a second half generated entirely by the target model, demonstrating that early interventions alter downstream trajectory dynamics. Increasing the target budget improves quality monotonically across evaluated mixed-rollout budgets on Krea Realtime and Helios, surpassing standalone target-only quality on Krea Realtime at both 36-

block and 72-block horizons, though it reduces quality on MAGI-1 [18] where the base target model exhibits severe standalone drift (Appendix D).

These observations establish two coupled offline decisions: how many blocks the target should generate under a compute constraint and where those blocks should be placed. For any specified budget k, evenly spaced target blocks globally minimize the longest uninterrupted draft run by a direct partitioning argument (§4.3, Appendix B.1). Because target positions depend only on the horizon H and budget k, the schedule can be fixed before generation begins, eliminating online scoring overhead while enabling deterministic budget selection from an end-to-end latency target (§4.3).

We introduce **StrideAR**, a training-free mixed-rollout decoder that implements this principle through a predetermined, balanced strided schedule. Given a rollout horizon of H blocks and a target budget of k, StrideAR assigns exactly k blocks to the target model, with the remaining $H-k$ blocks generated by the drafter. Target positions are fixed before generation and spaced evenly across the rollout, minimizing the longest uninterrupted draft run without learned routers, auxiliary reward models, or per-block runtime scoring.

We evaluate StrideAR on Krea Realtime 14B [15] with a Self-Forcing 1.3B drafter and on Helios 14B [26] with its distilled 14B drafter. Across evaluated budgets at $H = 36$, StrideAR achieves 2.04–$2.67\times$ and 4.37–$10.54\times$ speedups over target-only generation, respectively (Figure 1), outperforming SDVG in both latency and quality at every matched operating point. Extending Krea Realtime rollouts to $H = 72$ (53.6 s), StrideAR achieves $1.89\times$–$2.71\times$ speedups over target-only generation while surpassing standalone target-only VisionReward [25] at $k \geq 10$.

This paper makes the following contributions:

- **Target-block allocation as trajectory optimization:** We formulate mixed-rollout decoding under a compute budget and contrast online perceptual routing, contiguous boundary allocation, and strided periodic scheduling within a unified framework (§2.4).
- **Empirical and theoretical characterization of coupled rollout drift:** Through controlled placement ablations (§3.1) and a bit-exact 2×2 shared-prefix factorial experiment at $H = 72$ (§3.2), we show that per-block routing provides no advantage over random selection, that allocation effects interact non-additively across rollout halves, and that coupled two-mode drift (§4.1) explains why balanced interleaving outperforms both contiguous schedules and standalone target-only rollouts.
- **StrideAR and long-horizon evaluation:** We introduce StrideAR, a training-free strided decoder with offline latency-constrained budget selection (§4), achieving $1.89\times$–$10.54\times$ speedups over target-only generation and up to $1.84\times$ over SDVG across Krea Realtime and Helios while exceeding target-only VisionReward on Krea Realtime at both $H = 36$ and $H = 72$ (§5).

2. Related Work

2.1. Speculative Decoding

Speculative decoding accelerates autoregressive generation by using a smaller model to propose future outputs and a larger target model to verify them. In language models, this verification is defined over a shared discrete vocabulary, allowing proposed tokens to be evaluated directly against the target distribution. Early methods preserve the target distribution through modified rejection sampling [2, 12], while later work improves proposal quality or reduces serialization in the draft–verify loop. Medusa [1] predicts multiple future tokens with auxiliary heads, EAGLE [13] and EAGLE-3 [14] construct stronger drafters from target-model features, DFlash [3] uses block-diffusion drafting, and Speculative Speculative Decoding [11] overlaps drafting with verification. Despite these differences, they operate in the same basic regime: the proposal consists of discrete tokens for which the target model provides a tractable token-level distribution. StrideAR addresses a different setting. In block-autoregressive video diffusion, each autoregressive step produces a continuous video block, so the token-level verification mechanism used in language-model speculative decoding is not directly available.

2.2. Speculative Decoding for Visual Autoregressive Models

Speculative decoding has also been extended to visual autoregressive models that generate discrete visual tokens. LANTERN [10] and GSD [20] relax token-level verification to account for redundancy among visually similar tokens, while SJD [23] and SCD [21] use Jacobi-style decoding without a separately trained drafter; SCD additionally demonstrates large speedups on token-autoregressive video generation. VVS [5] reduces target evaluations by skipping verification on a scheduled subset of steps. Despite these differences, all of these methods retain a discrete token sequence for which target probabilities are available, allowing verification to be exact, relaxed, or selectively skipped. The discrete-token verification framework does not extend directly to continuous video blocks. In block-autoregressive video generation, target allocation instead determines which model generates each committed block of the evolving trajectory.

2.3. Speculative Decoding for Block-Autoregressive Video

Block-autoregressive video diffusion generates a continuous latent block at each autoregressive step, as in Self-Forcing [7] and Wan-style models [22]. Unlike discrete-token generation, these models do not expose a tractable target probability for each generated block, so standard token-level acceptance tests cannot be applied directly. SDVG [6] addresses this by generating candidate blocks with a smaller drafter and using an auxiliary perceptual model to decide online whether each candidate should be retained or regenerated by the target. StrideAR changes this allocation mechanism. The target budget and target positions are fixed before generation, eliminating content-dependent routing and its associated scoring cost. This allocation also differs from methods such as T-Stitch [16], which switch between small and large models across denoising timesteps within a single sample. StrideAR instead allocates the target across autoregressive blocks, so the choice of model determines which blocks are committed to the trajectory and therefore condition subsequent generation. Speculative Forcing [4] instead accelerates few-step autoregressive generators by treating intermediate clean estimates as drafts and freezing tokens early once a verifier accepts them, reducing denoising steps per token within a single model rather than allocating blocks between a target and a drafter.

2.4. Problem Formulation: Target-Block Allocation

Consider a block-autoregressive video diffusion generator that synthesizes a rollout of H latent video blocks $B_{0:H-1} = (b_0, \dots, b_{H-1})$ sequentially, where each block b_t conditions on a context z_t constructed from previously committed blocks. We are given a high-capacity target model M_T with effective per-block latency c_T and a smaller drafter M_D with per-block latency $c_D \ll c_T$.

An allocation policy π assigns each block $t \in \{0, \dots, H-1\}$ to either the target or the drafter, realizing a rollout schedule $\sigma \in \{T, D\}^H$ with initial target anchor $\sigma_0 = T$ and target-block count $k(\sigma) = \sum_{t=0}^{H-1} \mathbb{I}\{\sigma_t = T\}$. Let $\mathcal{Q}(\pi)$ denote the expected trajectory quality over prompts and diffusion noise, and let $\mathrm{Lat}(\pi)$ denote the expected end-to-end rollout latency. Given a latency budget Λ, target-block allocation is the constrained trajectory optimization problem $\max_\pi \mathcal{Q}(\pi)$ subject to $\mathrm{Lat}(\pi) \le \Lambda$ (1). Over mixed-rollout operating points ($1 \le k < H$), end-to-end latency is governed by:

$$\mathrm{Lat}(\pi) \approx Hc_D + \mathbb{E}[k(\sigma)](c_T - c_D) + Hc_{\text{score}}\,\mathbb{I}\{\text{online routing}\}, \tag{2}$$

where c_{score} captures the per-block runtime overhead of candidate decoding, reward-model evaluation, and discarded draft generation in closed-loop routers (whereas standalone target-only execution at $k = H$ bypasses dual-model context updates).

We contrast three structural families for solving Eq. (1), summarized in Table C7:

Online Perceptual Routing: Generates a candidate block with M_D at every step, decodes candidate frames, and evaluates an auxiliary reward model $s_\phi(\tilde{b}_t)$ to trigger target regeneration whenever the score falls below a threshold τ. This incurs per-block scoring and wasted-draft overhead ($c_{\text{score}} > 0$) while leaving the maximum uninterrupted draft run $L_{\max}$ unbounded up to $H - k$.

Contiguous Boundary Allocation: Fixes a budget of k target blocks offline ($c_{\text{score}} = 0$) and concentrates them at the start (prefix warmup) or end (tail anchoring) of the rollout, leaving a single contiguous stretch of $H - k$ drafter blocks.

Strided Periodic Scheduling (StrideAR): Fixes k evenly spaced target anchors offline before generation ($c_{\text{score}} = 0$, no discarded drafts), partitioning drafter execution into k short intervals of length at most $\lceil H/k \rceil - 1$.

3. Empirical Analysis of Target-Block Allocation

In block-autoregressive generation, each committed block updates the conditioning context and influences subsequent blocks along the rollout. Allocating a fixed target-computation budget across a horizon of H blocks therefore governs not only local frame fidelity, but also how generation errors propagate through the evolving trajectory. Before introducing StrideAR, we examine two empirical properties of mixed-model rollouts that motivate offline strided scheduling: the dependence of generation quality on target-anchor spacing rather than online per-block scoring (§3.1), and the non-additive interaction of block allocations across halves of the rollout (§3.2).

3.1. Target Placement and Perceptual Routing

Closed-loop speculative decoders such as SDVG generate a candidate block with the drafter at every step and invoke an auxiliary reward model (ImageReward [24]) to decide online whether the candidate should be retained or regenerated by the target. To isolate whether per-block scoring identifies beneficial intervention points, we first compare SDVG's router against uniform random acceptance at a matched 75% acceptance rate ($\tau = -0.7$) on Krea Realtime ($H = 36$, Table C4). With the initial block fixed to the target in both policies, router-guided selection scores 0.07160 VisionReward versus 0.07313 for random acceptance (paired $t = +0.75$, not significant). Pooled across five matched acceptance rates from 75% to 89%, the router's paired difference over random selection is -0.0011 (95% CI $[-0.0037, +0.0015]$). Removing the forced first block

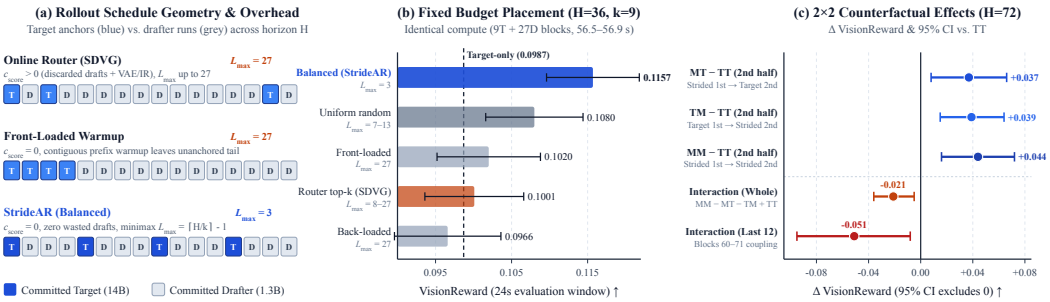


Figure 2. Empirical mechanism of StrideAR. **(a)** Online routing (SDVG) drafts and scores every step—discarding rejected drafts and permitting long unanchored draft runs ($L_{\max}$ up to 27)—whereas StrideAR fixes evenly spaced target anchors offline ($c_{\text{score}} = 0$, minimax $L_{\max} = \lceil H/k \rceil - 1 = 3$). **(b)** At fixed compute ($H = 36$, $k = 9$, 56.6 s), quality increases monotonically as $L_{\max}$ contracts, with balanced placement significantly outperforming all alternatives ($p_{\text{Holm}} \leq 0.034$) and standalone target-only generation. **(c)** Bit-exact 2×2 shared-prefix rollouts at $H = 72$ reveal that a strided first half elevates a target-only second half by $+0.037$ VisionReward (MT $-$ TT) with a non-additive interaction of -0.051 over the final 12 blocks.

lowers random acceptance to 0.06719, indicating that quality retention under SDVG stems from initial-block anchoring and the aggregate target budget rather than per-block perceptual scores.

To isolate the effect of anchor geometry at identical compute, Table C5 and Figure 2(b) compare five placement policies at a strictly fixed budget of $k = 9$ target blocks ($H = 36$, 75% draft share) on Krea Realtime without online scoring overhead.

All five policies execute the exact same number of target and drafter forwards, running within 1% end-to-end latency (56.5–56.9 s). Balanced placement ($L_{\max} = 3$) outperforms every alternative after Holm correction ($p_{\text{Holm}} \leq 0.034$). Quality degrades as target anchors become more concentrated and the longest uninterrupted drafter run $L_{\max}$ grows: every schedule whose maximum draft run reaches 27 blocks (front-loaded, router top-k, and back-loaded; 0.0966–0.1020) scores below both distributed schedules whose draft runs remain at or below 13 blocks (uniform random at 0.1080 and balanced at 0.1157). Moreover, balanced placement exceeds standalone target-only VisionReward (0.1157 versus 0.0987 under the same evaluation protocol): across the 39% of prompts on which standalone target-only generation degrades below the drafter over the rollout, StrideAR retains a $+0.0323$ VisionReward gain over target-only, while on the 55% of prompts where the 14B target scores higher than the drafter, periodic target reanchoring preserves target-only parity (-0.0032, not significant). Table C5 is evaluated under the 24-second VisionReward window without lazy target KV caching, which accounts for its 56.66 s latency and 0.1157 score.

3.2. Non-Additive Allocation Interactions Across the Rollout

If each target block contributed an independent, additive quality increment, the effect of second-half model allocation would be invariant to the schedule executed in the first half. To test whether early and late allocations act additively, we evaluate a 2×2 factorial design at $H = 72$ across 50 prompts on Krea Realtime, crossing first-half (blocks 0–35) and second-half (blocks 36–71) allocation between target-only (T) and balanced strided allocation (M, with 9 target blocks per half). All four arms (TT, TM, MT, MM) use identical per-block diffusion noise $\epsilon_{0:71}$, while matched second-half continuations (TT versus TM, and MT versus MM) branch from bit-identical 36-block prefix states (verified bit-for-bit against uninterrupted rollouts; Table C8, Figure 2(c)).

The results establish two properties of long-horizon mixed rollouts. First, replacing a target-only first half with a strided mixed schedule increases the VisionReward of a second half generated *entirely by the target model* by $+0.037$ (MT $-$ TT). Second, first-half and second-half allocations exhibit a statistically significant negative interaction of -0.021 over the full rollout and -0.051 over the final 12 blocks (blocks 60–71). Because the final 12-block window contains only second-half frames, this interaction is not an artifact of averaging scores across the two halves; rather, the committed context produced by early strided blocks alters the trajectory evolution of subsequent blocks. By comparison, substituting a single block in isolation yields a statistically unresolved whole-video difference of $+0.021$ (95% CI $[-0.001, +0.042]$), whereas sustained strided allocation across the 36-block prefix produces a clear downstream gain on the subsequent target-only half.

4. Method: StrideAR

The empirical findings in Section 3 establish two requirements for offline schedule design: target anchors must be distributed to prevent long uninterrupted drafter runs (§3.1), and early interleaving is required to stabilize the conditioning context inherited by later blocks (§3.2). We first formalize these requirements through a two-mode trajectory-drift model (§4.1), and then derive StrideAR's balanced placement rule and latency-constrained budget selector (§§4.2–4.3).

4.1. Trajectory-Drift Model and Scheduling Objective

To formalize when interleaving target and drafter blocks outperforms both standalone generators, consider a minimal two-state trajectory-drift model with state variables $x_t, y_t \geq 0$ ($x_0 = y_0 = 0$), representing target-accumulated drift and drafter-accumulated drift prior to block $t \in \{0, \ldots, H-1\}$:

$$\begin{aligned} &\text{Target block } (\sigma_t = T): x_{t+1} = x_t + \alpha, \quad y_{t+1} = \gamma y_t, \\ &\text{Drafter block } (\sigma_t = D): x_{t+1} = \beta x_t, \quad y_{t+1} = y_t + \delta, \end{aligned} \tag{3}$$

where $\alpha, \delta > 0$ are the per-block drift increments of the target and drafter, and $\beta, \gamma \in (0, 1]$ control how strongly each model contracts the drift accumulated by the other. Over a horizon of H blocks, a schedule $\sigma \in \{T, D\}^H$ with initial anchor $\sigma_0 = T$ incurs cumulative surrogate loss $\mathcal{L}_H(\sigma) = \sum_{t=0}^{H-1}(x_{t+1} + y_{t+1})$.

Proposition 1 (Additive versus Coupled Drift)**.** *Under Eq. (3) (proof in Appendix B.2):*

Uncoupled additive drift *($\beta = 1, \gamma = 1$): If neither model contracts the other's drift, then $\mathcal{L}_H(\sigma) = \sum_{s=0}^{H-1}(H - s)\big[\alpha\,\mathbb{I}\{\sigma_s = T\} + \delta\,\mathbb{I}\{\sigma_s = D\}\big]$. Consequently, (i) the 2×2 cross-segment interaction on $\mathcal{L}_H$ is identically zero for all schedules, and (ii) for any budget k, $\mathcal{L}_H$ is minimized by a contiguous schedule placing the remaining $k - 1$ target blocks contiguously at the beginning ($\{1, \ldots, k-1\}$ when $\alpha \leq \delta$) or at the end ($\{H - k + 1, \ldots, H - 1\}$ when $\alpha \geq \delta$).*

Cross-contraction *($\beta < 1, \gamma < 1$): If each model contracts the other's drift, then for any schedule σ with maximum target run length q and maximum drafter run length m, both drift states remain bounded for all $1 \leq t \leq H$ by*

$$x_t \leq \frac{\alpha q}{1 - \beta}, \qquad y_t \leq \frac{\delta m}{1 - \gamma}. \tag{4}$$

At fixed run bounds (q, m), $\mathcal{L}_H(\sigma)$ grows at most linearly in H, whereas pure target-only (T^H) and pure drafter-only (D^H) rollouts grow quadratically as $\frac{1}{2}\alpha H(H+1)$ and $\frac{1}{2}\delta H(H+1)$. Consequently, σ achieves strictly lower surrogate loss than both standalone generators whenever

$$H + 1 > \frac{2}{\min(\alpha, \delta)}\left(\frac{\alpha q}{1 - \beta} + \frac{\delta m}{1 - \gamma}\right). \tag{5}$$

Empirically, these two modes reflect the complementary failure regimes in §3.1—where standalone target-only degrades below the drafter on 39% of prompts (x_t) while the drafter trails on 55% (y_t)—and under cross-contraction ($\beta, \gamma < 1$) for $k \leq \lfloor H/2 \rfloor$ ($q = 1$), minimizing Eq. (4) reduces directly to minimizing $m = L_{\max}(\sigma)$.

4.2. Mixed-Rollout Decoding and Balanced Placement

To minimize $L_{\max}(\sigma)$ without runtime scoring overhead, StrideAR fixes a deterministic schedule $\mathcal{A}_k \subseteq \{0, \ldots, H-1\}$ of k target anchor positions before generation begins. At each block index $j \in \{0, \ldots, H-1\}$, the decoder dispatches execution directly to the assigned model: at anchor positions ($j \in \mathcal{A}_k$) the target model M_T generates and commits the block, and at all remaining positions ($j \notin \mathcal{A}_k$) the drafter M_D generates and commits it. Both models condition on the shared context constructed from previously committed blocks, so target anchors continue and re-anchor the evolving mixed rollout rather than reconstructing a separate target-only trajectory. Because $\mathcal{A}_k$ is known prior to inference, StrideAR never generates or discards candidate draft blocks at target positions, and lazy target context construction avoids redundant target updates during drafter steps.

The initial block ($j = 0$) is always assigned to the target model, and the remaining $k - 1$ target blocks are spaced evenly across the horizon H. For $1 \leq k \leq H$, the StrideAR anchor set is defined as

$$\mathcal{A}_k = \left\{ \left\lfloor \frac{jH}{k} + \frac{1}{2} \right\rfloor \;\middle|\; j = 0, \ldots, k-1 \right\}. \tag{6}$$

Because $\mathcal{A}_k$ depends only on (H, k), StrideAR requires no learned router, auxiliary reward model, or runtime perceptual evaluation.

4.3. Minimax Draft-Run Bound and Budget Selection

Proposition 2 (Minimax Draft-Run Bound). Partitioning the $H - k$ drafted blocks across the k intervals induced by $\mathcal{A}_k$ globally minimizes the longest uninterrupted drafter run among all schedules with k target anchors and $0 \in \mathcal{A}_k$ (proof in Appendix B.1):

$$L_{\max}(\mathcal{A}_k) = \left\lceil \frac{H}{k} \right\rceil - 1. \tag{7}$$

Substituting $m = \lceil H/k \rceil - 1$ and $q = 1$ ($k \leq H/2$) into Eq. (4) yields the tightest uniform drift bound attainable at budget k. At $H = 36$, setting $k \in \{2, 5, 7, 9\}$ restricts every uninterrupted drafter run to at most $17, 7, 5$, and 3 blocks; at $H = 72$, scaling the budget proportionally to $k \in \{4, 10, 14, 18\}$ preserves the exact same maximum drafter run bounds $L_{\max} \in \{17, 7, 5, 3\}$.

Over mixed-rollout budgets ($1 \leq k \leq \lfloor H/2 \rfloor$), because StrideAR executes k target blocks and $H - k$ drafter blocks with zero scoring overhead ($c_{\text{score}} = 0$), end-to-end latency follows the linear relation $\text{Lat}(k) \approx Hc_D + k(c_T - c_D)$ (8), where $c_D = 0.79$ s is the per-block drafter latency and $c_T = 2.80$ s is the effective per-block target latency including lazy context synchronization on Krea Realtime

(matching measured mixed-rollout latencies within 0.6 s at $H = 36$ and 0.4 s at $H = 72$; standalone target-only execution at $k = H$ bypasses dual-model context updates and runs at 87.6 s for $H = 36$ and 175.9 s for $H = 72$). Unlike closed-loop routers whose forward-pass counts vary with prompt content, StrideAR fixes execution counts prior to inference. For $\Lambda \geq (H-1)c_D + c_T$, because $q = 1$ for $1 \leq k \leq \lfloor H/2 \rfloor$, the drift bound in Eq. (4) contracts monotonically with k via $m = \lceil H/k \rceil - 1$, yielding the offline budget selector (Appendix B.3):

$$k(\Lambda) = \min\left(\left\lfloor \frac{H}{2} \right\rfloor, \left\lfloor \frac{\Lambda - Hc_D}{c_T - c_D} \right\rfloor\right). \tag{9}$$

Across both horizons on Krea Realtime, maintaining the same stride ratios k/H produces identical bounds on $L_{\max}$ $(17, 7, 5, 3)$ and nearly identical speedups ($1.89\times$–$2.71\times$), while whole-video VisionReward increases monotonically with k throughout the evaluated range ($k \leq H/4$).

5. Evaluation

5.1. Experimental Setup

Models and hardware. We evaluate StrideAR on two streaming block-autoregressive video diffusion models, which commit and decode each block before generating the next. The first pairs Krea Realtime 14B [15], a Wan2.1-T2V-14B model adapted for causal block-autoregressive generation through Self-Forcing, with the released Self-Forcing 1.3B DMD drafter [8]. The second pairs Helios-Base 14B [26] with Helios-Distilled 14B, a distilled few-step variant of the same model. Helios is a chunk-autoregressive diffusion transformer fine-tuned from Wan2.1-T2V-14B that generates 33-frame chunks conditioned on a multi-scale window of previous chunks and a first-frame anchor; the target denoises each chunk with 50 steps and classifier-free guidance, while the drafter uses six steps over three pyramid stages without guidance. Both pairs use their original checkpoints without additional training. All experiments run on a single NVIDIA H100 80GB GPU in BF16 with FlashAttention-3 [19]; Krea Realtime uses lazy target KV caching. A third pair, MAGI-1 24B with MAGI-1 4.5B, is reported as a case study in Appendix D.

Baselines. We compare StrideAR against three baseline families under otherwise identical configurations (prompts, seeds, rollout horizon, denoising schedule, precision, and hardware): (1) Target-only ($k = H$), running the 14B target at every step; (2) Draft-only ($k = 0$), running the drafter across the entire rollout (Krea Realtime); and (3) online perceptual routing (SDVG [6]), which drafts a candidate block at every step, scores it online with ImageReward, and regenerates it with the target when the score falls below a threshold τ. We use SDVG's released implementation and select its thresholds on held-out prompts to approximate StrideAR's target budgets (realized forward counts in Table C3). Fixed-budget placement alternatives at matched k (uniform random, front-loaded, back-loaded, and router top-k) are analyzed in §3.1.

Workloads and metrics. Our primary benchmark generates 36-block rollouts on 150 prompts per model pair, with every method run on the same prompts: Krea Realtime produces 26.6-second 832×480 videos (426 frames at 16 fps) from MovieGenVideoBench prompts [17], and Helios produces 74-second 640×384 videos (1,185 frames at 16 fps) from HeliosBench prompts. We sweep target budgets $k \in \{2, 5, 7\}$ ($L_{\max} \in \{17, 7, 5\}$, 5.6%–19.4% target share) and additionally evaluate $k = 9$ ($L_{\max} = 3$, 25.0%). To evaluate trajectory stability over extended horizons where autoregressive drift compounds, we additionally evaluate 72-block rollouts (53.6 s videos) on Krea Realtime across 50 prompts at matched stride densities $k \in \{4, 10, 14, 18\}$ ($L_{\max} \in \{17, 7, 5, 3\}$). Generation quality is measured by whole-video VisionReward, computed over 24 frames spanning each video, and Restricted VBench Quality-6 [9]: VBench's Quality Score restricted to subject consistency, background consistency, motion smoothness, dynamic degree, aesthetic quality and imaging quality, with leaderboard normalization and dynamic degree weighted one half. Systems performance is reported as mean end-to-end latency per video (prompt submission to final decoded frame, excluding model loading) and speedup relative to target-only generation.

5.2. Main Results ($H = 36$)

Table 1 and Figure 1 summarize end-to-end quality and latency on Krea Realtime and Helios at $H = 36$. On Krea Realtime, StrideAR achieves 2.04–$2.67\times$ speedup over target-only generation and is 1.63–$1.73\times$ faster than the corresponding SDVG configurations across the 150-prompt primary sweep ($k \in \{2, 5, 7\}$, Table 1). Its slowest setting ($k = 7$, $2.04\times$) is faster than SDVG's fastest ($1.54\times$), and at every matched budget it also scores higher on VisionReward. At $k = 5$ (13.9% target share), StrideAR reaches 0.0943 VisionReward in 39.0 s, above both standalone target-only generation (0.0914, 87.6 s) and every SDVG operating point. In the 100-prompt ablation run, increasing the budget to $k = 9$ ($L_{\max} = 3$) raises VisionReward to 0.1005 at $1.89\times$ speedup against its matched target-only baseline (46.84 s), compared with 0.0875 and 70.95 s for SDVG at the same budget. On Helios, where the drafter is a few-step distillation of the target, StrideAR achieves 4.37–$10.54\times$ speedup over target-only generation and is 1.10–$1.84\times$ faster than SDVG at every matched budget, with higher VisionReward at each of them.

Table 1. End-to-end quality and latency on Krea Realtime and Helios (one H100 each, $H = 36$). SDVG thresholds are selected on held-out prompts to approximate StrideAR's target budgets (realized forward counts in Table C3). Restricted VBench Quality-6 aggregates six VBench quality dimensions with VBench's leaderboard normalization and weights (§5.1). VR is whole-video VisionReward; speedups are relative to target-only generation. Higher is better for all quality columns.

Model & Method	Target Share	$L_{\max}$	Quality-6	Subject consist.	Background consist.	Motion smooth.	Dynamic degree	Aesthetic	Imaging	VisionReward ↑	Latency (s) ↓	Speedup ↑
Krea Realtime 14B, Self-Forcing 1.3B drafter ($H = 36$)												
Target-only	100.0%	0	78.06	83.74	88.70	98.29	82.00	58.56	69.11	0.0914	87.6	1.00×
Draft-only	0.0%	36	76.77	83.96	88.61	97.85	73.00	57.89	68.56	0.0828	29.0	3.02×
SDVG τ_1' (-2.2696)	—	—	76.95	83.29	89.17	97.74	78.00	57.73	67.63	0.0855	56.8	1.54×
StrideAR ($k = 2$)	5.6%	17	77.25	83.74	89.41	97.84	77.00	58.23	68.09	0.0876	32.8	**2.67**×
SDVG τ_2' (-2.1560)	—	—	77.01	83.42	89.23	97.76	78.00	57.68	67.73	0.0872	63.5	1.38×
StrideAR ($k = 5$)	13.9%	7	77.32	84.47	89.74	97.79	75.00	58.34	68.23	0.0943	39.0	**2.25**×
SDVG τ_4' (-1.9430)	—	—	76.91	83.44	89.26	97.86	76.00	57.67	67.78	0.0886	70.3	1.25×
StrideAR ($k = 7$)	19.4%	5	77.85	84.61	89.92	97.92	78.00	58.70	68.39	0.0972	42.9	**2.04**×
Helios 14B, Helios-Distilled 14B drafter ($H = 36$)												
Target-only	100.0%	0	78.98	80.45	90.00	97.52	100.00	64.90	63.59	0.1487	2911.5	1.00×
Draft-only	0.0%	36	76.95	84.77	92.06	98.11	83.33	60.58	55.20	0.1085	123.4	23.59×
SDVG τ_2 (-1.190)	—	—	75.89	85.06	92.04	98.41	66.67	60.30	56.64	0.1112	507.3	5.74×
StrideAR ($k = 2$)	5.6%	17	77.05	83.73	91.11	98.32	83.33	60.67	57.43	0.1223	276.2	**10.54**×
SDVG τ_5 (-0.859)	—	—	75.88	84.90	92.04	98.37	66.67	60.48	56.70	0.1156	665.4	4.38×
StrideAR ($k = 5$)	13.9%	7	77.17	85.40	91.90	98.29	75.00	62.39	57.61	0.1270	509.6	**5.71**×
SDVG τ_7 (-0.713)	—	—	75.69	85.39	92.52	98.36	58.33	61.83	57.31	0.1199	735.8	3.96×
StrideAR ($k = 7$)	19.4%	5	77.73	86.63	92.13	98.30	75.00	62.69	58.62	0.1375	666.5	**4.37**×

5.3. Generation Quality and Trajectory Stability

The quality gains of StrideAR on Krea Realtime follow from the coupled drift dynamics analyzed in Sections 3 and 4.1. While the 14B target produces sharper per-frame texture early in the rollout, unanchored 14B generation accumulates progressive drift over extended rollouts. Conversely, the 1.3B Self-Forcing drafter is trained to suppress autoregressive exposure bias, maintaining stable scene structure at the cost of finer visual detail. By bounding every drafter run to at most $L_{\max} = \lceil H/k \rceil - 1$ blocks and re-anchoring periodically with the 14B target, StrideAR mitigates both drift tendencies under the mechanism of Proposition 1. Across the 150-prompt Krea benchmark at $H = 36$, whole-video VisionReward increases monotonically from draft-only (0.0828) through $k = 2, 5, 7$ ($0.0876 \to 0.0943 \to 0.0972$) and reaches 0.1005 at $k = 9$ in the 100-prompt ablation run, surpassing standalone 14B generation (0.0914) for all $k \geq 5$.

Table 1 reports the full quality breakdown. On Krea Realtime, Restricted VBench Quality-6 rises from 77.25 at $k = 2$ to 77.85 at $k = 7$, approaching the target-only score of 78.06; at $k = 9$ (100-prompt ablation run), StrideAR retains 99.8% of the target-only score (77.87 versus 78.06). Helios follows the same direction: Quality-6 rises from 77.05 to 77.73 and VisionReward from 0.1223 to 0.1375 over the same budget range, and at every budget StrideAR scores above the corresponding SDVG configuration on both metrics.

5.4. Systems Performance and Overhead Breakdown

To isolate why StrideAR achieves higher speedups than online routing at matched target budgets, we decompose end-to-end latency on Krea Realtime at $H = 36$ and $k = 9$ by execution stage (Appendix C). StrideAR completes generation in 46.84 s, compared with 70.95 s for SDVG, a 1.51× speedup. SDVG spends 26.16 s (37% of its total latency) on online routing, including candidate decoding for scoring, ImageReward evaluation, and host transfers, while drafter (14.2 versus 14.3 s), target (18.0 versus 17.0 s), and VAE (9.8 s each) execution remain comparable. Because StrideAR fixes $\mathcal{A}_k$ offline before generation begins ($c_{\text{score}} = 0$), target blocks are dispatched directly to M_T without drafting or scoring a candidate first, and lazy context synchronization updates target states only at anchor transitions. In streaming, StrideAR and SDVG have nearly identical time to first frame (2.097 versus 2.098 s), but StrideAR reduces mean block-to-block latency from 1.915 to 1.193 s. Over all mixed-rollout budgets ($1 \leq k \leq H/2$), StrideAR's measured wall-clock latency matches the linear model $\text{Lat}(k) \approx Hc_D + k(c_T - c_D)$ ($c_D = 0.79$ s, $c_T = 2.80$ s) within 0.6 s at $H = 36$ and 0.4 s at $H = 72$, enabling direct offline budget selection via Eq. (9).

5.5. Ablation Studies

Target budget and architecture comparison. Varying $k \in \{2, 5, 7, 9\}$ at $H = 36$ yields monotonic quality scaling with k on both pairs as $L_{\max}(\mathcal{A}_k)$ contracts from 17 to 3. The two architectures illustrate both operating regimes of Proposition 1: on Krea Realtime, where the 14B target exhibits long-horizon drift contracted by the Self-Forcing drafter, strided interleaving exceeds standalone target quality at $k \geq 5$; on Helios, where the 14B target remains stronger than its distilled drafter (VisionReward 0.1487 versus 0.1085), StrideAR approaches the target-only ceiling as $L_{\max}$ decreases ($0.1223 \to 0.1270 \to 0.1375 \to 0.1395$ for $k = 2, 5, 7, 9$).

Placement geometry and online routing. Replaying SDVG's router-selected positions without online scoring isolates placement from routing cost: at $k = 7$ and $k = 9$, router-selected schedules leave draft runs of 24–26 blocks on Krea Realtime and 18–22 on Helios and score 0.0087–0.0179 VisionReward below StrideAR, while enabling the router online adds 15–25 s per video on Krea Realtime and about 120 s on Helios without changing the generated outputs. Fixed-budget placement ablations ($k = 9$, Table C5) and comparisons against random acceptance at matched rates (Table C4) are analyzed in §3.1; full results appear in Appendix C.

5.6. Long-Horizon Rollouts ($H = 72$)

To test whether balanced strided scheduling maintains trajectory quality when the generation horizon is doubled, Table C9 evaluates Krea Realtime over $H = 72$ blocks (53.6 s videos) across 50 prompts at target budgets $k \in \{4, 10, 14, 18\}$. These budgets preserve the same target fractions (5.6%–25.0%) and maximum uninterrupted draft run bounds $L_{\max} \in \{17, 7, 5, 3\}$ evaluated at $H = 36$.

At $H = 72$, standalone target-only generation scores 0.0599 VisionReward and draft-only generation 0.0399. Across the StrideAR sweep, whole-video VisionReward increases monotonically with the target budget from 0.0495 ($k = 4$, $2.71\times$ speedup) to 0.0780 ($k = 18$, $1.89\times$ speedup). Budgets $k = 10$ ($L_{\max} = 7$) and $k = 14$ ($L_{\max} = 5$) match or slightly exceed standalone target-only quality ($+0.0025$ and $+0.0053$) at $2.30\times$ and $2.07\times$ speedup. At $k = 18$ ($L_{\max} = 3$), StrideAR scores $+0.0181$ above target-only generation (95% CI $[+0.004, +0.032]$; Holm-adjusted $p = 0.052$ across the four budgets) and $+0.0380$ above draft-only generation, while cutting end-to-end latency from 175.9 s to 93.2 s. Across six 12-block rollout windows, the advantage of $k = 18$ over target-only generation is concentrated in the middle and late segments of the trajectory as standalone target drift accumulates. Because exposure drift compounds progressively after the shared early prefix, we test replication on a held-out 50-prompt confirmation split at $H = 72$: while early-half parity dilutes the whole-video average ($+0.006$, 95% CI $[-0.013, +0.025]$), the late-horizon advantage over blocks 36–71 replicates strongly at $+0.044$ (95% CI $[+0.016, +0.072]$).

6. Discussion and Limitations

Cross-model operating regimes. The results across Krea Realtime, Helios, and MAGI-1 delineate the empirical regimes in which mixed-rollout scheduling improves upon standalone generation. On Krea Realtime, the 14B target produces sharper per-frame detail early in the rollout but accumulates exposure drift over extended horizons, whereas the 1.3B Self-Forcing drafter is trained explicitly to stabilize rollout conditioning. The cross-contraction regime of Proposition 1 (§4.1) provides a mathematical mechanism for how balanced interleaving can achieve lower cumulative trajectory loss than either standalone generator, matching StrideAR's higher whole-video VisionReward relative to standalone 14B generation at $H = 36$ for $k \geq 5$ (1.89–$2.25\times$ faster) and at $H = 72$ for $k = 18$ ($1.89\times$ faster). On Helios ($H = 36$), where the 14B base model remains more stable than the distilled drafter across the horizon, StrideAR operates in the standard acceleration regime, monotonically recovering target quality as the maximum uninterrupted draft run $L_{\max}$ decreases. In contrast, on MAGI-1, where the base target model suffers severe standalone rollout degradation, injecting additional target blocks lowers quality relative to drafter-only generation. Mixed-rollout scheduling therefore requires that the target model act as a stabilizing anchor on the conditioning trajectory rather than introducing dominant rollout drift.

Offline schedule geometry versus per-block routing. In discrete-token speculative decoding, exact token-level verification guarantees that rejected drafts do not alter the target distribution. Continuous block-autoregressive video diffusion lacks a tractable block-level likelihood, forcing online routers such as SDVG to rely on single-block perceptual scores (ImageReward) to trigger target regeneration. Controlled comparisons at matched target budgets (§3.1) show that the router-minus-random VisionReward difference is -0.0011 across five acceptance thresholds, while offline router top-k selection creates long unanchored draft spans ($L_{\max}$ up to 27) that degrade downstream quality. Because autoregressive drift is a cumulative property of the committed prefix (§3.2), bounding the maximum uninterrupted draft run across the horizon is more effective than reacting locally to single-block perceptual scores after drift has already entered the context.

Limitations. StrideAR requires the target and drafter to share a compatible latent space and autoregressive block boundary so that committed blocks can be exchanged without re-encoding across incompatible representations. Second, while Eq. (9) selects the largest feasible target budget $k(\Lambda) \leq \lfloor H/2 \rfloor$, minimizing the worst-case drift bound $B(k)$ under measured per-block execution costs (c_D, c_T) (Appendix B.3), the resulting stride is uniform across prompts. Although uniform balanced placement outperforms post-hoc prompt-specific top-k routing under single-block ImageReward (§3.1), lightweight trajectory-level signals that estimate cumulative state divergence without decoding pixel frames remain an open direction for adaptive scheduling. Finally, Proposition 1 models cumulative drift through a linear two-state recurrence; extending this formulation to nonlinear perceptual metrics and combining block-level strided scheduling with intra-block step allocation [4, 16] are promising directions for future work.

7. Conclusion

We presented StrideAR, a training-free scheduling method for block-autoregressive video generation that allocates target computation under a latency constraint. Controlled placement ablations and a 2×2 shared-prefix factorial experiment demonstrate that target-block placement governs long-horizon rollout quality and that early and late allocation decisions interact non-additively. A two-mode trajectory-drift model establishes sufficient conditions under which interleaving achieves lower cumulative surrogate loss than either standalone generator, motivating a balanced

strided schedule that attains the minimax draft-run bound $L_{\max}(\mathcal{A}_k) = \lceil H/k \rceil - 1$ without online routing or perceptual scoring. Across Krea Realtime and Helios, StrideAR achieves up to $10.54\times$ speedup over target-only generation and up to $1.84\times$ over SDVG, while surpassing standalone 14B target-only VisionReward on Krea Realtime at both $H = 36$ and $H = 72$. These findings establish target-block allocation as a trajectory scheduling problem that governs both inference efficiency and the long-horizon stability of autoregressive video generation.

Large language models were used strictly for copy-editing and polishing the manuscript text; all code, experiments, mathematical derivations, and scientific analyses were conducted and verified by the authors.

References

[1] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pages 5209–5235, 2024. 2

[2] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023. 2

[3] Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. 2

[4] Hyunwook Choi, Seungryong Kim, Dongyoon Han, and Sangdoo Yun. Speculative forcing for efficient autoregressive video generation. *arXiv preprint arXiv:2610.22347*, 2026. 3, 8

[5] Haotian Dong, Ye Li, Rongwei Lu, Chen Tang, Shu-Tao Xia, and Zhi Wang. VVS: Accelerating speculative decoding for visual autoregressive generation via partial verification skipping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 2

[6] Yuezhou Hu and Jintao Zhang. Speculative decoding for autoregressive video generation. *arXiv preprint arXiv:2604.17397*, 2026. 1, 3, 6

[7] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 1, 3

[8] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self-forcing released checkpoints. Hugging Face repository `https://huggingface.co/gdhe17/Self-Forcing`, file `checkpoints/self_forcing_dmd.pt`, revision `2f8b7792`, 2025. 6

[9] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21807–21818, 2024. 6

[10] Doohyuk Jang, Sihwan Park, June Yong Yang, Yeonsung Jung, Jihun Yun, Souvik Kundu, Sung-Yub Kim, and Eunho Yang. LANTERN: Accelerating visual autoregressive models with relaxed speculative decoding. In *International Conference on Learning Representations (ICLR)*, 2025. 2

[11] Tanishq Kumar, Tri Dao, and Avner May. Speculative speculative decoding. In *International Conference on Learning Representations (ICLR)*, 2026. 2

[12] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 19274–19286, 2023. 2

[13] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pages 28935–28948, 2024. 2

[14] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 2

[15] Erwann Millon and Krea AI. Krea realtime 14b: Real-time, long-form AI video generation. Technical blog post, `https://www.krea.ai/blog/krea-realtime-14b`; model `krea/krea-realtime-video` on Hugging Face, 2025. 2, 6

[16] Zizheng Pan, Bohan Zhuang, De-An Huang, Weili Nie, Zhiding Yu, Chaowei Xiao, Jianfei Cai, and Anima Anandkumar. T-Stitch: Accelerating sampling in pre-trained diffusion models with trajectory stitching. In *International Conference on Learning Representations (ICLR)*, 2025. 3, 8

[17] 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. 6

[18] Sand.ai, Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, W. Q. Zhang, et al. MAGI-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025. 2

[19] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 6

[20] Junhyuk So, Juncheol Shin, Hyunho Kook, and Eunhyeok Park. Grouped speculative decoding for autoregressive image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15375–15384, 2025. 2

[21] Junhyuk So, Hyunho Kook, Chaeyeon Jang, and Eunhyeok Park. Speculative coupled decoding for training-free loss-

less acceleration of autoregressive visual generation. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. 2

[22] 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. 3

[23] Yao Teng, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Accelerating auto-regressive text-to-image generation with training-free speculative Jacobi decoding. In *International Conference on Learning Representations (ICLR)*, 2025. 2

[24] Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 3

[25] Jiazheng Xu, Yu Huang, Jiale Cheng, Yuanming Yang, Jiajun Xu, Yuan Wang, Wenbo Duan, Shen Yang, Qunlin Jin, Shurun Li, Jiayan Teng, Zhuoyi Yang, Wendi Zheng, Xiao Liu, Dan Zhang, Ming Ding, Xiaohan Zhang, Xiaotao Gu, Shiyu Huang, Minlie Huang, Jie Tang, and Yuxiao Dong. VisionReward: Fine-grained multi-dimensional human preference learning for image and video generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11269–11277, 2026. 2

[26] Shenghai Yuan, Yuanyang Yin, Zongjian Li, Xinwei Huang, Xiao Yang, and Li Yuan. Helios: Real real-time long video generation model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026. 2, 6

B. Theoretical Proofs

B.1. Proof of Proposition 2 (Minimum Longest Draft Run)

Let $\mathcal{A}_k = \{a_0, a_1, \ldots, a_{k-1}\} \subseteq \{0, \ldots, H-1\}$ with $0 = a_0 < a_1 < \cdots < a_{k-1}$ denote any schedule of k target blocks ($1 \le k \le H$), and set $a_k = H$. The k target blocks partition the $H-k$ drafter blocks into k contiguous runs of lengths $L_j = (a_{j+1} - a_j) - 1$ for $j = 0, \ldots, k-1$, satisfying $\sum_{j=0}^{k-1} L_j = H - k$. By the pigeonhole principle, the maximum run length $L_{\max} = \max_{0 \le j < k} L_j$ satisfies

$$L_{\max} \ge \left\lceil \frac{H-k}{k} \right\rceil = \left\lceil \frac{H}{k} \right\rceil - 1.$$

Under the StrideAR anchor placement $a_j = \lfloor jH/k + 1/2 \rfloor$, elementary bounds on the floor function give $a_{j+1} - a_j \le \lceil H/k \rceil$ for every $j \in \{0, \ldots, k-1\}$, and hence $L_j = a_{j+1} - a_j - 1 \le \lceil H/k \rceil - 1$. Thus $\mathcal{A}_k$ attains the global minimum $L_{\max}(\mathcal{A}_k) = \lceil H/k \rceil - 1$. □

B.2. Proof of Proposition 1 (Additive versus Coupled Drift)

Part 1 (Uncoupled additive drift, $\beta = \gamma = 1$). When $\beta = \gamma = 1$, iterating Eq. (3) from $x_0 = y_0 = 0$ yields $x_{t+1} = \sum_{s=0}^{t} \alpha\, \mathbb{I}\{\sigma_s = T\}$ and $y_{t+1} = \sum_{s=0}^{t} \delta\, \mathbb{I}\{\sigma_s = D\}$ for $t = 0, \ldots, H-1$. Summing $x_{t+1} + y_{t+1}$ over $t = 0, \ldots, H-1$ and exchanging the order of summation over $0 \le s \le t \le H-1$ gives

$$\begin{aligned}\mathcal{L}_H(\sigma) &= \sum_{t=0}^{H-1} \sum_{s=0}^{t} \big[\alpha\, \mathbb{I}\{\sigma_s = T\} + \delta\, \mathbb{I}\{\sigma_s = D\}\big] \\ &= \sum_{s=0}^{H-1} (H-s)\big[\alpha\, \mathbb{I}\{\sigma_s = T\} + \delta\, \mathbb{I}\{\sigma_s = D\}\big].\end{aligned}$$

For any schedule $\sigma = (p, s)$ formed by concatenating a first-half prefix $p \in \{T, D\}^{H/2}$ (indices $0 \le u < H/2$) and a second-half continuation $s \in \{T, D\}^{H/2}$ (indices $H/2 \le u < H$), the objective decomposes additively as $\mathcal{L}_H(p, s) = f_{\text{pre}}(p) + f_{\text{suf}}(s)$. Consequently, for any two prefixes p, p' and suffixes s, s', the 2×2 cross-difference vanishes identically:

$$\mathcal{L}_H(p', s') - \mathcal{L}_H(p', s) - \mathcal{L}_H(p, s') + \mathcal{L}_H(p, s) = 0.$$

Substituting $\mathbb{I}\{\sigma_s = D\} = 1 - \mathbb{I}\{\sigma_s = T\}$ yields $\mathcal{L}_H(\sigma) = \frac{1}{2}\delta H(H+1) + (\alpha - \delta)\sum_{s=0}^{H-1}(H-s)\,\mathbb{I}\{\sigma_s = T\}$. Because the positional weights $w_s = H - s$ are strictly decreasing in s, minimizing $\mathcal{L}_H(\sigma)$ subject to $\sigma_0 = T$ and $\sum_{s=0}^{H-1} \mathbb{I}\{\sigma_s = T\} = k$ assigns the remaining $k-1$ target blocks to the indices with largest weights $s \in \{1, \ldots, k-1\}$ when $\alpha < \delta$ (front-loaded), and to the indices with smallest weights $s \in \{H-k+1, \ldots, H-1\}$ when $\alpha > \delta$ (back-loaded); when $\alpha = \delta$, all schedules with budget k achieve identical loss.

Part 2 (Cross-contraction, $\beta < 1, \gamma < 1$). Let $q \ge 1$ and $m \ge 1$ denote the maximum contiguous run lengths of target and drafter blocks in σ. Suppose $x_{t_0} \le \frac{\beta\alpha q}{1-\beta}$ immediately prior to a contiguous target run of length $r \le q$. For each step $1 \le j \le r$ within the target run,

$$x_{t_0+j} = x_{t_0} + j\alpha \le \frac{\beta\alpha q}{1-\beta} + \alpha q = \frac{\alpha q}{1-\beta}.$$

If the target run is followed by at least one drafter block at step $t_0 + r$, the drafter update multiplies the state by $\beta < 1$, yielding $x_{t_0+r+1} = \beta x_{t_0+r} \le \frac{\beta\alpha q}{1-\beta}$ prior to the next target run. Since $x_0 = 0 \le \frac{\beta\alpha q}{1-\beta}$, induction over runs establishes $x_{t+1} \le \frac{\alpha q}{1-\beta}$ for all $0 \le t \le H-1$. By symmetric induction on drafter runs of length at most m followed by target contraction $\gamma < 1$, we obtain $y_{t+1} \le \frac{\delta m}{1-\gamma}$ for all $0 \le t \le H-1$, proving Eq. (4).

Summing Eq. (4) over $t = 0, \ldots, H-1$ bounds the cumulative loss of the interleaved schedule by

$$\mathcal{L}_H(\sigma) = \sum_{t=0}^{H-1} (x_{t+1} + y_{t+1}) \le H\left(\frac{\alpha q}{1-\beta} + \frac{\delta m}{1-\gamma}\right).$$

For standalone target-only generation (T^H), $y_{t+1} = 0$ and $x_{t+1} = \alpha(t+1)$, giving $\mathcal{L}_H(T^H) = \frac{1}{2}\alpha H(H+1)$; similarly, for standalone drafter-only generation (D^H), $\mathcal{L}_H(D^H) = \frac{1}{2}\delta H(H+1)$. Requiring $H\big(\frac{\alpha q}{1-\beta} + \frac{\delta m}{1-\gamma}\big) < \frac{1}{2}\min(\alpha, \delta) H(H+1)$ and dividing by $H > 0$ yields the sufficient condition in Eq. (5). □

B.3. Monotonic Drift Bound in the Sparse-Target Regime

For any balanced strided schedule $\mathcal{A}_k$ in the sparse-target regime $1 \le k \le \lfloor H/2 \rfloor$, consecutive target indices satisfy $a_{j+1} - a_j = \lfloor (j+1)H/k + 1/2 \rfloor - \lfloor jH/k + 1/2 \rfloor \ge \lfloor H/k \rfloor \ge 2$. Every pair of target blocks is therefore separated by at least one drafter block, fixing the maximum target run length to $q = 1$ while the maximum drafter run length is $m = L_{\max}(\mathcal{A}_k) = \lceil H/k \rceil - 1$ by Proposition 2. Substituting $q = 1$ and $m = \lceil H/k \rceil - 1$ into Eq. (4) gives the uniform per-step upper bound

$$B(k) = \frac{\alpha}{1-\beta} + \frac{\delta(\lceil H/k \rceil - 1)}{1-\gamma},$$

which is non-increasing in k for all $1 \le k \le \lfloor H/2 \rfloor$. Subject to the linear mixed-rollout latency constraint $\mathrm{Lat}(k) \approx Hc_D + k(c_T - c_D) \le \Lambda$, the largest feasible budget $k(\Lambda) = \min\big(\lfloor H/2 \rfloor, \lfloor (\Lambda - Hc_D)/(c_T - c_D) \rfloor\big)$ in Eq. (9) attains the minimum of $B(k)$ over all feasible budgets in $1 \le k \le \lfloor H/2 \rfloor$. □

C. Extended Results

C.1. Horizon and Hardware Controls

Table C1a reproduces SDVG's published configuration on its own rig and horizon, and Table C1b sweeps the horizon and the hardware.

Table C1a: Faithfulness reproduction of SDVG's published headline configuration ($\tau = -0.7$, the first row of its Table 1). Published cells (pub) are SDVG's Table 1 values (Hu & Zhang, 2026, Table 1): 1003 MovieGenVideoBench prompts, 2×A6000, $H = 9$, $\tau = -0.7$. Reproduction cells (repro) are our measurements on the same rig, horizon, and configuration, on the frozen evaluation split rather than a subset of their 1003 prompts. Pre-registered pass criterion: speedup within $\pm 0.15\times$ of published, accept within ± 5 points; the reproduction meets both (accept +0.4 points, speedup $-0.15\times$). The published draft-only endpoint (0.0644 VR, 25.7 s, 3.77×; Hu & Zhang, 2026, Table 1) was not part of the reproduction and is omitted. RTF and latency are the statistics defined in §5.1.

Arm	VR pub / repro	RTF pub / repro	Latency s (speedup) pub / repro	Accept % pub / repro	Scorer s/block pub / repro	Peak mem GB (repro)
Target-only	0.0788 / —	— / 0.0591	97.0 (1.00×) / 107.9 (1.00×)	—	—	39.91 / 16.52
SDVG τ=-0.7	0.0773 / —	— / 0.0952	60.9 (1.59×) / 75.0 (1.44×)	73.1 / 73.5	≈0.6 / 1.08	39.96 / 19.95

Table C1b: Runtime across hardware configurations and generation horizons. Video duration is fixed at 6.375 s for $H = 9$ and 26.625 s for $H = 36$, with identical durations across methods. RTF denotes mean per-video real-time factor; latency denotes mean generation time; speedup is computed from mean latencies (§5.1). On 2×A6000, SDVG's variable latency produces different rankings under mean RTF and mean latency: although StrideAR has lower mean latency at both horizons, SDVG has higher mean per-video RTF. VR here is the 1-frame-per-second VisionReward over the first 24 s used for this horizon and hardware sweep; every VR cell is an absolute score on that ruler, not a difference from target-only, so it differs from the whole-video VR of Table 1: on the same 1×H100, $H = 36$ videos, target-only scores 0.0987 here and 0.0914 there, SDVG $\tau = -0.7$ 0.1034 and 0.0954, and StrideAR $k = 9$ 0.1157 and 0.1005.

Rig	H	Arm	VR	RTF	Latency s (speedup)	Accept %	Scorer s/block	Peak mem GB
2×A6000	9	Target-only	0.0665	0.0591	107.9 (1.00×)	—	—	39.91 / 16.52
2×A6000	9	SDVG τ=-0.7	−0.0137	0.0952	75.0 (1.44×)	73.5	1.08	39.96 / 19.95
2×A6000	9	StrideAR k=3 (target-budget matched, anchors 0/3/6)	−0.0198	0.0930	68.5 (1.57×)	66.7	—	39.91 / 16.52
2×A6000	36	Target-only	0.0747	0.0614	433.8 (1.00×)	—	—	39.94 / 16.52
2×A6000	36	SDVG τ=-0.7	−0.0050	0.0834	366.5 (1.18×)	49.7	1.07	39.99 / 19.95
2×A6000	36	StrideAR k=19 (target-budget matched)	0.0049	0.0767	347.1 (1.25×)	47.2	—	39.94 / 16.52
1×H100	9	Target-only	0.0952	0.2550	26.9 (1.00×)	—	—	55.67
1×H100	9	SDVG τ=-0.7	0.0944	0.3379	19.5 (1.38×)	86.5	0.84	59.15
1×H100	9	StrideAR k=2 (target-budget matched, anchors 0/5)	0.0898	0.4203	16.0 (1.68×)	77.8	—	55.67
1×H100	36	Target-only	0.0987	0.3039	87.6 (1.00×)	—	—	45.50
1×H100	36	SDVG τ=-0.7	0.1034	0.3422	87.0 (1.01×)	63.1	0.66	48.88
1×H100	36	StrideAR k=9 (accept-nearest)	0.1157	0.5708	46.7 (1.88×)	75.0	—	45.38

C.2. Example Generations

Figure C1 shows frames from four evaluation prompts, chosen by hand for visual variety, generated by target-only, SDVG with its threshold set for $k = 7$, and StrideAR at $k = 7$. The videos come from the main evaluation; the times shown are the Table 1 means for each method.

C.3. Paired Comparisons with SDVG

Table C3 pairs each StrideAR budget with an SDVG configuration under three matching rules and reports the paired VisionReward difference.

C.4. Online Routing on Krea Realtime

These experiments precede the main evaluation and use SDVG's released ImageReward router on Krea Realtime. The router decides, for each drafted block, whether to keep it or regenerate it with the target; the first block is always regenerated.

The aggressive end of the published sweep leaves the router inactive. SDVG's paper sweeps the fixed threshold from $\tau = -0.7$, its headline configuration, to $\tau = -2.5$, its fastest operating point (2.09× at 88.9% acceptance on its own benchmark). At $\tau = -2.5$ on Krea Realtime at $H = 36$, the router rejected none of the drafted blocks. Every video's target share came only from the forced first block, so the measured draft share was 35/36 = 97.2% on every prompt. At $H = 18$, the draft share of 94.4% is the same arithmetic, 17/18.

Router-guided selection does not beat random selection. At the matched 75% rate, random acceptance scores 0.0015 higher than the router (paired $t = +0.75$, not significant). Repeating the comparison at five matched acceptance rates from 75% to 89% gives the same result at every rate (paired $|t| \leq 1.28$), with random acceptance scoring at least as high as the router at three of the five. Pooled over the four rates at which the two policies differ (at 89% both accept every drafted

block), the router's paired advantage over random acceptance is -0.0011 VisionReward, with a 95% confidence interval of $[-0.0037, +0.0015]$ over 100 prompts. The 95% confidence interval upper bound for per-block scoring is therefore $+0.0015$ (about 2% of the router's mean score), while the point estimate favors random acceptance. Removing the forced first block lowers quality to 0.06719, so the quality in SDVG's router comes from forcing the first block to the target and from the acceptance rate, not from ImageReward's per-block scores. SDVG's own ablation agrees: forcing the first block and accepting at random gives 0.0771, against 0.0773 for the full router (Hu & Zhang, 2026, Table 2).

C.5. Placement Geometry and Trajectory Stabilization on Krea Realtime

Table C5 compares five placement policies at a strictly fixed budget of $k = 9$ target blocks ($H = 36$, 75% draft share) on Krea Realtime. All policies execute the exact same number of target and drafter forwards, running within 1% end-to-end latency (56.5–56.9 s, measured without lazy target KV caching). Quality is evaluated using the 1-fps VisionReward protocol over the first 24 s (as in Table C1b), where target-only generation scores 0.0987.

Balanced placement outperforms every alternative after Holm correction for the four comparisons (adjusted $p \leq 0.034$). $L_{\max}$ bounds the longest stretch the drafter runs without target correction; schedules that share a bound need not score

Table C3: Paired comparisons under three frozen rules. Equal acceptance: τ'_j selected for StrideAR $k = j$'s draft share. Nearest target-forward: pairs by target forwards. Nearest-RTF: SDVG's fastest configuration (τ'_1, RTF 0.471) is slower than StrideAR's slowest ($k = 9$, RTF 0.571), so no pair meets a matching tolerance and every StrideAR point is compared with τ'_1. Differences are paired over the same prompts and use the whole-video VisionReward of Table 1.

Matching rule (exact configs)	SDVG	StrideAR	ΔVR, StrideAR−SDVG
Equal acceptance: k=2 vs τ'_1	τ'_1 · RTF 0.4711	RTF 0.8111	+0.0021
Equal acceptance: k=3 vs τ'_2	τ'_2 · RTF 0.4658	RTF 0.7591	+0.0040
Equal acceptance: k=5 vs τ'_3	τ'_3 · RTF 0.4353	RTF 0.6827	+0.0071
Equal acceptance: k=7 vs τ'_4	τ'_4 · RTF 0.3986	RTF 0.6214	+0.0086
Equal acceptance: k=9 vs τ'_5	τ'_5 · RTF 0.3956	RTF 0.5708	+0.0129
Pooled, equal acceptance	—	—	+0.0069
Nearest-RTF: k=2 vs τ'_1	τ'_1 · RTF 0.4711 (SDVG's fastest)	RTF 0.8111	+0.0021
Nearest-RTF: k=3 vs τ'_1	τ'_1 · RTF 0.4711	RTF 0.7591	+0.0048
Nearest-RTF: k=5 vs τ'_1	τ'_1 · RTF 0.4711	RTF 0.6827	+0.0088
Nearest-RTF: k=7 vs τ'_1	τ'_1 · RTF 0.4711	RTF 0.6214	+0.0117
Nearest-RTF: k=9 vs τ'_1	τ'_1 · RTF 0.4711	RTF 0.5708	+0.0149
Pooled, nearest-RTF	—	—	+0.0085
Nearest target-forward: $\tau'2$ vs k=2	1.7 tgt / 35 draft	2.0 tgt / 34 draft	+0.0013
Nearest target-forward: $\tau'3$ vs k=3	2.9 tgt / 35 draft	3.0 tgt / 33 draft	+0.0032
Nearest target-forward: $\tau'4$ vs k=5	4.3 tgt / 35 draft	5.0 tgt / 31 draft	+0.0057
Nearest target-forward: $\tau'5$ vs k=7	6.1 tgt / 35 draft	7.0 tgt / 29 draft	+0.0097
Nearest target-forward: $\tau'5$ vs k=9	6.1 tgt / 35 draft	9.0 tgt / 27 draft	+0.0129
Pooled, forward-count-matched	—	—	+0.0065

Table C4: Router-guided versus random acceptance on Krea Realtime at a matched 75% acceptance rate (SDVG threshold $\tau = -0.7$). VisionReward is paired across the same prompts; retention is relative to target-only generation.

Selection	First block forced to target	VisionReward ↑	Retention
ImageReward router (SDVG)	✓	0.07160	94.5%
Random acceptance	✓	0.07313	96.5%
Random acceptance	×	0.06719	88.7%

Table C5: Placement of nine target blocks on Krea Realtime ($H = 36$, 1×H100, same prompts and the same 75% draft share for every row). $L_{\max}$ is the longest uninterrupted draft run; for random and router-selected schedules it varies by prompt. VisionReward is the mean over prompts ± its 95% margin of error. Paired t statistics and two-sided p-values are for Balanced minus the row; Holm-adjusted p-values correct for the four comparisons.

Placement	$L_{\max}$	VisionReward ↑	Latency (s)	Paired t (vs. Balanced)	Two-sided p-value	Holm-adjusted p
Balanced (StrideAR)	**3**	**0.1157 ± 0.0061**	56.66	—	—	—
Uniform random	7–13	0.1080 ± 0.0064	56.55	+2.14	0.034	0.034
Front-loaded	27	0.1020 ± 0.0068	56.91	+2.45	0.015	0.030
Router top-k (SDVG scores)	8–27	0.1001 ± 0.0065	56.54	+2.61	0.010	0.030
Back-loaded	27	0.0966 ± 0.0070	56.76	+3.44	0.0008	0.0032

identically. Every schedule whose bound reaches 27 (front-loaded, back-loaded and router top-k; 0.0966–0.1020) scores below every schedule whose bound stays at or below 13 (balanced and uniform random; 0.1080–0.1157). Quality degrades as anchors become more concentrated and the longest uninterrupted draft run $L_{\max}$ grows: front-loaded ($L_{\max} = 27$, 0.1020), router top-k ($L_{\max} \in [8, 27]$, 0.1001) and back-loaded ($L_{\max} = 27$, 0.0966) all trail the distributed schedules, whereas balanced StrideAR ($L_{\max} = 3$, 0.1157) outperforms both the concentrated and the random placements ($L_{\max} \in [7, 13]$, 0.1080) without stochastic variance or scoring overhead. Moreover, StrideAR exceeds target-only VisionReward (0.1157 versus 0.0987) because the 14B target exhibits progressive drift on 39% of prompts, where the Self-Forcing 1.3B drafter remains temporally stable (+0.0323 gain retained by StrideAR), while on the 55% of prompts where the 14B target is stronger, periodic target re-anchoring preserves target-only parity (-0.0032, not significant).

C.6. Latency Breakdown and Streaming on Krea Realtime

Table C6a breaks down end-to-end generation latency by execution stage, and Table C6b reports single-request streaming performance (§5.4).

Table C6a: End-to-end latency breakdown for Krea Realtime on one H100 ($H = 36$, StrideAR $k = 9$, lazy target KV caching). Times are reported in milliseconds per video, except total latency in seconds. SDVG spends 26.16 s on online routing, while StrideAR eliminates this overhead with comparable drafter, target, and VAE execution costs.

Method	Drafter	Target	Router	VAE decode	Context building	Frame delivery & host	Total = latency (s)
Target-only	—	73,558	—	9,869	604	4,586	88.62
SDVG (τ_5')	14,297	17,008	26,163	9,800	832	2,851	70.95
StrideAR	14,156	18,039	**—**	9,819	659	4,167	**46.84**

Table C6b: Single-request streaming performance on Krea Realtime ($H = 36$, StrideAR $k = 9$). Each generated block represents 0.75 s of video. We report time to first frame, mean and 95th-percentile block delivery intervals, and real-time factors. StrideAR reduces mean block-to-block latency from 1.915 s to 1.193 s relative to SDVG, with nearly identical time to first frame.

Method	Time to first frame (s) ↓	Block-to-block mean (s) ↓	Block-to-block p95 (s) ↓	Steady-state RTF ↑	End-to-end RTF ↑
Target-only	1.803	2.384	2.396	0.315	0.300
SDVG (τ_5')	2.098	1.915	3.737	0.392	0.375
StrideAR	2.097	**1.193**	**2.792**	**0.629**	**0.568**

C.7. Allocation Families

Table C7 compares the three target-block allocation families of §2.4.

Table C7: Structural comparison of target-block allocation families across a rollout of H blocks with target budget k.

Property	Online Perceptual Routing (SDVG)	Contiguous Allocation (Warmup / Tail)	Strided Scheduling (StrideAR)
Target Positions $\mathcal{A}_k$	$\sigma_t = T$ if $s_\phi(\tilde{b}_t) < \tau$ ($\sigma_0 = T$)	$\{0, \dots, k-1\}$ or $\{0\} \cup \{H-k+1, \dots, H-1\}$	$\left\{\lfloor jH/k + \frac{1}{2} \rfloor : j = 0, \dots, k-1\right\}$
Decision Time	Per block, online at runtime	Fixed offline before generation	Fixed offline before generation
Runtime Overhead	Decode + score every draft ($c_{\text{score}} > 0$)	None ($c_{\text{score}} = 0$)	None ($c_{\text{score}} = 0$)
Longest Draft Run $L_{\max}$	Content-dependent, up to $H - k$	$H - k$	$\lceil H/k \rceil - 1$ (global minimum)

C.8. Counterfactual 2×2 Allocation Study

Table C8 reports the 2×2 allocation study of §3.2.

Table C8: Counterfactual 2×2 allocation study on Krea Realtime ($H = 72$, 50 prompts), crossing first-half and second-half schedules between target-only (T) and strided mixed (M, $k = 9$ per half).

Trajectory Schedule ($H = 72$, Blocks 36–71 vs. TT)	Δ VisionReward	95% Confidence Interval
MT − TT (Strided first half, Target-only second half)	+0.037	[+0.008, +0.066]
TM − TT (Target-only first half, Strided second half)	+0.039	[+0.015, +0.064]
MM − TT (Strided first half, Strided second half)	+0.044	[+0.016, +0.072]
Interaction (MM − MT − TM + TT), Whole Video	−0.021	[−0.036, −0.005]
Interaction (MM − MT − TM + TT), Final 12 Blocks	−0.051	[−0.095, −0.008]

C.9. Long-Horizon Rollouts ($H = 72$)

Table C9 reports the long-horizon rollouts of §5.6.

Table C9: Long-horizon rollouts on Krea Realtime ($H = 72$, 53.6 s videos, 50 prompts).

Method ($H = 72$, Krea 14B)	$L_{\max}$	Whole-Video VR ↑	Δ VR vs. Target-only	Δ VR vs. Draft-only	Latency (s) ↓	Speedup ↑
Target-only	0	0.0599	—	+0.0200	175.9	1.00×
Draft-only	72	0.0399	−0.0200	—	56.7	3.10×
StrideAR ($k = 4$)	17	0.0495	−0.0104	+0.0096	65.0	2.71×
StrideAR ($k = 10$)	7	0.0624	+0.0025	+0.0225	76.6	2.30×
StrideAR ($k = 14$)	5	0.0652	+0.0053	+0.0253	85.0	2.07×
StrideAR ($k = 18$)	3	0.0780	+0.0181 [+0.004, +0.032]	+0.0380	93.2	1.89×

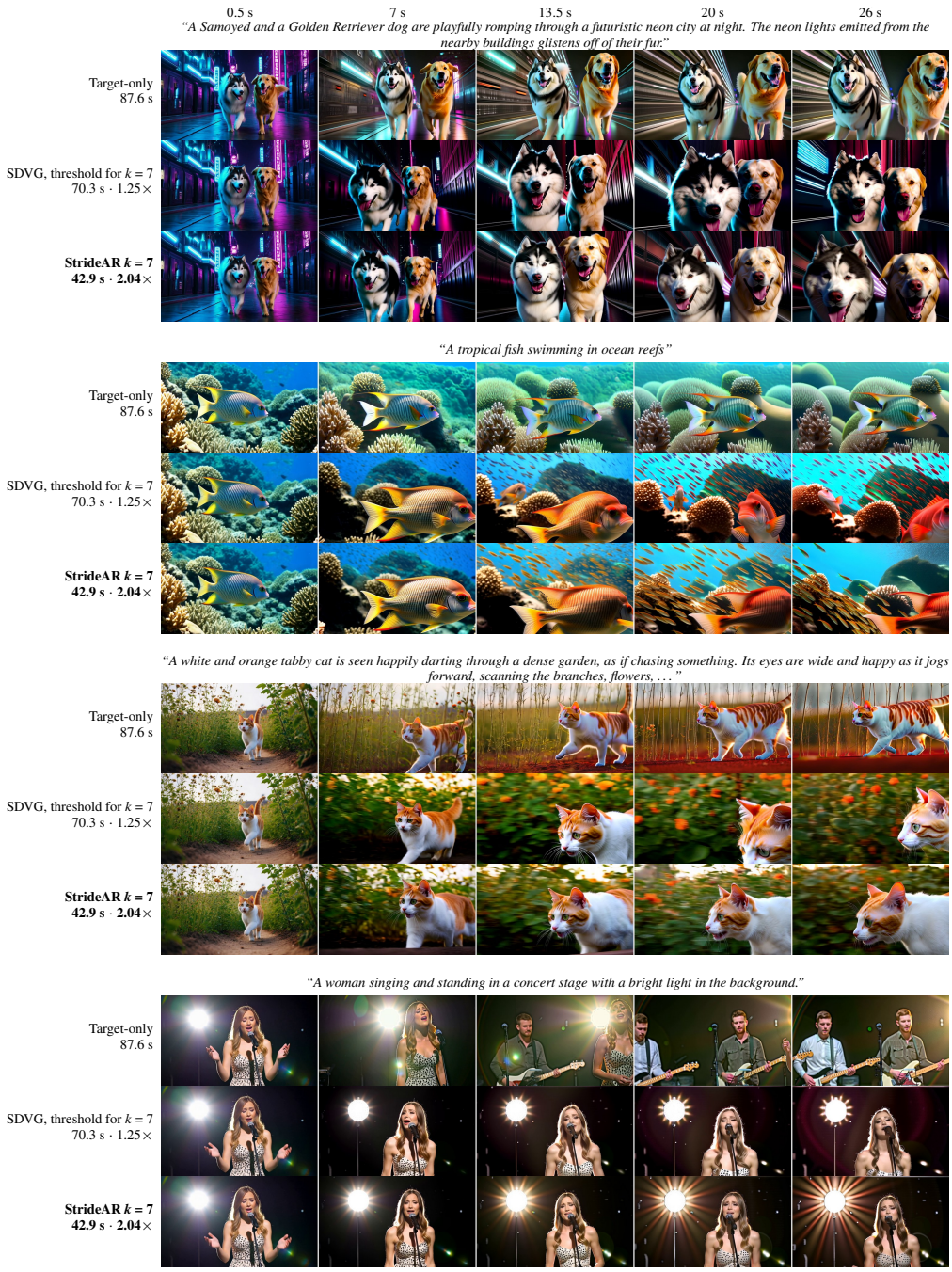


Figure C1: Example generations on Krea Realtime at $H = 36$ (26.6 s videos). Rows: target-only, SDVG with its threshold set for $k = 7$, and StrideAR $k = 7$, with the same prompt and seed in every row; columns are frames at 0.5, 7, 13.5, 20 and 26 s. Under each method is its mean generation time and speedup from Table 1 (one H100, lazy target KV caching, which leaves the generated frames unchanged).

D. MAGI-1: Target Budget and Generation Quality

We evaluate MAGI-1 24B as the target and MAGI-1 4.5B as the drafter. Under the evaluated settings, increasing target involvement reduces StrideAR's Quality-6 score. This experiment examines how target-budget selection depends on the relative quality of the two models.

D.1. Experimental Setup

We use the MAGI-1 24B and 4.5B base checkpoints without distillation or quantization. Both models use 32 denoising steps; the 4.5B checkpoint's default is 64. The workload consists of 150 image-to-video prompts initialized from VBench++ I2V first frames. Each video contains 36 one-second chunks at 24 fps and a resolution of 1280 $\times$ 720.

Experiments run on eight H200 GPUs with eight-way Ulysses context parallelism and no pipeline parallelism. Both models execute across all eight GPUs. KV caches remain on the GPUs, and the drafter's cache is reconstructed in the layout required by the target when switching models. End-to-end latency includes T5 text encoding.

We evaluate StrideAR at target budgets $k \in \{2, 5, 7\}$. SDVG thresholds are selected on held-out prompts to approximate the corresponding draft fractions; the realized fractions differ and are reported in Table D1. StrideAR speedups use target-only measurements on the same 150 prompts. SDVG speedups use the target-only reference from its separate evaluation run.

D.2. Quality and Latency

Draft-only generation completes in 856.4 seconds, compared with 2094.2 seconds for target-only generation. It scores 74.39 on Quality-6 and 0.0785 on VisionReward, compared with 70.74 and 0.0203 for target-only generation. StrideAR at $k = 2$ scores higher on Quality-6 (75.87) but lower on VisionReward (0.0634), and has the highest Quality-6 in Table D2.

Increasing StrideAR's target budget reduces Quality-6 from 75.87 at $k = 2$ to 74.93 at $k = 5$ and 72.17 at $k = 7$. Latency rises from 745.1 to 1114.0 seconds across the same budgets. StrideAR achieves 1.88-2.81$\times$ speedup over its target-only reference (Tables D1–D2).

These results establish that additional target computation does not necessarily improve generation quality.

Table D1: End-to-end quality and performance of StrideAR and SDVG on MAGI-1 (8$\times$H200), with $H = 36$. Speedups are relative to target-only generation; the separate SDVG rerun uses its own target-only reference.

Model	Method	VR	RTF	Latency (s)	Speedup	Accept (%)	Mem (GB)
MAGI-1 24B MAGI-1 4.5B drafter 1280$\times$720 · 36 chunks · 8$\times$H200	Target-only	0.0203	0.0172	2094.2	1.00$\times$	—	78.98
	Draft-only	0.0785	0.0420	856.4	2.45$\times$	—	72.30
	SDVG τ_1' (-1.5489)	0.0636	0.0280	1284.8	1.63$\times$	62	92.92
	StrideAR (k=2)	0.0634	0.0483	745.1	**2.81**$\times$	94	82.50
	SDVG τ_2' (-1.1082)	0.0560	0.0242	1485.3	1.41$\times$	75	92.92
	StrideAR (k=5)	0.0670	0.0383	940.4	**2.23**$\times$	86	88.70
	SDVG τ_3' (-1.0194)	0.0012	0.0193	1869.8	1.12$\times$	83	92.92
	StrideAR (k=7)	0.0143	0.0323	1114.0	**1.88**$\times$	81	89.83

Table D2: Generation quality on MAGI-1 across target-only, SDVG, and StrideAR operating points.

Model	Method	Quality-6	Subject consist.	Background consist.	Motion smooth.	Dynamic degree	Aesthetic	Imaging	VisionReward
MAGI-1 24B MAGI-1 4.5B drafter	Target-only	70.74	71.75	82.84	98.82	100.00	45.57	53.02	0.0203
	Draft-only	74.39	78.47	85.15	98.36	100.00	50.94	58.29	0.0785
	SDVG τ_1' (-1.5489)	72.92	77.36	87.17	99.05	100.00	50.70	46.64	0.0636
	StrideAR (k=2)	75.87	81.75	88.58	98.68	100.00	52.97	54.83	0.0634
	SDVG τ_2' (-1.1082)	72.79	79.26	86.04	98.78	100.00	50.60	46.27	0.0560
	StrideAR (k=5)	74.93	79.88	87.96	98.71	100.00	53.86	51.69	0.0670
	SDVG τ_3' (-1.0194)	75.21	83.74	84.35	98.91	100.00	54.01	52.75	0.0012
	StrideAR (k=7)	72.17	79.47	83.89	98.70	100.00	46.48	49.92	0.0143

D.3. Quality Across the Rollout

We evaluate VisionReward separately on consecutive six-second segments. Target-only scores decline from 0.156 in the first segment to -0.045 in the final segment. From the second segment onward, target-only generation scores below StrideAR at $k = 2$, with differences of 0.056–0.085.

The quality decline is therefore visible within the rollout as well as in the aggregate scores. The 24B model also scored below the 4.5B model in a separate text-to-video check using MAGI-1's upstream inference script. The cause of this ordering remains unresolved, and the conclusions here apply to the evaluated configuration.

Table D3: Mean VisionReward per 6-s segment on MAGI-1 ($H = 36$), with the slope per segment.

Method	**0–6 s**	**6–12 s**	**12–18 s**	**18–24 s**	**24–30 s**	**30–36 s**	**Slope per segment**
Target-only	0.156	0.016	0.004	−0.031	−0.054	−0.045	**−0.036**
StrideAR (k=2)	0.163	0.101	0.067	0.040	0.027	0.011	−0.029
StrideAR (k=5)	0.143	0.072	0.038	0.030	0.010	0.051	−0.019
StrideAR (k=7)	0.143	−0.026	−0.013	−0.051	−0.071	−0.070	−0.035