PALMER: A Parallel Attentive LSTM-Transformer Architecture for Partially Observable Reinforcement Learning

Group 10

Abstract

In this work, we propose PALMER, a novel reinforcement learning (RL) architecture that integrates parallel LSTM and Transformer memory streams through an adaptive gating mechanism to address partial observability in sequential decision-making. By combining short-term sequential modeling and long-term associative attention, PALMER aims to capture diverse temporal dependencies. We evaluate PALMER across four MiniGrid-Memory environments with varying memory demands and compare it against PPO-LSTM and PPO-TrXL baselines. Our experiments show that PALMER achieves faster convergence and competitive performance in structured settings, particularly outperforming in MiniGrid-MemoryS11 and S13. However, PALMER's gains diminish in stochastic tasks like MemoryS17Random, where its adaptive gating exhibits instability. We further analyze the dynamic memory usage via learned gate weights and identify key limitations, including limited training steps and lack of hyperparameter tuning. Our findings suggest PALMER is a promising hybrid memory model, but requires further refinement and extensive evaluation across diverse environments. Interactive visualizations and demo can be found in our Colab Notebook.

1 Introduction

Memory is a fundamental challenge in reinforcement learning (RL), particularly under conditions of partial observability. Agents must effectively retain and recall past observations to succeed in tasks involving delayed decision-making. Although recurrent neural network (RNN)-based policies such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Transformer-based models (Vaswani et al., 2017; Parisotto et al., 2020) have demonstrated success, each has notable limitations. LSTM networks excel at capturing local temporal dependencies but suffer from rapid memory decay and limited long-term credit assignment capabilities (Parisotto et al., 2020). Conversely, Transformers, effectively model global context through self-attention mechanisms but are computationally expensive and prone to instability during training (Parisotto et al., 2020; Liu et al., 2021).

Consequently, existing models struggle to simultaneously achieve efficient long-term temporal credit assignment and stable memory retention (Parisotto et al., 2020; Hochreiter and Schmidhuber, 1997). Transformers, while effective at modeling long-range dependencies, often face challenges in reinforcement learning due to training instability and lack of inductive bias for sequential processing (Pleines et al., 2025). On the other hand, LSTM-based agents are better suited for local temporal reasoning but tend to forget earlier observations in long-horizon tasks. These limitations highlight the need for novel architectures capable of leveraging the strengths of both local sequential and global associative memory processes without compromising training efficiency or stability (Goyal et al., 2021).

Inspired by cognitive neuroscience research, we observe that human memory involves distinct yet interconnected systems: working memory (short-term, sequential) managed predominantly by the prefrontal cortex (PFC), and episodic memory (long-term, associative) mediated primarily by the hippocampus (McClelland et al., 1995; Wang et al., 2018). This biological insight motivates the development of hybrid neural architectures that emulate the cooperative dynamics of these memory systems.

In this work, we introduce **PALMER**, a novel hybrid memory architecture combining parallel LSTM and Transformer modules. PALMER leverages the complementary strengths of both components by utilizing LSTMs to capture short-term sequential dynamics and Transformers to model

long-term associative relationships. These parallel memory streams are fused through a learned gating mechanism, dynamically weighting their contributions at each timestep. Inspired by biological models of the basal ganglia and cognitive routing mechanisms (Kriete et al., 2013; Graves et al., 2016), the gating mechanism allows PALMER to adaptively select between short-term and long-term memory based on contextual cues, improving temporal generalization in partially observable environments.

Our primary contributions are summarized as follows:

- We propose PALMER, a parallel attentive memory architecture combining LSTM and Transformer modules to enhance memory representation in RL.
- We integrate PALMER into the CleanRL framework and benchmark it comprehensively on the MiniGrid-Memory environments: `MemoryS11-v0`, `MemoryS13-v0`, `MemoryS13Random-v0`, and `MemoryS17Random-v0` (Chevalier-Boisvert et al., 2023).
- We perform a rigorous quantitative comparison against two strong baselines: PPO-LSTM (Hochreiter and Schmidhuber, 1997) and PPO-TrXL (Pleines et al., 2025), analyzing performance in terms of success rate and episode length.

2 Background and Related Work

Memory plays a central role in intelligent behavior, allowing agents to retain and leverage past experiences over time. In the context of reinforcement learning (RL), memory is crucial for decision-making in partially observable environments, enabling temporal credit assignment and planning across different timescales (Hassabis et al., 2017; Graves et al., 2016). Long-term memory capacity is not only a defining trait of human intelligence but also a foundational requirement for achieving artificial general intelligence (AGI).

The integration of memory in RL policies has seen widespread use of recurrent architectures. LSTM-based models (Hochreiter and Schmidhuber, 1997) offer fast and efficient local memory modeling and are widely adopted in recurrent PPO baselines (PPO-LSTM). However, LSTM struggles to capture long-range dependencies due to gradient decay over time.

In contrast, Transformers offer strong global modeling capabilities and long-range credit assignment. Architectures such as Gated Transformer-XL (GTrXL) (Parisotto et al., 2020) and PPO-TrXL (Pleines et al., 2025) integrate transformers into RL pipelines, enabling better temporal abstraction. Despite their advantages, Transformer-based models suffer from computational inefficiency and training instability.

To address these limitations, various hybrid and recurrent-Transformer architectures have been proposed. COBERL (Liu et al., 2021), SHM (Liu et al., 2022), BiGRUFormer (Shang et al., 2023), and MERLIN (Wayne et al., 2018) explore different designs that combine recurrence with attention mechanisms. Our work builds on these insights, introducing PALMER—a parallel LSTM-Transformer model with a learned gating mechanism to dynamically blend short-term and long-term memory representations.

The MiniGrid environment (Chevalier-Boisvert et al., 2018a) has become a standard benchmark for evaluating memory capacity in RL agents. Memory games such as MiniGrid-MemoryS13 and MemoryS17Random test agents' ability to retain key observations over long sequences. These environments simulate real-world memory demands like object permanence, spatial navigation, and key-goal dependencies. Notably, environments like Memory Gym (Pleines et al., 2025) and PopGym (Pitis and et al., 2022) further challenge agents with tasks requiring multi-scale and continual memory retention.

PPO-LSTM remains a strong baseline for memory-intensive RL tasks due to its simplicity and training stability. Recent advances introduced PPO-TrXL, a CleanRL-based implementation of PPO with Transformer-XL as a memory module (Pleines et al., 2025). It demonstrates competitive performance in memory environments while allowing plug-and-play integration with modern RL frameworks.

3 Method

3.1 Architecture Overview

PALMER is a hybrid memory architecture designed to address partial observability in reinforcement learning by integrating complementary memory systems for short- and long-term reasoning. As illustrated in Figure 1, PALMER consists of five

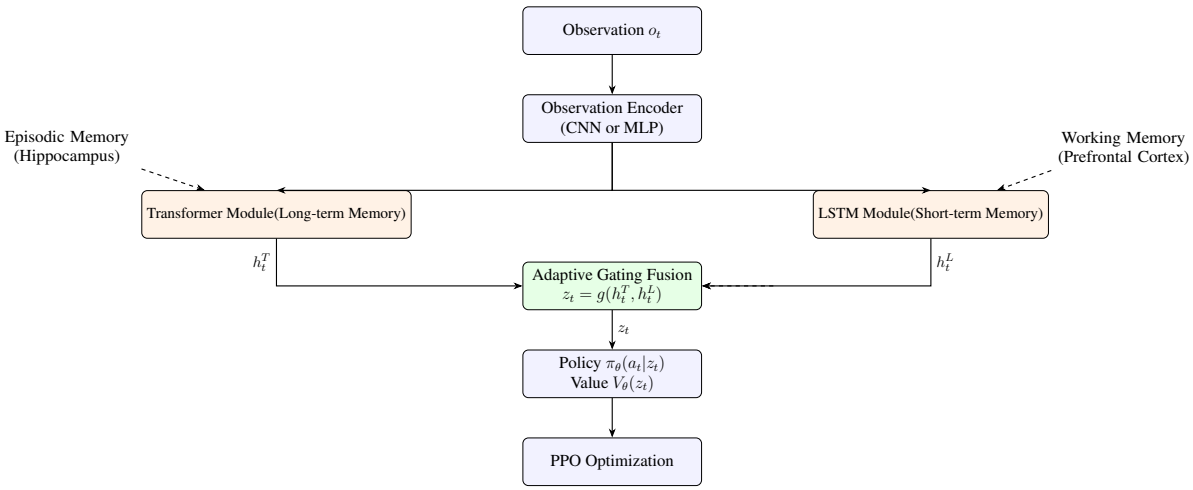


Figure 1: PALMER architecture. Observations are encoded and passed to parallel memory modules (LSTM for short-term, Transformer for long-term), fused via an adaptive gating mechanism, then used to produce policy and value predictions trained via PPO.

components. An observation encoder first maps raw inputs into fixed-size embeddings. These are processed in parallel by an LSTM for local sequential modeling and a Transformer for long-range attention.

The outputs from both streams are fused via an adaptive gating mechanism, which dynamically balances their contributions to form a context-aware representation. This fused output is then fed into policy and value heads for action selection and value estimation. The entire architecture is trained end-to-end using the PPO algorithm with clipped policy gradients for stable and efficient learning.

3.2 Observation Encoding

The first stage of PALMER processes the raw environment observation o_t into an intermediate representation $x_t \in R^d$, suitable for input to the memory modules. The encoding strategy depends on the nature of the input:

Visual Observations (Images) For environments where o_t is an RGB image or spatial feature map (e.g., MiniGrid top-down views), we use a convolutional neural network (CNN) encoder. The CNN consists of convolutional layers with ReLU activations, followed by flattening and a linear projection layer to produce a fixed-dimensional embedding:

$$x_t = \text{Proj}(\text{Flatten}(\text{Conv}(o_t))) \quad (1)$$

This allows the model to extract spatial and compositional features from raw pixels efficiently.

Vector Observations (State Vectors) In environments with low-dimensional inputs (e.g., symbolic agent states), the observation o_t is directly encoded using a multi-layer perceptron (MLP) with nonlinear activations:

$$x_t = \text{MLP}(o_t) = f(W_2 \cdot f(W_1 \cdot o_t + b_1) + b_2) \quad (2)$$

where $f(\cdot)$ denotes a non-linearity such as ReLU or GELU. The output embedding dimension is aligned with the visual encoder output and shared across both memory streams.

3.3 Memory Modules

PALMER features two parallel memory streams for encoding temporal context: a Transformer block for global, long-term dependencies, and an LSTM block for recent, short-term transitions. At each timestep t, both modules receive the same encoded observation x_t and produce distinct hidden representations h_t^T and h_t^L.

3.3.1 Transformer Block

The Transformer module captures long-range patterns using multi-head self-attention. Given a sequence $X = [x_1, \ldots, x_t]$, attention is computed from projections $Q = XW^Q$, $K = XW^K$, $V = XW^V$, and applied as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \quad (3)$$

Multiple attention heads are combined as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{4}$$

where each head is defined as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{5}$$

Each Transformer block includes masked self-attention (to prevent information leakage from future timesteps), pre-layer normalization, and a position-wise feed-forward network. The FFN applies a two-layer transformation:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{6}$$

Residual connections ensure gradient flow, and learnable positional encodings are added to preserve temporal order. The final output at timestep t is the long-term memory representation:

$$h_t^T = \text{Transformer}(x_1, \ldots, x_t) \tag{7}$$

3.3.2 LSTM Block

The LSTM module models short-term dependencies via gated recurrence, processing one observation embedding x_t at each timestep. It updates the hidden and cell states (h_{t-1}^L, c_{t-1}^L) from the previous step to produce current states (h_t^L, c_t^L):

$$(h_t^L, c_t^L) = \text{LSTM}(x_t, h_{t-1}^L, c_{t-1}^L) \tag{8}$$

The internal operations of the LSTM involve several gating mechanisms. The forget gate controls how much past memory to retain:

$$f_t = \sigma(W_f[h_{t-1}^L, x_t] + b_f) \tag{9}$$

The input gate and candidate memory vector determine how new information is incorporated:

$$i_t = \sigma(W_i[h_{t-1}^L, x_t] + b_i) \tag{10}$$

$$\tag{11}$$

$$\tilde{c}_t = \tanh(W_c[h_{t-1}^L, x_t] + b_c) \tag{12}$$

The cell state is then updated as a combination of retained and incoming memory:

$$c_t^L = f_t \odot c_{t-1}^L + i_t \odot \tilde{c}_t \tag{13}$$

The output gate modulates which part of the memory is exposed to the next layers:

$$o_t = \sigma(W_o[h_{t-1}^L, x_t] + b_o) \tag{14}$$

Finally, the hidden state is updated as:

$$h_t^L = o_t \odot \tanh(c_t^L) \tag{15}$$

Here, $\sigma(\cdot)$ denotes the sigmoid activation function and $\odot$ represents element-wise multiplication. This gating structure allows the LSTM to retain fine-grained temporal dependencies and respond to short-term variations, making it particularly effective for tasks that require immediate memory, such as sequential action planning or obstacle avoidance.

Both h_t^L and the Transformer output h_t^T are forwarded to the adaptive gating mechanism, described in Section 3.4, which determines their relative contributions to the agent's final policy representation.

3.4 Adaptive Gating Mechanism

To leverage the complementary strengths of short-term and long-term memory, PALMER employs an adaptive gating mechanism that fuses the outputs of the LSTM and Transformer modules. This mechanism dynamically regulates their relative contributions based on the current state and context of the agent.

At timestep t, let $h_t^L, h_t^T \in R^d$ denote the LSTM and Transformer outputs, respectively. PALMER computes the fused memory representation $z_t \in R^d$ as a convex combination:

$$z_t = g_t \odot h_t^T + (1 - g_t) \odot h_t^L \tag{16}$$

where $g_t \in (0, 1)^d$ is a learned gating vector, and $\odot$ denotes element-wise multiplication. This formulation ensures that each element of z_t is adaptively drawn from either memory stream.

The gating signal g_t is computed from the concatenated hidden states:

$$g_t = \sigma(W_g[h_t^T; h_t^L] + b_g) \tag{17}$$

where $[h_t^T; h_t^L] \in R^{2d}$ is the concatenated vector of the two memory outputs, $W_g \in R^{d \times 2d}$ and $b_g \in R^d$ are learnable parameters, and $\sigma(\cdot)$ is the element-wise sigmoid activation ensuring the output is in $(0, 1)^d$.

The gating vector g_t acts as a soft attention mechanism over memory types. For a given dimension i, $g_{t,i} \approx 1$ implies that the Transformer is favored, while $g_{t,i} \approx 0$ implies LSTM dominance. Since g_t is computed jointly from both streams, it is context-aware and can vary across time, environment states, and learning stages. This design enables PALMER to adaptively select memory resources in response to the temporal demands of each task.

3.5 Optimization via PPO

PALMER is optimized using the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017), which offers a balance between performance and stability in policy gradient methods. PPO constrains policy updates via a clipped surrogate objective to prevent destructive updates and improve sample efficiency.

At each training iteration, PALMER minimizes the following composite loss:

$$\begin{aligned}\mathcal{L}_{\text{PPO}}(\theta) = & -E_t\left[\min\left(r_t\hat{A}_t,\ \text{clip}(r_t,\ 1-\epsilon,\ 1+\epsilon)\hat{A}_t\right)\right] \\ & + c_v E_t\left[\left(V_\theta(s_t) - V_t^{\text{target}}\right)^2\right] \\ & - c_e E_t\left[\mathcal{H}(\pi_\theta(\cdot|s_t))\right]\end{aligned} \tag{18}$$

The first term is the clipped policy loss, where $r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the importance sampling ratio between current and old policy, and $\hat{A}_t$ is the estimated advantage obtained using Generalized Advantage Estimation (GAE).

The second term penalizes value estimation error. Here, $V_\theta(s_t)$ is the predicted value of state s_t, and V_t^{target} is the empirical return (bootstrapped from Monte Carlo rollouts or TD-lambda returns).

The third term encourages policy entropy to maintain sufficient exploration during training. $\mathcal{H}(\pi_\theta(\cdot|s_t))$ denotes the entropy of the action distribution.

The scalar coefficients c_v and c_e weight the contribution of the value and entropy losses, while ϵ is a hyperparameter (typically 0.2) controlling the clipping range of policy updates.

4 Experimental Setup

Environments We conduct experiments using four variants of the MiniGrid-Memory environment suite: `MiniGrid-MemoryS11-v0`, `MiniGrid-MemoryS13-v0`, `MiniGrid-MemoryS13Random-v0`, and `MiniGrid-MemoryS17Random-v0`. These environments are sourced from the MiniGrid benchmark suite (Chevalier-Boisvert et al., 2018b) and are designed to assess the agent's ability to retain and retrieve memory over varying temporal horizons.

In each episode, the agent starts in a small room where it briefly observes a single object. It must then navigate a narrow hallway ending in a split, where it is presented with two candidate objects. Only one of these matches the object initially observed. The agent's task is to correctly recall the observed object and navigate to the matching one at the end of the split. This setup explicitly requires the agent to rely on internal memory representations, making it a robust benchmark for evaluating temporal credit assignment and memory retention under partial observability.

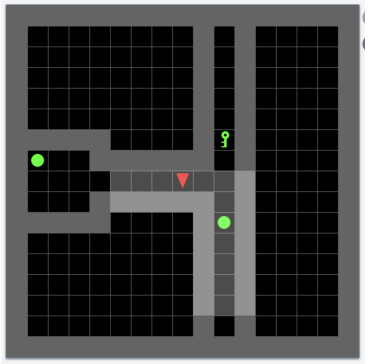


Figure 2: Illustration of the MiniGrid-Memory environment. The agent (red triangle) must remember and navigate to the goal (green ball) after visiting a distant distractor object.

Architectures. We evaluate three neural architectures in our experiments:

PPO-LSTM serves as a standard baseline recurrent architecture. It captures temporal dependencies efficiently over short sequences but typically struggles with longer-term memory demands (Hochreiter and Schmidhuber, 1997).

PPO-TrXL: integrates a Transformer-XL backbone into the PPO framework by leveraging recurrence-aware attention and memory caching, enabling the agent to model long-term temporal dependencies efficiently in partially observable environments (Pleines et al., 2025).

PALMER (our proposed method) leverages a parallel Transformer-LSTM architecture combined via adaptive gating mechanisms.

Evaluation Metrics. We assess the performance of all architectures using the following metrics:

- **Success Rate (primary metric)**: The percentage of episodes in which the agent successfully accomplishes its goal.
- **Mean Episode Length**: Average number of steps taken by the agent per episode, reflecting both efficiency and task completion speed.

- **Average Gate Value Over Steps**: Unique to the PALMER architecture, this metric quantifies how the model dynamically balances information between Transformer and LSTM modules.

Training Setup. All models were trained using the CleanRL framework (Huang et al., 2022), a simplified implementation of PPO for reproducible and efficient RL experiments. Training was performed on a GPU (NVIDIA GeForce RTX 4070).

Detailed hyperparameters for all architectures are provided in Appendix 7.

5 Experiment Results

The comparative performance analysis in the MiniGrid-MemoryS13 and MiniGrid-MemoryS13Random environments, depicted in Figure 3 and Figure 4, reveals nuanced differences between PALMER and baseline architectures. In the deterministic MiniGrid-MemoryS13 setting, PALMER demonstrates a notably faster convergence in both success rate and mean episode length compared to PPO-LSTM and PPO-TrXL. This advantage likely stems from its hybrid structure that combines the Transformer and LSTM, effectively utilizing both short-term and long-term dependencies early in training, thereby quickly learning critical decision paths. However, despite rapid initial improvements, PALMER exhibits instability at around the 0.5 success rate threshold, failing to surpass PPO-LSTM in stable performance levels. This instability might be attributed to the adaptive gating mechanism occasionally misallocating importance between memory modules, thus impeding stable policy optimization.

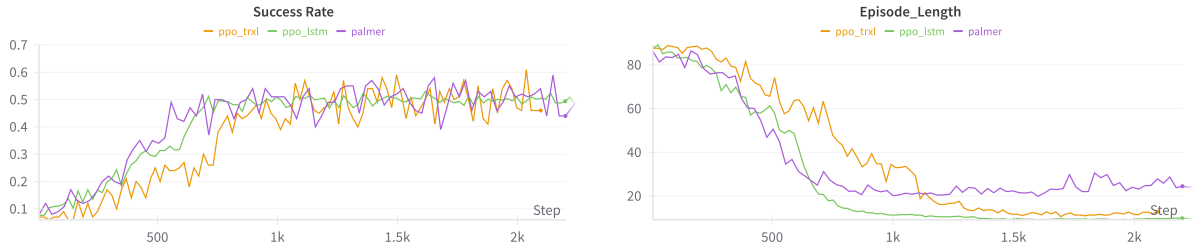
When evaluating mean episode length in the deterministic scenario, PALMER again shows rapid initial improvement, converging faster than the other architectures. Nevertheless, its stable episode length remains higher than PPO-LSTM and PPO-TrXL. This phenomenon suggests that while PALMER quickly identifies viable solutions, its adaptive gating may continuously shift attention between memory types unnecessarily, causing fluctuations and resulting in suboptimal efficiency. In the randomized MiniGrid-MemoryS13Random environment, PALMER's performance resides between PPO-LSTM and PPO-TrXL, neither excelling nor significantly lagging. The randomization introduces unpredictability, diminishing the systematic long-term advantage from which Transformer-based architectures, like PPO-TrXL, typically benefit. Consequently, PALMER's advantage from its hybrid memory architecture becomes less pronounced under uncertainty. The presence of randomness demands robust sequential decision-making, where PPO-LSTM maintains slight superiority due to its inherent strength in handling rapidly changing short-term information. PALMER, attempting to balance long-term context from Transformer modules and short-term context from LSTM units, occasionally falters as randomness challenges the gating mechanism's efficiency in dynamically adjusting between memory types.

Thus, while PALMER initially capitalizes on hybrid memory management for rapid learning, especially in structured environments, its adaptive gating mechanism may introduce instabilities during optimization under both deterministic and random scenarios. Addressing the balance and tuning of the gating mechanism might enhance PALMER's stability and robustness, especially under conditions of randomness.

For the MiniGrid-MemoryS11 environment (Appendix Figure 6), PALMER consistently outperforms both baseline architectures across metrics. It achieves a higher and more stable success rate earlier during training and converges quickly to shorter episodes, suggesting effective exploitation of hybrid memory in simpler, predictable settings. In contrast, PALMER's performance on the MiniGrid-MemoryS17Random environment (Appendix Figure 7) degrades notably, falling between PPO-LSTM and PPO-TrXL. The randomization introduces substantial variability and demands more robust generalization capabilities, highlighting PALMER's sensitivity to environmental stochasticity and its potential instability in highly uncertain scenarios.

The gating weight g, which determines the relative contributions of Transformer-based long-term memory and LSTM-based short-term memory in PALMER, is illustrated in Figure 5. Initially, all environments show a similar downward trend, indicating the model's early reliance on short-term memory provided by LSTM for immediate feedback and rapid adaptation. However, this trend gradually shifts, revealing subtle but significant divergences between environments as training progresses.

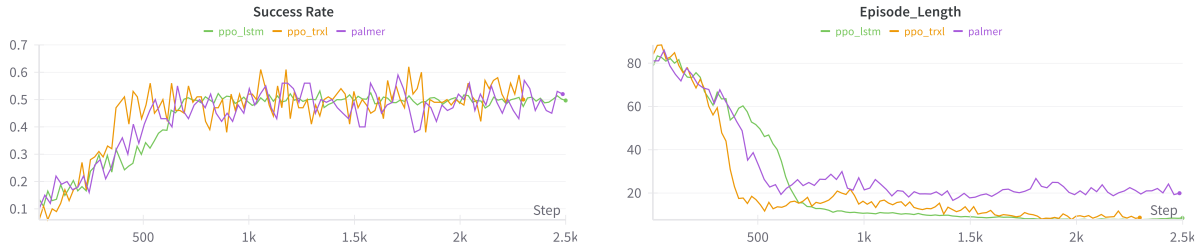
Specifically, in the static environments (S11,



(a) Success Rate over Steps

(b) Mean Episode Length over Steps

Figure 3: Performance of PALMER and baseline agents in MiniGrid-MemoryS13 environments.



(a) Success Rate over Steps

(b) Mean Episode Length over Steps

Figure 4: Performance of PALMER and baseline agents in MiniGrid-MemoryS13Random environments.

S13), gate values stabilize around a mid-point (approximately 0.41–0.42), suggesting a balanced fusion of both short-term sequential information and long-term episodic memory. This indicates that in stable, predictable contexts, PALMER effectively integrates memory at both timescales without heavily favoring one over the other, reflecting the steady environment dynamics that require balanced temporal reasoning.

Contrastingly, environments with randomization (S13Random, S17Random) show a distinct upward trend after an initial drop, particularly prominent in S17Random. This gradual increase towards the Transformer-dominant region (higher gate values approaching 0.43–0.46) signals a growing reliance on long-term episodic memory, likely due to the complexity and unpredictability of these environments. PALMER, therefore, adapts its strategy, favoring more extensive temporal context provided by the Transformer to maintain robust performance in the presence of randomness and variability.

Critically, the increasing gate values in random environments highlight an essential behavior of PALMER's adaptive gating mechanism: it dynamically allocates greater emphasis to episodic memory when confronted with more challenging, unpredictable scenarios. However, this adaptive flexibility can also introduce instability, as evidenced by performance fluctuations. The S17Random environment, notably, illustrates this instability, where PALMER does not consistently outperform the baselines. This suggests the gating mechanism, although beneficial for dynamic memory allocation, may sometimes compromise model stability when the environment complexity surpasses a certain threshold.

We further provide qualitative insights into agent behavior by analyzing rollout videos of trained policies in the MiniGrid-Memory environments. The agent behaviors can be viewed in our Colab demo[1]. At early stages of training, the agent exhibits minimal exploratory behavior—often remaining stationary or wandering aimlessly without reaching the target. This is expected, as the agent has not yet learned to associate visual cues with task objectives. As training progresses, however, we observe a clear behavioral shift: the agent begins to actively explore the left-side of the environment to observe the memory symbol (e.g., a colored object), and then navigates toward the correct target location on the right. Notably, with more training steps, the agent's movements become increasingly deliberate and efficient, with minimal hesitation or "thinking steps," suggesting it has learned a robust memory retrieval policy. This progression highlights the model's growing capacity to store task-relevant in-

[1] Colab Notebook Link

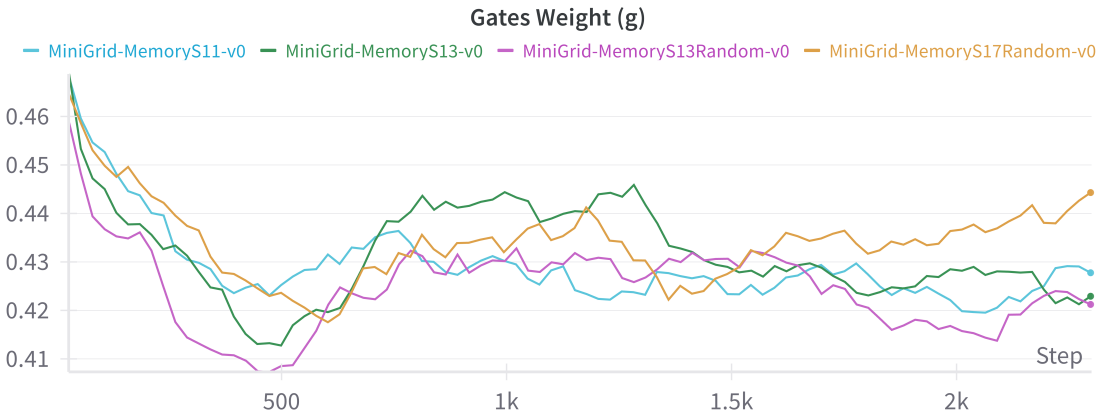


Figure 5: Average adaptive gate weights (g) over steps across MiniGrid memory environments. Higher values indicate greater reliance on the Transformer pathway, while lower values for more dependence on the LSTM module.

formation (e.g., target identity) and retrieve it at the appropriate moment.

Despite this improvement, occasional failure modes persist across all architectures. In particular, we observe cases where the agent confidently moves to the incorrect target, even after correctly observing the cue. This likely reflects imperfections in memory encoding or retrieval under policy noise or ambiguity in the gating mechanism (for PALMER). These errors underscore the difficulty of precise memory recall in long-horizon tasks and suggest a potential benefit in augmenting architectures with confidence-aware or error-corrective mechanisms. Overall, the demos offer qualitative confirmation of the agent's learning process and expose the nuanced limitations of memory-based reasoning under partial observability.

6 Limitations and Future Work

This study explores the effectiveness of the PALMER architecture in memory-intensive tasks within the MiniGrid-Memory environments. However, several limitations should be acknowledged and addressed in future work. Firstly, due to constraints in time and computational resources, we limited our training steps to approximately 2 million timesteps. Previous studies in memory-based reinforcement learning typically use significantly higher training budgets (Pleines et al., 2025) (e.g., around 100 million timesteps), suggesting that PALMER's full capabilities might not have been fully demonstrated under our limited training regime. Extended experiments with substantially more training steps are therefore recommended.

Secondly, the generalizability and robustness of PALMER could be more thoroughly evaluated by benchmarking it across other environments specifically designed to test long-term memory retention, such as Memory Gym (Meter, 2023), PopGym (Morad et al., 2023), DMLab-30 (Beattie et al., 2016), and ViZDoom (Kempka et al., 2016). Future research should systematically assess PALMER in these varied memory contexts to provide deeper insights into its versatility and long-term memory handling.

Thirdly, hyperparameter optimization was not exhaustively performed; the hyperparameters used were primarily derived from baseline architectures (PPO-TrXL and PPO-LSTM). To achieve a fair evaluation of PALMER's capabilities, extensive hyperparameter tuning specific to the PALMER architecture should be conducted to identify optimal settings and further enhance its performance.

Fourthly, we did not explicitly measure or compare training times across the different architectures tested. Recording and analyzing training times would provide valuable insights into PALMER's computational efficiency relative to baseline architectures, such as PPO-TrXL and PPO-LSTM.

Lastly, it would be highly beneficial to visualize the internal mechanisms of memory storage and forgetting within the agent's operation. Developing visualization tools or analyses to illustrate these processes could significantly enhance our understanding of PALMER's internal dynamics and inform further architectural improvements.

7 Conclusion

This study introduced PALMER, a novel reinforcement learning architecture that integrates parallel short-term (LSTM) and long-term (Transformer) memory modules via adaptive gating. Evaluations on MiniGrid-Memory environments indicate PALMER often converges faster and achieves competitive or superior performance in non-random settings compared to PPO-LSTM and PPO-TrXL. However, in random and more complex scenarios, PALMER demonstrated mixed results, suggesting sensitivity to environmental stochasticity and insufficient training duration. Future improvements include extending training steps significantly, performing hyperparameter optimization specifically for PALMER, and validating on diverse, memory-intensive benchmarks such as Memory Gym, PopGym, DMLab30, and ViZDoom.

References

Charles Beattie et al. 2016. Deepmind lab. *arXiv preprint arXiv:1612.03801*.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. 2018a. Minigrid: A minimalistic gridworld environment for openai gym. Https://github.com/Farama-Foundation/Minigrid.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. 2018b. Minimalistic gridworld environment for openai gym.

Maxime Chevalier-Boisvert, Lucas Willems, Suman Pal, Ahmed Touati, et al. 2023. MiniGrid: Minimalistic gridworld environment for openai gymnasium.

Anirudh Goyal, Alex Lamb, Jordan Hoffman, Jack Rae, Shakir Mohamed, Matthew Botvinick, Hugo Larochelle, and Yoshua Bengio. 2021. Coordination among neural modules through a shared global workspace. *Nature Communications*, 12(1):1–16.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2016. Hybrid computing using a neural network with dynamic external memory. In *Nature*, volume 538, pages 471–476.

Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. 2017. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Shengyi Huang et al. 2022. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms.

Michał Kempka et al. 2016. Vizdoom: A doom-based ai research platform for visual reinforcement learning. Accessed: 2025-04-09.

Trenton Kriete, David C Noelle, Jonathan D Cohen, and Randall C O'Reilly. 2013. Indirection and symbol-like processing in the prefrontal cortex and basal ganglia. *Proceedings of the National Academy of Sciences*, 110(41):16390–16395.

Bing Liu, Yao Meng, Xiaojie Yang, Qi Wang, Zhenyu Fu, and Dongyan Zhou. 2021. Behavior transformers: Cloning k modes with one stone. *arXiv preprint arXiv:2106.12008*.

Bing Liu et al. 2022. Structured memory for reinforcement learning. *arXiv preprint arXiv:2202.02330*.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

Marco Meter. 2023. Endless memory gym. Accessed: 2025-04-09.

Steven Morad et al. 2023. Popgym. Accessed: 2025-04-09.

Emilio Parisotto et al. 2020. Stabilizing transformers for reinforcement learning. *ICML*.

Silviu Pitis and et al. 2022. The environment suite for measuring multi-scale memory in rl. *arXiv preprint arXiv:2206.07710*.

Marco Pleines, Matthias Pallasch, Frank Zimmer, and Mike Preuss. 2025. Memory gym: Towards endless tasks to benchmark memory capabilities of agents. *Journal of Machine Learning Research*, 26:1–40. ArXiv preprint arXiv:2309.17201.

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

Yao Shang et al. 2023. Bigruformer: A bigru-enabled transformer for reinforcement learning. *arXiv preprint arXiv:2309.17207*.

Ashish Vaswani et al. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matthew Botvinick. 2018. Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, 21(6):860–868.

Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack W Rae, Piotr Mirowski, Joel Z Leibo, Adam Santoro, et al. 2018. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*.

A Hyperparameters

Hyperparameter	**Value**
Architecture Parameters	
LSTM Layers	3
LSTM Hidden Dimension	384
LSTM Sequence Length	128
Optimization Parameters	
Initial Learning Rate	2.75×10^{-4}
Final Learning Rate	1.0×10^{-5}
Number of Environments	32
Steps per Environment	512
Batch Size	16,384
Mini-batch Size	2,048
Anneal Steps	1.64×10^{8}
Update Epochs	3
Number of Mini-batches	8
Reinforcement Learning Parameters	
Discount Factor (γ)	0.995
GAE Lambda	0.95
Clipping Coefficient	0.1
Initial Entropy Coefficient	1.0×10^{-4}
Final Entropy Coefficient	1.0×10^{-6}
Value Function Coefficient	0.5
Max Gradient Norm	0.25
Reconstruction Coefficient	0.0

Table 1: LSTM Architecture Hyperparameters

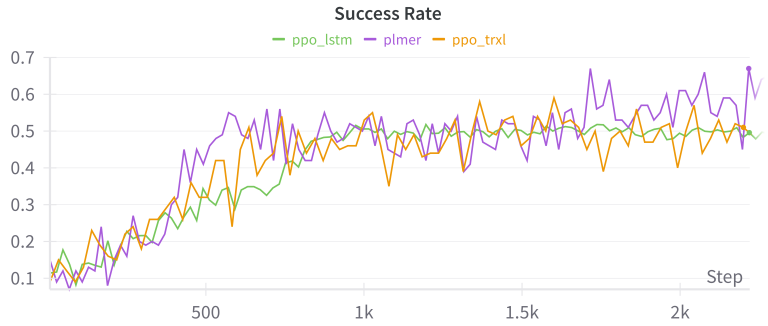
Hyperparameter	Value
Architecture Parameters	
Transformer Layers	3
Transformer Heads	4
Transformer Dimension	384
Memory Length	119
Positional Encoding	absolute
Optimization Parameters	
Initial Learning Rate	2.75×10^{-4}
Final Learning Rate	1.0×10^{-5}
Number of Environments	32
Steps per Environment	512
Batch Size	16,384
Mini-batch Size	2,048
Anneal Steps	1.64×10^{8}
Update Epochs	3
Number of Mini-batches	8
Reinforcement Learning Parameters	
Discount Factor (γ)	0.995
GAE Lambda	0.95
Clipping Coefficient	0.1
Initial Entropy Coefficient	1.0×10^{-4}
Final Entropy Coefficient	1.0×10^{-6}
Value Function Coefficient	0.5
Max Gradient Norm	0.25
Reconstruction Coefficient	0.0

Table 2: TransformerXL Architecture Hyperparameters

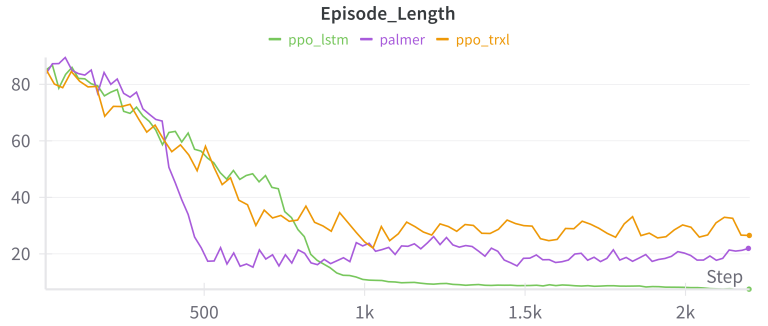
Hyperparameter	Value
Architecture Parameters	
LSTM Layers	3
LSTM Hidden Dimension	384
Transformer Layers	3
Transformer Heads	4
Transformer Dimension	384
Memory Length	119
Positional Encoding	absolute
Minimum Gate Value	0.2
Optimization Parameters	
Initial Learning Rate	2.75×10^{-4}
Final Learning Rate	1.0×10^{-5}
Number of Environments	32
Steps per Environment	512
Batch Size	16,384
Mini-batch Size	2,048
Anneal Steps	1.64×10^{8}
Update Epochs	3
Number of Mini-batches	8
Reinforcement Learning Parameters	
Discount Factor (γ)	0.995
GAE Lambda	0.95
Clipping Coefficient	0.1
Initial Entropy Coefficient	1.0×10^{-4}
Final Entropy Coefficient	1.0×10^{-6}
Value Function Coefficient	0.5
Max Gradient Norm	0.25
Reconstruction Coefficient	0.0

Table 3: PALMER Architecture Hyperparameters

B Evaluation Results

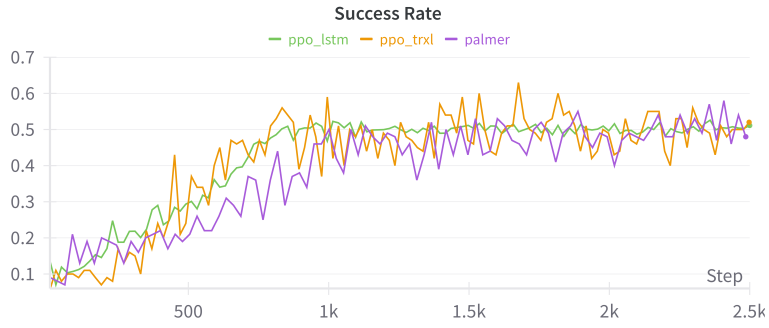


(a) Success Rate over Steps

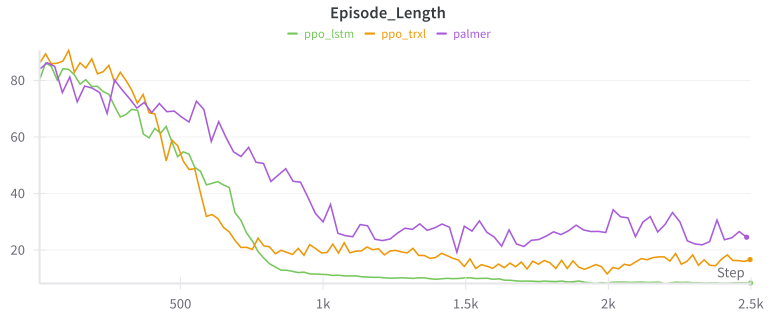


(b) Mean Episode Length over Steps

Figure 6: Performance of PALMER and baseline agents in MiniGrid-MemoryS11 environments.



(a) Success Rate over Steps



(b) Mean Episode Length over Steps

Figure 7: Performance of PALMER and baseline agents in MiniGrid-MemoryS17Random environments.