Not All Context Is Memory: Rethinking Time, Depth, and Space in VLA-Based Object Navigation

Behrad Rabiei **Yue Hu** **Yulun Tian** **Maani Ghaffari**
University of Michigan

Abstract: Vision language action (VLA) models for navigation add hundreds of visual tokens to their context at every step, so memory usage and inference latency grow with trajectory length. This history is retained because long horizon navigation requires global decisions that depend on observations made many steps earlier. We investigate whether this accumulated context functions as an effective memory. Further, we study context management along three dimensions, time (how quickly a past frame stops mattering), depth (which decoder layers use visual tokens), and space (which patches within a frame are worth keeping), on two state-of-the-art object goal navigation VLAs. To explain our empirical findings, we utilize an information-theoretic metric to quantify how much the removed context shifts the model's action distribution. Along time, the influence of a frame decreases with its age, and performance holds with as few as sixteen recent frames. Along depth, visual tokens influence actions only in the earlier decoder layers. Along space, none of three existing token selection criteria outperforms uniform random sampling at a matched budget. The strong recency bias along time indicates that the models do not effectively retrieve information from the distant past, a defining characteristic of an effective memory. Our results therefore suggest that retaining the full context does not by itself yield an effective memory, and that performance can be preserved by keeping minimally sufficient context along time and depth. A protocol built on these findings matches full-context performance on Habitat ObjectNav while reducing cache memory by 81% on average.

1 Introduction

Vision language action (VLA) models, which adapt pretrained multimodal language models to map camera observations and language instructions directly to robot actions, have become a leading approach to both manipulation [1, 2, 3] and navigation [4, 5, 6, 7]. Because they keep the autoregressive decoder of their backbone, they also keep its KV cache, so that attention over the history is not recomputed at every control step. Each new frame adds hundreds of visual tokens to this cache. Object goal navigation (ObjectNav) runs for hundreds of steps, during which the agent is expected to draw on earlier observations to avoid re-exploring visited areas. Memory use and per-step latency therefore rise throughout an episode while the robot's onboard compute stays fixed.

Existing work in token pruning mostly focuses on static or open-loop settings such as image and video question answering [8, 9]. State-of-the-art methods rank visual tokens by attention [10, 11], select a subset with minimal redundancy [12, 13], remove all visual tokens at a fixed decoder layer [14], or exploit redundancy between consecutive views in manipulation [15, 16, 17]. Even so, random pruning remains a baseline that no single method beats consistently [18], and beyond an intermediate decoder layer no tested criterion performs better than random [19]. Navigation models that compress their context [5, 6, 20, 21, 22, 23] are evaluated almost entirely on instruction following, where the instruction lays out the route in detail. In contrast, ObjectNav only specifies a target category, making this one of the tasks in which context should matter most. Nevertheless, to our knowledge, there is not yet a conclusive study that investigates whether VLA-based ObjectNav policies possess an effective memory and how existing token pruning strategies perform in this closed-loop setting.

10th Conference on Robot Learning (CoRL 2026), Austin TX, USA.

In this work, we study two ObjectNav VLAs, LongNav-R1 [4], built on Qwen3-VL-2B [24], and Uni-NaVid [6], built on EVA-CLIP [25] and Vicuna-7B [26], both of which already compress their visual history (Appendix D). For each model we prune the context along time, depth, and space, and explain each result by how much the removed context shifts the model's action distribution. We find that tokens from old frames or deep layers contribute almost nothing to the predicted action, and no selection criterion outperforms random sampling at a matched budget. An effective memory must retrieve information from the distant past, yet the strong recency bias we observe shows that this retrieval does not occur. These models therefore retain their full history without using it as memory, and when tokens must be pruned, where to prune matters more than how. We distill these findings into a context management protocol that matches full-context performance while reducing cache memory by 81% on average.

2 Formulation and Approach

Object goal navigation. We study ObjectNav as formulated in Habitat [27]. Given a target object category $g \in \mathcal{G}$, at each step t the agent receives an egocentric RGB observation o_t and selects an action a_t from the discrete set $\mathcal{A} = \{\texttt{forward}, \texttt{left}, \texttt{right}, \texttt{stop}\}$. An episode is successful if the agent calls `stop` within 1 m geodesic distance of a visible instance of g before the step limit of $T_{\max} = 500$.

Policy and KV cache. Each observation is encoded into n visual embeddings, which enter the decoder as visual tokens alongside a fixed set of task tokens and, optionally, the tokens of previous actions (Appendix A). A causal transformer decoder with layers $l \in \{1, \dots, L\}$ consumes these tokens and outputs a distribution over $\mathcal{A}$. At every layer, each visual token contributes a key and a value, which we call an *entry* and index by the triple (l, t', i), the layer l, the frame $t' \leq t$ the token belongs to, and its position $i \in \{1, \dots, n\}$ within that frame. Entries are what the decoder attends over and what the KV cache stores, so they are the unit we manage. Processing a block of tokens to populate its entries is called prefill. LongNav-R1 keeps the entries in a KV cache that persists across the episode and prefills only the n tokens of each new frame, so every past entry remains resident in memory. Uni-NaVid keeps no cache across steps and prefills its entire context at every step, a cost it bounds by merging historical tokens.

Context management. The index (l, t', i) exposes three dimensions along which entries can be evicted, namely *time*, along the age $t - t'$, *depth*, along the layer l, and *space*, along the position i.

Problem 1 (Context management under a memory budget)**.** Let $\mathcal{F}_t = \{(l, t, i) : 1 \leq l \leq L,\ 1 \leq i \leq n\}$ denote the visual entries of the current frame and let $\mathcal{S}_{t-1}$ denote the visual entries retained at the previous step, with $\mathcal{S}_0 = \emptyset$. At every step t, choose the visual entries $\mathcal{S}_t$ to retain such that

$$\mathcal{S}_t \subseteq \mathcal{S}_{t-1} \cup \mathcal{F}_t, \qquad |\mathcal{S}_t| \leq B, \tag{1}$$

where B is a budget on retained visual entries across all decoder layers. The problem is closed loop, since an eviction at step t changes the subsequent actions and observations of the agent.

Budget allocation. Along time, we retain the entries of the K most recent frames, current frame included, and evict older ones. Along depth, we evict every visual entry at layers $l \geq l_h$, where l_h is the information horizon [19], the depth beyond which visual entries no longer contribute to the action. Sweeping K and l_h determines how much history and how many layers the policy needs. We state budgets in units of frames, where a budget of k frames corresponds to kLn entries.

Token selection. Within the allocated region, a utility function u scores sets of entries, and we retain the subset that maximizes it at the budget. We compare four utilities (Appendix B). *Attention* based selection follows FastV [10], *diversity* based selection follows DivPrune [12], and *duplication* based selection follows DART [13]. *Random* selects a uniform sample without an explicit utility function.

Influence. Navigation success is observed only when an episode ends, so we need a per step measure of how much the policy draws on a part of its context. Since context affects the outcome only through the actions the policy selects, we measure how far the action distribution shifts when that

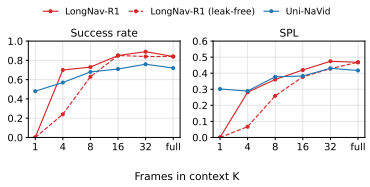
part is removed. Let $\mathcal{V}_t$ be the set of all visual entries at step t, let $\mathcal{R}^{\geq l}$ and $\mathcal{R}^{<l}$ be the entries of a set $\mathcal{R}$ at layers at or above l and below l, and let $\pi(\cdot \mid \mathcal{S})$ be the policy acting with its visual entries restricted to $\mathcal{S}$. The influence of $\mathcal{R} \subseteq \mathcal{V}_t$ from layer l onward is

$$\mathrm{Infl}_t(\mathcal{R}, l) = D_{\mathrm{KL}}\big(\pi(\cdot \mid \mathcal{V}_t^{<l} \cup \mathcal{R}^{\geq l}) \,\big\|\, \pi(\cdot \mid \mathcal{V}_t^{<l})\big)\,. \tag{2}$$

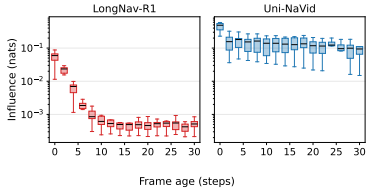
Both policies see the full visual context below layer l, so the measure isolates what $\mathcal{R}$ contributes at layers l and above. Unlike attention weights or hidden states, the action distribution is also comparable across the two models. Along time, $\mathrm{Infl}_t(\mathcal{F}_{t'}, 1)$ measures how much a single frame of age $t - t'$ still moves the action. Along depth, $\mathrm{Infl}_t(\mathcal{F}_t, l)$ measures whether the current frame is still read at depth l.

3 Experiments

Setup. LongNav-R1 [4] has $L = 28$ decoder layers and Uni-NaVid [6] has $L = 32$. We evaluate on a fixed subset of 100 episodes of the HM3D-v2 validation split [28], additionally evaluate the protocol on 100 episodes of MP3D [29], and compute the influence measures on 10 episodes. We report success rate (SR) and success weighted by path length (SPL), with details in Appendix D.



(a) Window sweep



(b) Influence against age

Figure 1: Time. (a) SR and SPL against the window K. (b) $\mathrm{Infl}_t(\mathcal{F}_{t'}, 1)$ against frame age.

Context management along time. The *window* retains every entry of the K most recent frames, current frame included, and evicts the rest. Both models show a strong recency bias (Figure 1a). We run LongNav-R1 in the *leaky* regime, which evicts entries from the existing cache as a deployed system would, and in the *leak free* regime, which rebuilds the cache from the retained tokens alone (Appendix C). In the leaky regime, $K = 16$ holds SR and $K = 32$ holds both SR and SPL. The influence measure explains the recency bias (Figure 1b). For LongNav-R1, the median influence falls by two orders of magnitude between the current frame and age 10 and then stays at a floor, and $K = 16$ is the smallest window in the sweep that covers the frames above it. For Uni-NaVid, the influence of the current frame is an order of magnitude higher, and its historical frames decay more slowly. In both models, frames beyond the window barely move the action, so information from the distant past is not effectively retrieved. Since such retrieval is a defining characteristic of an effective memory, retaining the full context does not by itself yield one.

Context management along depth. The *cutoff* evicts every visual entry at layers $l \geq l_{\mathrm{cut}}$, current frame included, and retains everything below. Both models reach full context performance well before their last layer (Figure 3a). LongNav-R1 fails every episode for $l_{\mathrm{cut}} \leq 15$ and matches full context from $l_{\mathrm{cut}} = 18$ onward, and Uni-NaVid from $l_{\mathrm{cut}} = 17$. The influence of the current frame is consistent with these horizons (Figure 3b). For LongNav-R1 it is flat up to $l_{\mathrm{cut}} = 17$ and falls by two orders of magnitude at $l_{\mathrm{cut}} = 18$. The plateau is an artifact of the leaky probe, in which

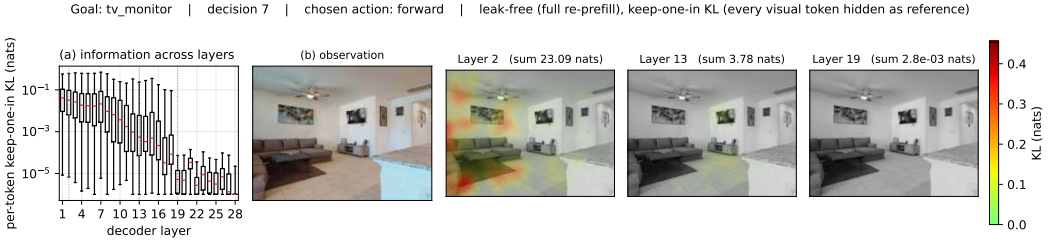
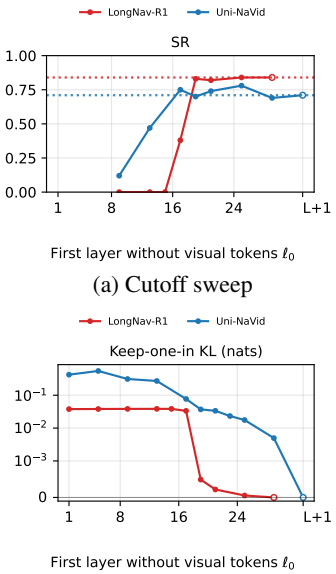


Figure 2: Per token influence of the current frame on LongNav-R1 (leak free) at one step: distribution over tokens per layer (left) and influence of each patch at layers 2, 13, and 19 (right).

the text tokens prefilled before the decision token still carry the frame, so the measure captures only what the decision token reads directly (Appendix C). The per token influence in the leak free regime confirms the horizon without this artifact (Figure 2). For Uni-NaVid the curve declines gradually, with a small residual in the last layers that does not affect SR or SPL.

Token selection. A *selector* keeps the current frame whole and reduces the history to a budget of k frames with one of the four utilities. On LongNav-R1 no selector outperforms random sampling (Figure 4a). FastV and random track each other, while DivPrune and DART lie far below both at every budget. Random sampling keeps the same fraction of every frame, whereas DivPrune and DART score entries by content alone. Since consecutive frames are near duplicates, both criteria evict tokens of the most recent frames and spend the budget on older frames whose influence is minimal (Figure 1b). On Uni-NaVid the four selectors lie within a narrow band, since its built in merging leaves little for a selector to remove. Within a frame, none of the three criteria is positively correlated with influence on LongNav-R1 (Appendix E.3).



(a) Cutoff sweep

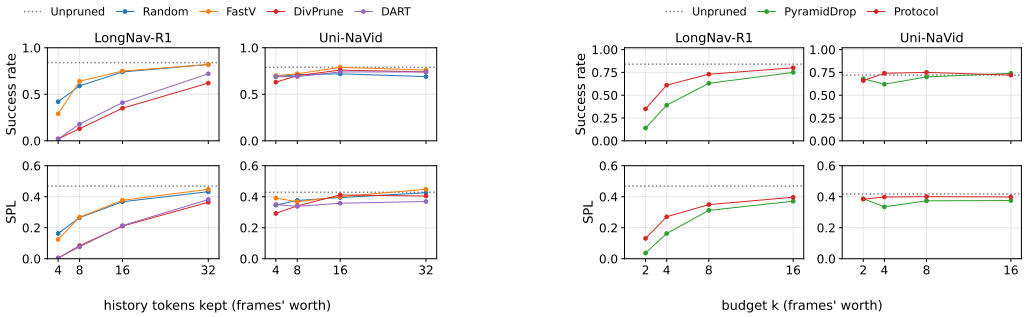
(b) Current frame influence

Figure 3: Depth. (a) SR against the first layer l_{cut} without visual entries. (b) $\text{Infl}_t(\mathcal{F}_t, l_{\text{cut}})$.

Protocol. The *protocol* combines the three findings. At and above l_h every visual entry is evicted, and below l_h the current frame is kept whole, frames older than K are evicted, and the remaining history is reduced to a budget of k frames by random selection. We set $K = 32$ for both models, $l_h = 18$ for LongNav-R1, and $l_h = 20$ for Uni-NaVid as a margin for its residual influence. The baseline is PyramidDrop [11] run as published at a matched budget of historical entries. On LongNav-R1 the protocol outperforms PyramidDrop at every budget in both metrics (Figure 4b), and at $k = 16$ it approaches full context in SR. On Uni-NaVid the protocol is at or near full context at every budget. The ordering and further details of the baseline are in Appendix E.4, including results on MP3D.

4 Conclusion and Ongoing Work

The growing visual context of a navigation VLA largely does not function as memory. Both models show a strong recency bias and do not effectively retrieve information from the distant past, which is a defining characteristic of an effective memory. Actions instead depend on a small, recent, and shallow part of the context, and no selection criterion beats uniform random sampling. A protocol built on these measurements matches full context performance on HM3D-v2 and MP3D at constant memory and latency, without retraining. We are now refining its budget allocation, repeating the ablations on more recent navigation VLAs, and moving from simulation to a physical robot.



(a) Selectors

(b) Protocol and PyramidDrop

Figure 4: HM3D-v2: SR (top) and SPL (bottom) against budget k. Dotted line: full context.

References

[1] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[2] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[3] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pages 29982–30009, 2025.

[4] Y. Hu, A. Xi, Q. Xiao, S. Isaacson, H. X. Liu, R. Vasudevan, and M. Ghaffari. Longnav-r1: Horizon-adaptive multi-turn rl for long-horizon vla navigation, 2026. URL https://arxiv.org/abs/2602.12351.

[5] J. Zhang, K. Wang, R. Xu, G. Zhou, Y. Hong, X. Fang, Q. Wu, Z. Zhang, and H. Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. *arXiv preprint arXiv:2402.15852*, 2024.

[6] J. Zhang, K. Wang, S. Wang, M. Li, H. Liu, S. Wei, Z. Wang, Z. Zhang, and H. Wang. Uninavid: A video-based vision-language-action model for unifying embodied navigation tasks. *arXiv preprint arXiv:2412.06224*, 2024.

[7] Z. Qi, Z. Zhang, Y. Yu, J. Wang, and H. Zhao. Vln-r1: Vision-language navigation via reinforcement fine-tuning. *arXiv preprint arXiv:2506.17221*, 2025.

[8] K. Tao, C. Qin, H. You, Y. Sui, and H. Wang. Dycoke: Dynamic compression of tokens for fast video large language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18992–19001. IEEE, 2025.

[9] Y. Yang, Z. Zhao, S. N. Shukla, A. Singh, S. K. Mishra, L. Zhang, and M. Ren. Streammem: Query-agnostic kv cache memory for streaming video understanding. *arXiv preprint arXiv:2508.15717*, 2025.

[10] L. Chen, H. Zhao, T. Liu, S. Bai, J. Lin, C. Zhou, and B. Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35. Springer, 2024.

[11] L. Xing, Q. Huang, X. Dong, J. Lu, P. Zhang, Y. Zang, Y. Cao, C. He, J. Wang, F. Wu, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024.

[12] S. R. Alvar, G. Singh, M. Akbari, and Y. Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9392–9401. IEEE, 2025.

[13] Z. Wen, Y. Gao, S. Wang, J. Zhang, Q. Zhang, W. Li, C. He, and L. Zhang. Stop looking for "important tokens" in multimodal language models: Duplication matters more. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9972–9991, 2025.

[14] Z. Lin, M. Lin, L. Lin, and R. Ji. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 5334–5342, 2025.

[15] S. Xu, Y. Wang, C. Xia, D. Zhu, T. Huang, and C. Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. *Advances in Neural Information Processing Systems*, 38:164448–164473, 2026.

[16] T. Jiang, X. Jiang, Y. Ma, X. Wen, B. Li, K. Zhan, P. Jia, Y. Liu, S. Sun, and X. Lang. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. *arXiv preprint arXiv:2509.12594*, 2025.

[17] Z. Liu, Y. Chen, H. Cai, T. Lin, S. Yang, Z. Liu, and B. Zhao. Bridging the semantic-action gap in visual token pruning for efficient vla inference. *arXiv preprint arXiv:2511.16449*, 2025.

[18] T. Peng, Y. Du, P. Ji, S. Dong, K. Jiang, M. Ma, Y. Tian, J. Bi, Q. Li, W. Du, et al. Can visual input be compressed? a visual token compression benchmark for large multimodal models. *arXiv preprint arXiv:2511.02650*, 2025.

[19] Y. Wang, J. Wu, Z. Ni, L. Yang, Y. Liu, C. Yang, Y. Wen, L. He, X. Tang, H. Liu, et al. When token pruning is worse than random: Understanding visual token information in vllms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 31910–31919, 2026.

[20] M. Wei, C. Wan, X. Yu, T. Wang, Y. Yang, X. Mao, C. Zhu, W. Cai, H. Wang, Y. Chen, et al. Streamvln: Streaming vision-and-language navigation via slowfast context modeling. *arXiv preprint arXiv:2507.05240*, 2025.

[21] L. Zhang, X. Hao, Q. Xu, Q. Zhang, X. Zhang, P. Wang, J. Zhang, Z. Wang, S. Zhang, and R. Xu. Mapnav: A novel memory representation via annotated semantic maps for vlm-based vision-and-language navigation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13032–13056, 2025.

[22] Y. Lu, S. Sun, N. Liu, B. Jiang, Y. Zhang, J. Chen, and C. Du. Step-nav: Spatial-temporal efficient visual token pruning for vision-and-language navigation with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40:24097–24105, 03 2026. doi: 10.1609/aaai.v40i29.39588.

[23] Q. Wang, Y. Liang, M. Li, T. Zhou, and C. Rasmussen. History-conditioned spatio-temporal visual token pruning for efficient vision-language navigation. *arXiv preprint arXiv:2603.06480*, 2026.

[24] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[25] Q. Sun, Y. Fang, L. Wu, X. Wang, and Y. Cao. Eva-clip: Improved training techniques for clip at scale. *arXiv preprint arXiv:2303.15389*, 2023.

[26] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[27] K. Yadav, J. Krantz, R. Ramrakhya, S. K. Ramakrishnan, J. Yang, A. Wang, J. Turner, A. Gokaslan, V.-P. Berges, R. Mootaghi, O. Maksymets, A. X. Chang, M. Savva, A. Clegg, D. S. Chaplot, and D. Batra. Habitat challenge 2023. https://aihabitat.org/challenge/2023/, 2023.

[28] K. Yadav, R. Ramrakhya, S. K. Ramakrishnan, T. Gervet, J. Turner, A. Gokaslan, N. Maestre, A. X. Chang, D. Batra, M. Savva, A. W. Clegg, and D. S. Chaplot. Habitat-matterport 3d semantics dataset, 2023. URL https://arxiv.org/abs/2210.05633.

[29] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

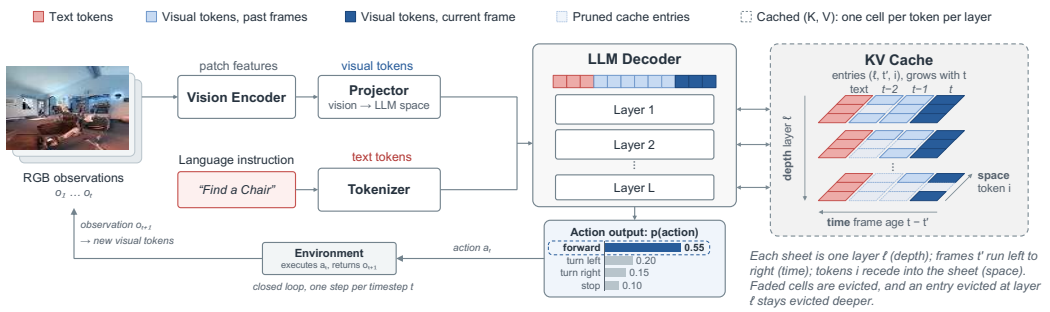


Figure 5: VLA architecture and the point of intervention. Each observation is encoded into n visual tokens, appended to the context $\mathcal{C}_t$, and processed by an L layer causal transformer decoder whose keys and values are cached. We evict visual entries from the cache, while the encoder, projector, and decoder are unchanged.

A Architecture and KV Cache

Architecture. A navigation VLA, illustrated in Figure 5, consists of a vision encoder, a projector, and a causal transformer decoder [30] that produces the action. At step t the vision encoder E maps the observation to a grid of patch features and the projector P maps these into the decoder's embedding space, $x_t = P(E(o_t)) \in \mathbb{R}^{n \times d}$, yielding n visual tokens for frame t, where d is the decoder hidden dimension. LongNav-R1 uses up to $n = 300$ and $d = 2048$. Uni-NaVid uses $d = 4096$ and compresses 256 tokens to $n = 64$ for the current frame, and its built in token merging compresses historical frames further. The current frame's visual tokens are concatenated with a fixed set of task tokens τ_g encoding the goal category and instruction, the visual tokens of all previous frames, and optionally the tokens of previous actions, to form the context

$$\mathcal{C}_t = (\tau_g,\ x_1, \ldots, x_t,\ [\,a_1, \ldots, a_{t-1}\,])\,, \tag{3}$$

where square brackets denote an optional component. Uni-NaVid conditions only on task and visual tokens, whereas LongNav-R1 additionally conditions on its action history. The decoder outputs a distribution over $\mathcal{A}$ by decoding the action as text from the language head. Because tokens are appended in temporal order, a token from frame t' can never attend to a later frame.

KV cache and prefill. At layer l a token contributes a key $k^{(l)}_{t',i}$ and a value $v^{(l)}_{t',i}$ to the attention of every later token, and this pair is the entry (l, t', i) of Section 2. Because an entry depends only on tokens that precede (t', i), it does not change when new frames are appended, so inference can compute it once and store it. This store is the KV cache, and processing a block of tokens to populate its entries is called prefill. A model with a persistent cache, such as LongNav-R1, prefills only the n tokens of each new frame and attends against the cached entries of everything earlier, which reduces per step compute from $\mathcal{O}(|\mathcal{C}_t|^2)$ to $\mathcal{O}(|\mathcal{C}_t|)$ at the cost of keeping every past token resident in memory. The cache occupies $\mathcal{O}(Ld\,|\mathcal{C}_t|)$ with $|\mathcal{C}_t| = |\tau_g| + nt + (t-1)$, and the visual history nt dominates since the task and action tokens number in the hundreds. A model without a persistent cache, such as Uni-NaVid, prefills its entire context at every step, which avoids the memory growth but pays for the whole history, a cost it bounds by merging historical tokens. On the same evaluation episode, LongNav-R1 accumulates 36,300 visual tokens over 121 steps, occupying 4.16 GB of cache, while Uni-NaVid reaches the goal in 101 steps and retains only 322 visual tokens after merging, which would occupy 0.17 GB were its history cached. Removing a token from $\mathcal{C}_t$ removes its entries from the cache at the corresponding layers, or, for a model without a persistent cache, from the context that is prefilled, so context management and cache management are the same operation.

B Token Selection Utilities

Scoring. Every entry of a token receives the same score, computed at the first decoder layer, so a token is kept or evicted at all layers together and every selector holds the same number of entries at a given budget. Ranking is over all frames of the history jointly, so a token can be evicted for being outscored by, or duplicating, a token of another frame. For an entry e, let $\alpha_t(e)$ be the attention at that layer from the last text token of $\mathcal{C}_t$ to the token of e, averaged over heads, and let h_e be the hidden state of the token of e entering that layer, which at layer 1 is the projected embedding $x_{t',i}$.

Utilities. *Attention* based selection keeps the entries the text reads most, as in FastV [10],

$$u^{\mathrm{att}}(\mathcal{S}) = \sum_{e \in \mathcal{S}} \alpha_t(e). \tag{4}$$

Diversity based selection ignores attention and keeps a subset that covers the hidden states under the cosine distance δ,

$$u^{\mathrm{div}}(\mathcal{S}) = \min_{\substack{e,e' \in \mathcal{S} \\ e \neq e'}} \delta(h_e, h_{e'}), \tag{5}$$

which DivPrune [12] maximizes greedily. *Duplication* based selection, DART [13], fixes a small set of pivot entries Π, always keeps them, and evicts the entries most similar to any pivot,

$$u^{\mathrm{dup}}(\mathcal{S}) = -\sum_{e \in \mathcal{S} \setminus \Pi} \max_{p \in \Pi} \cos(h_e, h_p). \tag{6}$$

Random selection takes $u^{\mathrm{rand}}(\mathcal{S}) = 0$, so the retained set is a uniform sample. Attention and duplication are additive over entries and are maximized exactly by keeping the highest scoring entries. All three criteria were designed for a single image and applied once, and we apply them at every step as the history grows. *PyramidDrop* [11] applies attention based selection at stage layers $l_1, \ldots, l_s$ with a geometric budget, each stage keeping a fraction r of the entries kept by the previous one. We run it as published, over the whole history with no window.

Implementation. LongNav-R1 keeps a KV cache that persists across the episode, so evicting an entry deletes it from that cache. We leave its built in filter, which drops near duplicate tokens within a frame at ingestion, enabled except where noted. Uni-NaVid keeps no cache across steps. It stores the merged visual features of each frame and prefills the decoder from them at every step, so eviction removes features before prefill, and its historical frames hold fewer than $n = 64$ tokens after merging.

C Influence

Interpretation. In Equation 2, both distributions are read from the logits of the action tokens at the language head. The policy that sees $\mathcal{R}$ is the reference, so the measure is large when the policy without $\mathcal{R}$ places little mass on actions the policy with $\mathcal{R}$ favors, and zero exactly when the two agree. At $l = 1$ it compares $\mathcal{R}$ alone to the empty visual context and is a pointwise analogue of the mutual information between the action and $\mathcal{R}$, with the model's prediction under the empty context standing in for the marginal. It differs from the attention scores of Appendix B in what it measures. Attention quantifies how much an entry is read, influence how much the action changes when the entry is present. Along space, $\mathcal{R} = \{(l', t, i) : l' \geq l\}$ is a single token of the current frame, following the layer wise analysis of [19]. Rendered over the token grid at each l, the measure shows where in the image and at what depth information enters the action, and for these maps we disable LongNav-R1's filter so that the full grid is present. Frame level influence costs $t - 1$ additional forward passes per step, so we compute it on a subset of episodes.

Information leakage. Evicting an entry from a persistent cache does not remove all of its information. Every cached entry is computed once, by attending over the entries of all earlier tokens at the layer below, and is never updated. The entries of a later frame or action token that attended to $(l-1, t', i)$ therefore still carry its information after every entry of token (t', i) is evicted. Eviction

severs only the direct path. In the *leaky* regime, entries are dropped from the existing cache, which is what a deployed system does. In the *leak free* regime, the cache is rebuilt by prefilling only the retained tokens from scratch, with the same positions as before, so that no evicted token enters any state. The leak free regime costs a full prefill per step and is not deployable, but it is the correct reference for how much of its history a model needs, and the gap between the two regimes measures leakage. The window sweep and the per frame influence curve run LongNav-R1 in both regimes. Every other experiment runs it in the leaky regime, since the depth and space strategies and the protocol are meant to be deployed as they are evaluated. Uni-NaVid prefills its whole context at every step, so for it the two regimes coincide.

D Experimental Details

Task. In the ObjectNav formulation of Habitat [27], an agent is placed at a random pose in an unknown indoor environment and given a target object category $g \in \mathcal{G}$ (e.g., *chair*, *bed*). The actions `forward`, `left`, `right`, and `stop` correspond to a $0.25\,\mathrm{m}$ translation, a 30° rotation in either direction, and episode termination. An episode is successful if the agent calls `stop` within $1\,\mathrm{m}$ geodesic distance of an instance of g that is visible from its position, and fails if it stops elsewhere or exhausts the step limit of $T_{\max} = 500$.

Models. LongNav-R1 uses a Qwen3-VL-2B [24] backbone with $L = 28$ decoder layers and maintains a KV cache that persists across the episode. Uni-NaVid is the official 7B checkpoint with $L = 32$ decoder layers. It prefills a single turn prompt at every step from its stored per frame features, and we use its released sampled decoding at temperature 0.2. LongNav-R1's built in filter is enabled throughout except for the per token maps. Decoder layers are indexed from 1, as in Section 2.

Benchmark. We evaluate on the HM3D-v2 validation split for object goal navigation [28], and additionally evaluate the protocol on the MP3D validation split [29]. To keep the sweeps tractable we fix a subset of 100 episodes from each, drawn once with a fixed seed, and use the same manifest, goal prompt, action space, and 500 step cap for every experiment and both models. The influence measures require additional forward passes at every step, so we compute them on a fixed subset of 10 episodes. We report success rate (SR) and success weighted by path length (SPL) as episode means.

Protocol configuration. Task and action tokens are never evicted. For LongNav-R1, $l_h = 18$ is the measured horizon, and eviction deletes entries from the persistent cache. For Uni-NaVid we set $l_h = 20$, above the measured horizon of 17, as a margin for the residual influence that Appendix E.2 found past that layer. Evicted history tokens are removed from its input, and visual tokens are masked from attention from layer 20 on.

Baseline. PyramidDrop runs with attention based stages at layers 1, 8, 15, and 22 for LongNav-R1 and 1, 9, 17, and 25 for Uni-NaVid, over the whole history and with the current frame never evicted. At every step its stage retention ratio r is chosen so that the historical entries retained, summed over all layers, equal k frames' worth, for $k \in \{2, 4, 8, 16\}$. At the same total, the protocol holds more history below the horizon and none above it, whereas PyramidDrop spreads its retention over every layer and every frame of the episode and ranks tokens by attention, which Appendix E.3 shows to be uninformative about which tokens the action depends on.

E Additional Results and Discussion

E.1 Time

The leak free regime of LongNav-R1 matches full context at $K = 16$ but degrades faster below it than the leaky regime, and the two meet only at $K = 1$, where LongNav-R1 fails every episode (Figure 1a). The gap between the regimes is information leakage. Leak free performance at a given K reflects those K frames alone, while leaky performance also reflects the information their entries

absorbed from the frames before them, a contribution that is largest at $K = 4$. A deployed system evicts from its live cache and retains this absorbed information, so we set the window of the protocol from the leaky curve. Uni-NaVid, for which the two regimes coincide, holds its SR to $K = 8$ and degrades gradually below it. At $K = 1$ it still succeeds in about half of the episodes, whereas LongNav-R1 succeeds in none, so the two models differ in how much they rely on the current frame alone.

In the influence curve of Figure 1b, LongNav-R1's median decreases monotonically with age and does not recover at any later age, since the upper whiskers beyond age 10 stay near the floor. Uni-NaVid's current frame influence is an order of magnitude above LongNav-R1's, consistent with its behavior at $K = 1$, and its historical frames decay with a spread of more than an order of magnitude at every age and no floor reached within the plotted range. The window that preserves navigation performance is in both cases the one that retains the frames whose influence has not yet decayed.

E.2 Depth

LongNav-R1 fails every episode for $l_{\text{cut}} \leq 15$, recovers part of its success at $l_{\text{cut}} = 17$, and matches full context in both SR and SPL from $l_{\text{cut}} = 18$ onward, while Uni-NaVid degrades more gradually below its transition (Figure 3a). In the influence curve of Figure 3b, Uni-NaVid declines by an order of magnitude between $l_{\text{cut}} = 1$ and $l_{\text{cut}} = 17$ and keeps a small residual into the last layers that does not affect SR or SPL. In both models the drop in influence and the recovery of navigation performance occur at the same layer, and since the two measurements are independent, their agreement indicates that the horizon is a property of the model rather than of either measurement.

Figure 6 shows the per token influence of the current frame at every decoder layer of LongNav-R1 for two decisions from each of two episodes, with the cache rebuilt at that step and the full 15×20 grid of 300 tokens present. Every decision drops to the floor at layer 18, the frame level horizon. Below it, the distribution over tokens differs from decision to decision. In one it declines with depth from the first layer, while in the other three it is flat or small in the early layers and peaks in the middle layers, so the per token measure is not monotone in depth the way the frame level measure is. The maps explain the peak. In the second layer, influence is either spread broadly over the scene or close to zero. By layer 13 it has concentrated on the structure that decides the action, namely the target television at both decisions of the television episode, and the edge of the door at step 9 and the floor through the doorway ahead at step 16 of the chair episode. The model therefore reads the goal relevant part of the image in its middle layers.

E.3 Space

We run the four selectors at $k \in \{4, 8, 16, 32\}$ with the whole episode in context (Figure 4a). On LongNav-R1, FastV is slightly above random from $k = 8$ and slightly below at $k = 4$, and both approach full context at $k = 32$, while DivPrune and DART fail almost every episode at $k = 4$. On Uni-NaVid the four selectors are at or near full context even at $k = 4$, since its built in merging leaves only a few tokens per historical frame and a budget of 4 frames' worth is already a large fraction of its entire history.

The budget sweep shows that the criteria misallocate the budget across frames, but not whether they identify the right tokens within a frame. To test this, we compute at every step of the 10 episode subset the per token influence of the current frame at $l = 1$ and the score each utility assigns to the same token, namely attention for FastV, distance to the nearest retained token for DivPrune, dissimilarity to the pivots for DART, and a uniform draw for random. Figure 7 shows the Spearman rank correlation between the two over the tokens of a frame, aggregated over steps. On LongNav-R1 none of the three criteria is positively correlated with influence. FastV is centered near zero, DivPrune slightly below it, and DART below it, so the tokens DART ranks highest for retention tend to be the tokens on which the action does not depend. On Uni-NaVid FastV is weakly positive and DivPrune and DART are centered near zero. Figure 8 shows one observation with the four maps. Influence concentrates on the target television. Attention varies little across the frame. DivPrune's

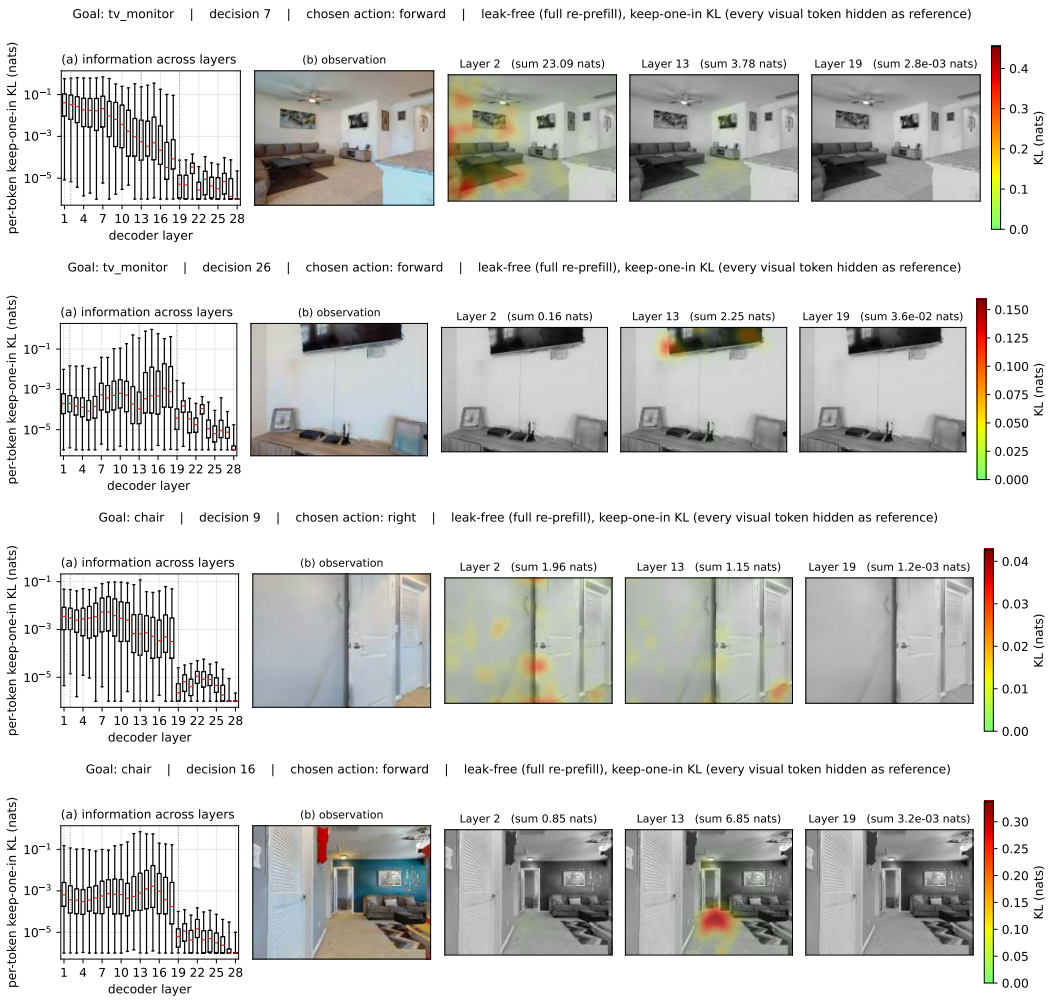


Figure 6: Per token influence of the current frame for LongNav-R1 in the leak free regime, with all historical frames removed and the built in filter disabled. Rows are two decisions from a television episode followed by two from a chair episode, with the goal, step, and chosen action in the row title. Left, the distribution of influence over the 300 tokens at every decoder layer. Right, the observation and influence maps at three layers, with the total over the grid in each title.

distance is high at the window, the plant, and the edges of the mantel, the regions least similar to their neighbors, and DART's dissimilarity is spread across the scene. Both are low on the television, whose patches resemble one another and are scored as redundant. The tokens with the highest influence are thus the ones a diversity or duplication criterion is designed to remove. This extends to closed loop navigation the finding of [19] for image question answering, and adds that a criterion can do worse than random when its scoring implicitly reallocates the budget toward the frames the policy does not use, or, within a frame, toward the tokens it does not read.

E.4 Protocol

On MP3D (Figure 9) the ordering of the three curves is the same as on HM3D-v2. On LongNav-R1 the protocol outperforms PyramidDrop at every budget in both metrics, and the gap widens as the budget shrinks, as expected if where the budget is spent matters more when the budget is smaller. On Uni-NaVid the protocol is at or near full context at every budget and PyramidDrop falls below it at intermediate budgets. MP3D is the harder benchmark for both models, with lower full context SR and SPL than HM3D-v2.

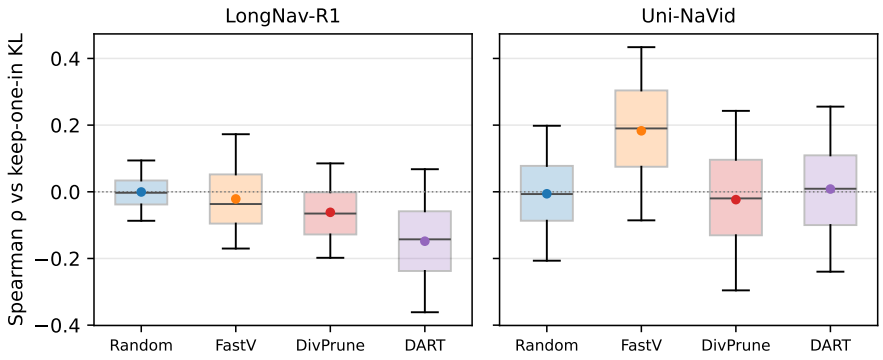


Figure 7: Spearman rank correlation between each utility's per token score and the per token influence of the current frame, computed over the tokens of a frame at every step of 10 episodes. Boxes show the distribution over steps and dots mark the mean. Random scores are a uniform draw and give the reference at zero.

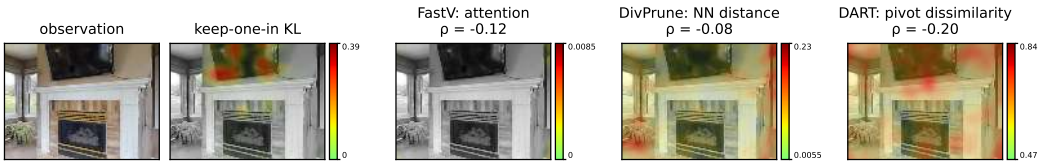


Figure 8: One observation from a television episode of LongNav-R1 with, from left to right, the per token influence of the current frame and the score maps of FastV, DivPrune, and DART, each with its rank correlation to influence. Influence concentrates on the target, whereas the three criteria score other regions of the frame highest.

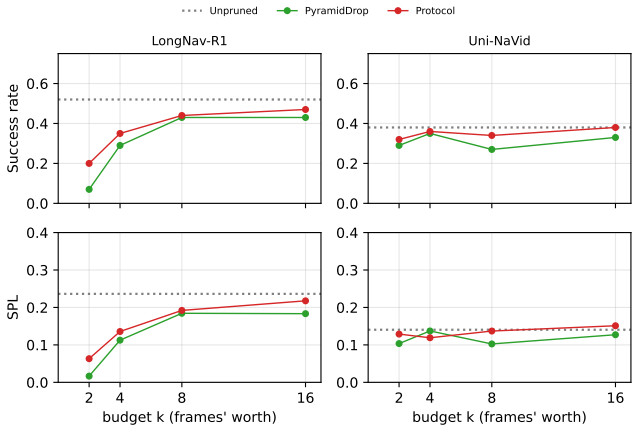


Figure 9: Navigation performance on MP3D against the budget k for the protocol and PyramidDrop. Top: SR, bottom: SPL, left: LongNav-R1, right: Uni-NaVid. The dotted line is the full context model.

Figure 10 reports memory and inference time at $k = 16$ as a function of the episode step. For the full context LongNav-R1 all three quantities grow with the step count, since each new frame is prefilled against every cached entry. Under the protocol they are constant, so the resource footprint of the policy is fixed by K, k, and l_h rather than by episode length. PyramidDrop holds the same number of entries, but its inference time still grows because it rescores the whole history by attention at every step, whereas eviction by age and random sampling cost nothing per step. The space finding thus saves compute as well as memory. For Uni-NaVid, whose built in merging already bounds the history, the gains are correspondingly smaller.

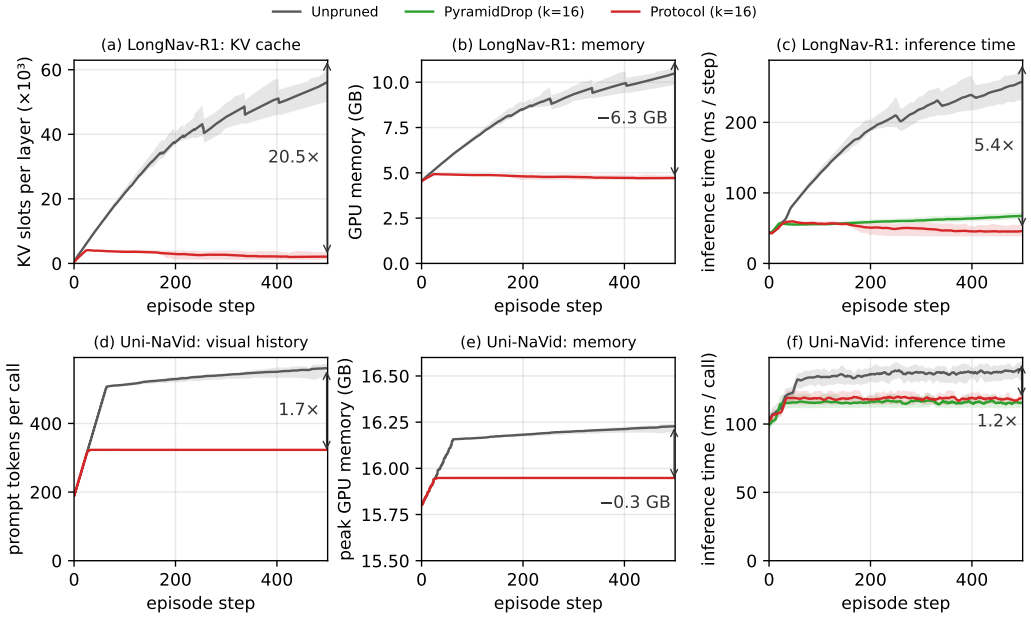


Figure 10: Memory and inference time of the protocol at $k = 16$ as a function of the episode step, averaged over the 100 HM3D-v2 episodes, with shaded bands showing one standard deviation. Top: LongNav-R1, (a) cache entries per layer, (b) GPU memory, and (c) inference time per step. Bottom: Uni-NaVid, (d) visual tokens in the prompt, (e) peak GPU memory, and (f) inference time per call. PyramidDrop holds the same number of history entries as the protocol and is shown for inference time only. Annotations give the ratio or difference at the step cap.