Learning to Focus: Prioritizing Informative Histories with Structured Attention Mechanisms in Partially Observable Reinforcement Learning

Anonymous Submission

Abstract

Transformers have shown strong ability to model long-term dependencies and are increasingly adopted as world models in model-based reinforcement learning (RL) under partial observability. However, unlike natural language corpora, RL trajectories are sparse and reward-driven, making standard self-attention inefficient because it distributes weight uniformly across all past tokens rather than emphasizing the few transitions critical for control. To address this, we introduce structured inductive priors into the self-attention mechanism of the dynamics head: (i) per-head **memory-length priors** that constrain attention to task-specific windows, and (ii) **distributional priors** that learn smooth Gaussian weightings over past state–action pairs. We integrate these mechanisms into UniZero, a model-based RL agent with a Transformer-based world model that supports planning under partial observability. Experiments on the Atari 100k benchmark show that most efficiency gains arise from the Gaussian prior, which smoothly allocates attention to informative transitions, while memory-length priors often truncate useful signals with overly restrictive cut-offs. In particular, Gaussian Attention achieves a 77% relative improvement in mean human-normalized scores over UniZero. These findings suggest that in partially observable RL domains with non-stationary temporal dependencies, discrete memory windows are difficult to learn reliably, whereas smooth distributional priors flexibly adapt across horizons and yield more robust data efficiency. Overall, our results demonstrate that encoding structured temporal priors directly into self-attention improves the prioritization of informative histories for dynamics modeling under partial observability.

1 Introduction

Reinforcement learning (RL) Sutton and Barto [2018] provides a principled framework for sequential decision making, but real-world tasks often violate the Markov assumption and exhibit only partial observability. Such settings are naturally modeled as Partially Observable Markov Decision Processes (POMDPs), which require agents to leverage observation–action histories to reduce uncertainty and achieve robust control Sondik [1971], Kaelbling et al. [1998].

Model-based RL addresses this challenge by learning an explicit world model of environment dynamics Sutton and Barto [2018], which can be used to plan or imagine future trajectories. A seminal example is MuZero Schrittwieser et al. [2020], which learns a joint representation, dynamics, and value model in latent space, paired with Monte Carlo Tree Search Kocsis and Szepesvári [2006] to achieve state-of-the-art performance in board games and Atari. More recently, UniZero Pu et al. [2025] replaced MuZero's recurrent dynamics with a Transformer backbone, using masked self-attention to capture long-range dependencies in latent state–action sequences and improve sample efficiency under partial observability.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Embodied World Models for Decision Making.

Despite this architectural shift, UniZero often remains sample-inefficient in low-data regimes because it inherits assumptions from natural language modeling: namely, that sequential data are abundant, balanced, and richly interdependent. In reality, RL trajectories consist of long stretches of uninformative transitions, sparse rewards, skewed return distributions, and a limited number of interactions Janner et al. [2021], Andrychowicz et al. [2017]. In Transformer-based world models, standard self-attention treats all past tokens within the history as equally relevant, making it hard to identify the sparse transitions that actually drive reward. Unlike language modeling, where vast corpora make even rare dependencies learnable Brown et al. [2020], RL agents operate on scarce and noisy trajectories, requiring attention mechanisms that explicitly prioritize informative segments of history Ni et al. [2023].

To address this limitation, we enhance the UniZero world model by introducing two structured temporal priors into the self-attention layers of its dynamics head. The dynamics head predicts the next latent state z_{t+1} and immediate reward r_t based on attention-weighted histories. The first prior, a **memory-length prior**, restricts each attention head to a learnable contiguous window, approximating the minimal history required for accurate prediction. The second, a **distributional prior**, applies smooth Gaussian weighting over past tokens, emphasizing those most informative for immediate outcomes. We instantiate these as **Adaptive Attention** (memory-length prior), **Gaussian Attention** (distributional prior), and their combination, **Gaussian Adaptive Attention**.

On the Atari 100k benchmark, Gaussian Attention yields a 77% relative improvement in human-normalized mean score over UniZero's standard self-attention. This gain stems from its ability to allocate weight smoothly across past transitions, capturing relevant temporal dependencies without imposing sharp cutoffs. In contrast, Adaptive Attention often misestimates the true dependency horizon, either truncating important signals or including irrelevant ones, reducing sample efficiency. Combining the two mechanisms degrades performance: the hard span mask truncates Gaussian tails, negating its smooth weighting benefits. These results highlight a general guideline for model-based RL under partial observability: smooth distributional priors offer more robust and data-efficient dynamics modeling than rigid memory-length priors.

Our contributions are as follows:

- We propose two structured temporal priors for self-attention in world models: a memory-length prior enforcing per-head learnable look-back windows, and a distributional prior introducing smooth Gaussian weightings over histories.
- We integrate these mechanisms into the UniZero agent and demonstrate on Atari 100k that Gaussian Attention achieves substantial gains in human-normalized mean and median scores, with negligible computational overhead.
- We analyze the complementary behavior of hard and smooth priors, showing how Gaussian priors reliably capture diverse temporal dependencies while memory-length priors offer benefits in limited cases.
- Through systematic ablations across Atari games, we isolate the effects of each prior and its regularization, confirming robustness to initialization and low additional computational cost.

2 Background

MDPs and POMDPs. A Markov Decision Process (MDP) is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s' \mid s, a)$ denotes the transition probability, $R(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor Sutton and Barto [2018]. The agent seeks a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected discounted return $E[\sum_{t=1}^{\infty} \gamma^t R(s_t, a_t)]$, satisfying the Bellman optimality equation. A Partially Observable MDP (POMDP) extends this formulation with an observation space $\mathcal{O}$ and observation probabilities $O(o \mid s, a)$, since the true state is not directly observable, and thus is defined by $(\mathcal{S}, \mathcal{A}, \mathcal{O}, P, R, O, \gamma)$. To act under partial observability, the agent maintains a belief distribution b over states, updated after action a and observation o as $b_{a,o}(s') \propto O(o \mid s', a) \sum_s P(s' \mid s, a) b(s)$. Not all observations are equally informative, and a central objective in planning under partial observability is to identify a minimal subset of history sufficient for predicting future transitions and rewards. Influence-Based Abstraction (IBA) formalizes this by identifying d-separating observation sets that render the future conditionally independent of the remaining history Oliehoek et al. [2012], echoing state-abstraction principles in RL Givan et al. [2003].

Deep Reinforcement Learning. Deep Reinforcement Learning (DRL) integrates classical RL with deep neural networks to handle high-dimensional state and action spaces. Foundational value-based methods include DQN and Double DQN Mnih et al. [2015], Van Hasselt et al. [2016], while policy-gradient and actor–critic methods such as REINFORCE, A3C, TRPO, and PPO have become standard benchmarks Williams [1992], Mnih et al. [2016], Schulman et al. [2015, 2017]. More recently, entropy-regularized off-policy algorithms such as SAC have improved stability and exploration in continuous and high-dimensional domains Haarnoja et al. [2018]. These advances underpin the extension of DRL to increasingly complex, partially observable, and real-world tasks.

Transformers. Transformers Vaswani et al. [2017] have emerged as powerful alternatives to recurrent neural networks (RNNs) for long-sequence modeling. Given an input sequence of length N, each token is projected into queries $Q \in \mathbb{R}^{N\times d_k}$, keys $K \in \mathbb{R}^{N\times d_k}$, and values $V \in \mathbb{R}^{N\times d_v}$. Self-attention then aggregates contextual information via:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{d_k}}\Big)V. \tag{1}$$

Since self-attention is permutation-invariant, positional encodings, either fixed or learned, are added to token embeddings to inject order information Devlin et al. [2019]. By combining global context aggregation with positional encodings, Transformers effectively capture long-range dependencies that truncated RNNs fail to model Dai et al. [2019]. This has made Transformer architectures compelling candidates for world models in RL, where long-horizon planning and memory are critical Ni et al. [2023], Robine et al. [2023].

MuZero. MuZero Schrittwieser et al. [2020] achieves superhuman performance in board games and Atari by integrating Monte Carlo Tree Search (MCTS) with a learned latent dynamics model. At each time step t, MuZero employs:

1. **Encoder:** $z_t^0 = h_\theta(o_{1:t})$, mapping the observation history to a latent state.
2. **Dynamics head:** $(z_{t+1}, r_t) = g_\theta(z_t, a_t)$, unrolling latent states and predicting rewards recurrently.
3. **Prediction head:** $(\pi_t, v_t) = f_\theta(z_t)$, producing policy logits and value estimates.

Although powerful, MuZero's recurrent dynamics suffer from vanishing gradients and a fixed unroll horizon, which limit its ability to capture long-range dependencies Bengio et al. [1994].

UniZero. UniZero Pu et al. [2025] retains MuZero's overall world model tuple $W = (h_\theta, g_\theta, f_\theta)$ but parameterizes g_θ and f_θ with a Transformer backbone. Unlike MuZero, whose encoder produces only a single latent state summarizing the entire history, UniZero encodes each observation individually as $z_i = h_\theta(o_i)$, yielding a sequence $z_{1:t}$. The sequence of observation–action pairs $\big[(z_1, a_1), \ldots, (z_t, a_t)\big]$ is then processed by L stacked Transformer layers, each with h attention heads. Masked self-attention ensures that token i attends only to past tokens, preventing future leakage.

The outputs from all heads are concatenated and projected through a final linear layer, integrating the diverse subspaces captured by each head. This allows UniZero to capture dependencies far beyond MuZero's fixed horizon, though at quadratic complexity in the sequence length, the number of layers L, and the number of heads h. Moreover, because self-attention initially treats all past tokens as equally relevant, the model must learn relevance weights during training, often leading to sample inefficiency.

The final Transformer layer outputs the next latent state z_{t+1} and immediate reward r_t, which are passed to the unchanged prediction head f_θ to produce π_t and v_t. UniZero, like MuZero, is trained via joint model–policy optimization, maintaining a soft-target world model $\hat{W} = (\hat{h}_\theta, \hat{g}_\theta, \hat{f}_\theta)$ to stabilize learning Eysenbach et al. [2022]. By leveraging global temporal context, UniZero improves long-horizon performance, but its uniform attention weighting motivates the structured temporal priors we introduce in this work.

3 Related Work

RL in POMDPs. Under partial observability, model-free methods typically rely on recurrent networks to infer hidden states Hausknecht and Stone [2015], whereas model-based approaches learn latent world models for planning. Early frameworks such as predictive state representations Littman and Sutton [2001] have evolved into deep generative models such as Dreamer, which combine variational inference and recurrent state-space models to compactly represent belief states and enable efficient long-horizon planning Ha and Schmidhuber [2018], Hafner et al. [2020].

Memory Mechanisms in DRL. Many deep RL methods explicitly incorporate memory to handle partial observability. Simple approaches stack the last k frames Mnih et al. [2015], recurrent architectures summarize the entire action–observation history into fixed-size states Hausknecht and Stone [2015], and external differentiable memories further expand capacity but often introduce training instability Graves et al. [2016]. Influence-Aware Memory (IAM), inspired by Influence-Based Abstraction (IBA) Oliehoek et al. [2012], learns gating mechanisms that selectively retain past observations predictive of future outcomes Suau et al. [2022].

Transformer-based World Models. Recent Transformer adaptations in RL leverage self-attention to capture long-range dependencies, but most do not incorporate inductive priors tailored to RL sequences. On the model-free side, methods such as GTrXL and Transformer-XL stabilize attention via gating and relative encodings Parisotto et al. [2020], Dai et al. [2019], Decision Transformer reframes control as return-conditioned masked attention over past trajectories Chen et al. [2021], and Adaptive Span Transformer reduces computation by learning per-head context lengths without building an explicit dynamics model Kumar et al. [2020]. On the model-based side, hybrids such as IRIS Micheli et al. [2023] and TransDreamer Chen et al. [2022] integrate Transformers into latent world models, rolling out imagined trajectories for planning to achieve strong sample efficiency. However, most existing Transformer-based world models in RL rely on fixed or NLP-inspired positional encodings (e.g., sinusoidal or relative embeddings), which emphasize computational efficiency rather than task relevance. In contrast, we introduce structured temporal priors to better align attention with reward-relevant dependencies.

4 Dynamics Modelling with Self-Attention Priors

In UniZero's world model, the dynamics function aggregates past latents and actions up to time t into a history h_t, and predicts the next latent and reward:

$$(\hat{z}_{t+1}, \hat{r}_t) = g_\theta(z_{\le t}, a_{\le t}) = g_\theta(h_t), \tag{2}$$

where relevance is computed via self-attention with weights $\{\alpha_{ij}\}_{j=1}^{i}$ (with i the current query and j the key). Under partial observability, however, only a limited window of context and a sparse set of key events truly drive accurate predictions. To better align attention with these reward-relevant dependencies, we introduce two structured temporal priors into the attention mechanism: (i) a **memory-length prior** that enforces a learnable finite look-back span, and (ii) a **distributional prior** that softly emphasizes tokens according to a Gaussian saliency distribution. Our goal is to bias self-attention toward histories that matter most for predicting dynamics and rewards, thereby improving sample efficiency in low-data, partially observable RL settings.

4.1 Memory-Length Prior

Many partially observable environments admit a finite effective memory: only the most recent n steps are needed to predict the next reward Littman and Sutton [2001], Mnih et al. [2016]. Imposing this prior focuses the model on a minimal history window, reducing redundant computation over distant tokens. Formally,

$$\mathbb{E}[r_t \mid h_{1:t}, a_t] \;=\; \mathbb{E}[r_t \mid h_{t-n+1:t}, a_t]. \tag{3}$$

We implement this using **Adaptive Attention** Sukhbaatar et al. [2019]. Each head h learns a scalar parameter s_h, transformed via softplus into a positive span $L_h = \mathrm{softplus}(s_h)$. A hard mask over relative positions is then constructed:

$$M_{ij}^{(h)} = \begin{cases} 0, & i - j \le L_h, \\ -\infty, & i - j > L_h, \end{cases} \tag{4}$$

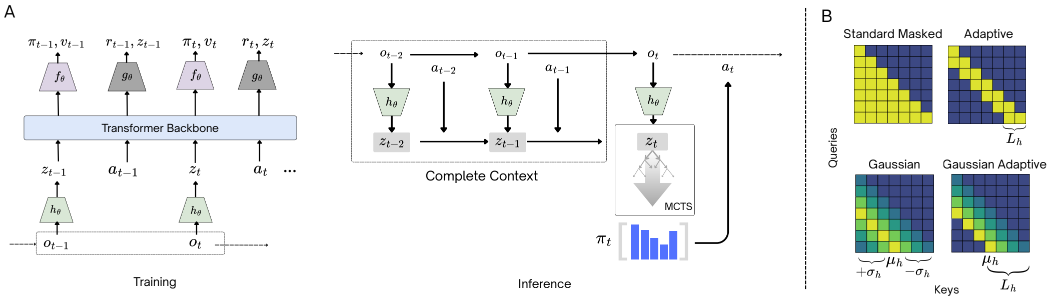

Figure 1: Model framework and attention priors. (**A**) UniZero world model with Transformer dynamics: observation–action sequences are encoded into latents, processed by self-attention, and used for dynamics and decision heads. (**B**) Attention priors: standard masking (left), memory-length prior (adaptive span), distributional prior (Gaussian bias), and their combination (Gaussian Adaptive). Yellow indicates high attention bias; dark blue indicates zero. Together, these priors bias self-attention toward reward-relevant temporal context.

so that queries at i can only attend within their learned look-back span. Attention weights become

$$\text{Attention}^{(h)} = \text{softmax}\big(\frac{Q^{(h)}K^{(h)\top}}{\sqrt{d_k}} + M^{(h)}\big). \tag{5}$$

To prevent trivial solutions where all spans grow without bound, we apply an ℓ_1 penalty, encouraging the model to learn minimal but sufficient spans Tibshirani [1996], Givan et al. [2003]. Each attention head h produces a context vector

$$c_t^{(h)} = \sum_{j=1}^{t} \alpha_{tj}^{(h)} [z_j; a_j], \tag{6}$$

where $[z_j; a_j]$ denotes the concatenated latent state and action at step j. By constraining spans L_h, different heads specialize at distinct temporal scales, yielding a multi-scale representation when their context vectors are combined into h_t.

4.2 Distributional Prior

In partially observable settings, only a sparse subset of tokens carries predictive signal for $(\hat{z}_{t+1}, \hat{r}_t)$. We capture this *distributional prior* by learning a Gaussian positional kernel.

Each head h learns parameters $\mu_h, \sigma_h > 0$, defining

$$G_{ij}^{(h)} = -\frac{(i - j - \mu_h)^2}{2\sigma_h^2}. \tag{7}$$

This is added to the scaled dot-product logits:

$$\text{Attention}^{(h)} = \text{softmax}\Big(\frac{Q^{(h)}K^{(h)\top}}{\sqrt{d_k}} + G^{(h)}\Big), \tag{8}$$

so that queries at i privilege tokens at offset μ_h with sharpness σ_h Ioannides et al. [2024]. Unlike spans, μ_h and σ_h are unconstrained: σ_h may expand to broad attention or shrink to narrow focus, thereby giving each head a smooth, learned saliency profile through $G^{(h)}$. Different heads capture different offsets and spreads, producing complementary temporal filters that are concatenated into h_t.

4.3 Combining Priors

Finally, we combine the two priors by defining

$$B_{ij}^{(h)} = G_{ij}^{(h)} + M_{ij}^{(h)}, \tag{9}$$

and apply it within the attention mechanism by adding $B^{(h)}$ as a bias term to the scaled dot-product before the softmax. This **Gaussian Adaptive Attention** enforces a finite horizon while retaining smooth saliency within it, thereby combining the strengths of memory-length and distributional priors.

5 Experiments

We evaluate our Transformer-based world model augmented with attention priors on the Atari 100k benchmark, a widely used testbed for sample efficiency in model-based reinforcement learning. This suite spans diverse reward densities, horizon lengths, and stochastic dynamics. Our evaluation considers both aggregate performance and the contribution of each prior through controlled ablations.

5.1 Experimental Setup

Agents are trained on 26 Atari environments for 100k interaction steps, with results averaged over five independent runs using random seeds (1–5). Performance is reported as human-normalized scores, following the protocol of Łukasz Kaiser et al. [2020]. UniZero supports both *Single-Task* (ST) training, where a separate model is learned per environment, and *Multi-Task* (MT) training, where a shared latent space spans multiple environments. To isolate the effects of the proposed attention priors, all experiments are conducted in the ST setting. Unless otherwise noted, we adopt the default hyperparameters of UniZero from Pu et al. [2025], ensuring strict comparability with prior work. Full architectural details, training configurations, and hyperparameters are provided in Appendix A, together with instructions for reproducing all reported results.

Table 1: Raw Atari 100k scores comparing our attention-biased UniZero variants against reproduced UniZero and MuZero baselines. MuZero results are from LightZero reproductions in Pu et al. [2025] (three seeds), while Random and Human scores are from Pu et al. [2025]. All "Ours" results are averaged over five seeds. **Bold** entries denote the superior method between the *UniZero ST* baseline and our attention-biased methods, while underlined values indicate the overall best-performing method.

Game	Random	Human	MuZero	UniZero ST (Baseline)	Adaptive UniZero (Ours)	Gaussian UniZero (Ours)	Gaussian Adaptive UniZero (Ours)
Alien	227.8	7127.7	300.0	468.5	$\underline{\mathbf{570.6}}$	483.3	509.6
Amidar	5.8	1719.5	$\underline{90.0}$	57.2	57.9	**71.2**	53.4
Assault	222.4	742.0	$\underline{609.0}$	341.9	423.5	**486.8**	333.7
Asterix	210.0	8503.7	$\underline{1400.0}$	495.3	500.1	**619.9**	333.6
BankHeist	14.2	753.1	$\underline{223.0}$	91.3	13.3	**165.1**	0.7
BattleZone	2360.0	37187.5	$\underline{7587.0}$	**6000.0**	5872.5	5361.6	5297.6
Boxing	0.1	12.1	$\underline{20.0}$	0.1	−9.5	**2.4**	−11.3
Breakout	1.7	30.5	3.0	3.7	0.8	$\underline{\mathbf{5.1}}$	0.5
ChopperCommand	811.0	7387.8	1050.0	1169.0	872.5	$\underline{\mathbf{1263.4}}$	735.2
CrazyClimber	10780.5	35829.4	$\underline{22060.0}$	7418.9	4326.6	**7966.6**	2020.0
DemonAttack	152.1	1971.0	$\underline{4601}$	236.3	187.4	**267.0**	166.4
Freeway	0.0	29.6	$\underline{12.0}$	0.0	0.7	0.1	**2.6**
Frostbite	65.2	4334.7	260.0	239.8	$\underline{\mathbf{261.2}}$	236.7	162.2
Gopher	257.6	2412.5	346.0	606.7	646.4	$\underline{\mathbf{798.8}}$	240.0
Hero	1027.0	30826.4	$\underline{3315.0}$	1483.0	1422.2	699.6	**2414.4**
Jamesbond	29.0	302.8	90.0	201.7	156.7	$\underline{\mathbf{362.0}}$	75.9
Kangaroo	52.0	3035.0	200.0	842.6	488.6	$\underline{\mathbf{1636.4}}$	367.9
Krull	1598.0	2665.5	$\underline{5191.0}$	2539.8	2647.5	**3108.8**	1964.0
KungFuMaster	258.5	22736.3	6100.0	2019.0	8546.5	$\underline{\mathbf{9424.5}}$	644.3
MsPacman	307.3	6951.6	1010.0	643.9	$\underline{\mathbf{1103.3}}$	726.6	394.7
Pong	−20.7	14.6	−15.0	−14.5	−19.6	$\underline{\mathbf{-7.1}}$	−20.3
PrivateEye	24.9	69571.3	$\underline{100.0}$	**93.3**	−60.1	57.6	80.0
Qbert	163.9	13455.0	1700.0	677.2	941.5	$\underline{\mathbf{1741.8}}$	356.3
RoadRunner	11.5	7845.0	$\underline{4400.0}$	1941.3	**2164.5**	1948.4	1400.0
Seaquest	68.4	42054.7	466.0	384.1	293.2	$\underline{\mathbf{485.7}}$	273.3
UpNDown	533.4	11693.2	1213.0	2018.0	1374.7	$\underline{\mathbf{2373.8}}$	1246.4
Normalized Mean	0.000	1.000	$\underline{0.44}$	0.13	0.095	**0.23**	0.00
Normalized Median	0.000	1.000	$\underline{0.13}$	0.05	0.05	**0.10**	0.02

Attention Prior Initialization. We initialize all attention priors to align with typical temporal dependencies in Atari trajectories. For Adaptive Attention, each head begins with a span of $L_h^0 = 0.3\,L_{\max} \approx 6$, following the recommendations of Kumar et al. [2020]. Gaussian Attention is initialized with mean offset $\mu_h = 6$ and standard deviation $\sigma_h = 1$, while Gaussian Adaptive Attention learns both μ_h and σ_h but applies a hard cutoff at $L_h^0 = 10$. To ensure comparable starting conditions, initial distributional logits are sampled from $\mathcal{N}(\mu_h, \sigma_h^2)$, exactly matching the Gaussian prior.

Baselines. We compare against two established model-based RL baselines implemented in the LightZero framework Niu et al. [2023]: (i) *MuZero* Schrittwieser et al. [2020], which combines latent dynamics with Monte Carlo Tree Search, and (ii) *UniZero* Pu et al. [2025], which replaces MuZero's recurrent core with a Transformer backbone. Both baselines are trained for 100k steps per environment under identical hyperparameters, ensuring that performance differences arise solely from the proposed priors.

5.2 Performance Results

Table 1 reports Atari 100k results against UniZero (ST) and MuZero. Gaussian UniZero delivers the best overall performance, improving HNS from 0.13 to 0.23 (+77%) and HMS from 0.05 to 0.10 (+100%), outperforming the baseline in 19 of 26 games. Adaptive and Gaussian Adaptive variants yield inconsistent or weaker results, with Adaptive only matching the baseline on HMS. Overall, smooth Gaussian priors provide consistent sample-efficiency gains, while rigid span cutoffs hurt performance. See full learning curves in Appendix B. .

Gaussian Attention consistently outperforms alternatives because it distributes weights smoothly across short- and mid-range temporal offsets, effectively capturing both immediate and moderately delayed dependencies Ni et al. [2023]. By contrast, Adaptive Attention's hard spans often misestimate the relevant horizon, either truncating delayed yet informative signals or incorporating irrelevant context. Combining Gaussian weighting with a hard cutoff further degrades performance: truncating the Gaussian kernel removes useful tails and produces conflicting priors. Together, these findings suggest a general guideline for model-based RL under partial observability: smooth, learnable positional priors offer a more robust and flexible mechanism for temporal modeling than rigid memory windows. Future directions include extending Gaussian priors to multi-task settings, where shared temporal structure across games could further improve generalization.

5.3 Ablation Studies

To isolate the contributions of each prior, we conduct ablations on four representative Atari games: Pong, MsPacman, Jamesbond, and Freeway, which span diverse observation complexities, reward structures, and temporal dependencies.

Regularization Ablation. We compare three penalties on the learned span vector L_h, each with penalty coefficient $\lambda = 0.025$ as in Kumar et al. [2020]:

① Max-norm ℓ_{max}: enforces $\|L_h\|_\infty \leq c$, restricting each head to the most recent tokens Srivastava et al. [2014].

② ℓ_1: adds $\lambda \sum_j L_{h,j}$, encouraging sparsity by driving many spans to zero while letting a few grow.

③ ℓ_2: adds $\lambda \sum_j L_{h,j}^2$, softly shrinking spans while preserving long-range context.

In practice, max-norm favors purely short-term attention; ℓ_1 produces a bimodal mix of very short and very long spans; and ℓ_2 encourages balanced recency while retaining moderate long-range dependencies. Figure 2 illustrates these effects: max-norm performs best in short-horizon tasks, ℓ_2 dominates in mid-horizon settings, and ℓ_1 occasionally excels in long-horizon environments by retaining sparse but wide spans. Overall, ℓ_2 generalizes most robustly, striking a balance between stability and flexibility.

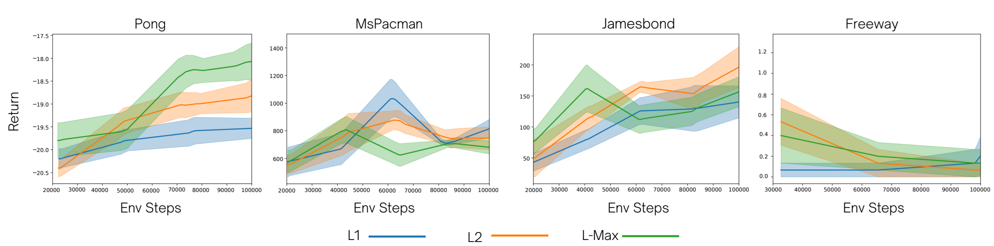

Figure 2: Regularization ablation. Comparison of ℓ_1 (blue), ℓ_2 (orange), and max-norm (green) penalties across four Atari games over five seeds. Shaded regions denote standard error. Each scheme exhibits task-specific strengths, but ℓ_2 achieves the most consistent performance overall.

Initial Parameter Sensitivity. We further analyze sensitivity to initialization. As shown in Table 2, performance is largely robust to different initial spans (L_h) and Gaussian centers (μ_h), both of which adapt quickly during training. By contrast, the Gaussian width σ_h has a stronger effect: narrower widths ($\sigma_h = 1$) consistently yield superior results across environments, while wider priors ($\sigma_h = 3$) underperform. This suggests that initializing with a tight positional prior is more critical than precise initialization of span or offset, providing a better inductive starting point for efficient learning.

Table 2: Ablation on initialization of attention priors. Mean $\pm$ standard error over five seeds on four Atari games. Varying initial spans L_h or offsets μ_h has little effect, while narrow Gaussian widths ($\sigma_h = 1$) consistently improve performance. **Bold** entries mark the best result per game.

	Pong	MsPacman	Jamesbond	Freeway
$L_h = 2$	-18.5 ± 0.4	716.7 ± 58.9	180.0 ± 49.8	2.2 ± 2.1
$L_h = 6$	-19.6 ± 0.4	$\mathbf{1103.3 \pm 345.8}$	156.7 ± 29.8	0.7 ± 0.6
$L_h = 10$	-18.7 ± 0.5	633.3 ± 47.4	130.0 ± 34.7	$\mathbf{2.7 \pm 2.1}$
$\mu_h = 2$	$\mathbf{-6.9 \pm 1.8}$	805.3 ± 112.6	293.3 ± 49.6	0.0 ± 0.0
$\mu_h = 6$	-7.9 ± 1.0	726.7 ± 98.2	$\mathbf{362.1 \pm 53.1}$	0.1 ± 0.1
$\mu_h = 10$	-10.5 ± 1.0	894.7 ± 101.8	290.0 ± 58.4	0.1 ± 0.1
$\sigma_h = 1$	-7.9 ± 1.0	726.7 ± 98.2	$\mathbf{362.1 \pm 53.1}$	0.1 ± 0.1
$\sigma_h = 3$	-15.1 ± 0.7	638.7 ± 47.6	196.7 ± 24.4	0.0 ± 0.0

Compute and Memory Overhead. All proposed priors incur negligible overhead, with at most a 0.002% increase in MFLOPs per forward pass. Table 3 shows that parameter counts and FLOPs remain effectively unchanged relative to UniZero, demonstrating that the efficiency gains of adaptive and Gaussian attention come at no meaningful computational cost. Full details on resources and training times are provided in Appendix A.

Table 3: Overhead analysis. Parameter counts (in millions), MFLOPs per Transformer forward pass, and relative increase over the vanilla UniZero baseline.

Model	Total Parameters (M)	Transformer Parameters (M)	MFLOPs	Δ MFLOPs (%)
Baseline	20.77	14.18	454.611	—
Adaptive	20.77	14.18	454.615	$\mathbf{+0.001}$
Gaussian	20.77	14.18	454.619	$\mathbf{+0.002}$
Gaussian Adaptive	20.77	14.18	454.619	$\mathbf{+0.002}$

Limitations. Our evaluation is restricted to Atari, leaving open whether the proposed attention priors generalize to continuous-control or multi-task settings. In addition, the learned look-back spans require regularization to avoid collapse to trivial extremes, which may limit adaptability in environments with highly variable temporal dependencies. Future work should investigate more flexible temporal priors and evaluate their robustness across broader RL domains, including continuous-control benchmarks such as Tassa et al. [2018].

6 Conclusion

In NLP, Transformers benefit from massive, balanced corpora where long-range dependencies recur frequently, allowing self-attention to capture them implicitly. In contrast, model-based RL agents must identify the few reward-relevant dependencies hidden within sparse and correlated trajectories under limited supervision. This mismatch makes standard self-attention sample-inefficient, as it spreads its focus across many uninformative transitions rather than concentrating on the critical ones. We addressed this by incorporating two inductive priors into UniZero's dynamics head: a **memory-length prior**, restricting each head to a finite span, and a **distributional prior**, implemented as a smooth Gaussian positional prior.

Experiments on Atari-100k demonstrate that Gaussian positional priors substantially improve sample efficiency, delivering a 100% relative gain in human-normalized median score, while hard span cutoffs degrade performance by truncating delayed yet informative signals. These results suggest a broader principle: smooth, learnable temporal priors align better with the irregular dependency structure of RL trajectories than rigid memory windows. Looking ahead, structured temporal priors in self-attention promise to improve robustness and data efficiency in Transformer world models, with potential benefits extending beyond Atari to continuous control, multi-task learning, and other domains with complex temporal dependencies.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in Neural Information Processing Systems*, 30, 2017.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*, 34:15084–15097, 2021.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988. Association for Computational Linguistics, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Russ R Salakhutdinov. Mismatched no more: Joint model-policy optimization for model-based rl. *Advances in Neural Information Processing Systems*, 35:23230–23243, 2022.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial intelligence*, 147(1-2):163–223, 2003.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476, 2016.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

Matthew J Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI fall symposia*, volume 45, page 141, 2015.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv: Learning*, 2016.

Georgios Ioannides, Aman Chadha, and Aaron Elkins. Gaussian adaptive attention is all you need: Robust contextual representations across multiple modalities. *CoRR*, 2024.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, volume 34, pages 1273–1286, 2021.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer Berlin Heidelberg, 2006.

Shakti Kumar, Jerrod Parker, and Panteha Naderian. Adaptive transformers in rl. *arXiv preprint arXiv:2004.03761*, 2020.

Michael Littman and Richard S Sutton. Predictive representations of state. *Advances in Neural Information Processing Systems*, 14, 2001.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937. PMLR, 2016.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment. *Advances in Neural Information Processing Systems*, 36:50429–50452, 2023.

Yazhe Niu, Yuan Pu, Zhenjie Yang, Xueyan Li, Tong Zhou, Jiyuan Ren, Shuai Hu, Hongsheng Li, and Yu Liu. Lightzero: A unified benchmark for monte carlo tree search in general sequential decision scenarios. *Advances in Neural Information Processing Systems*, 36:37594–37635, 2023.

Frans Oliehoek, Stefan Witwicki, and Leslie Kaelbling. Influence-based abstraction for multiagent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1422–1428, 2012.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, pages 7487–7498. PMLR, 2020.

Yuan Pu, Yazhe Niu, Zhenjie Yang, Jiyuan Ren, Hongsheng Li, and Yu Liu. Unizero: Generalized and efficient planning with scalable latent world models. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. In *The Eleventh International Conference on Learning Representations*, 2023.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Edward Jay Sondik. *The optimal control of partially observable markov processes*. PhD thesis, Stanford University, Stanford, CA, 1971.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Miguel Suau, Jinke He, Elena Congeduti, Rolf A. N. Starre, Aleksander Czechowski, and Frans A. Oliehoek. Influence-aware memory architectures for deep reinforcement learning in pomdps. *Neural Computing and Applications*, 37(19):13145–13161, 2022. doi: 10.1007/s00521-022-07691-7.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 331–335. Association for Computational Linguistics, 2019.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, second edition, 2018.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020.

A Implementation Details

Encoder Archictecture. We adopt the UniZero encoder architecture Pu et al. [2025], which builds on the convolutional backbone of LightZero Niu et al. [2023] and adds a final linear projection to produce a 768-dimensional (D) latent state. To improve training stability under partial observability, we incorporate simplicial normalization (SimNorm) Hansen et al. [2023], which normalizes each latent segment via a learnable temperature-controlled mappings.

Transformer Backbone and Prediction Heads. Our Transformer backbone follows the nanoGPT architecture described in Pu et al. [2025], stacking multiple self-attention and feed-forward layers to process sequences of timestep inputs. All of our proposed inductive biases are implemented directly within the self-attention module of each Transformer layer. At each step, the latent state (after SimNorm) and the corresponding action are embedded into a common D-dimensional space via learnable `nn.Embedding` (or a linear layer for continuous actions) and summed with learnable positional embeddings. The Transformer outputs context-enriched representations that are sent to two separate two-layer MLPs with GELU Hendrycks and Gimpel [2016] activations: the dynamics head predicts the next latent state (of dimension D, followed by SimNorm) and the reward distribution (discrete support size), while the decision head predicts policy logits (action-space size) and value distribution (support size).

Hyperparameters and Environments. Table 4 summarizes all architectural and training parameters used in our experiments. Most values such as latent dimension, Transformer depth, MCTS settings, and optimizer configuration are inherited from UniZero Pu et al. [2025], with additional entries for our attention-bias hyperparameters. All Atari environments are provided through the ALE interface via Gymnasium v0.28, using the standard NoFrameskip variants with sticky actions enabled, matching the settings in the UniZero framework. We select the environments from the Atari 100k benchmark Łukasz Kaiser et al. [2020].

Training Details. All reported results are averaged over 5 random seeds, with error bars as described in Appendix B. Atari environments are provided through the ALE interface (Gymnasium 0.28, sticky actions enabled), ensuring consistency with prior work. All experiments were conducted with a configuration of a single NVIDIA Tesla A100 / V100 GPU, $15-20$ CPU cores, and $60-80$ GB of total RAM. Training an Atari agent for $100{,}000$ environment steps requires approximately $4-5$ hours, with agent evaluations every $10{,}000$ steps (starting after the $20{,}000$th step). We observed stable results across A100 and V100 GPUs. Training configurations can be found in the `zoo/atari/config` directory, where each attention model has a different configuration file within UniZero. See README file in the codebase for details on how to train an agent.

Table 4: Key Hyperparameters. The values are aligned with those in Pu et al. [2025] for Atari environments. The section on **Attention** refers to the newly added parameters.

Hyperparameter	Value
Planning	
Number of MCTS Simulations (sim)	50
Inference Context Length (H_{infer})	4
Temperature	0.25
Dirichlet Noise (α)	0.3
Dirichlet Noise Weight	0.25
Coefficient c_1	1.25
Coefficient c_2	19652
Environment and Replay Buffer	
Replay Buffer Capacity	1,000,000
Sampling Strategy	Uniform
Observation Shape (Atari)	(3, 64, 64) (stack1)
Reward Clipping	True
Number of Frames Stacked	1 (stack1)

(continued)

Hyperparameter	**Value**
Frame Skip	4
Game Segment Length	400
Data Augmentation	False
Architecture	
Latent State Dimension (D)	768
Number of Transformer Heads	8
Number of Transformer Layers (N)	2
Dropout Rate (p)	0.1
Activation Function	LeakyReLU (encoder); GELU (others)
Reward/Value Bins	101
SimNorm Dimension (V)	8
SimNorm Temperature (τ)	1
Optimization	
Training Context Length (H)	10
Replay Ratio	0.25
Buffer Reanalyze Frequency	1/50
Batch Size	64
Optimizer	AdamW Loshchilov and Hutter [2019]
Learning Rate	1×10^{-4}
Next Latent State Loss Coefficient	10
Reward Loss Coefficient	1
Policy Loss Coefficient	1
Value Loss Coefficient	0.5
Policy Entropy Coefficient	1×10^{-4}
Weight Decay	10^{-4}
Max Gradient Norm	5
Discount Factor	0.997
Soft Target Update Momentum	0.05
Hard Target Network Update Frequency	100
Temporal Difference (TD) Steps	5
Evaluation Frequency	10k Collector Steps
Attention	
Attention Type	*causal*, *gaussian*, *adaptive* or *gaam*
Rotary Positional Embeddings	False
Initial Gaussian Mean Offset μ_0^h (`init_adaptive_mu`)	6.0 (Varied across ablations)
Initial Gaussian Standard Deviation σ_0^h (`init_adaptive_sigma`)	1.0 (Varied across ablations)
Max Adaptive Span (`max_adaptive_span`)	20.0
Initial Adaptive Span L_h^0 (`init_adaptive_span`)	6.0 (Adaptive), 10.0 (Gaussian Adaptive)
Adaptive Span Regularization Parameter (`adapt_span_loss`)	0.025
Adaptive Span Ramp R (`adapt_span_ramp`)	3.0

B Learning Curves and Learned Biases

Figure 3: Learning Curves for Attention-Biased UniZero. Each panel plots the mean evaluation return (solid line) and standard error (shaded band) over five random seeds for three variants: Gaussian attention (blue), Adaptive attention (orange), and Gaussian Adaptive attention (green). The grey dotted horizontal line in each subplot marks the UniZero baseline's final return at the 100,000th environment step.

In Pong, the learned parameters reveal clear differences between the inductive priors (Figure 4).

Adaptive attention. The learned memory spans L_h (initialized at 6) drift inconsistently across heads and layers. Some collapse to very short horizons, while others expand far beyond the relevant dependency range. This instability indicates that Adaptive attention struggles to capture Pong's narrow but stable temporal dependencies.

Gaussian attention. By contrast, Gaussian attention learns mean offsets μ_h that remain close to the initialization ($\mu \approx 6$), while widths σ_h expand moderately beyond 1.0. This produces smooth, head-specific kernels that emphasize a few recent steps but still leverage informative tails. These stable parameters align well with Pong's true dependency horizon and explain the stronger performance of this variant.

Gaussian Adaptive attention. This mechanism combines both priors, but the hard cutoff imposed by L_h (initialized at 10) often truncates the Gaussian kernel. Although the learned μ_h and σ_h resemble those of Gaussian attention, the span clips the tails, removing the soft weighting needed to capture delayed signals. As a result, Gaussian Adaptive inherits the instability of Adaptive rather than the robustness of Gaussian.

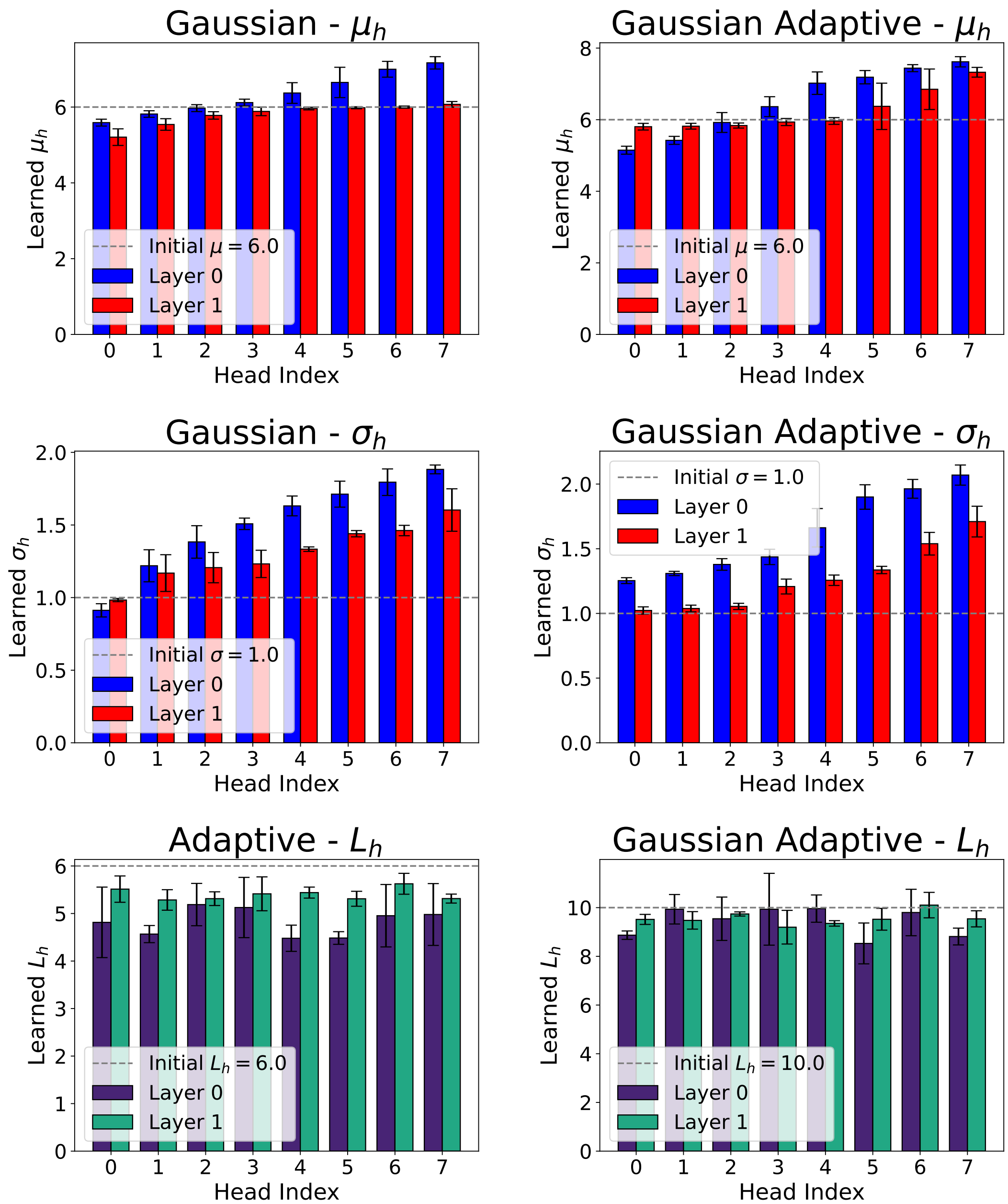

Figure 4: Learned adaptive and Gaussian-based attention parameters in Pong. The six subplots report the learned values across attention heads and layers, compared against their initialization (dashed lines). Top row: learned Gaussian mean offsets (μ_h) for Gaussian (left) and Gaussian Adaptive (right) attention. Middle row: learned Gaussian standard deviations (σ_h) for Gaussian (left) and Gaussian Adaptive (right) attention. Bottom row: learned adaptive memory lengths (L_h) for Adaptive (left) and Gaussian Adaptive (right) attention. Each bar shows the mean over 5 random seeds, with error bars indicating standard deviations. These plots illustrate how different inductive biases (Gaussian, Adaptive, and Gaussian Adaptive) evolve during training and how learned parameters adapt relative to their initial values.

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 main claims are accurate, since the abstract and introduction emphasize the role of biased self-attention for dynamics modelling under partial observability, and the paper substantiates this with both the method (Gaussian and adaptive priors) and empirical evidence across Atari environments.

   Guidelines:

   - The answer NA 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 NA 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: Yes, there is a subsection discussing the limitations of our experiments.

   Guidelines:

   - The answer NA 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: [NA]

Justification: We show no theoretical results in this paper, but we leverage theoretical assumptions to drive the empirical results.

Guidelines:

- The answer NA 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: The details of the experiments (setup and environments), architecture and hyperparameters used are widely described throughout the paper and appendices.

   Guidelines:

   - The answer NA 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: Our experiments are based on standard Atari benchmark environments, which are publicly available. We provide open access to our code, building upon existing open-source frameworks Pu et al. [2025], Niu et al. [2023], and include instructions in the README to reproduce the main experimental results. The release contains scripts and configuration files for training and evaluation, along with details on environment setup, ensuring faithful reproduction of the reported results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All the hyperparameter and experimental setup details are described throughout the paper and Appendix A.

   Guidelines:

   - The answer NA 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: [Yes]

   Justification: We report standard deviation error bars for all experiments. In addition, for the full learning curves that provide the final Atari scores, we report standard error bars, which are included in Appendix B.

   Guidelines:

   - The answer NA 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: This details about the computational resources and training times are detailed in Appendix A

   Guidelines:

   - The answer NA 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: We have read and reviewed the NeurIPS Code of Ethics. Our work is purely methodological, evaluated on a standard Atari benchmark, and does not involve human subjects, sensitive data, or any form of deployment with ethical or societal risk. We confirm that our work adheres to the Code without deviation.

   Guidelines:

   - The answer NA 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: [NA]

Justification: Our paper presents a methodological contribution evaluated in controlled Atari benchmark environments. The work does not involve deployment in real-world applications, sensitive domains, or datasets with societal implications. As such, it poses no immediate positive or negative societal impacts beyond advancing reinforcement learning research.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our work does not pose risks of misuse that would require special safeguards such as releasing pretrained models, large language models, image generators, or specific datasets.

Guidelines:

- The answer NA 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 reused code from Pu et al. [2025], Niu et al. [2023], which we properly acknowledged in the README. The original code was released under Apache License, Version 2.0 (Apache-2.0), and we respected the license terms.

Guidelines:

- The answer NA 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: [NA]

Justification: Our work does not introduce any new datasets, benchmarks, or other assets.

Guidelines:

- The answer NA 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: [NA]

Justification: Our research does not involve crowdsourcing or human subjects.

Guidelines:

- The answer NA 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: [NA]

Justification: Our work does not involve human subjects or crowdsourcing.

Guidelines:

- The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs were used exclusively for formatting, writing and editing.

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.