Learning-to-Memorize: Dynamic Context Management for Long-Horizon Autoregressive Video Generation

Haowei Zhu[1,2], **Qijie Wang**[1], **Jia Li**[2], **Xing Wang**[2], **Tianyu Zhao**[2], **Jiexi Wang**[2], **Xurui Peng**[2], **Xinglong Wu**[2], **Junhai Yong**[1], **Bin Wang**[1*]
[1] Tsinghua University [2] ByteDance Seed

Abstract

Autoregressive video generation has achieved impressive results on short clips, yet generating long videos remains challenging due to error accumulation over extended horizons, where each predicted frame depends on potentially imperfect previous frames. Existing approaches typically rely on static context management strategies, such as sliding-window caches that discard early frames or fixed sink frames that permanently preserve initial content. These designs either lose critical long-range context or anchor generation to outdated information, which can lead to degraded fidelity, motion stagnation, and reduced diversity. In this paper, we propose **Learning to Memorize (L2M)**, a learned context memory management framework that dynamically retains, evicts, and evolves historical frames according to predicted importance. The framework consists of an *importance prediction router* that estimates the relevance of each historical frame, *degradation-aware training* that improves robustness to noisy or corrupted context, and a *dynamic memory initialization and evolution* mechanism that updates long-term memory using exponential moving average (EMA) importance scores. This allows high-quality frames to replace stale anchors and enables memory to adapt naturally to scene evolution. Trained with a teacher-student distillation objective and sparsity regularization, L2M makes efficient use of a fixed KV-cache budget while preserving the most informative contextual information. Extensive experiments demonstrate that L2M improves long-horizon video generation quality and temporal consistency over existing baselines, providing a learned strategy for context memory management in long autoregressive video generation.

1 Introduction

Recent advances in video diffusion models Yang et al. [2024], Wan et al. [2025], HaCohen et al. [2024], Ho et al. [2022], Blattmann et al. [2023], Polyak et al. [2024] have demonstrated remarkable capabilities in synthesizing short video clips (e.g., 5–10 seconds) with both high visual fidelity and coherent motion dynamics. Building upon this progress, emerging interactive applications such as world models Parker-Holder and Fruchter [2025], Ye et al. [2025] and real-time content creation Shin et al. [2025] increasingly demand *long-horizon* autoregressive video generation, where frames are streamed sequentially over extended durations (e.g., 1–2 minutes).

Self Forcing Huang et al. [2025] and its variants Yang et al. [2025b], Liu et al. [2025], Cui et al. [2025] have become standard paradigms in autoregressive video generation, leveraging causal attention masks and Key-Value (KV) caches from previously generated frames. However, this autoregressive formulation is inherently susceptible to *error accumulation over long horizons* Yin et al. [2025],

*Corresponding author: wangbins@tsinghua.edu.cn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Huang et al. [2025], Wang et al. [2025], depends on previously generated frames that may contain errors, leading to progressive fidelity degradation, color drift, and texture blurring.

To mitigate error accumulation, recent methods have explored two principal directions. The first direction introduces history corruption during training Chen et al. [2024], Song et al. [2025], adding noise to previous frames to improve robustness. While effective to some degree, a discrepancy remains between artificially injected noise and the actual distribution of generated errors, leaving models vulnerable to long-horizon drift. The second direction, inspired by the attention sink mechanism from large language models (LLMs) Xiao et al. [2023], retains a small set of initial tokens as permanent anchors to stabilize the attention distribution Yesiltepe et al. [2025]. Deep Forcing Yi et al. [2025] extends this idea with deep sinks and participative compression, while LongLive Yang et al. [2025b] and Rolling Forcing Liu et al. [2025] incorporate sink-aware training. PackForcing Mao et al. [2026] introduces a three-partition KV cache with token compression for bounded memory.

Despite these advances, we identify a fundamental limitation shared by existing methods: their context management strategies are **static** and **rule-based**. Specifically, (1) FIFO policies discard frames solely based on recency, without considering their semantic importance; (2) fixed sink strategies permanently retain initial frames, regardless of whether they contain useful information, such as during transitions or occlusions; and (3) in long-horizon dynamic scenes, early sink frames often become outdated as subsequent content diverges significantly, reducing their relevance as anchors. Recent methods Yi et al. [2025], Mao et al. [2026], Zhang et al. [2026] that select tokens based on similarity are also suboptimal, as they tend to prioritize recently generated frames, which are naturally more similar to each other, rather than identifying truly informative historical content.

In this paper, we propose **Learning-to-Memorize (L2M)**, a learned context management framework that dynamically determines which historical frames should be retained, evicted, or promoted during long-horizon autoregressive video generation. Our key insight is that context management should be content-adaptive rather than governed by fixed rules: the model learns to predict which historical frames are most valuable for future generation and manages the limited KV-cache budget accordingly.

The framework consists of three main components. First, an *importance prediction router* evaluates the relevance of each historical frame relative to the current frame through cross-frame feature interactions, predicting an importance score of each context frames. Second, a *degradation-aware training* scheme augments historical context with simulated degradations such as noise, blur, or occlusion, teaching the router to assign lower importance to unreliable frames and improving inference-time robustness. Third, a *dynamic memory evolving* mechanism maintains exponential moving average (EMA) importance scores and promotes high-quality newly generated frames into the long-term memory sink set while replacing stale anchors during sampling, allowing the memory to evolve naturally with the video content. The entire framework is trained with a teacher-student distillation objective, where a teacher operating on full context provides supervision for a student operating on router-pruned context, combined with sparsity regularization to encourage efficient cache utilization. Our contributions are summarized as follows:

- We identify a fundamental limitation of static context management in autoregressive video generation and propose a *learned* context management framework, **L2M**, which dynamically retains, evicts, and updates historical frames based on predicted importance.
- We introduce a lightweight importance prediction router together with degradation-aware training via context perturbation and a dynamic memory mechanism that initializes and continuously evolves long-term context. These components jointly enable content-adaptive memory management in long video generation.
- Extensive experiments demonstrate that L2M consistently outperforms existing baselines on long-horizon video generation, improving temporal consistency and visual fidelity, and establishing a new learned context management method in autoregressive video generation.

2 Related Work

Video Diffusion and Long Video Generation. Early video generation extended 2D image diffusion models with temporal modules Ho et al. [2022], Blattmann et al. [2023], Singer et al. [2022], while recent Diffusion Transformers (DiTs) Peebles and Xie [2023], Brooks et al. [2024] treat videos as spatiotemporal patches for scalable 3D attention. State-of-the-art models such as CogVideoX Yang

et al. [2024], MovieGen Polyak et al. [2024], Wan Wan et al. [2025], and LTX-Video HaCohen et al. [2024] produce high-fidelity short clips, and flow-matching techniques Lipman et al. [2022], Liu et al. [2022] accelerate convergence. However, joint spatiotemporal modeling for minute-level videos remains computationally intractable, motivating memory-efficient autoregressive strategies Qiu et al. [2023], Ge et al. [2023], Hong et al. [2023], Henschel et al. [2025] that either rely on hierarchical planning or multi-stage pipelines, often at the cost of complexity.

Autoregressive Video Generation and Context Management. Autoregressive approaches synthesize frames sequentially using KV caches, with methods like CausVid Yin et al. [2025], Diffusion Forcing Chen et al. [2024], and Self Forcing Huang et al. [2025] bridging the train-inference gap or improving robustness Song et al. [2025]. Extensions such as Rolling Forcing Liu et al. [2025], LongLive Yang et al. [2025b], and Self Forcing++ Cui et al. [2025] enable minute-scale generation while maintaining visual continuity. Despite these advances, existing methods largely rely on hand-crafted cache-management rules, including FIFO eviction, fixed sink retention, and attention-based dynamic selection. L2M learns a frame-importance predictor and additionally updates long-term memory.

Attention Sinks and KV Cache Optimization. The attention sink phenomenon, first observed in large language models Xiao et al. [2023], concentrates attention on initial tokens, and retaining these sinks in a sliding window stabilizes long-context generation. Various KV cache compression methods have been proposed for efficiency, including H2O Zhang et al. [2023], SnapKV Li et al. [2024], LaCache Shi et al. [2025], and D2O Ghadia et al. [2025]. Video generation methods Yi et al. [2025], Mao et al. [2026], Li et al. [2026], Fu et al. [2026] extend sinks with deep tokens or spatiotemporal compression. All these approaches rely on static or heuristic criteria, whereas L2M learns frame importance via a trainable router and dynamically evolves long-term memory, achieving content-adaptive context management within a single-stage causal framework while maintaining visual fidelity and natural content evolution.

3 Method

3.1 Overview

We propose a learned context memory management framework for long-horizon autoregressive video generation. Given an autoregressive video generation model that generates the current latent chunk z_b conditioned on historical context $\mathcal{C}_b$, our goal is to learn a context selection policy

$$\widetilde{\mathcal{C}}_b = \pi(z_b, \mathcal{C}_b), \quad |\widetilde{\mathcal{C}}_b| \leq K, \tag{1}$$

where b indexes the current generation chunk and K denotes the cache budget measured in latent frames. Unlike prior approaches that define π through static rules (e.g., FIFO eviction or fixed sinks), we *learn* π from data such that it can dynamically decide which historical frames to retain, evict, and promote based on their content and predicted utility for future generation.

Our framework consists of three core components: (1) an *importance prediction router* that predicts the importance of each historical frame (Section 3.3); (2) *degradation-aware training* that simulates context degradation to improve router robustness (Section 3.4); and (3) *dynamic memory management mechanism* that maintains an evolving long-term memory set at inference time (Section 3.5). The overall framework is illustrated in Figure 1.

3.2 Preliminaries: Autoregressive Video Generation

We build upon the autoregressive video diffusion framework Huang et al. [2025], Yin et al. [2025]. A video consisting of T latent frames is divided into $N_c = \lceil T/B_f \rceil$ non-overlapping chunks, each containing B_f latent frames (typically $B_f = 3$). We use $b \in \{1, \ldots, N_c\}$ to index chunks and $i \in \{1, \ldots, T\}$ to index individual latent frames. Specifically, the b-th chunk is denoted by $z_b = [x_{(b-1)B_f+1}, \ldots, x_{(bB_f,T)}] \in \mathbb{R}^{B_f \times C \times H \times W}$, where $x_i \in \mathbb{R}^{C \times H \times W}$ denotes the i-th latent frame. During the generation of chunk b, each transformer layer l attends to the KV pairs cached from all previously generated latent frames:

$$\mathcal{C}_b^l = \left\{ (K_i^l, V_i^l) \right\}_{i=1}^{(b-1)B_f}, \tag{2}$$

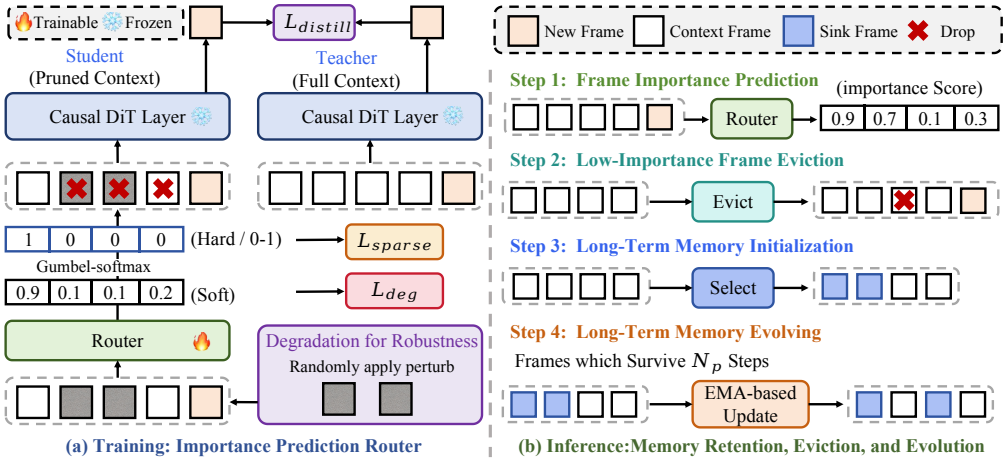

Figure 1: Overview of L2M. **(a) Training phase**: Given the current chunk feature $\bar{f}_b$ and historical latent frame features $\{\bar{f}_i\}_{i<(b-1)B_f+1}$, the importance-aware frame router predicts importance scores for each historical latent frame. Binary frame selection masks $m_i \in \{0, 1\}$ are obtained via Gumbel-Softmax discretization. The pruned context is fed to the student network, which is supervised by the teacher operating on the full context. Degradation-aware training augments a subset of historical frames with degradations, encouraging the router to assign them lower retention probabilities. **(b) Inference phase**: The router predicts frame importance scores, and low-scoring latent frames are evicted to meet the cache budget. High-scoring frames are dynamically selected as long-term memory and periodically updated using maintained EMA scores, continuously promoting informative historical frames.

For each latent frame i, the cached KV tensors at transformer layer l are denoted by $K_i^l, V_i^l \in \mathbb{R}^{hw \times N_h \times d_h}$, where h and w denote the spatial token dimensions of each latent frame, N_h is the number of attention heads, and d_h is the per-head dimension. We omit the batch dimension for simplicity. The KV cache of chunk b is obtained by concatenating the KV tensors of its constituent latent frames along the token dimension.

As generation proceeds, the KV cache grows linearly. For minute-long videos, this quickly exceeds available GPU memory, necessitating a context management strategy to bound the cache size. Prior approaches employ either FIFO eviction Huang et al. [2025], Yin et al. [2025], where the oldest chunk is discarded when the cache is full, or attention sinks Yi et al. [2025], Yang et al. [2025b], where initial chunks are permanently retained. Our method instead *learns* which frames to retain.

3.3 Importance-aware Frame Router

The core component of our framework is a lightweight frame router that evaluates the importance of each historical frame for generating the current chunk. Let $\bar{f}_b$ denote the average-pooled feature of the current chunk z_b along the temporal dimension. For each historical latent frame x_i, its corresponding feature f_i is averaged along the spatial token dimension to obtain $\bar{f}_i$. To capture content-dependent relevance, we first concatenate $\bar{f}_t$ and $\bar{f}_i$ along the channel dimension and pass the result through a lightweight multi-layer perceptron (MLP). The router then produces the importance scores:

$$s_i = \text{Router}(\bar{f}_b, \bar{f}_i) = \text{Softmax}\left(\text{MLP}([\bar{f}_b \oplus \bar{f}_i])\right) \in \mathbb{R}^2. \tag{3}$$

We use the predicted probability of the first category as the importance score of each historical frame, i.e., $r_i = [s_i]_0$, where a higher score indicates a higher likelihood of retaining the frame. During training, the Softmax output is combined with the Gumbel-Softmax trick [Jang et al., 2016] to produce a differentiable approximation of discrete selection, while at inference time a standard Softmax is applied. After normalization, $m_{i,0} = 1$ indicates that the frame is retained, while $m_{i,1} = 1$ indicates it is pruned. The predicted variables m_i are then applied to the historical context to produce the pruned context for the student network during training or for memory-efficient inference.

However, directly discarding or zeroing out frames with $m_i = 0$ would interfere with batch parallelism and is non-differentiable. To address this, we employ a differentiable attention masking strategy Rao

et al. [2021] that completely prevents pruned frames from affecting retained tokens while maintaining a fixed token count. Detailed description of this masking strategy is provided in Appendix C.1.

Context Pruning The frame-level selection masks m_i are applied to the corresponding historical KV pairs to construct the pruned context $\widetilde{\mathcal{C}}_b^l$. Specifically, all spatial tokens belonging to a latent frame are jointly retained or removed, ensuring that frame-level temporal positions remain consistent.

$$\widetilde{\mathcal{C}}_b^l = \left\{ (K_i^l, V_i^l) \mid i < (b-1)B_f + 1,\ m_i = 1 \right\}. \tag{4}$$

During training, the pruned context is fed to the student network, while the full context is used by the teacher network for supervision. At inference time, frames with $m_i = 0$ are evicted to satisfy the fixed cache budget.

Learning Objective We train the router under a teacher–student distillation framework, with the router being the sole trainable parameter of our approach. The teacher network, operating on the full context $\mathcal{C}_{t-1}$, produces target features y^*, while the student network, operating on the pruned context $\tilde{\mathcal{C}}_{t-1}$, produces output features y. We apply both distillation and sparsity losses to each transformer layer, and an additional distillation loss is applied to the model's noise predictions $\hat{\epsilon}$. The distillation loss encourages the student to match the teacher's output:

$$\mathcal{L}_{\text{distill}} = \|y - y^*\|_2^2. \tag{5}$$

A sparsity regularization term is added to encourage the router to retain as few frames as possible while maintaining generation quality:

$$\mathcal{L}_{\text{sparse}} = \frac{1}{N_b} \sum_{i=1}^{N_b} m_{i,\text{retain}}. \tag{6}$$

where N_b is the total number of historical frames. These two objectives together encourage the router to prune as many frames as possible while minimizing information loss, ensuring that the student network retains sufficient context for high-quality generation. To save memory and enable efficient training, gradients are computed only for a single sampled denoising step Huang et al. [2025] and the last frame chunk.

Temporal RoPE Adjustment The key cache for each frame encodes temporal position information via the use of 3D Rotary Position Embeddings (RoPE) Su et al. [2024]. When context frames are evicted, gaps in the temporal positions of the remaining keys are introduced, which can disrupt the positional continuity that the Transformer attention mechanism relies on. To address this, we apply a *continuous temporal RoPE adjustment* to the re-encoded frames. Specifically, for each frame, we compute the difference Δ_i between its original temporal position in the sequence and its new expected position after eviction. Using the RoPE temporal frequency vector $\boldsymbol{\omega}_t$, the temporal component of the key is adjusted as:

$$K_i^{l,(\text{time})} \leftarrow K_i^{l,(\text{time})} \odot \exp\left(\mathrm{i}\omega_t \Delta_i\right). \tag{7}$$

where i is the imaginary unit and $\odot$ denotes element-wise multiplication. This operation rotates K_i in the complex plane to align the relative temporal positions of sink and tail tokens, preserving the positional continuity required for accurate attention computation. This multiplicative adjustment is computationally efficient, has been widely adopted in prior works Yi et al. [2025], Mao et al. [2026], and introduces only negligible overhead.

Why Frame Pruning Instead of Token Pruning In our framework, the frame is the minimal unit for pruning, rather than individual tokens as in previous methods. This design choice is motivated by the observation that retaining a large number of tokens from multiple frames can introduce significant temporal gaps between early and recent frames in the RoPE-encoded key cache. Prior methods Yi et al. [2025] attempt to compact temporal positions by adjusting only the sink tokens while setting all other tokens to the same temporal position. However, this creates ambiguity in the 3D RoPE encoding: tokens from different frames but at the same spatial location are mapped to identical positional embeddings. We find that, compared to such fine-grained token pruning with position encoding errors, using coarser-grained frame pruning preserves accurate temporal encodings and yields better performance. Additional comparison experiments are provided in Appendix E.

3.4 Degradation-aware Training

A key challenge in long-horizon autoregressive generation is that the historical context at inference time is not clean. It has been generated by the model itself and may contain accumulated errors, visual artifacts, or degraded representations. These errors tend to accumulate and amplify as the generated sequence grows. If the router is trained solely on clean context, it may over-trust historically early frames that have become unreliable due to error accumulation. Simulating such inference-time degraded context during training would require full long-context rollouts, incurring prohibitively high computational cost.

Inspired by Diffusion Forcing Chen et al. [2024], we explicitly simulate context degradation during training. For a randomly selected subset $\mathcal{N}$ of historical frames, we apply random feature perturbations as the degradation operator:

$$\hat{f}_i = f_i + \beta \cdot \epsilon, \tag{8}$$

where ϵ is random noise and β controls the perturbation strength. The degraded context $\hat{\mathcal{C}}_{t-1}$ is constructed by replacing the selected frames with their degraded versions. The perturbed historical Keys are used as inputs to the Router, while both the perturbed Keys and Values are fed into the Student's attention computation. The Teacher, in contrast, uses the original, unperturbed historical KV cache to provide supervision.

To encourage the router to discount unreliable frames, we define a degradation-aware loss:

$$\mathcal{L}_{\text{deg}} = \sum_{i \in \mathcal{N}} s^{\text{noisy}}_{i,\text{retain}}. \tag{9}$$

where $s^{\text{noisy}}_{i,\text{retain}}$ denotes the importance score predicted by the router for the degraded version of frame i. This loss teaches the router to distinguish between "unreliable" frames and "genuinely valuable" frames, improving its robustness to degraded context during inference.

The overall training objective combines all three losses:

$$\mathcal{L} = \mathcal{L}_{\text{distill}} + \lambda_1 \mathcal{L}_{\text{sparse}} + \lambda_2 \mathcal{L}_{\text{deg}}, \tag{10}$$

where λ_1 controls the sparsity of context selection and λ_2 balances robustness.

3.5 Dynamic Memory Management at Inference

During inference, the KV cache has a fixed budget, requiring a strategy to determine which frames to retain as generation progresses. Our dynamic memory management framework consists of three mechanisms: importance-based eviction, dynamic long-term memory initialization, and evolving.

Importance-based Frame Eviction Unlike FIFO eviction, which removes the oldest frames, our approach discards frames based on their predicted importance. Specifically, we preserve a set of sink frames $\mathcal{S}$ and recent frames $\mathcal{R}$, while treating the remaining frames as candidates $\mathcal{C}_{\text{cand}}$, where $\mathcal{S}$, $\mathcal{R}$, and $\mathcal{C}_{\text{cand}}$ are disjoint sets of latent frame indices, and $|\mathcal{S}| + |\mathcal{R}| + |\mathcal{C}_{\text{cand}}| = N_h$, where N_h denotes the total number of historical latent frames. Importance scores are computed for frames in $\mathcal{C}_{\text{cand}}$ using the router (Eq. 3), and frames with the lowest scores are evicted first. This strategy ensures that the most informative frames are retained under a fixed memory budget.

Dynamic Long-term Memory Initialization Previous methods often fix the initial frames as permanent sinks, motivated by observations in LLMs Xiao et al. [2023] that attention tends to concentrate on early tokens. However, we observe that in autoregressive video generation, attention is frequently distributed across frames from diverse temporal positions (see Figure 9).

Our analysis suggests that the effectiveness of sink frames is primarily attributed to their content quality and reliability, rather than their temporal order. Therefore, sink selection should be content-driven rather than position-dependent. Instead of fixing the initial frames as sinks, we dynamically initialize the sink set. At the first eviction event, we select the top-k frames (with $k = |\mathcal{S}|$) according to their importance scores as the initial sink set:

$$\mathcal{S}_0 = \text{TopK}\big(\{s_i\}_{i \in \mathcal{C}_{t-1}},\ k\big). \tag{11}$$

This design ensures that sink frames correspond to the most informative historical content, rather than being biased toward early positions.

Long-term Memory Evolving Fixing the sink set after initialization can be suboptimal for long videos with significant motion or scene changes, as early long-term memory may become outdated. To enable the sink set to adapt to evolving content, we maintain an exponential moving average (EMA) of the importance scores for each frame:

$$\bar{s}_i^{(t)} = \alpha\, s_i^{(t)} + (1-\alpha)\, \bar{s}_i^{(t-1)}, \tag{12}$$

where $s_i^{(t)}$ denotes the instantaneous importance score at step t, and α is the EMA coefficient. The EMA captures the sustained importance of a frame over time, smoothing out transient fluctuations. Smaller values of α place more emphasis on earlier, more reliable estimates, thereby mitigating error accumulation in later stages.

To further ensure robustness, a frame is considered eligible for long-term memory only if it persists for at least N_p consecutive steps without being evicted. The sink set is then updated by selecting the top-k frames from the union of existing sinks and eligible candidates:

$$\mathcal{S}_t = \mathrm{TopK}\big(\{\bar{s}_i^{(t)}\}_{i \in \mathcal{S}_{t-1} \cup \mathcal{P}_t},\ k\big), \tag{13}$$

where $\mathcal{P}_t$ denotes the set of eligible frames at step t.

This dynamic update mechanism allows the sink set to evolve with the video content: newly emerged high-quality frames can gradually replace outdated ones, enabling adaptive memory refinement while preserving long-term stability.

Why Dynamic Long-term Memory Matters The proposed long-term memory evolving mechanism addresses two complementary scenarios. In *stable scenes*, the sink set maintains long-term consistency by retaining frames that act as reliable anchors, thereby suppressing error accumulation and mitigating instability caused by local noise. In *dynamic scenes* with significant motion or scene transitions, a static sink set may impose outdated constraints, forcing the model to regress toward obsolete states. In contrast, the evolving mechanism allows newly generated frames to enter the long-term memory, enabling the context to adapt to the natural progression of the video.

4 Experiments

4.1 Experimental Settings

Implementation Details We implement L2M on top of the Self Forcing Huang et al. [2025] framework as our base model. The importance-aware frame router is realized as a 4-layer MLP with hidden dimension 768 and GeLU activations, taking as input the concatenation of the current chunk feature and each historical frame feature. For inference, we set the KV cache budget K to 21 latent frames, the sink set size $|\mathcal{S}|$ to 14 latent frames, the EMA coefficient α to 0.5, and the long-term memory update threshold N_p to 6 steps. The training objective weights are set to $\lambda_1 = 1.0$ and $\lambda_2 = 1.0$. To simulate degraded context, we apply feature perturbation to historical frames with a probability of 0.2 and $\beta = 0.1$. The router is jointly trained for 1000 steps with a batch size of 32 using the AdamW optimizer Loshchilov and Hutter [2017] with a learning rate of 1×10^{-4}. Training is conducted on 16 GPUs and takes approximately 6 hours. This setup balances training efficiency with effective learning of content-adaptive context management.

Evaluation Protocol Following standard practice Huang et al. [2025], Yi et al. [2025], Cui et al. [2025], we evaluate on VBench-Long Huang et al. [2024] using 128 prompts from MovieGen Polyak et al. [2024]. Each prompt is refined using Qwen2.5-7B-Instruct Yang et al. [2025a]. We report nine metrics from VBench-Long, including Aesthetic Quality, Background Consistency, Dynamic Degree, Imaging Quality, Motion Smoothness, Subject Consistency, Temporal Flickering, Quality Score, and Overall Consistency. Overall Consistency is computed by averaging ViCLIP Wang et al. [2023] scores over 5-second video segments. These metrics enable a comprehensive evaluation of the generated videos from multiple perspectives, including visual quality, temporal consistency, and semantic coherence. We evaluate all methods on 30-second and 60-second videos, and additionally report 120-second results in Table 4.

Table 1: **Quantitative comparison on long video generation.** We evaluate our method against open-source autoregressive video diffusion baselines on 30-second and 60-second videos using multiple quality metrics from VBench-Long Huang et al. [2024]. All metrics follow the convention that higher values indicate better performance. The best results are in **bold** and the second best are underlined.

Model	Aesthetic Quality	Background Consistency	Dynamic Degree	Imaging Quality	Motion Smoothness	Subject Consistency	Temporal Flickering	Quality Score	Overall Consistency
30 seconds									
CausVid	61.80	90.70	54.69	66.98	98.06	87.69	96.80	79.18	22.92
Rolling Forcing	59.51	91.58	35.16	**69.93**	**98.43**	89.23	96.94	78.50	23.69
Self Forcing	60.28	90.81	44.53	68.73	97.97	87.08	**97.00**	78.39	24.10
Deep Forcing	61.17	91.90	**59.38**	68.08	97.70	87.56	96.12	79.37	24.55
L2M	**62.69**	**93.48**	57.81	69.54	98.17	**90.99**	96.71	**81.15**	**24.78**
60 seconds									
CausVid	61.43	89.41	54.69	66.57	98.12	85.48	96.96	78.49	22.30
Rolling Forcing	58.57	90.52	35.94	**69.88**	**98.36**	86.74	96.85	77.66	23.01
Self Forcing	57.01	87.52	33.59	67.08	98.06	81.22	**97.33**	75.23	21.58
Deep Forcing	59.74	92.84	53.12	67.20	97.66	89.91	96.21	79.17	24.24
L2M	**62.17**	**93.20**	**56.25**	69.43	98.15	**90.29**	96.70	**80.73**	**24.54**

4.2 Main Results

Quantitative Comparison Table 1 shows that L2M achieves the best overall trade-off between visual quality and long-term consistency on both 30-second and 60-second videos. It achieves the best results in aesthetic quality, background consistency, subject consistency, and overall consistency, while maintaining competitive imaging quality and motion smoothness. These results demonstrate the effectiveness of dynamic memory management in preserving useful long-range context while maintaining natural motion. Compared with Self Forcing, which uses FIFO-based memory eviction, our method better preserves visual fidelity and temporal coherence. Compared with Deep Forcing, which relies on a static sink and attention-score-based eviction, L2M further improves quality and consistency, showing the advantage of learnable importance prediction. Although Rolling Forcing is trained with sink tokens and yields relatively stable appearance, it suffers from limited dynamics, especially on longer videos. In contrast, L2M maintains stronger motion dynamics while avoiding obvious quality degradation.

Qualitative Results Figure 2 presents qualitative comparisons with existing methods. Self Forcing, which employs FIFO-based memory eviction, exhibits noticeable color drift and texture degradation in later frames. CausVid also shows oversaturation and visual distortion in later stages. Rolling Forcing, trained with attention sinks, produces relatively stable results; however, it exhibits weaker background consistency. Deep Forcing uses a static long-term memory and evicts frames based on attention scores, maintaining better visual quality but still exhibits identity inconsistency and gradual image quality degradation over time. In contrast, L2M predicts learnable importance scores and dynamically manages memory, maintaining consistent visual quality throughout generation while preserving natural motion dynamics.

Router Prediction Patterns We visualize the router prediction patterns in Figure 3. We observe structured patterns across frame chunks: early high-quality frames and their neighboring chunks tend to receive higher importance scores, while intermediate frames are often assigned low scores and appear redundant.

This partly explains why prior methods that retain initial and nearby frames while discarding middle ones can achieve reasonable performance. However, not all early or temporally adjacent frames are equally important. The importance distribution varies across layers, and some such frames still receive low scores, indicating they should not be persistently preserved. This highlights the limitation of static heuristics and motivates our dynamic eviction and evolving memory design, which selectively retains truly informative frames based on learned importance.

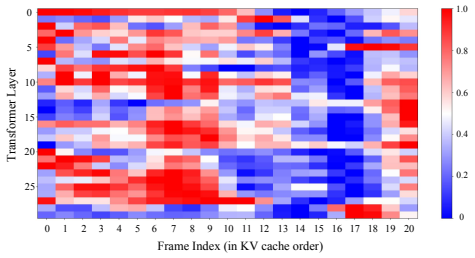

Figure 3: Visualization of layer-wise min-max normalized predicted importance scores.

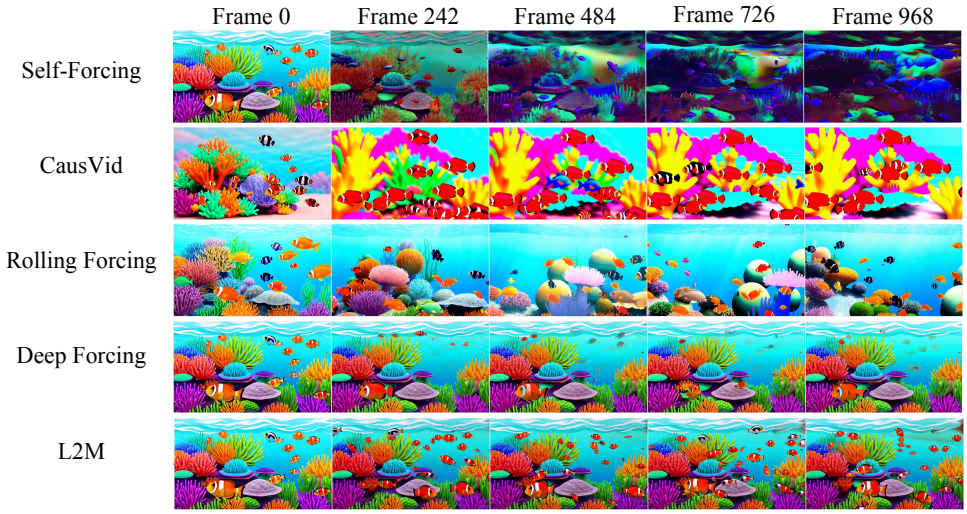

Figure 2: Qualitative comparison with baseline methods. Our method demonstrates superior visual quality and stronger overall consistency.

4.3 Ablation Studies

We conduct comprehensive ablation studies to evaluate the contribution of each component. More ablation results are provided in Appendix E.

Effect of Individual Components Table 2 shows the contribution of each component on 30-second videos. Starting from the base model, introducing the router yields clear gains in both Quality Score and Overall Consistency, indicating that, compared to the naive FIFO strategy, routing can better select informative context for generation. Incorporating degradation-aware training further improves performance, suggesting enhanced robustness to long-sequence content degradation. Enabling dynamic sink initialization brings additional gains, demonstrating that long-term memory selection should be content-aware rather than position-based. Finally, allowing the dynamic sink to evolve during generation achieves the best performance, confirming that adaptive memory updating is critical for maintaining visual quality while preserving long-term consistency.

Table 2: **Ablation study on individual components of our method.** Results are evaluated on 30-second videos.

Router	Degradation	Dynamic Sink Init.	Dynamic Sink Evolving	Quality Score↑	Overall Cons.↑
				78.39	24.10
✓				80.32	24.44
✓	✓			80.56	24.52
✓	✓	✓		80.88	24.70
✓	✓	✓	✓	**81.15**	**24.78**

Comparison with Alternative Importance Estimation To validate the effectiveness of our importance prediction router, we compare it with two baselines: random scoring and attention-based scoring. Random selection achieves a Quality Score of 79.92 and an Overall Consistency of 24.43, indicating that unstructured memory management fails to preserve useful context in long video generation. Attention-based scoring improves to 80.68 and 24.55, suggesting that attention weights provide a reasonable proxy for frame importance. In contrast, our learned importance predictor achieves the best performance, reaching 81.15 in Quality Score and 24.78 in Overall Consistency, demonstrating the effectiveness of learning frame importance over heuristic or random strategies.

Qualitative Analysis of Dynamic Memory We further provide a qualitative ablation study in Figure 4 to analyze the effect of dynamic long-term memory. We observe that incorporating dynamic

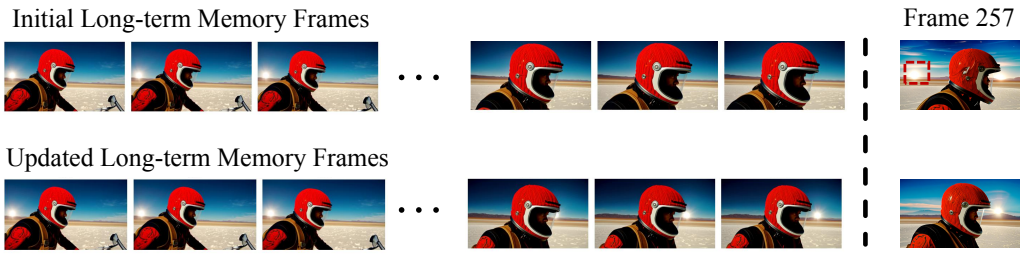

Figure 4: Qualitative analysis of dynamic long-term memory evolution. After memory updates, newly emerged informative frames are incorporated into sink memory, enabling better motion tracking and reducing long-term generation artifacts.

Table 3: ViCLIP score on consecutive 5-second segments of 30-second videos. Overall Consistency is the average of the six segment scores. Δ is the drop from the first segment to the last. Higher is better for segment and overall scores; lower is better for Δ. Best results are in **bold**.

Method	0–5 s	5–10 s	10–15 s	15–20 s	20–25 s	25–30 s	$\Delta \downarrow$	Overall $\uparrow$
CausVid	24.62	23.27	22.87	22.35	22.32	22.10	2.52	22.92
Rolling Forcing	25.19	24.51	23.92	23.08	22.91	22.50	2.69	23.69
Self Forcing	25.28	**25.28**	24.70	24.52	23.11	21.79	3.49	24.11
Deep Forcing	25.28	24.74	24.47	24.39	24.24	24.20	1.08	24.55
L2M	**25.28**	24.79	**24.72**	**24.71**	**24.65**	**24.54**	**0.74**	**24.78**

sink memory evolving consistently improves generation quality. This improvement mainly stems from introducing more content-relevant, rather than position-biased, information into long-term memory, while continuously updating the memory to adapt to evolving scenes.

As shown in Figure 4, taking the first-layer long-term memory as an example, we observe that after memory updates, frames capturing the positional changes of the sun are incorporated into the sink memory. This enables the model to better track the motion of the sun, avoiding temporal regression artifacts and duplicated suns that appear in the variant without long-term memory updates.

Quantifying Error Accumulation. To quantify temporal error accumulation, we report ViCLIP scores for consecutive 5-second segments over the 30-second generation horizon, tracking the evolution of text-video consistency. As shown in Table 3, all methods exhibit lower ViCLIP scores in the final segment than in the initial one, but with different degradation rates. Self Forcing, Deep Forcing, and L2M start at the same score of 25.28, yet diverge as generation proceeds. Self Forcing suffers the largest decline among these three methods, dropping by 3.49 points, while CausVid, Rolling Forcing, and Deep Forcing decline by 2.52, 2.69, and 1.08 points, respectively. In contrast, L2M maintains a nearly stable score after the first 5–10 s segment, decreasing from 24.79 to 24.54 in the final segment, with the smallest overall decline of 0.74. L2M also achieves the highest segment-wise scores from 10 s onward and the best overall score of 24.78. These results highlight the benefit of learned context selection and EMA-based memory updates in preserving informative historical frames and mitigating the long-horizon degradation caused by static or FIFO-based context management.

5 Conclusion

We proposed **L2M**, a learned context management framework for long autoregressive video generation. By dynamically selecting and updating historical frames based on their predicted importance, L2M overcomes the limitations of static caches and fixed sinks. Through importance-aware prediction, degradation-aware training, and dynamic memory updating, L2M better utilizes a fixed KV-cache budget to improve long-range consistency and visual fidelity. Extensive experiments demonstrate its advantages over existing baselines.

References

Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22563–22575, 2023.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. URL `https://openai.com/research/video-generation-models-as-world-simulators`.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

Jaemin Cho, Abhay Zala, and Mohit Bansal. Dall-eval: Probing the reasoning skills and social biases of text-to-image generation models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3043–3054, 2023.

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. *arXiv preprint arXiv:2510.02283*, 2025.

Xiaomeng Fu, Jia Li, Yiming Hu, Yong Wang, Hayden Kwok-Hay So, Jiao Dai, Xiangxiang Chu, and Jizhong Han. Towards memory-efficient autoregressive video generation via instance-specific parametric absorption. In *European Conference on Computer Vision*, pages 448–466. Springer, 2026.

Songwei Ge, Seungjun Nah, Guilin Liu, Tyler Poon, Andrew Tao, Bryan Catanzaro, David Jacobs, Jia-Bin Huang, Ming-Yu Liu, and Yogesh Balaji. Preserve your own correlation: A noise prior for video diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22930–22941, 2023.

Ravi Ghadia, Avinash Kumar, Gaurav Jain, Prashant J. Nair, and Poulami Das. Dialogue without limits: Constant-sized kv caches for extended responses in llms. *ArXiv*, abs/2503.00979, 2025. URL `https://api.semanticscholar.org/CorpusID:276741873`.

Yoav HaCohen, Nisan Chiprut, Benny Brazowski, Daniel Shalem, Dudu Moshe, Eitan Richardson, Eran Levin, Guy Shiran, Nir Zabari, Ori Gordon, Poriya Panet, Sapir Weissbuch, Victor Kulikov, Yaki Bitterman, Zeev Melumian, and Ofir Bibi. Ltx-video: Realtime video latent diffusion. *arXiv preprint arXiv:2501.00103*, 2024.

Roberto Henschel, Levon Khachatryan, Hayk Poghosyan, Daniil Hayrapetyan, Vahram Tadevosyan, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Streamingt2v: Consistent, dynamic, and extendable long video generation from text. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2568–2577, 2025.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Susung Hong, Junyoung Seo, Heeseong Shin, Sunghwan Hong, and Seungryong Kim. Large language models are frame-level directors for zero-shot text-to-video generation. In *First Workshop on Controllable Video Generation@ ICML24*, 2023.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint arXiv:2506.08009*, 2025.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jia Li, Xiaomeng Fu, Xurui Peng, Weifeng Chen, Youwei Zheng, Tianyu Zhao, Jiexi Wang, Fangmin Chen, Xing Wang, and Hayden Kwok-Hay So. Train short, inference long: Training-free horizon extension for autoregressive video generation. *arXiv preprint arXiv:2602.14027*, 2026.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. *arXiv preprint arXiv:2509.25161*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Siwei Lyu. Deepfake detection: Current challenges and next steps. pages 1–6, 2020.

Xiaofeng Mao, Shaohao Rui, Kaining Ying, Bo Zheng, Chuanhao Li, Mingmin Chi, and Kaipeng Zhang. Packforcing: Short video training suffices for long video sampling and long context inference. In *European Conference on Computer Vision*, pages 576–593. Springer, 2026.

Ranjita Naik and Besmira Nushi. Social biases through the text-to-image generation lens. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, pages 786–808, 2023.

Jack Parker-Holder and Shlomi Fruchter. Genie 3: A new frontier for world models, 2025. URL `https://deepmind.google/discover/blog/genie-3-a-new-frontier-for-world-models/`.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

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, Menghan Xia, Yong Zhang, Yingqing He, Xintao Wang, Ying Shan, and Ziwei Liu. Freenoise: Tuning-free longer video diffusion via noise rescheduling. *arXiv preprint arXiv:2310.15169*, 2023.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

Candace Ross, Boris Katz, and Andrei Barbu. Measuring social biases in grounded vision and language embeddings. *arXiv preprint arXiv:2002.08911*, 2020.

Dachuan Shi, Yonggan Fu, Xiangchi Yuan, Zhongzhi Yu, Haoran You, Sixu Li, Xin Dong, Jan Kautz, Pavlo Molchanov, and Yingyan Celine Lin. Lacache: Ladder-shaped kv caching for efficient long-context modeling of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Joonghyuk Shin, Zhengqi Li, Richard Zhang, Jun-Yan Zhu, Jaesik Park, Eli Schechtman, and Xun Huang. Motionstream: Real-time video generation with interactive motion controls. *arXiv preprint arXiv:2511.01266*, 2025.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

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

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

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

Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, et al. Internvid: A large-scale video-text dataset for multimodal understanding and generation. *arXiv preprint arXiv:2307.06942*, 2023.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, et al. Longlive: Real-time interactive long video generation. *arXiv preprint arXiv:2509.22622*, 2025b.

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. *arXiv preprint arXiv:2408.06072*, 2024.

Deheng Ye, Fangyun Zhou, Jiacheng Lv, Jianqi Ma, Jun Zhang, Junyan Lv, Junyou Li, Minwen Deng, Mingyu Yang, Qiang Fu, et al. Yan: Foundational interactive video generation. *arXiv preprint arXiv:2508.08601*, 2025.

Hidir Yesiltepe, Tuna Han Salih Meral, Adil Kaan Akan, Kaan Oktay, and Pinar Yanardag. Infinity-rope: Action-controllable infinite video generation emerges from autoregressive self-rollout. *arXiv preprint arXiv:2511.20649*, 2025.

Jung Yi, Wooseok Jang, Paul Hyunbin Cho, Jisu Nam, Heeji Yoon, and Seungryong Kim. Deep forcing: Training-free long video generation with deep sink and participative compression. *arXiv preprint arXiv:2512.05081*, 2025.

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 *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22963–22974, 2025.

Lvmin Zhang, Shengqu Cai, Muyang Li, Gordon Wetzstein, and Maneesh Agrawala. Frame context packing and drift prevention in next-frame-prediction video diffusion models. *Advances in Neural Information Processing Systems*, 38:30546–30566, 2026.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

Xiangqing Zheng, Chengyue Wu, Kehai Chen, and Min Zhang. Locot2v-bench: Benchmarking long-form and complex text-to-video generation. *arXiv preprint arXiv:2510.26412*, 2025.

A Broader Impact

This work improves long-horizon autoregressive video generation by introducing a learned dynamic memory management mechanism. On the positive side, it can benefit a wide range of applications such as video editing, content creation, simulation, and storytelling by enabling more temporally consistent and visually coherent long video synthesis under limited memory budget.

On the potential negative side, as with most generative video models pretrained on large-scale internet data, our method may inherit and amplify existing biases and stereotypes present in the training distribution Cho et al. [2023], Naik and Nushi [2023], Ross et al. [2020]. Such biases may manifest in generated visual content, potentially affecting fairness and representation in downstream applications.

In addition, improved long-horizon video generation capabilities may also increase the risk of misuse, including the creation of highly realistic DeepFakes Lyu [2020] and other synthetic media that could be used for misinformation or impersonation. While our method reduces the computational and memory cost of long video generation, lowering the barrier to content creation, it also emphasizes the importance of responsible deployment.

We therefore encourage the development of appropriate governance frameworks, usage guidelines, and detection mechanisms to ensure that generative models are used in a safe and responsible manner, and to mitigate potential negative societal impacts.

B Limitations and Future Work

While L2M substantially improves context management for autoregressive video generation, several limitations remain. First, the Router introduces additional computational overhead during both training and inference, although this cost is largely mitigated by its lightweight design. Second, the current framework operates at the frame level; extending it to finer-grained, token-level management could further improve cache efficiency and flexibility. Third, the degradation-aware training relies on hand-crafted degradation operators. Learning the degradation distribution directly from real generation errors may lead to more robust and adaptive behavior. Future work may explore hierarchical memory architectures with multiple levels of compression, as well as extensions to interactive generation settings, where user inputs dynamically influence context management decisions.

C More Implementation Details

C.1 Attention with Masking

To enable end-to-end optimization, the sampling score s_i in Eq. 3 are converted into a binary mask using the Gumbel-Softmax trick. This operator yields one-hot outputs while remaining differentiable, thereby allowing gradient-based training.

In addition, when token pruning is applied within a mini-batch, the learned masks are typically irregular, resulting in different samples retaining varying numbers of tokens. Physically removing the pruned tokens would therefore lead to variable sequence lengths across samples, which is incompatible with efficient batch computation.

Simply zeroing out pruned tokens is also insufficient to suppress their influence, as they can still affect other tokens through the softmax normalization in self-attention:

$$\mathbf{A} = \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{C}} \right). \tag{14}$$

To fully eliminate such influence, following the strategy in Rao et al. [2021], attention masking is employed to compute the masked attention matrix:

$$\mathbf{P} = \mathbf{Q}\mathbf{K}^T/\sqrt{C} \in \mathbb{R}^{N_q \times N_k}, \tag{15}$$

$$\mathbf{G}_{ij} = \begin{cases} 1, & i = j, \\ \hat{\mathbf{D}}_j, & i \neq j, \end{cases} \quad 1 \leq i \leq N_q,\ 1 \leq j \leq N_k, \tag{16}$$

$$\tilde{\mathbf{A}}_{ij} = \frac{\exp(\mathbf{P}_{ij})\mathbf{G}_{ij}}{\sum_{k=1}^{N_k} \exp(\mathbf{P}_{ik})\mathbf{G}_{ik}}, \quad 1 \leq i \leq N_q,\ 1 \leq j \leq N_k. \tag{17}$$

This construction can be interpreted as a graph in which $\mathbf{G}_{ij} = 1$ indicates that token j is allowed to contribute to the update of token i. Self-loops are added for numerical stability. Importantly, when $\hat{\mathbf{D}}_j = 0$, token j only preserves its self-loop and does not affect any other token. As a result, $\tilde{\mathbf{A}}$ is equivalent to performing attention over the retained tokens only, while keeping a constant $N_q \times N_k$ shape during training.

C.2 Pseudo Algorithm

We provide the pseudo algorithm of our method in Algorithm 1.

D Differences from Existing Work

Although recent autoregressive video generation methods have substantially improved long-horizon synthesis, they mostly differ from our approach in that they rely on *static* context management policies. For example, CausVid Yin et al. [2025] and Self Forcing Huang et al. [2025] primarily focus on bridging the train-inference gap and improving robustness in causal generation, but they still manage historical context using hand-crafted cache policies rather than learning which frames are truly informative. As a result, their retained context is determined by predefined rules instead of content-aware importance, which may preserve redundant frames while discarding useful ones.

Rolling Forcing Liu et al. [2025] extends autoregressive generation to longer sequences by introducing rolling memory and fixed sink-style context retention, which improves temporal stability but remains heuristic in nature. Similarly, Deep Forcing Yi et al. [2025] augments the memory with deeper long-term tokens, and Pack Forcing Mao et al. [2026] compresses context through packing strategies to reduce cache usage. However, both methods still depend on static or heuristic criteria for deciding what to keep, and thus cannot adapt their memory organization to different scenes, motions, or video stages. In contrast, our method learns an explicit importance router that dynamically ranks frames based on their predicted contribution to future generation.

More broadly, unlike prior sink-based or FIFO-based approaches that either permanently preserve early frames or evict frames according to recency, L2M introduces a learned and evolving memory mechanism. It dynamically initializes the sink set from content-relevant frames and continuously updates long-term memory using predicted importance scores, enabling the model to retain informative historical context while discarding redundant or outdated frames. This makes L2M fundamentally different from existing long-video generation methods, which either rely on fixed memory heuristics or use compression rules that are not content-adaptive.

E More Experiments

Longer Video Generation Extrapolation We further evaluate L2M on 120-second video generation, corresponding to a $24\times$ extrapolation beyond the 5-second training horizon. As shown in Table 4, L2M maintains strong performance despite the substantially extended generation duration, achieving the highest Quality Score (80.30) and Overall Consistency (24.47). It also obtains the best results in Aesthetic Quality, Dynamic Degree, Imaging Quality, and Subject Consistency. These results demonstrate that the proposed dynamic context management strategy generalizes effectively to generation horizons far beyond those encountered during training.

We further extend the evaluation horizon to 240 seconds. Notably, the advantage of L2M over its Self Forcing backbone becomes progressively larger as the generation horizon increases. The

Algorithm 1 L2M: Learned Context Management for Long-Horizon Autoregressive Video Generation

Require: Router R, teacher ϕ_t, student ϕ_s, total latent frames T, frames per chunk B_f, cache budget K, sink budget K_s, recent budget K_r, patience N_p, EMA coefficient α, sparsity weight λ_1, degradation weight λ_2

$N_c \leftarrow \lceil T/B_f \rceil$
Initialize memory $\mathcal{C}_0 \leftarrow \emptyset$, sink set $\mathcal{S} \leftarrow \emptyset$, recent set $\mathcal{R} \leftarrow \emptyset$, and EMA scores $\bar{s}_i^{(0)} \leftarrow 0$
for $b = 1$ to N_c **do**
  $n_b \leftarrow \min((b-1)B_f, T)$
  $B_b \leftarrow \min(B_f, T - (b-1)B_f)$
  $z_b \leftarrow [x_{(b-1)B_f+1}, \ldots, x_{(b-1)B_f+B_b}]$
  Extract current chunk feature $\bar{f}_b$ and historical frame features $\{\bar{f}_i\}_{i=1}^{n_b}$
  if $n_b > 0$ **then**
    if training **then**
      Sample degraded subset $\mathcal{N} \subseteq \{1, \ldots, n_b\}$
      for $i = 1$ to n_b **do**
        if $i \in \mathcal{N}$ **then**
          $\hat{\bar{f}}_i \leftarrow \bar{f}_i + \beta\epsilon$ ▷ Eq. (8)
          $s_i \leftarrow R(\bar{f}_b, \hat{\bar{f}}_i)$
        else
          $s_i \leftarrow R(\bar{f}_b, \bar{f}_i)$
        end if
        $m_i \leftarrow \mathrm{GumbelSoftmax}(s_i)$ ▷ Eq. (3) ▷ Frame-level selection mask
      end for
      $\tilde{\mathcal{C}}_{b-1} \leftarrow \{(\mathbf{K}_i^l, \mathbf{V}_i^l) \mid m_{i,\mathrm{retain}} = 1\}$
      Adjust Temporal RoPE of the historical key cache ▷ Eq. (7)
      $y^* \leftarrow \phi_t(z_b, \mathcal{C}_{b-1})$
      $y \leftarrow \phi_s(z_b, \tilde{\mathcal{C}}_{b-1})$ ▷ Use Eq. (17) for parallel, differentiable forward passes
      $\mathcal{L}_{\mathrm{distill}} \leftarrow \|y - y^*\|_2^2$ ▷ Eq. (5)
      $\mathcal{L}_{\mathrm{sparse}} \leftarrow \frac{1}{n_b}\sum_{i=1}^{n_b} m_{i,\mathrm{retain}}$ ▷ Eq. (6)
      $\mathcal{L}_{\mathrm{deg}} \leftarrow \sum_{i\in\mathcal{N}} s_{i,\mathrm{retain}}^{\mathrm{noisy}}$ ▷ Eq. (9)
      $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{distill}} + \lambda_1\mathcal{L}_{\mathrm{sparse}} + \lambda_2\mathcal{L}_{\mathrm{deg}}$ ▷ Eq. (10)
      Update router R by minimizing $\mathcal{L}$
    else
      Compute frame importance scores $\{s_i^{(b)}\}_{i=1}^{n_b} \leftarrow R(\bar{f}_b, \{\bar{f}_i\}_{i=1}^{n_b})$
      if $\mathcal{S} = \emptyset$ **then**
        $\mathcal{S} \leftarrow \mathrm{TopK}(\{s_i^{(b)}\}_{i=1}^{n_b}, K_s)$ ▷ Eq. (11)
      end if
      Update EMA scores $\bar{s}_i^{(b)} \leftarrow \alpha s_i^{(b)} + (1-\alpha)\bar{s}_i^{(b-1)}$ ▷ Eq. (12)
      $\mathcal{P}_b \leftarrow \{i \mid i$ survives for N_p consecutive steps$\}$
      $\mathcal{S} \leftarrow \mathrm{TopK}\big(\{\bar{s}_i^{(b)}\}_{i\in\mathcal{S}\cup\mathcal{P}_b}, K_s\big)$ ▷ Eq. (13)
      $\mathcal{C}_{\mathrm{cand}} \leftarrow \mathcal{C}_{b-1} \setminus (\mathcal{S} \cup \mathcal{R})$
      Evict the lowest-scoring frames in $\mathcal{C}_{\mathrm{cand}}$ until $|\mathcal{S}| + |\mathcal{R}| + |\mathcal{C}_{\mathrm{cand}}| \le K - B_b$ ▷ Make room for the incoming chunk
      Adjust Temporal RoPE of the historical key cache ▷ Eq. (7)
      Generate the current chunk z_b using the managed context $(\mathcal{S}, \mathcal{R}, \mathcal{C}_{\mathrm{cand}})$
      Update memory $\mathcal{C}_b$ with the newly generated chunk
      Update recent set $\mathcal{R}$ with the most recent K_r frames in $\mathcal{C}_b$
    end if
  else
    Generate the first chunk z_1 without historical context
    Initialize memory $\mathcal{C}_1$ with the newly generated chunk
  end if
end for

Table 4: **Quantitative comparison on 120-second long video generation.** We evaluate our method against open-source autoregressive video diffusion baselines using multiple quality metrics from VBench-Long Huang et al. [2024]. All metrics follow the convention that higher values indicate better performance. The best results are in **bold** and the second best are underlined.

Model	Aesthetic Quality	Background Consistency	Dynamic Degree	Imaging Quality	Motion Smoothness	Subject Consistency	Temporal Flickering	Quality Score	Overall Consistency
CausVid	61.13	88.46	45.31	66.20	98.15	83.73	97.08	77.22	21.60
Rolling Forcing	57.86	89.87	36.72	69.47	**98.31**	85.03	96.80	77.06	22.59
Self Forcing	51.14	84.00	28.91	63.09	98.10	74.33	**97.52**	71.48	17.04
Deep Forcing	61.48	**93.06**	44.53	68.88	98.13	89.56	97.00	79.59	24.33
L2M	**61.88**	92.93	**53.12**	**69.56**	98.14	**89.72**	96.71	**80.30**	**24.47**

Table 5: **Quantitative comparison on 240-second long video generation.** Higher is better for all metrics. The best results are in **bold**, and the second-best results are underlined.

Model	Aesthetic Quality	Background Consistency	Dynamic Degree	Imaging Quality	Motion Smoothness	Subject Consistency	Temporal Flickering	Quality Score	Overall Consistency
CausVid	60.67	87.78	44.53	66.00	98.21	82.54	97.24	76.80	21.12
Rolling Forcing	57.40	89.48	36.72	69.36	98.27	84.11	96.76	76.68	22.40
Self Forcing	44.89	81.27	24.22	58.75	97.92	68.82	97.46	67.81	13.76
Deep Forcing	59.70	89.80	42.19	66.84	97.68	83.82	96.40	76.62	23.33
LongLive	61.23	92.39	40.62	69.38	**98.31**	89.04	**97.58**	79.43	24.03
L2M	**61.61**	**92.74**	**52.34**	**69.60**	98.16	**89.48**	96.77	**80.16**	**24.30**

improvements in Quality Score grow from 2.76 at 30 seconds to 5.50, 8.82, and 12.35 at 60, 120, and 240 seconds, respectively. Similarly, the gains in Overall Consistency increase from 0.68 to 2.96, 7.43, and 10.54. At 240 seconds, L2M maintains a Quality Score of 80.16 and an Overall Consistency of 24.30, whereas Self Forcing degrades to 67.81 and 13.76, respectively. L2M also outperforms the strongest competing method at this horizon, LongLive, which obtains 79.43 and 24.03 on the two metrics. These results provide direct evidence that adaptive context management becomes increasingly important over extended generation horizons: under a fixed KV-cache budget, the model must preserve informative context while preventing accumulated errors and outdated information from propagating to subsequent frames.

Human Evaluation We further conduct a two-alternative forced-choice (2AFC) user study comparing L2M with CausVid, Rolling Forcing, Deep Forcing, and Self Forcing. For each pair of videos, participants indicate their preference according to four criteria: (1) scene consistency throughout the video; (2) the naturalness and dynamics of subject and camera motion; (3) consistency in the identity and appearance of the main subject; and (4) overall realism and visual quality. Ten participants each evaluate 30 pairs generated from randomly sampled MovieGen prompts, with no prompt repeated within a participant's session, resulting in 300 pairwise comparisons in total.

Table 6 reports the percentage of votes favoring L2M over each baseline. L2M is preferred over every competing method across all four criteria, with preference rates ranging from 76.2% to 100.0%. In particular, when compared with CausVid, Deep Forcing, and Self Forcing, L2M receives more than 90% of the votes for scene consistency, subject consistency, and overall quality. These results corroborate the quantitative evaluation and demonstrate that the improvements of L2M are also clearly perceptible to human observers.

Comparison on Additional Benchmark To assess whether the benefits of L2M generalize beyond the MovieGen prompt distribution used in VBench-Long, we further evaluate all methods on LoCoT2V-Bench Zheng et al. [2025]. This benchmark features long-form, multi-scene prompts with substantially greater linguistic and structural complexity. Following our main evaluation setting, we generate 30-second videos and evaluate them using the official Perceptual Quality (PQ) metric, which measures frame-level visual fidelity.

As shown in Table 7, L2M achieves the highest PQ score of 77.37, outperforming the strongest baseline, Rolling Forcing, by 1.84 points. It also surpasses Deep Forcing, CausVid, and Self Forcing by 3.56, 6.61, and 6.93 points, respectively. These results demonstrate that L2M remains effective at preserving visual quality under a distinct and more challenging prompt distribution, supporting its generalization beyond VBench-Long.

Table 6: Human evaluation using two-alternative forced choice (2AFC). Each entry denotes the percentage of votes favoring L2M over the corresponding baseline. Higher is better. The best results are in **bold**, and the second-best results are underlined.

Baseline	Scene Consistency	Motion Quality	Subject Consistency	Overall Quality
CausVid	90.7	**100.0**	90.7	**96.0**
Rolling Forcing	80.8	76.2	84.5	78.8
Self Forcing	93.0	97.6	93.0	93.0
Deep Forcing	**94.0**	85.0	**96.0**	94.0

Table 7: **Quantitative comparison on LoCoT2V-Bench.** We report Perceptual Quality (PQ) for 30-second videos. Higher is better. The best result is in **bold**, and the second-best result is underlined.

Model	**Perceptual Quality** ↑
CausVid	70.76
Rolling Forcing	75.53
Self Forcing	70.44
Deep Forcing	73.81
L2M	**77.37**

Compatibility with Additional Baseline To investigate whether L2M transfers to different baseline models, we integrate it into LongLive by replacing its original fixed sink-and-recent-window policy with our learned context selection mechanism, while keeping the underlying video generator unchanged. As shown in Table 8, L2M improves the Quality Score from 80.73 to 81.20 and Overall Consistency from 24.58 to 24.84. These results demonstrate that L2M can be readily incorporated into other autoregressive long-video generation frameworks without retraining their underlying generators.

Inference Complexity Analysis Our method introduces an additional Router for importance prediction. We analyze its overhead in terms of both computation and memory. Specifically, we evaluate the inference cost for generating 30-second videos on an A100 GPU. Results are averaged over 10 samples. The results show that the Router incurs only negligible computational overhead. This efficiency mainly stems from two factors: (1) the Router is lightweight, and (2) it is executed only at the first timestep rather than at every timestep. In addition, our method requires tracking the lifespan of context frames and maintaining an EMA score for importance estimation. However, the associated memory overhead is also negligible.

We further analyze the computational overhead introduced by each component of L2M. On 30-second video generation, L2M increases the inference time from 78.55 s to 79.21 s, corresponding to an overhead of only 0.66 s (0.84%), while peak GPU memory increases from 36.05 GB to 36.32 GB (0.75%). More than 95% of the additional runtime is attributable to router forwarding. The EMA score update and sink-set modification together require less than 0.1 s. This low overhead is achieved because the lightweight router is evaluated only at the first denoising timestep and its predictions are reused throughout the remaining sampling process.

Overall, our approach achieves improved performance with almost no additional cost in inference time or memory usage.

Effect of EMA coefficient α We investigate the effect of the EMA coefficient α in Eq. 12. A smaller α gives more weight to historical importance scores, resulting in more stable but less adaptive long-term memory selection. In contrast, a larger α emphasizes recent observations, making the sink set more responsive to changing content but also more prone to instability.

As shown in Figure 5, a moderate value of α offers the best trade-off between stability and adaptability. In particular, $\alpha = 0.5$ achieves the best performance, with the highest Quality Score and Overall Consistency. Both smaller and larger values lead to slight performance drops, suggesting that balanced temporal smoothing is important for effective dynamic memory updating.

Table 8: **Integration of L2M into LongLive.** Results are evaluated on 30-second videos using 128 MovieGen prompts. Higher is better for both metrics.

Method	**Quality Score ↑**	**Overall Consistency ↑**
LongLive	80.73	24.58
LongLive + L2M	**81.20**	**24.84**

Table 9: Inference cost comparison on 30-second video generation.

Method	**Memory (GB)**	**Inference Time (s)**
Baseline	36.05	78.55
Ours	36.32	79.21

Comparison of Long-term Memory Update Interval N_p We further study the long-term memory update interval N_p, which controls the lifespan of frames when updating the long-term memory. An overly small N_p leads to excessively frequent memory updates, causing the model to rely on importance scores computed over short temporal windows that may not accurately reflect a frame's long-term contribution to video quality. In contrast, an overly large N_p delays memory updates, resulting in stale context that cannot adapt to scene evolution in a timely manner. As shown in Figure 6, $N_p = 6$ achieves the best performance, striking a balance between stability and adaptability in long-term memory updating.

Comparison of Sink Set Size We evaluate the effect of sink set size on model performance, as shown in Figure 7. We observe that performance nearly saturates at a sink size of 14 (our default setting). Further decreasing the sink size leads to a gradual performance drop, while increasing it does not bring any improvement and may slightly degrade performance. This is mainly because a larger sink size forces more redundant frames to be preserved as long-term memory, reducing the efficiency of context utilization.

Comparison of Recent Window Size In our method, we preserve the three most recent frames as contextual frames during each eviction step, and only perform pruning on frames excluding the recent and sink frames. We further study the effect of different recent window lengths $|R|$. As shown in Figure 8, we find that $|R| = 3$ yields the best performance. Reducing $|R|$ leads to a noticeable performance drop, likely due to the disruption of temporal continuity in the video. This observation is also consistent with the results in Figure 3, where the most recent three frames consistently receive the highest importance scores.

Comparison of Cache Budget We further analyze the sensitivity of L2M to the cache budget, using $K \in \{18, 21, 24, 27, 30\}$. As shown in Table 10, L2M achieves the highest Quality Score of 81.15 at $K = 21$. Overall Consistency improves modestly with larger cache budgets, increasing from 24.72 at $K = 18$ to 24.84 at $K = 30$. However, the score at $K = 21$ is already within 0.06 of this maximum while using 30% fewer cache slots. Because increasing K also raises KV-cache memory consumption and attention computation, we adopt $K = 21$ as the default setting, which provides the best trade-off between efficiency and generation quality. These results also demonstrate that the effectiveness of L2M does not rely on an excessively large context budget.

Comparison of Degradation Strength. We further investigate the sensitivity of L2M to the degradation strength used during training. As shown in Table 11, removing degradation-aware training reduces the Quality Score and Overall Consistency to 80.66 and 24.68, respectively. Applying moderate perturbations with the default setting ($p = 0.2$, $\beta = 0.1$) yields the best results, reaching 81.15 and 24.78. Increasing the perturbation strength beyond this setting slightly degrades performance to 80.92 and 24.73. These results suggest that moderate degradation provides effective robustness regularization, whereas overly strong perturbations may distort historical features and hinder importance estimation.

Frame-wise versus Token-wise Eviction In this paper, we find that using frames as the minimal unit for context retention and eviction is more reasonable than using tokens. Specifically, we compare

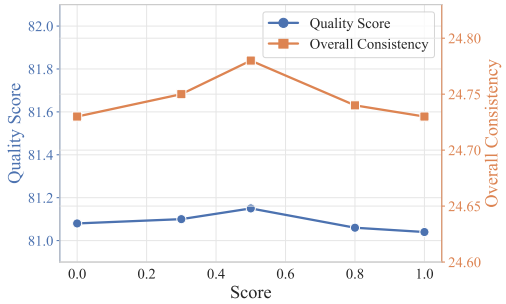

Figure 5: **Comparison of EMA coefficient α.** Results on 30-second videos under different α.

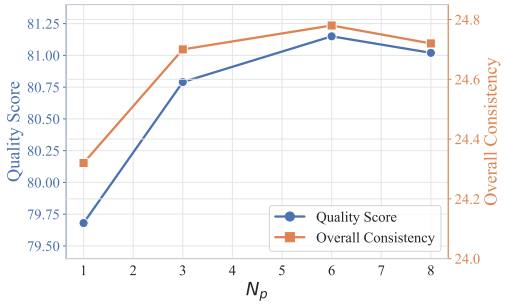

Figure 6: **Comparison of update interval N_p.** Results on 30-second videos under different N_p.

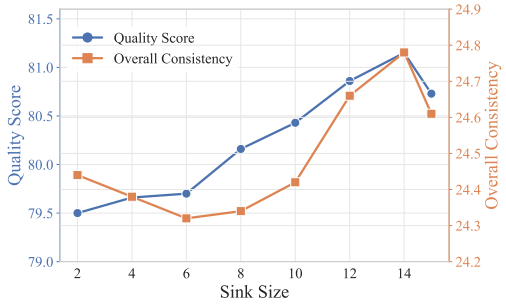

Figure 7: **Comparison of sink size.** Results on 30-second videos under different sink size.

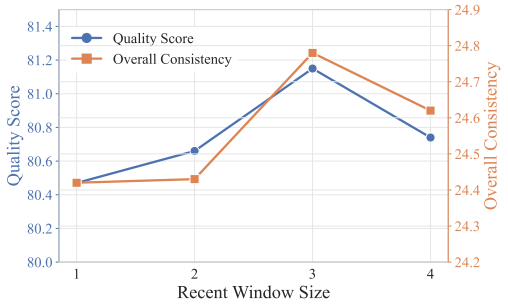

Figure 8: **Comparison of recent window size.** Results on 30-second videos under different recent window size.

against a token-wise eviction strategy and obtain a quality score of 80.45 on 30-second videos, which is significantly lower than our frame-wise method (81.15). This degradation mainly arises because token-wise eviction introduces large temporal gaps in positional encodings across different frames. Although applying temporal RoPE adjustment can partially compensate for this issue, it forces tokens from different frames to be mapped into overly similar temporal positions, which in turn confuses the model's contextual understanding and leads to inferior generation quality.

Attention Pattern Analysis In this work, we observe that attention weight distributions vary significantly across different samples and network layers. Unlike prior observations in StreamLLM Xiao et al. [2023], where attention responses are predominantly concentrated at the beginning of the sequence, we find that high-attention regions can emerge at different positions depending on the sample and the layer, as illustrated in Figure 9.

This suggests that, in autoregressive video generation, sink tokens are not necessarily confined to the initial positions of the sequence, but may instead be distributed across the entire context. In practice, earlier frames and nearby frames often exhibit higher relevance, which explains their generally stronger attention responses. However, we also observe clear discrepancies in high-response regions across layers (e.g., between layer 15 and layer 25), indicating that attention patterns are both content-dependent and layer-specific.

These findings motivate the use of a content-aware, dynamically adaptive sink mechanism, rather than a position-based static sink.

Consistency of Importance Prediction across Diffusion Timesteps In our design, the importance Router is evaluated only at the first timestep, rather than being recomputed at every timestep. The predicted frame-wise importance pattern is then reused throughout subsequent steps. This design enables immediate token eviction when a new frame begins to be generated, thereby avoiding additional memory overhead. It also eliminates the extra computational cost associated with repeatedly running the Router at each timestep.

Table 10: **Comparison of cache budget K.** Higher is better for both metrics. The best results are in **bold**, and the second-best results are underlined.

Cache Budget (K)	Quality Score ↑	Overall Consistency ↑
18	80.98	24.72
21	**81.15**	24.78
24	81.05	24.79
27	<u>81.10</u>	<u>24.83</u>
30	81.02	**24.84**

Table 11: **Comparison of degradation strength.** Higher is better for both metrics. The best results are in **bold**.

Training Setting	Quality Score ↑	Overall Consistency ↑
No degradation ($p = 0, \beta = 0$)	80.66	24.68
Default ($p = 0.2, \beta = 0.1$)	**81.15**	**24.78**
Stronger degradation	80.92	24.73

As shown in Figure 10, we verify that the importance score patterns predicted from features at different timesteps are highly consistent, and lead to comparable final video quality. This robustness mainly stems from our training strategy, where the Router is exposed to noisy features from randomly sampled timesteps. As a result, the Router learns to make reliable decisions under varying noise levels. Consequently, the importance scores predicted at the first timestep generalize well across the entire sampling process.

Additional Context Pruning Moreover, as shown in Figure 10, we observe that the magnitude of token importance scores gradually decreases as the diffusion process progresses. This trend suggests that tokens at later timesteps tend to be more redundant. This observation is consistent with prior findings that early timesteps primarily determine the coarse structure and semantic content of the generated frames, while later timesteps focus on refining fine-grained details. Consequently, redundancy is typically higher in the later stages of the diffusion process. Motivated by this insight, we further explore reducing the attention window size at subsequent timesteps. Our results show that removing an additional 10% of context tokens in later steps results in only a marginal performance drop of 0.3%, while further reducing computational cost.

Table 12: Results of additional context pruning at later timesteps.

Context Pruning	Quality Score	Overall Consistency
L2M	81.15	24.78
Uniform pruning (10%)	80.90	24.72
Uniform pruning (20%)	80.68	24.67

Additional Comparisons with Existing Methods We provide additional visual comparisons with existing methods, as shown in Figure 11. Our results exhibit better overall consistency and richer motion dynamics.

Router Reliability and Error Analysis Incorrect routing decisions may propagate errors to subsequent frames. To reduce its dependence on individual predictions, L2M incorporates three complementary safeguards: (1) degradation-aware training teaches the router to identify and down-weight unreliable historical features; (2) EMA smoothing aggregates importance scores over time, reducing sensitivity to transient prediction noise; and (3) the N_p-step survival criterion prevents newly generated frames from being prematurely promoted to long-term memory.

We directly assess router reliability through a controlled diagnostic in which the current query feature and context positions are fixed. The router assigns average retention scores of 0.61, 0.24, and 0.19 to clean, synthetically degraded, and rollout-degraded frames, respectively. The substantially lower scores for both types of degraded frames demonstrate that the router learns to suppress not only

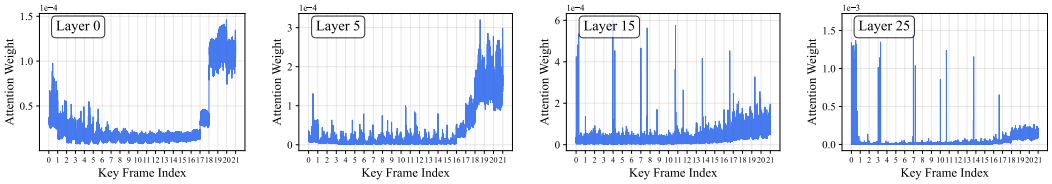

Figure 9: Visualization of attention weights across layers. Query-averaged attention maps illustrate how each key frame is attended during the generation of the final chunk.

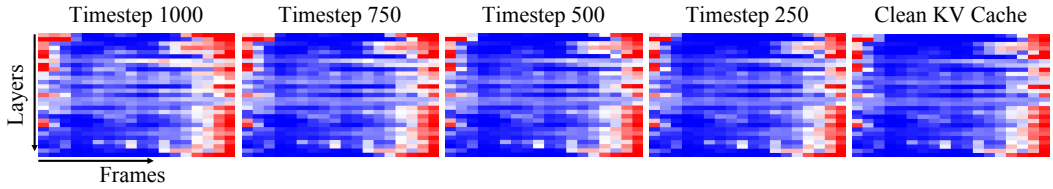

Figure 10: Consistency of importance scores across diffusion timesteps. The stable importance patterns observed at different timesteps (1000, 750, 500, 250, and the final clean KV cache) indicate that predictions made by the Router at the initial timestep ($t = 1000$) remain reliable throughout the entire denoising process.

synthetic perturbations encountered during training but also artifacts arising from actual autoregressive rollouts.

Nevertheless, occasional routing errors may still propagate, particularly when the historical context contains semantic inconsistencies such as identity drift. Although the proposed mechanisms reduce this risk, they cannot eliminate it entirely.

Failure Cases Our method does not require additional training of the base model, enabling efficient and low-cost performance improvements. However, this design also limits the upper bound of achievable gains. Although L2M consistently outperforms existing baselines, it may still suffer from abrupt content changes in scenes with drastic motion or rapid scene transitions. In addition, since our approach operates as a dynamic memory regulation mechanism, poor-quality frames stored in the initial memory cache may not be evicted promptly and can propagate erroneous information to subsequent generation, thereby degrading future visual quality.



Figure 11: Qualitative comparison with existing methods. Our method achieves better visual quality, overall consistency patterns.

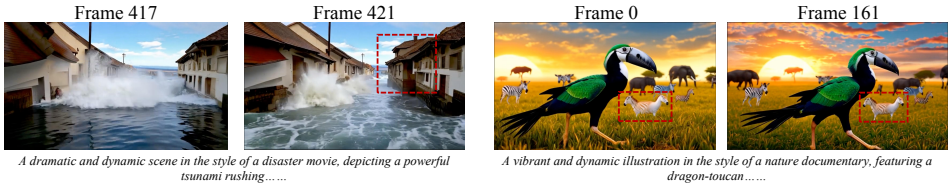

Figure 12: **Failure cases of L2M.** In the first example, rapid camera movement and highly dynamic scene changes cause abrupt content inconsistency, where a new house artifact appears after the waves sweep across the scene. In the second example, an incorrectly generated two-headed zebra appears in several initial frames and remains in subsequent generations due to delayed memory eviction.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly describe our main contributions, including dynamic long-term memory management, importance prediction router, and extensive evaluations on long video generation benchmarks. The claims are consistent with experimental results presented in the main paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in the Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical proofs. It focuses on empirical methods and experimental validation.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed model architecture, training settings, inference procedure, and evaluation protocol, sufficient to reproduce the main results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is provided in the attach file.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper includes full training details, hyperparameters, data processing, and evaluation settings in the main paper and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: We follow standard evaluation protocols for video generation benchmarks and report averaged results, variance across runs is small and not the primary focus of this work.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide details on training hardware (GPU type, number of devices), training time, and inference cost in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work complies with the NeurIPS Code of Ethics, involving no human subjects or sensitive data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss potential positive impacts in improving long video generation and possible negative impacts such as misuse for deepfake generation, along with mitigation considerations.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk generative models with direct deployment safeguards beyond standard research usage.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use publicly available datasets and pretrained models with properly cited sources and respect their licenses as stated in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce new datasets or external assets beyond the proposed method and model.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve human subject experiments or crowdsourcing studies.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: A large language model (Qwen2.5-7B-Instruct) is used only for prompt refinement and does not influence the core method design or evaluation.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.