Mastering Memory Tasks with World Models

April 23, 2025

Reference

Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering memory tasks with world models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=1vDArHJ68h.

RNNs vs. Transformers for Long Sequences

- **World models** in RL capture environment dynamics
  - Enable agents to perceive, simulate, and plan
  - Learn from past experiences to predict future states
- **Key challenges**:
  - Credit assignment problem
  - Memorizing and recalling past experiences
  - Learning long-range dependencies
- **RNNs** (common in MBRL):
  - Handle sequential data
  - Limited by vanishing gradients
- **Transformers**:
  - Successful in language modeling
  - Quadratic complexity in sequence length
  - Unstable during training on long sequences
- **State-Space Models (SSMs)**:
  - Effectively capture dependencies in long sequences
  - S4 model redefines long-range sequence modeling
  - Can handle dependencies up to 16K in length

Introducing Recall to Imagine (R2I) [Samsami et al., 2024]

- **R2I**: First MBRL approach using a variant of S4
  - Built upon DreamerV3
  - Empowers agents with long-term memory
  - Computationally efficient (up to 9x faster)
- **Key contributions**:
  - Memory-enhanced MBRL agent using modified S4
  - State-of-the-art performance in memory domains:
    - POPGym
    - Behavior Suite (BSuite)
    - Memory Maze (outperforms humans)
  - Maintains strong performance in standard benchmarks:
    - Atari
    - DeepMind Control Suite (DMC)
  - Comprehensive ablation studies validate design decisions
- R2I offers a general solution for tasks requiring long-term memory or credit assignment

State-Space Models: Continuous View

- SSMs describe hidden state evolution over time:

$$x'(t) = \mathbf{A}\,x(t) \;+\; \mathbf{B}\,u(t),$$
$$y(t) = \mathbf{C}\,x(t) \;+\; \mathbf{D}\,u(t).$$

- Key components:
  - $x(t) \in \mathbb{C}^N$: hidden "memory" of size N
  - **A**: controls state evolution
  - **B**: injects new input $u(t)$
  - **C**, **D**: read out state to form output y
- Continuous-time model must be discretized for ML tasks

Discretization and S4 Architecture

- Discretizing with time step Δ gives recurrence:

$$x_n = \bar{\mathbf{A}}\, x_{n-1} \;+\; \bar{\mathbf{B}}\, u_n,$$
$$y_n = \bar{\mathbf{C}}\, x_n \;+\; \bar{\mathbf{D}}\, u_n,$$

- **S4 innovations**:
  - Uses diagonal + low-rank (DPLR) structure for **A**
  - **HiPPO initialization**:
    - Decomposes $u(t)$ into infinite basis functions
    - Enables capturing long-range dependencies
  - Instead of stepping through time one step at a time (as an RNN would), S4 shows that the entire mapping

$$(u_{1:T},\, x_0) \;\longmapsto\; (y_{1:T},\, x_T)$$

    can be performed as a single 1D convolution over the sequence.
  - **Parallel scan**: Efficient computation of entire sequence

S4 Advantages for RL World Models

- **S4 as a fusion**:
  - Combines CNNs, RNNs, and classical SSMs
  - Outperforms Transformers in inference speed and memory consumption
  - Recurrent inference mode provides efficiency
- **Benefits for RL world models**:
  - **Long-term credit assignment** becomes feasible
  - **Training speed** increases significantly
  - **Stability** improves on long rollouts
  - Avoids vanishing gradients (RNNs)
  - Prevents instabilities at long sequence lengths (Transformers)
- Impressive empirical results on benchmarks involving long dependencies
- Recent refinements focus on understanding and improving S4
- Ideal backbone for world models in MBRL

Non-recurrent Representation Model in R2I

- **Original coupled (recurrent) setup**:
  - Representation model: $q_\theta(z_t \mid h_t, o_t)$
  - Sequence model: $(h_t, x_t) = f_\theta(h_{t-1}, z_{t-1}, a_{t-1})$
  - Circular dependency: z_t depends on h_t, which depends on z_{t-1}
  - Forces strict time-step ordering, preventing parallelization
- **Breaking the cycle with non-recurrent representation**:
  - Simplify to $q_\theta(z_t \mid o_t)$ by dropping h_t dependency
  - Each posterior z_t inferred only from observation o_t
  - All $z_1, z_2, \ldots, z_T$ can be computed in parallel
- **Parallel sequence modeling with SSMs**:
  - Feed full batch of latents and actions: $h_{1:T}, x_{1:T} = f_\theta((a_{1:T}, z_{1:T}), x_0)$
  - S4-style SSM enables convolutional/parallel-scan implementation
  - Entire sequence processed in one parallel operation
- **Benefits**: Full parallelism, no performance loss, much faster world model updates

R2I Architecture: S3M + Prediction Heads

- **S3M** (Sequence-State-Space Model) is the core world model:
  - **Representation model**: $q_\theta(z_t \mid o_t)$ infers latent state
  - **Dynamics model**: Implicit in training losses
  - **Sequence model**: Updates deterministic belief state h_t
- **Three prediction heads** built on top of S3M:
  - **Observation predictor**: $p_\theta(\hat{o}_t \mid z_t, h_t)$
  - **Reward predictor**: $p_\theta(\hat{r}_t \mid z_t, h_t)$
  - **Continuation predictor**: $p_\theta(\hat{c}_t \mid z_t, h_t)$
- At each time step t, S3M processes:

$$(h_t, x_t) = f_\theta((a_{t-1}, z_{t-1}), x_{t-1})$$

  where x_t represents all internal SSM layer states and h_t is the final output

How the Sequence Model Works

- f_θ **architecture** - stack of SSM layers:
  - Each layer applies discrete-time SSM update:

$$x_n = \bar{\mathbf{A}}\, x_{n-1} + \bar{\mathbf{B}}\, u_n, \quad y_n = \bar{\mathbf{C}}\, x_n + \bar{\mathbf{D}}\, u_n$$

    where u_n is the concatenation (a_{n-1}, z_{n-1})
  - Processing pipeline for each layer:

$$\text{SSM} \to \text{GeLU} \to \text{GLU} \to \text{LayerNorm}$$

- **Key advantages**:
  - Modality-agnostic sequence core with specialized I/O
  - Parallel processing of long sequences: $\mathcal{O}(L \log L)$ complexity
  - Stable training with rich temporal dynamics

R2I: Recurrent World Model Architecture

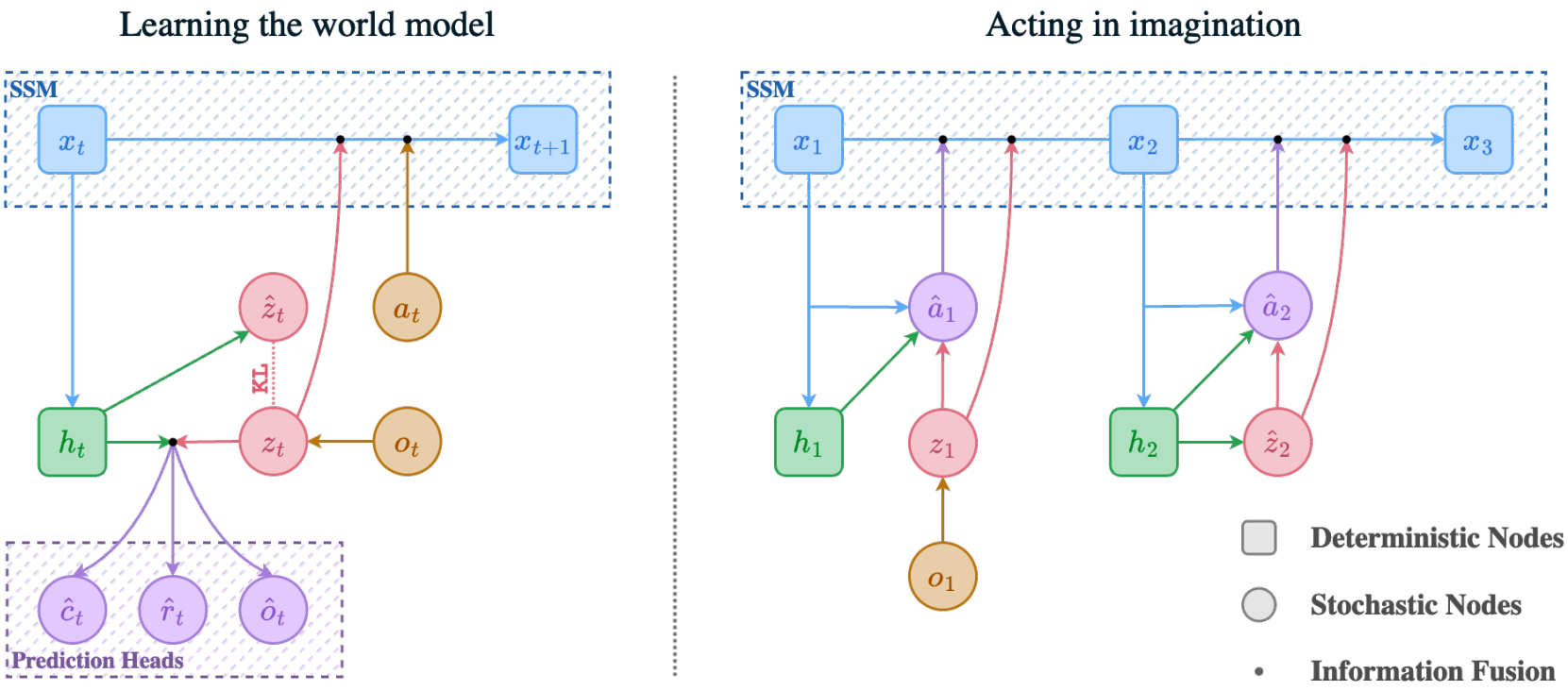

Figure: Graphical representation of R2I. **(Left)** The world model encodes past experiences, transforming observations and actions into compact latent states. Reconstructing the trajectories serves as a learning signal for shaping these latent states. **(Right)** The policy learns from trajectories based on latent states imagined by the world model. The representation corresponds to the full state policy, and we have omitted the critic for the sake of simplifying the illustration.

Training Details

- **Optimization objective**:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{z_{1:T} \sim q_\theta} \sum_{t=1}^{T} \mathcal{L}^{\text{pred}}(\theta, h_t, o_t, r_t, c_t, z_t) + \mathcal{L}^{\text{rep}}(\theta, h_t, o_t) + \mathcal{L}^{\text{dyn}}(\theta, h_t, o_t)$$

- **Loss components**:
  - **Prediction loss**:

$$\mathcal{L}^{\text{pred}} = -\beta_{\text{pred}}(\ln p_\theta(o_t \mid z_t, h_t) + \ln p_\theta(r_t \mid z_t, h_t) + \ln p_\theta(c_t \mid z_t, h_t))$$

  - **Dynamics loss**:

$$\mathcal{L}^{\text{dyn}} = \beta_{\text{dyn}} \max(1, \text{KL}[\text{sg}(q_\theta(z_t \mid o_t)) \,\|\, p(z_t \mid h_t)])$$

  - **Representation loss**:

$$\mathcal{L}^{\text{rep}} = \beta_{\text{rep}} \max(1, \text{KL}[q_\theta(z_t \mid o_t) \,\|\, \text{sg}(p(z_t \mid h_t))])$$

SSMs Computational Modeling

- **Parallelizability options**:
  - Convolution (Gu et al., 2021) vs. parallel scan (Smith et al., 2023)
  - Parallel scan chosen for several key advantages
- **Key reasons for choosing parallel scan**:
  - Essential to pass hidden states x_t to policy in memory environments
  - Avoids burn-in steps needed with convolution mode
  - Enables scaling of sequence length across distributed devices
  - Facilitates resetting of hidden states between episodes

Method	Training	Inference step	Imagination step	Parallel	State Reset
Attention	$\mathcal{O}(L^2)$	$\mathcal{O}(L^2)$	$\mathcal{O}((L+H)^2)$	✓	✓
RNN	$\mathcal{O}(L)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	×	✓
SSM (Conv)	$\mathcal{O}(L)$	$\mathcal{O}(1)$	$\mathcal{O}(L)$	✓	×
SSM (Par.Scan)	$\mathcal{O}(L)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	✓	✓

Table: Asymptotic runtimes of different architectures. L is sequence length and H is imagination horizon. SSMs with parallel scan achieve optimal complexity by combining recurrence with parallel computation.

SSMs Computational Modeling (continued)

- **Computational complexity comparison**:
  - Attention models: Quadratic complexity $\mathcal{O}(L^2)$ during training
  - RNNs: Linear complexity but sequential processing (not parallelizable)
  - SSM with convolution: Requires burn-in steps, leading to $\mathcal{O}(L)$ imagination complexity
  - SSM with parallel scan: Combines best properties - parallel processing with $\mathcal{O}(1)$ imagination steps
- **Hidden state reset capability**:
  - Critical when sampling sequences with multiple episodes from buffer
  - Hidden states must reset from terminal states to initial states in new episodes
  - Improves early training performance when episodes are short

World Model States and Policy Conditioning

- **Two distinct states in R2I's world model**:
  - **Deterministic state** h_t: Output of final SSM layer at time t
    - Analogous to what DreamerV3 feeds to reconstruction heads and GRU
  - **SSM hidden state** x_t: Packs every layer's internal memory at time t
    - Only x_t is carried forward to next SSM update; h_t is not
- **Policy conditioning variants**:
  - Output-state policy: $\pi(a_t \mid z_t, h_t)$
  - Hidden-state policy: $\pi(a_t \mid z_t, x_t)$
  - Full-state policy: $\pi(a_t \mid z_t, h_t, x_t)$
- **Key empirical findings**:
  - In memory-intensive tasks, using (z_t, h_t) breaks policy learning
  - Using (z_t, x_t) restores stable, high-performance learning
  - Full-state policy (z_t, h_t, x_t) fails due to non-stationarity in joint distribution

Practical Recipe for Policy Conditioning

- **Task-dependent policy conditioning**:
  - **Vector or non-memory tasks**: Use output-state policy (z_t, h_t)
  - **Memory-heavy tasks**: Use hidden-state policy (z_t, x_t)
  - Full-state policy (z_t, h_t, x_t) not recommended due to non-stationarity
- **Actor-Critic training**:
  - Adopts same scheme from DreamerV3
  - Imagined rollouts generated by policy inside learned world model
  - Actor maximizes expected imagined returns
  - Critic (value network) predicts those returns
  - Gradients flow through both policy and world model
- **Key insight**: When separating SSM's internal memory x_t from output h_t, policies trained on x_t (plus z_t) are far more stable in long-horizon, memory-demanding tasks

Computational Efficiency Analysis

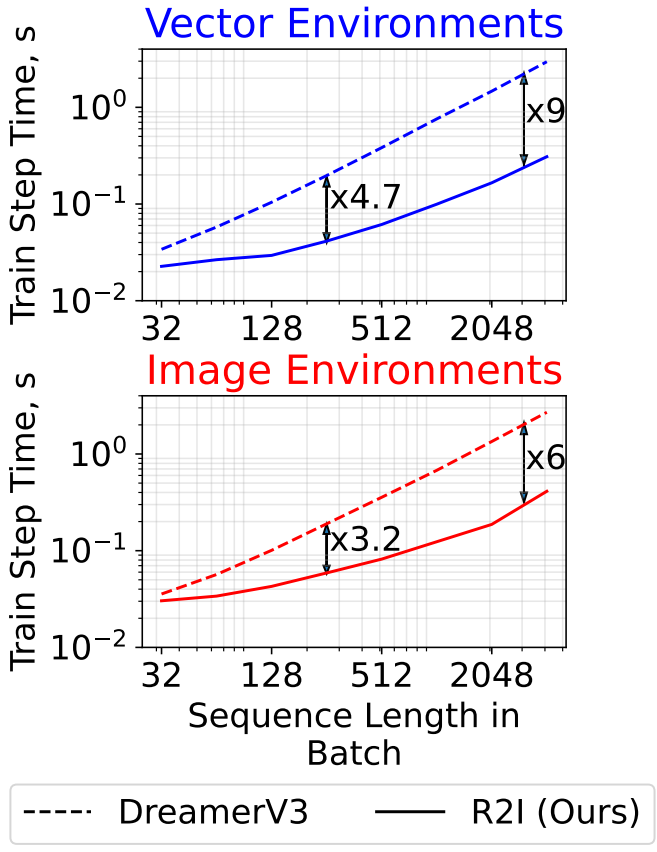

Figure: Computational time taken by DreamerV3 and R2I (lower is preferred)

Quantifying Memory of R2I

- **Memory stress tests** using BSuite tasks:
  - **Memory Length**: Remember initial cue throughout episode
    - First observation determines correct final action
    - Challenge: Maintain memory across variable episode lengths
  - **Discounting Chain**: Link early action to delayed reward
    - Initial action choice determines reward after fixed delay
    - Challenge: Credit assignment across time
- **Key result**: R2I achieves significantly higher success rates across wider range of episode lengths and reward delays.

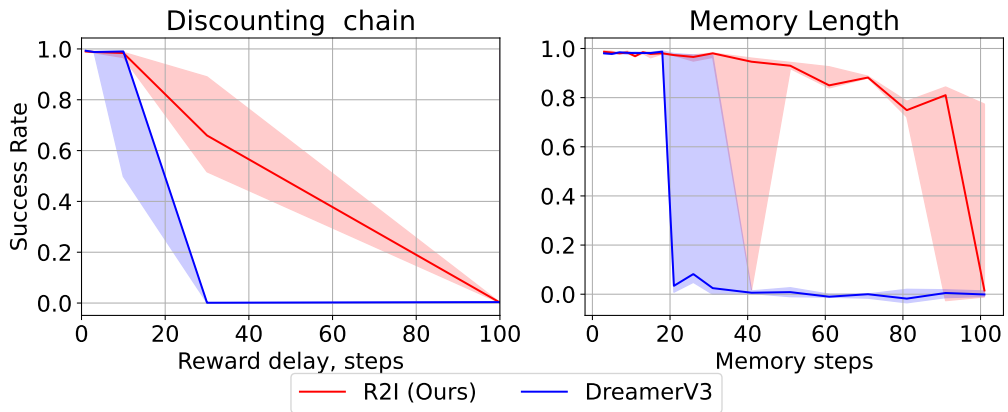

Figure: Success rate comparison between R2I and DreamerV3 on BSuite tasks

Quantifying Memory of R2I (cont.)

- **POPGym benchmark study**:
  - Collection of RL environments for POMDP challenges
  - Focus on navigation, noise robustness, and memory
- **Selected memory-intensive environments**:
  - **RepeatPrevious**: Recall and reproduce past actions
  - **Autoencode**: Memorize and reconstruct observations
  - **Concentration**: Track multiple objects simultaneously
- **Difficulty levels** (Easy, Medium, Hard):
- **Key result**: R2I establishes new SOTA performance in these memory-intensive tasks

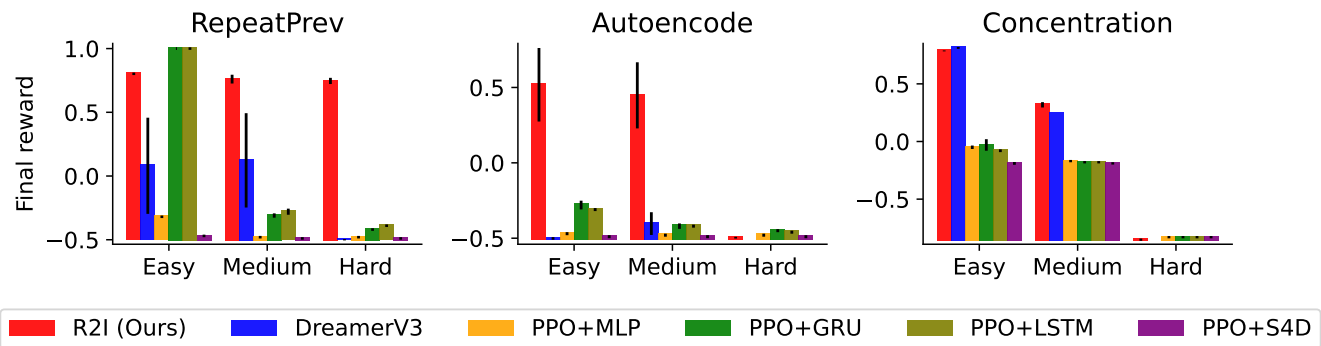

Figure: Performance comparison of R2I against baseline methods on POPGym memory-intensive tasks across different difficulty levels

Combined with BSuite results, demonstrates R2I significantly pushes memory limits

Evaluating Long-term Memory In Complex 3D Tasks

- **Memory Maze** (Pasukonis et al., 2022):
  - Randomized 3D mazes with multiple objects to navigate to
  - Requires agent to remember object locations, maze layout, and position
  - Episodes extend up to 4K environment steps
  - Ideal agent with long-term memory only needs to explore each maze once
  - Existing memory-augmented RL algorithms significantly underperform humans

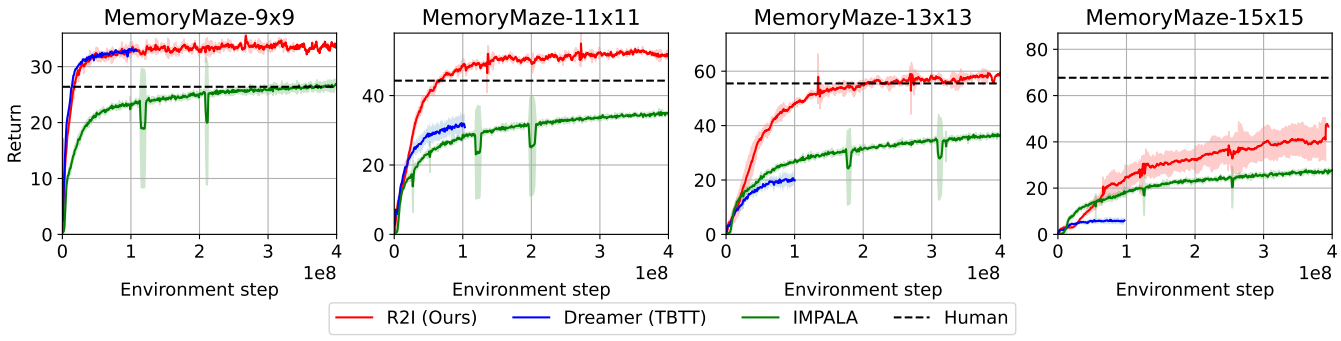

Figure: Scores in Memory Maze after 400M environment steps. R2I outperforms baselines across difficulty levels, becoming the domain's new SOTA. Due to its enhanced computational efficiency, R2I was trained during a fewer number of days compared to Dreamer

Generality of R2I in non-memory domains

- **Evaluating R2I on standard benchmarks**:
  - Tested on Atari and DMC - widely used non-memory RL benchmarks
  - Essential to verify R2I maintains general capabilities
- **Key finding**: R2I performs similarly to DreamerV3
- **Conclusion**: R2I enhances memory capabilities without sacrificing performance on standard tasks

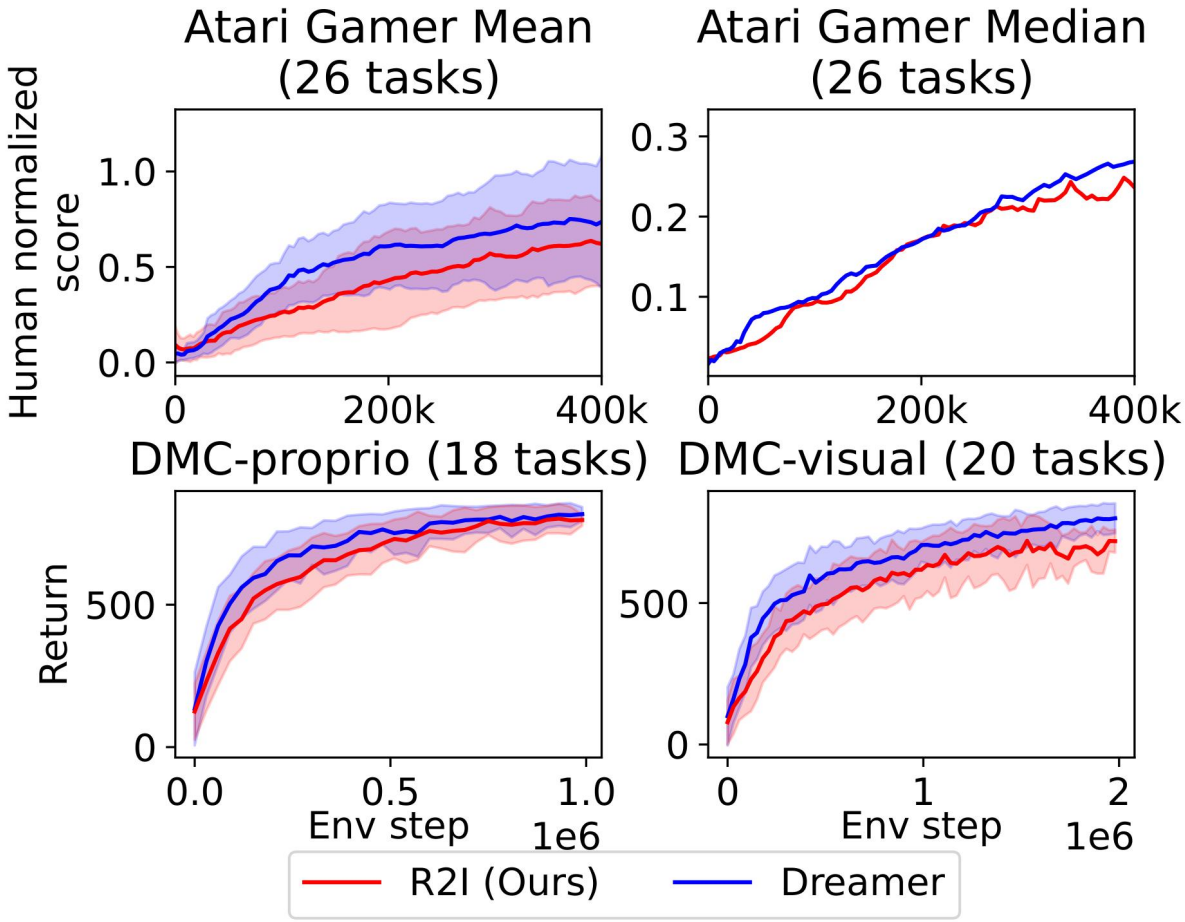

Figure: Performance comparison of R2I against baseline methods on standard