Achieving Precise Control with Slow Hardware: Model-Based Reinforcement Learning for Action Sequence Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Current reinforcement learning (RL) models are often claimed to explain animal behavior. However, they are designed for artificial agents that sense, think, and react much faster than the brain, and they tend to fail when operating under human-like sensory and reaction times. Despite using slow neurons, the brain achieves precise and low-latency control through a combination of predictive and sequence learning. The basal ganglia is hypothesized to learn compressed representations of action sequences, allowing the brain to produce a series of actions for a given input. We present the Hindsight-Sequence-Planner (HSP), a model of the basal ganglia and the prefrontal cortex that operates under "brain-like" conditions: slow information processing with quick sensing and actuation. Our "temporal recall" mechanism is inspired by the prefrontal cortex's role in sequence learning, where the agent uses an environmental model to replay memories at a finer temporal resolution than its processing speed while addressing the credit assignment problem caused by scalar rewards in sequence learning. HSP employs model-based training to achieve model-free control, resulting in precise and efficient behavior that appears low-latency despite running on slow hardware. We test HSP on various continuous control tasks, demonstrating that it not can achieve comparable performance 'human-like' frequencies by relying on significantly fewer observations and actor calls (actor sample complexity).

1 Introduction

Biological and artificial agents must learn behaviors that maximize rewards to thrive in complex environments. Reinforcement learning (RL), a class of algorithms inspired by animal behavior, facilitates this learning process (1). The connection between neuroscience and RL is profound. The Temporal Difference (TD) error, a key concept in RL, effectively models the firing patterns of dopamine neurons in the midbrain (2; 3; 4). Additionally, a longstanding goal of RL algorithms is to match and surpass human performance in control tasks (5; 6; 7; 8; 9; 10)

However, most of these successes are achieved by leveraging large amounts of data in simulated environments and operating at speeds orders of magnitude faster than biological neurons. For example, the default timestep for the Humanoid task in the MuJoCo environment (11) in OpenAI Gym (12) is 15 milliseconds. In contrast, human reaction times range from 150 milliseconds (13) to several seconds for complex tasks (14). When RL agents are constrained to human-like reaction times, even state-of-the-art algorithms struggle to perform in simple environments.

The primary reason for this difficulty is the implicit assumption in RL that the environment and the agent operate at a constant timestep. Consequently, in embodied agents, all components—sensors,

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

compute units, and actuators—are synchronized to operate at the same frequency. Typically, this frequency is limited by the speed of computation in artificial agents (15). As a result, robots often require fast onboard computing hardware (CPU or GPU) to achieve higher control frequencies (16; 17; 18).

In contrast, biological agents achieve precise and seemingly fast control using much slower hardware. This is possible because biological agents effectively decouple the computation frequency from the actuation frequency, allowing them to achieve high actuation frequencies even with slow computational speeds. Consequently, biological agents demonstrate robust, adaptive, and efficient control.

To allow the RL agent to observe and react to changes in the environment quickly, RL algorithms are forced to set a high frequency. Even in completely predictable environments, when the agent learns to walk or move, a small timestep is required to account for the actuation frequency required for the task, but it is not necessary to observe the environment as often or compute new actions as frequently. As a result, RL algorithms suffer from many problems such as low sample efficiency, failure to learn tasks with sparse rewards, jerky control, high compute cost, and catastrophic failure due to missing inputs.

In this work, we propose Hindsight-Sequence-Planner (HSP), a model for sequence learning based on the role of the basal ganglia (BG) and the prefrontal cortex (PFC). Our model learns open-loop control utilizing a slow hardware and low attention, and hence also low energy. Additionally, the algorithm utilizes a simultaneously learned model of the environment during its training but can act without it for fast and cheap inference. We demonstrate the algorithm achieves competitive performance on difficult continuous control tasks while utilizing a fraction of observations and calls to the policy. To the best of our knowledge, HSP is the first to achieve this feat.

2 Neural Basis for Sequence Learning

Unlike artificial RL agents, learning in the brain does not stop once an optimal solution has been found. During initial task learning, brain activity increases as expected, reflecting neural recruitment. However, after training and repetition, activity decreases as the brain develops more efficient representations of the action sequence, commonly referred to as muscle memory (19). This phenomenon is further supported by findings that sequence-specific activity in motor regions evolves based on the amount of training, demonstrating skill-specific efficiency and specialization over time (20).

The neural basis for action sequence learning involves a sophisticated interconnection of different brain regions, each making a distinct contribution:

1. **Basal ganglia** (BG): Action chunking is a cognitive process by which individual actions are grouped into larger, more manageable units or "chunks," facilitating more efficient storage, retrieval, and execution with reduced cognitive load (21). Importantly, this mechanism allows the brain to perform extremely fast and precise sequences of actions that would be impossible if produced individually. The BG plays a crucial role in chunking, encoding entire behavioral action sequences as a single action (22; 21; 23; 24; 25; 26). Dysfunction in the BG is associated with deficits in action sequences and chunking in both animals (27; 28; 29) and humans (30; 31; 21). However, the neural basis for the compression of individual actions into sequences remains poorly understood.

2. **Prefrontal cortex** (PFC): The PFC is critical for the active unbinding and dismantling of action sequences to ensure behavioral flexibility and adaptability (32). This suggests that action sequences are not merely learned through repetition; the PFC modifies these sequences based on context and task requirements. Recent research indicates that the PFC supports memory elaboration (33) and maintains temporal context information (34) in action sequences. The prefrontal cortex receives inputs from the hippocampus.

3. **Hippocampus** (HC) replays neuronal activations of tasks during subsequent sleep at speeds six to seven times faster. This memory replay may explain the compression of slow actions into fast chunks. The replayed trajectories from the HC are consolidated into long-term cortical memories (35; 36). This phenomenon extends to the motor cortex, which replays motor patterns at accelerated speeds during sleep (37).

3 Related Work

3.1 Model-Based Reinforcement Learning

Model-Based Reinforcement Learning (MBRL) algorithms leverage a model of the environment, which can be either learned or known, to enhance RL performance (38). Broadly, MBRL algorithms have been utilized to:

1. Improve Data Efficiency: By augmenting real-world data with model-generated data, MBRL can significantly enhance data efficiency (39; 40; 41).
2. Enhance Exploration: MBRL aids in exploration by using models to identify potential or unexplored states (42; 43; 44).
3. Boost Performance: Better learned representations from MBRL can lead to improved asymptotic performance (45; 46).
4. Transfer Learning: MBRL supports transfer learning, enabling knowledge transfer across different tasks or environments (47; 48).
5. Online Planning: Models can be used for online planning with a single-step policy (49). However, this approach increases model complexity as each online planning step requires an additional call to the model, making it nonviable for energy and computationally constrained agents like the brain and robots.

Compared to online planning, our algorithm maintains a model complexity of zero after training, eliminating the need for any model calls post-training. This significantly reduces the computational and energy requirements, making it more suitable for practical applications in constrained environments. Additionally, the performance of online planning algorithms relies heavily on the accuracy of the model. In contrast, our approach can leverage even an inaccurate model to learn a better-performing policy than online planning, using the same model.

3.2 Macro-Actions

Reinforcement Learning (RL) algorithms that utilize macro-actions demonstrate many benefits, including improved exploration and faster learning (50). However, identifying effective macro-actions is a challenging problem due to the curse of dimensionality, which arises from large action spaces. To address this issue, some approaches have employed genetic algorithms (51) or relied on expert demonstrations to extract macro-actions (52). However, these methods are not scalable and lack biological plausibility.

In contrast, our approach learns macro-actions using the principles of RL, thus requiring little overhead while combining the flexibility of primitive actions with the efficiency of macro-actions.

3.3 Action Repetition and Frame-skipping

To overcome the curse of dimensionality while gaining the benefits of macro-actions, many approaches utilize frame-skipping and action repetition, where macro-actions are restricted to a single primitive action that is repeated. Frame-skipping and action repetition serve as a form of partial open-loop control, where the agent selects a sequence of actions to be executed without considering the intermediate states. Consequently, the number of actions is linear in the number of time steps (53; 54; 55; 56; 57).

For instance, FiGaR (56) shifts the problem of macro-action learning to predicting the number of steps that the outputted action can be repeated. TempoRL (55) improved upon FiGaR by conditioning the number of repetitions on the selected actions. However, none of these algorithms can scale to continuous control tasks with multiple action dimensions, as action repetition forces all actuators and joints to be synchronized in their repetitions, leading to poor performance for longer action sequences.

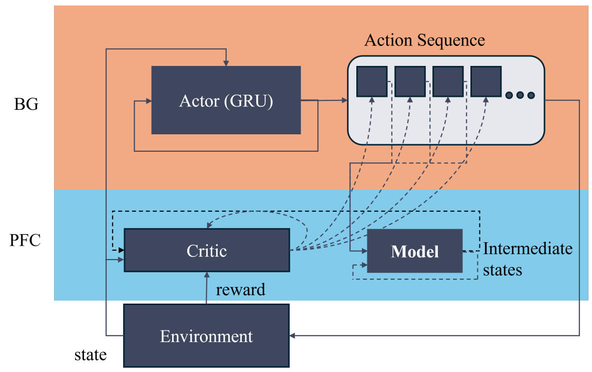

Figure 1: The Hindsight-Sequence-Planner (HSP) model. The HSP takes inspiration from the function of the basal ganglia (BG) (Top/Orange) and the prefrontal cortex (PFC) (Bottom/Blue). We train an actor with a gated recurrent unit that can produce sequences of arbitrary lengths given a single state. This is achieved by utilizing a critic and a model that acts at a finer temporal resolution during training/replay to provide an error signal to each primitive action of the action sequence.

4 Hindsight Sequence Planner

Based on the insights presented in Section 2, we introduce a novel reinforcement learning model capable of learning sequences of actions (macro-actions) by replaying memories at a finer temporal resolution than the action generation, utilizing a model of the environment during training.

Components

The Hindsight-Sequence-Planner (HSP) algorithm learns to plan "in-the-mind" using a model during training, allowing the learned action-sequences to be executed without the need for model-based online planning. This is achieved using an actor-critic setting where the actor and critic operate at different frequencies, representing the observation/computation and actuation frequencies, respectively. Essentially, the critic is only used during training/replay and can operate at any temporal resolution, while the actor is constrained to the temporal resolution of the slowest component in the sensing-compute-actuation loop. Denoting the actor's timestep as t' and the critic's timestep as t, our algorithm includes three components:

$$\begin{aligned}\text{Model} &: s_{t+1} = \mathbf{m}_\phi(s_t, a_t) \\ \text{Critic} &: q_t = \mathbf{q}_\psi(s_t, a_t) \\ \text{Actor} &: a_{t'} = a_t, a_{t'+t}, a_{t'+2t}.. \sim \pi_\omega(s_{t'})\end{aligned} \tag{1}$$

We denote individual actions in the action sequence generated by actor using the notation $\pi_\omega(s_{t'})_t$ to represent the action $a_{t'+t}$

1. **Model**: Learns the dynamics of the environment, predicting the next state s_{t+1} given the current state s_t and primitive action a_t.
2. **Critic**: Takes the same input as the model but predicts the Q-value of the state-action pair.
3. **Actor**: Produces a sequence of actions given an observation at time t'. Observations from the environment can occur at any timestep t or t', where we assume $t' > t$. Specifically, in our algorithm, $t' = Jt$ where $J > 1; J \in \mathbb{Z}$.

Each component of our algorithm is trained in parallel, demonstrating competitive learning speeds.

We follow the Soft-Actor-Critic (SAC) algorithm (58) for learning the actor-critic. Exploration and uncertainty are critical factors heavily influenced by timestep size and planning horizon. Many model-free algorithms like DDPG (59) and TD3 (60) explore by adding random noise to each action during training. However, planning a sequence of actions over a longer timestep can result in additive noise, leading to poor performance during training and exploration. The SAC algorithm addresses this by maximizing the entropy of each action in addition to the expected return, allowing our algorithm to automatically lower entropy for deeper actions farther from the observation.

Learning the Model

The model is trained to minimize the Mean Squared Error of the predicted states. For a trajectory $\tau = (s_t, a_t, s_{t+1})$ drawn from the replay buffer $\mathcal{D}$, the predicted state is taken from $\tilde{s}_{t+1} \sim \mathbf{m}\phi(s_t, a_t)$. The loss function is:

$$\mathcal{L}_\phi = \mathbb{E}_{\tau\sim\mathcal{D}}(\tilde{s}_{t+1} - s_{t+1})^2 \tag{2}$$

For this work, the model is a feed-forward neural network with two hidden layers. In addition to the current model $\mathbf{m}_\phi$, we also maintain a target model $\mathbf{m}_{\phi^-}$ that is the exponential moving average of the current model.

Learning Critic

The critic is trained to predict the Q-value of a given state-action pair $\tilde{q}_t = \mathbf{q}_\psi(s_t, a_t)$ using the target value from the modified Bellman equation:

$$\hat{q}_t = r_t + \gamma\mathbb{E}_{a_{t+1}\sim\pi_\omega(s_{t+1})_0}[\mathbf{q}_{\psi^-}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\omega(a_{t+1}|s_{t+1})] \tag{3}$$

Here, $\mathbf{q}_{\psi^-}$ is the target critic, which is the exponential moving average of the critic. Following the SAC algorithm, we train two critics and use the minimum of the two $\mathbf{q}_{\psi^-}$ values to train the current critics. The loss function is:

$$\mathcal{L}_\psi = \mathbb{E}_{\tau\sim\mathcal{D}}[(\tilde{q}_{tk} - \hat{q}_t)^2]\forall k \in 1, 2 \tag{4}$$

Both critics are feed-forward neural networks with two hidden layers.

Learning Policy

The HSP policy utilizes two hidden layers followed by a Gated-Recurrent-Unit (GRU) (61) that takes as input the previous action in the action sequence, followed by two linear layers that output the mean and standard deviation of the Gaussian distribution of the action. This design allows the policy to produce action sequences of arbitrary length given a single state and the last action.

A naive approach to training a sequence of actions would be to augment the action space to include all possible actions of the sequence length. However, this quickly leads to the curse of dimensionality, as each sequence is considered a unique action, dramatically increasing the policy's complexity. Additionally, such an approach ignores the temporal information of the action sequence and faces the difficult problem of credit assignment, with only a single scalar reward for the entire action sequence.

To address these problems, we use different temporal scales for the actor and critic. The critic assigns value to each segment of the action sequence, bypassing the credit assignment problem caused by the single scalar reward. However, using collected transitions to train the action sequence is impractical, as changing the first action in the sequence would render all future states inaccurate. Thus, the model populates intermediate states, which the critic then uses to assign value to each primitive action in the sequence.

Therefore, given a trajectory $\tau = (a_{t-1}, s_t, a_t, s_{t+1})$, we first produce the J-step action sequence using the policy: $\tilde{a}_{t:t+J} = \pi_\phi(s_t)$. We then iteratively apply the target model to get the intermediate states $\tilde{s}_{t+1:t+J-1}$. Finally, we use the critic to calculate the loss for the actor as follows:

$$\mathcal{L}_\omega = \mathbb{E}_{\tau\sim\mathcal{D}}\left[\alpha \log \pi_\omega(\tilde{a}_t|s_t) - \mathbf{q}_\psi(s_t, \tilde{a}_t) + \sum_{j=1}^{J} \alpha \log \pi_\omega(\tilde{a}_{t+j}|\tilde{s}_{t+j}) - \mathbf{q}_\psi(\tilde{s}_{t+j}, \tilde{a}_{t+j})\right] \tag{5}$$

5 Experiments

Overview

We evaluate our HSP approach on several continuous control tasks, comparing it against the SAC baseline and the TempoRL algorithm (55). Our focus is on environments with multi-dimensional actions, ranging from the simple LunarLanderContinuous (2 action dimensions) to the complex Humanoid environment (17 action dimensions). This allows us to highlight the benefits of HSP over traditional action repetition approaches. We utilize the OpenAI gym (62) implementation of the MuJoCo environments (11).

Experiemental Setup

We train HSP with four different action sequence lengths (ASL), $J = 2, 4, 8, 16$, referred to as HSP-J. During training, HSP is evaluated based on its J value, processing states only after every J actions. All hyperparameters are identical between HSP and SAC, except for the actor update frequency: HSP updates the actor every 4 steps, while SAC updates every step. Thus, SAC has four more actor update steps compared to HSP. Additionally, HSP learns a model in parallel with the actor and critic.

Learning Curves

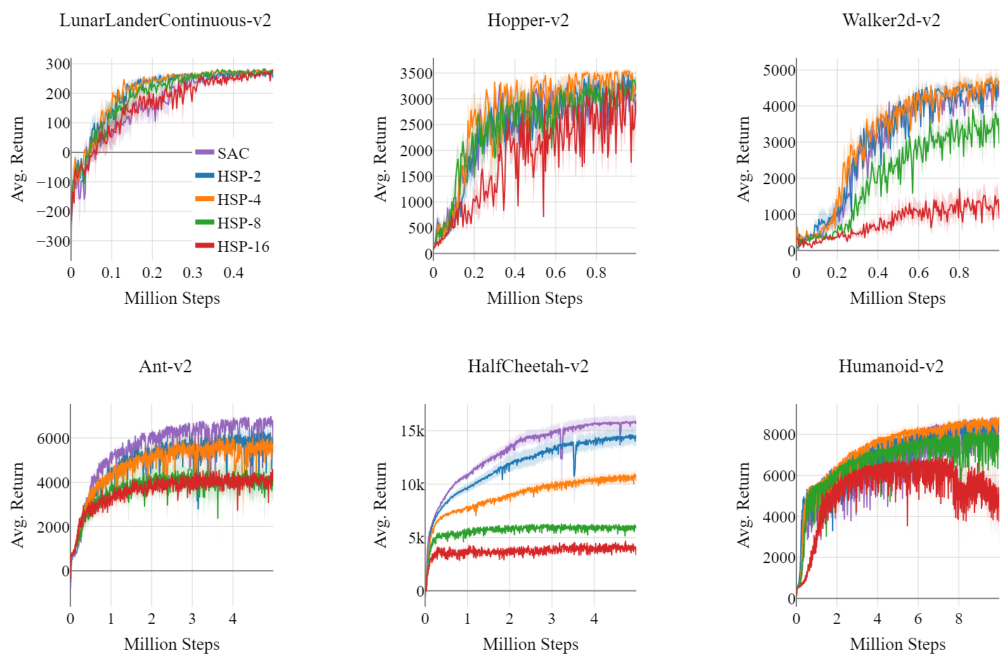

Figure 2: Learning curves of HSP-J and Soft-Actor Critic (SAC) (58) over continuous control tasks. HSP and SAC are evaluated under different settings: SAC receives input after every primitive action, while HSP receives input after J primitive actions. Yet it demonstrates competitive performance on all environments, even outperforming SAC on LunarLander, Hopper and Humanoid environments. HSP demonstrates stable learning even with the added model and generative replay training. All curves are averaged over 5 trials, with shaded regions representing standard error.

Figure 6 presents the learning curves of HSP and SAC across six continuous control tasks. We observe that HSP outperforms SAC in four out of six tasks (excluding Ant and HalfCheetah). Notably, HSP-16 achieves competitive performance on LunarLander and Hopper tasks, showcasing the algorithm's capability to learn long action sequences from scratch. Surprisingly, HSP also outperforms SAC in the Humanoid environment with fewer inputs and actor updates while concurrently learning a model, demonstrating the efficacy of the algorithm on environments with higher action dimensions.

Action Sequence Length (ASL) Performance

Learning curves alone do not fully capture HSP's performance and benefits. For instance, HSP-16 shows poor performance on Ant in the learning curve, yet it demonstrates competitive performance when tested on shorter action sequences. Figure 3 presents the performance of trained algorithms across different action sequence lengths (ASL).

We select the largest J that shows competitive performance (greater than 75% of the SAC when evaluated on primitive actions) for each environment and test it for sequence lengths up to 30. For SAC and HSP, we fix the length of action sequences while TempoRL is designed to dynamically pick the best ASL, therefore we report the avg. action sequence length for TempoRL. HSP demonstrates

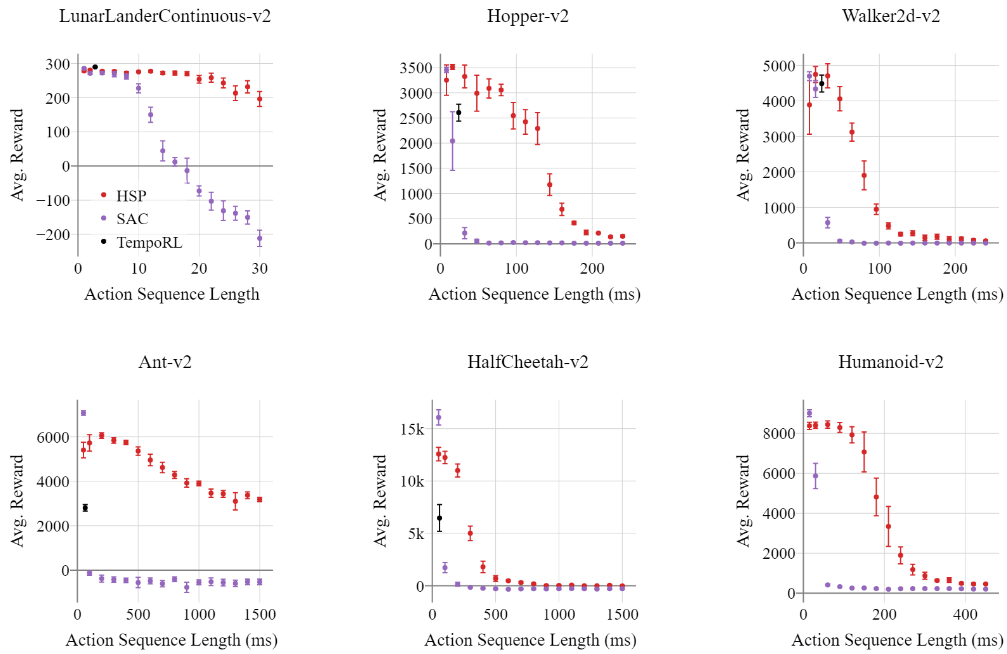

Figure 3: Performance of HSP, SAC, and TempoRL (55) at different Action Sequence Lengths (ASL). SAC and TempoRL repeat the same action for the duration, while HSP can perform a sequence of actions. Since it implements dynamic action repetition, we present the average ASL for TempoRL instead of a range of ASL. HSP demonstrates robust performance even at human-like reaction times (>150ms). All markers are averaged over 5 trials, with the error bars representing standard error. Going from left to right then top to bottom, the selected training ASL J for HSP are: 16, 16, 4, 16, 4, 8.

competitive performance on longer action sequences, approaching human-like reaction times in some environments. Unlike SAC, which fails with action sequences of 2 or 3, HSP shows a gradual degradation of performance. Additionally, HSP generalizes well in environments like LunarLander and Ant, even though the actor is trained only on sequence lengths of 16.

Comparing HSP to TempoRL, we find that TempoRL prefers shorter repetitions and struggles in more difficult environments. TempoRL does not incentivize longer actions and suffers from the curse of dimensionality to some extent, as it needs to learn the number of repetitions for each unique state-action pair. Furthermore, action repetitions are not suitable for multi-dimensional actions, as they force synchronized repetition across all actuators resulting in poor performance in environments with high action dimensions like Ant and HalfCheetah environments.

Comparison to Model-based Online Planning

In addition to action repetition, model-based online planning is another approach that allows the RL agent to reduce its observational frequency. However, it often requires a highly accurate model of the environment and incurs increased model complexity due to the use of the model during control. Despite these challenges, comparing HSP to model-based online planning is essential since it is useful when the actor cannot produce long sequences of actions and does not require the hyper-parameter J. With access to an accurate model of the environment, the agent's performance might generalize to arbitrary ASL.

Since HSP incorporates a model of the environment that is learned in parallel, we compare the performance of the HSP actor utilizing the actor-generated action sequences against model-based online planning, where the actor produces only a single action between each simulated state.

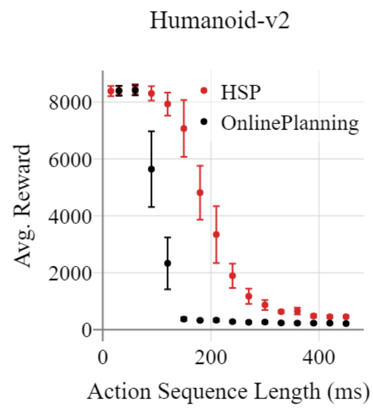

Figure 4: Performance of HSP and model based online planning on different ASL. Both HSP and Online Planning utilize the same actor and model. HSP utilizes the actor to generate a sequence of actions while online planning utilizes the actor and the model to generate a sequence of actions. The same model is used to train the HSP action sequences. Yet, we find that while the model is not accurate enough to sustain performance for longer sequences, it can train the actor to produce accurate action sequences.

Figure 4 shows the performance of online planning using the model in HSP versus the action sequences generated by the HSP policy. We see that HSP can learn action sequences that perform better than model-based online planning using the same model. Thus, HSP can leverage inaccurate models to learn accurate action sequences, further reducing the required computational complexity during training. We hypothesize that this superior performance is due to the fact that the actor learns a J-step action sequence concurrently, while online planning only produces one action at a time. Consequently, HSP is able to learn and produce long, coherent action sequences, whereas single-step predictions tend to drift, similar to the "hallucination" phenomenon observed in transformer-based language models.

Generative Replay in Latent Space

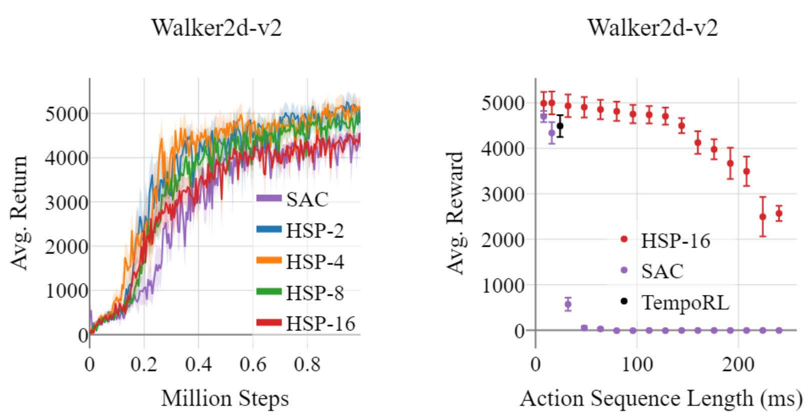

Figure 5: Left: Learning curve of HSP with latent state-space on the Walker2d-v2 environment. Right: Performance of latent HSP-16 on different ASL, compared to SAC and TempoRL. Utilizing a latent representation for state space is especially beneficial for the Walker2d environment so that it outperforms SAC even when training upto sequence lengths of $J = 16$.

Previous studies have shown that generative replay benefits greatly from latent representations (63). Recently, Simplified Temporal Consistency Reinforcement Learning (TCRL) (64) demonstrated that learning a latent state-space improves not only model-based planning but also model-free RL algorithms. Building on this insight, we introduced an encoder to encode the observations in our algorithm. We provide the complete implementation details in the Appendix.

We did not observe any benefits of using the encoder and temporal consistency for HSP in most environments (results in the appendix). However, for the Walker environment, utilizing the latent space for generative replay significantly improved performance, making it competitive even at 16 steps (128ms) (Figure 5).

6 Discussion, Limitations and Future Work

We introduce the Hindsight-Sequence-Planner (HSP) algorithm, a biologically plausible model for sequence learning. It represents a significant step towards achieving robust control at brain-like speeds. The key contributions of HSP include its ability to generate long sequences of actions from a single state, its resilience to reduced input frequency, and its lower computational complexity per primitive action.

The current RL framework encourages synchrony between the environment and the components of the agent. However, the brain utilizes components that act at different frequencies and yet is capable of robust and accurate control. HSL provides an approach to reconcile this difference between neuroscience and RL, while remaining competitive on current RL benchmarks. HSP offers substantial benefits over traditional RL algorithms, particularly in the context of autonomous agents such as self-driving cars and robots. By enabling operation at slower observational frequencies and providing a gradual decay in performance with reduced input frequency, HSP addresses critical issues related to sensor failure and occlusion, and energy consumption. Additionally, HSP generates long sequences of actions from a single state, which can enhance the explainability of the policy and provide opportunities to override the policy early in case of safety concerns. HSP also learns a latent representation of the action sequence, which could be used in the future to interface with large language models for multimodal explainability and even hierarchical reinforcement learning and transfer learning.

Limitations

Despite its advantages, HSP has some limitations. It shows slightly reduced performance in the Ant and HalfCheetah environments, which we believe can be mitigated through improved models and hyperparameter tuning. HSP also requires more computational resources during training due to the parallel training of an environment model and introduces more hyperparameters, particularly the training ASL (J). In this work, we do not optimize the neural network architecture of the actor to reduce the compute, as a result, the total compute per primitive action is still larger than SAC. However, we believe producing a sequence of actions will be more efficient than producing a single primitive action per state after optimization. Larger ASL values may not perform well in stochastic environments. Moreover, HSP currently uses a constant ASL, but ideally, the ASL should adapt based on the environment's predictability.

Future Work

We believe the HSP model contributes to both artificial agents and the study of biological control. Future work will incorporate biological features like attention mechanisms and knowledge transfer. Additionally, HSP can benefit from existing Model-Based RL approaches as it naturally learns a model of the world. In deterministic environments, a capable agent should achieve infinite horizon control for tasks like walking and hopping from a single state. This is an important research direction that is currently underexplored, as many environments are partially observable or have some degree of stochasticity. Current approaches rely on external information at every state, which increases energy consumption and vulnerability to adversarial or missing inputs. Truly autonomous agents will need to impl ement multiple policies simultaneously, and simple tasks like walking can be performed without input states if learned properly. Our future work will focus on extending the action sequence horizon until deterministic tasks can be performed using a single state and implementing a mechanism to dynamically pick the action sequence horizon based on context and predictability of the state. Serotonin is an important neuromodulator that has been demonstrated to signal the availability of time and resources in the brain to enable the decision on the planning horizon and the use of compute (65). In the future, we hope to introduce a mechanism to replicate the effect of serotonin in HSP.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward," *Science*, vol. 275, pp. 1593–1599, 1997.

[3] W. Schultz, "Neuronal reward and decision signals: From theories to data," *Physiological Reviews*, vol. 95, pp. 853–951, 7 2015.

[4] J. Y. Cohen, S. Haesler, L. Vong, B. B. Lowell, and N. Uchida, "Neuron-type-specific signals for reward and punishment in the ventral tegmental area," *Nature 2012 482:7383*, vol. 482, pp. 85–88, 1 2012.

[5] OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with large scale deep reinforcement learning," 12 2019.

[6] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature 2020 588:7839*, vol. 588, pp. 604–609, 12 2020.

[7] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature 2023 620:7976*, vol. 620, pp. 982–987, 8 2023.

[8] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, L. Gilpin, P. Khandelwal, V. Kompella, H. C. Lin, P. MacAlpine, D. Oller, T. Seno, C. Sherstan, M. D. Thomure, H. Aghabozorgi, L. Barrett, R. Douglas, D. Whitehead, P. Dürr, P. Stone, M. Spranger, and H. Kitano, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature 2022 602:7896*, vol. 602, pp. 223–228, 2 2022.

[9] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature 2019 575:7782*, vol. 575, pp. 350–354, 10 2019.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature 2015 518:7540*, vol. 518, pp. 529–533, 2 2015.

[11] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.

[12] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis, "Gymnasium," Mar. 2023.

[13] A. Jain, R. Bansal, A. Kumar, and K. Singh, "A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students," *International journal of applied and basic medical research*, vol. 5, no. 2, pp. 124–127, 2015.

[14] R. Limpert, *Brake design and safety*. SAE international, 2011.

[15] B. Katz, J. D. Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 6295–6301, 5 2019.

[16] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *International Journal of Robotics Research*, vol. 43, pp. 572–587, 4 2024.

[17] Q. Li, G. Dong, R. Qin, J. Chen, K. Xu, and X. Ding, "Quadruped reinforcement learning without explicit state estimation," in *2022 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1989–1994, IEEE, 2022.

[18] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, *et al.*, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," *arXiv preprint arXiv:2304.13653*, 2023.

[19] T. Wiestler and J. Diedrichsen, "Skill learning strengthens cortical representations of motor sequences," *Elife*, vol. 2, p. e00801, 2013.

[20] N. F. Wymbs and S. T. Grafton, "The human motor system supports sequence-specific representations over multiple training-dependent timescales," *Cerebral cortex*, vol. 25, no. 11, pp. 4213–4225, 2015.

[21] N. Favila, K. Gurney, and P. G. Overton, "Role of the basal ganglia in innate and learned behavioural sequences," *Reviews in the Neurosciences*, vol. 35, no. 1, pp. 35–55, 2024.

[22] X. Jin, F. Tecuapetla, and R. M. Costa, "Basal ganglia subcircuits distinctively encode the parsing and concatenation of action sequences," *Nature neuroscience*, vol. 17, no. 3, pp. 423–430, 2014.

[23] X. Jin and R. M. Costa, "Shaping action sequences in basal ganglia circuits," *Current opinion in neurobiology*, vol. 33, pp. 188–196, 2015.

[24] G. S. Berns and T. J. Sejnowski, "How the basal ganglia make decisions," in *Neurobiology of decision-making*, pp. 101–113, Springer, 1996.

[25] G. S. Berns and T. J. Sejnowski, "A computational model of how the basal ganglia produce sequences," *Journal of cognitive neuroscience*, vol. 10, no. 1, pp. 108–121, 1998.

[26] E. Garr, "Contributions of the basal ganglia to action sequence learning and performance," *Neuroscience & Biobehavioral Reviews*, vol. 107, pp. 279–295, 2019.

[27] A. J. Doupe, D. J. Perkel, A. Reiner, and E. A. Stern, "Birdbrains could teach basal ganglia research a new song," *Trends in neurosciences*, vol. 28, no. 7, pp. 353–363, 2005.

[28] X. Jin and R. M. Costa, "Start/stop signals emerge in nigrostriatal circuits during sequence learning," *Nature*, vol. 466, no. 7305, pp. 457–462, 2010.

[29] M. Matamales, Z. Skrbis, M. R. Bailey, P. D. Balsam, B. W. Balleine, J. Götz, and J. Bertran-Gonzalez, "A corticostriatal deficit promotes temporal distortion of automatic action in ageing," *ELife*, vol. 6, p. e29908, 2017.

[30] J. G. Phillips, E. Chiu, J. L. Bradshaw, and R. Iansek, "Impaired movement sequencing in patients with huntington's disease: a kinematic analysis," *Neuropsychologia*, vol. 33, no. 3, pp. 365–369, 1995.

[31] L. Boyd, J. Edwards, C. Siengsukon, E. Vidoni, B. Wessel, and M. Linsdell, "Motor sequence chunking is impaired by basal ganglia stroke," *Neurobiology of learning and memory*, vol. 92, no. 1, pp. 35–44, 2009.

[32] C. F. Geissler, C. Frings, and B. Moeller, "Illuminating the prefrontal neural correlates of action sequence disassembling in response–response binding," *Scientific Reports*, vol. 11, no. 1, p. 22856, 2021.

[33] M. A. Immink, M. Pointon, D. L. Wright, and F. E. Marino, "Prefrontal cortex activation during motor sequence learning under interleaved and repetitive practice: a two-channel near-infrared spectroscopy study," *Frontiers in Human Neuroscience*, vol. 15, p. 644968, 2021.

[34] D. Shahnazian, M. Senoussi, R. M. Krebs, T. Verguts, and C. B. Holroyd, "Neural representations of task context and temporal order during action sequence execution," *Topics in Cognitive Science*, vol. 14, no. 2, pp. 223–240, 2022.

[35] M. C. Zielinski, W. Tang, and S. P. Jadhav, "The role of replay and theta sequences in mediating hippocampal-prefrontal interactions for memory and cognition," *Hippocampus*, vol. 30, no. 1, pp. 60–72, 2020.

[36] P. Malerba, K. Tsimring, and M. Bazhenov, "Learning-induced sequence reactivation during sharp-wave ripples: a computational study," in *Advances in the Mathematical Sciences: AWM Research Symposium, Los Angeles, CA, April 2017*, pp. 173–204, Springer, 2018.

[37] D. B. Rubin, T. Hosman, J. N. Kelemen, A. Kapitonava, F. R. Willett, B. F. Coughlin, E. Halgren, E. Y. Kimchi, Z. M. Williams, J. D. Simeral, *et al.*, "Learned motor patterns are replayed in human motor cortex during sleep," *Journal of Neuroscience*, vol. 42, no. 25, pp. 5007–5020, 2022.

[38] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, *et al.*, "Model-based reinforcement learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.

[39] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 10674–10681, 2021.

[40] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *Advances in neural information processing systems*, vol. 32, 2019.

[41] J. Wang, W. Li, H. Jiang, G. Zhu, S. Li, and C. Zhang, "Offline reinforcement learning with reverse model-based imagination," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29420–29432, 2021.

[42] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*, pp. 2778–2787, PMLR, 2017.

[43] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," *arXiv preprint arXiv:1507.00814*, 2015.

[44] N. Savinov, A. Raichuk, R. Marinier, D. Vincent, M. Pollefeys, T. Lillicrap, and S. Gelly, "Episodic curiosity through reachability," *arXiv preprint arXiv:1810.02274*, 2018.

[45] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[46] S. Levine and V. Koltun, "Guided policy search," in *International conference on machine learning*, pp. 1–9, PMLR, 2013.

[47] A. Zhang, H. Satija, and J. Pineau, "Decoupling dynamics and reward for transfer learning," *arXiv preprint arXiv:1804.10689*, 2018.

[48] R. Sasso, M. Sabatelli, and M. A. Wiering, "Multi-source transfer learning for deep model-based reinforcement learning," *arXiv preprint arXiv:2205.14410*, 2022.

[49] A. Fickinger, H. Hu, B. Amos, S. Russell, and N. Brown, "Scalable online planning via reinforcement learning fine-tuning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16951–16963, 2021.

[50] A. McGovern, R. S. Sutton, and A. H. Fagg, "Roles of macro-actions in accelerating reinforcement learning," 1997.

[51] Y.-H. Chang, K.-Y. Chang, H. Kuo, and C.-Y. Lee, "Reusability and transferability of macro actions for reinforcement learning," *ACM Transactions on Evolutionary Learning and Optimization*, vol. 2, no. 1, pp. 1–16, 2022.

[52] H. Kim, M. Yamada, K. Miyoshi, T. Iwata, and H. Yamakawa, "Reinforcement learning in latent action sequence space," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5497–5503, IEEE, 2020.

[53] S. Kalyanakrishnan, S. Aravindan, V. Bagdawat, V. Bhatt, H. Goka, A. Gupta, K. Krishna, and V. Piratla, "An analysis of frame-skipping in reinforcement learning," *ArXiv*, vol. abs/2102.03718, 2021.

[54] A. Srinivas, S. Sharma, and B. Ravindran, "Dynamic action repetition for deep reinforcement learning," in *AAAI*, 2017.

[55] A. Biedenkapp, R. Rajan, F. Hutter, and M. Lindauer, "Temporl: Learning when to act," in *International Conference on Machine Learning*, pp. 914–924, PMLR, 2021.

[56] S. Sharma, A. Srinivas, and B. Ravindran, "Learning to repeat: Fine grained action repetition for deep reinforcement learning," *ArXiv*, vol. abs/1702.06054, 2017.

[57] H. Yu, W. Xu, and H. Zhang, "Taac: Temporally abstract actor-critic for continuous control," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29021–29033, 2021.

[58] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[59] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[60] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, pp. 1587–1596, PMLR, 2018.

[61] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[62] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[63] G. M. Van de Ven, H. T. Siegelmann, and A. S. Tolias, "Brain-inspired replay for continual learning with artificial neural networks," *Nature communications*, vol. 11, no. 1, p. 4069, 2020.

[64] Y. Zhao, W. Zhao, R. Boney, J. Kannala, and J. Pajarinen, "Simplified temporal consistency reinforcement learning," in *International Conference on Machine Learning*, pp. 42227–42246, PMLR, 2023.

[65] K. Doya, K. W. Miyazaki, and K. Miyazaki, "Serotonergic modulation of cognitive computations," *Current Opinion in Behavioral Sciences*, vol. 38, pp. 116–123, 2021.

[66] D. Yarats and I. Kostrikov, "Soft actor-critic (sac) implementation in pytorch." `https://github.com/denisyarats/pytorch_sac`, 2020.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope.

   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: Limitations are provided in Section 6

   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: The paper does not include theoretical 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: We provide a complete algorithm and list of hyperparameters in the appendix. Additionally we also released the code and trained models.

   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: We include code to reproduce the 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: We provide code, and list of all hyperparameters in appendix

   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: Error bars are reported for all results presented.

   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: A description of compute resources used is provided in the appendix

   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: the research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   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: [Yes]

    Justification: See section 6

    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: the paper poses no such risks.

    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 cite original papers for each algorithm and envrionment used

    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: [Yes]

    Justification: The documentation for the released code is provided

    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: paper does not involve crowdsourcing nor research with 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: the paper does not involve crowdsourcing nor research with human subjects.

    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.

A Appendix / supplemental material

A.1 HSP Algorithm

Algorithm 1: Hindsight Sequence Planner

Input: $\phi, \psi_1, \psi_2, \omega$. Initial parameters

$\bar{\phi} \leftarrow \phi, \bar{\psi}_1 \leftarrow \psi_1, \bar{\psi}_2 \leftarrow \psi_2$; `// Initialize target network weights`
$D \leftarrow \emptyset$; `// Initialize an empty replay pool`
for *each iteration* **do**
  $\{a_t, a_{t+1}, \ldots, a_{t+J-1}\} \sim \pi_\omega(\{a_t, a_{t+1}, \ldots, a_{t+J-1}\}|s_t)$; `// Sample action sequence from the policy`
  for *each action a_t in the sequence* **do**
    $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$; `// Sample transition from the environment`
    $D \leftarrow D \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$; `// Store transition in the replay pool`
  end
  for *each gradient step* **do**
    $\phi \leftarrow \phi - \lambda_{\mathbf{m}} \nabla_\phi \mathcal{L}_\phi$; `// Update the model parameters`
    for $i \in \{1, 2\}$ **do**
      $\psi_i \leftarrow \psi_i - \lambda_Q \nabla_{\psi_i} \mathcal{L}_{\psi_i}$; `// Update the Q-function parameters`
    end
    $\{a_t, a_{t+1}, \ldots, a_{t+J-1}\} \sim \pi_\omega(\{a_t, a_{t+1}, \ldots, a_{t+J-1}\}|s_t)$; `// Sample action sequence from the policy`
    if *iteration mod actor_update_frequency == 0* **then**
      for $j \in \{1, \ldots, J\}$ **do**
        $s_{j+1} \sim \mathbf{m}_{\bar{\phi}}(s_{j+1}|s_j, a_j)$; `// Sample transition from the target model`
      end
      $\phi \leftarrow \omega - \lambda_\pi \nabla_\omega L_\omega$; `// Update policy weights`
    end
    $\alpha \leftarrow \alpha - \lambda \nabla_{\hat{\alpha}} \mathcal{L}(\alpha)$; `// Adjust temperature`
    for $i \in \{1, 2\}$ **do**
      $\bar{\psi}_i \leftarrow \tau \psi_i + (1 - \tau)\bar{\psi}_i$; `// Update target network weights`
    end
    $\bar{\phi} \leftarrow \tau \phi + (1 - \tau)\bar{\phi}$; `// Update target model weights`
  end
end

Output: $\phi, \psi_1, \psi_2, \omega$; `// Optimized parameters`

Hyperparameters

The table below lists the hyperparameters that are common between every environment used for all our experiments for the SAC and HSP algorithms:

A.2 Implementation Details

Due to its added complexity during training, HSP requires longer wall clock time for training when compared to SAC. We performed a minimal hyperparameter search over the actor update frequency parameter on the Hopper environment (tested values: 1, 2, 4, 8, 16). All the other hyperparamters were picked to be equal to the SAC implementation. We also did not perform a hyerparameter search over the size of GRU for the actor. It was picked to have the same size as the hidden layers of the feed forward network of the actor in SAC. The neural network for the model was also picked to have the same architecture as the actor from SAC, thus it has two hidden layers with 256 neurons. Similarly the encoder for the latent HSP implementation was also picked to have the same architecture. For the latent HSP implementation we also add an additional replay buffer to store transitions of length 5, to implement the temporal consistency training for the model. This was done for simplicity of the implementation, and it can be removed since it is redundant to save memory.

Hyperparameter	Value	description
Hidden Layer Size	256	Size of the hidden layers in the feed forward networks of Actor, Critic, Model and Encoder networks
Updates per step	1	Number of learning updates per one step in the environment
Target Update Interval	1	Inverval between each target update
γ	0.99	Discount Factor
τ	0.005	Update rate for the target networks (Critic and Model)
Learning Rate	0.0003	Learning rate for all neural networks
Replay Buffer Size	10^6	Size of the replay buffer
Batch Size	256	Batch size for learning
Start Time-steps	10000	Initial number of steps where random policy is followed

Table 1: List of Common hyperparameters

Environment	max Timestep	Eval frequency
LunarLanderContinuous-v2	500000	2500
Hopper-v2	1000000	5000
Walker2d-v2	1000000	5000
Ant-v2	5000000	5000
HalfCheetah-v2	5000000	5000
Humanoid-v2	10000000	5000

Table 2: List of environment-specific hyperparameters

All experiments were performed on a GPU cluster the Nvidia 1080ti GPUs. Each run was performed using a single GPU, utilizing 8 CPU cores of Intel(R) Xeon(R) Silver 4116 (24 core) and 16GB of memory.

We utilize the pytorch implementation of SAC (`https://github.com/denisyarats/pytorch_sac`) (66).

A.3 Latent State Space Experiments

Following the TCRL implementation, we use two encoders: an online encoder $\mathbf{e}_\theta$ and a target encoder $\mathbf{e}_{\theta^-}$, which is the exponential moving average of the online encoder:

$$\text{Encoder} : e_t = \mathbf{e}_\theta(s_t) \tag{6}$$

Thus, the model predicts the next state in the latent space. Additionally, we introduce multi-step model prediction for temporal consistency. Following the TCRL work, we use a cosine loss for model prediction. The model itself predicts only a single step forward, but we enforce temporal consistency by rolling out the model H-steps forward to predict $\tilde{e}_{t+1:t+1+H}$.

Specifically, for an H-step trajectory $\tau = (z_t, a_t, z_{t+1})_{t:t+H}$ drawn from the replay buffer $\mathcal{D}$, we use the online encoder to get the first latent state $e_t = \mathbf{e}_\theta(o_t)$. Then conditioning on the sequence of actions $a_{t:t+H}$, the model is applied iteratively to predict the latent states $\tilde{e}_{t+1} = \mathbf{m}_\phi(\tilde{e}_t, a_t)$. Finally, we use the target encoder to calculate the target latent states $\hat{e}_{t+1:t+H+1} = \mathbf{e}_{\theta^-}(o_{t+1:t+1+H})$. The Loss function is defined as:

$$\mathcal{L}_{\theta,\phi} = \mathbb{E}_{\tau\sim\mathcal{D}}\left[\sum_{h=0}^{H} -\gamma^h \left(\frac{\tilde{e}_{t+h}}{||\tilde{e}_{t+h}||_2}\right)^T \left(\frac{\hat{e}_{t+h}}{||\hat{e}_{t+h}||_2}\right)\right] \tag{7}$$

We set $H = 5$ for our experiments. Both the encoder and the model are feed-forward neural networks with two hidden layers.

Here, we provide complete learning curves for the latent space HSP.

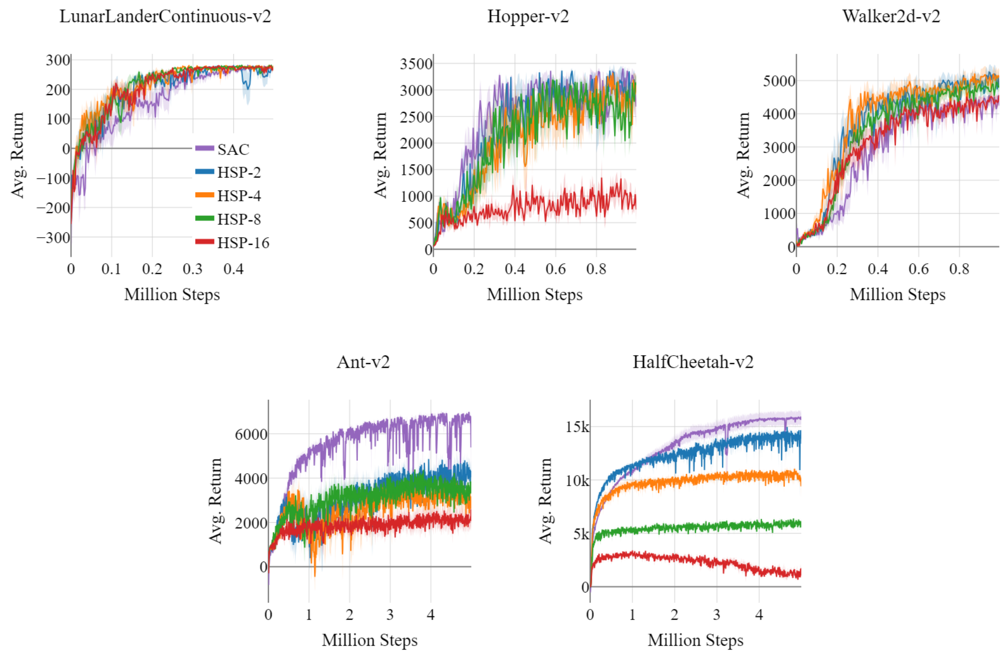

Figure 6: Learning curves of Latent HSP-n and Soft-Actor Critic (SAC) over continuous control tasks.