Extended Abstract

Motivation Long-horizon goal-conditioned reinforcement learning (GCRL) is hard because of sparse rewards, compounding errors, and credit assignment. As the horizon grows, a policy must decide both where to go and how to get there, so the action search space explodes while the lone goal reward is too weak and delayed to shape behavior. Early mistakes accumulate, pushing the agent into unseen, unrecoverable states, and distant outcomes make credit assignment hard. Planning in raw observation and action spaces entangles long-term intent with short-term execution. We instead learn a structured latent space and combine it with world models to decompose long-horizon control into manageable steps, bridging the gap between distant goals and the low-level actions needed to reach them, which we study on AntMaze.

Method Our method does all long-horizon planning in a learned latent space, giving the policy short-horizon subproblems. A masked, dual-input encoder embeds full states and goals into a shared latent space, over which we build a graph from observed latent transitions. Shortest-path search produces a sequence of latent subgoals between any start and goal, and a learned world model imagines waypoints across disjoint components when no path exists. A goal-conditioned PPO policy is conditioned on the current state, the next latent subgoal, and a latent goal, while the planner advances and replans as the agent progresses.

Implementation The encoder is a masked MLP mapping proprioception, position, and a mask flag to a latent, where the flag masks either the proprioception or the goal. It is trained on an offline dataset with an InfoNCE loss over temporally nearby states, forward- and inverse-dynamics losses, and reconstruction and state-goal alignment terms. We summarize the latent space with K-means centroids as graph nodes, connect them by observed transitions, and run Dijkstra for shortest paths. Disconnected pairs are bridged by world models that predict the latent trajectory between current and goal states. At inference the planner encodes the state and goal, snaps them to the nearest nodes, generates the subgoal sequence, and advances to the next subgoal within a distance threshold, replanning when the goal changes or the episode resets.

Results We evaluate each configuration on 256 episodes of antmaze by the mean goals reached. Every raw-state policy reaches 0.51 to 0.65 goals and clears at least one goal in 34 to 46% of episodes, while every latent-state policy collapses to near zero regardless of how the goal or subgoal is encoded. We believe this reflects an optimization and exploration failure under sparse reward rather than a broken representation: a decoder recovers state from the latent, so the information is present but the policy fails to exploit it. Among raw-state methods, graph planning with inverse-model imagination is strongest at 0.65 goals (46.1%), narrowly ahead of the baseline at 0.62 (44.1%). Planning thus yields a marginal gain, and only with raw-state input, leaving trainable latent-state control as the key bottleneck.

Discussion The main finding is that a good representation is not necessarily a usable one: the latent decodes accurately to state and yields a graph of helpful subgoals, yet the policy still cannot learn to act in it. Where the structure is useful is with the imagined subgoals, which fill in waypoints across disconnected graph regions and break a long, sparsely rewarded reach into shorter completable segments, so inverse-imagination planning outperforms the baseline. However, the encoder, graph, and world models are trained offline and frozen, which caps how much the structure can help. Added structure only helps if the agent can use it, and we believe end-to-end training would yield much larger gains where gradients flow from the PPO objective directly into the encoder.

Conclusion We study whether long-horizon goal-conditioned RL can be made easier by moving planning and learning into a learned latent space. The latent we learn is representative of the original data and its imagined subgoals give a small edge over baselines, but acting directly in the latent space fails, and the modular pipeline limits how much the structure can help. We read this as evidence that representation, planning, and control should be learned together rather than in sequence, and we believe end-to-end training is the most promising path forward.

MIRAGE: Model Imagined Reachability for Augmented Graph Expansion

Samrat Sahoo
Department of Computer Science
Stanford University
samrat@stanford.edu

Abhinav Sattiraju
Department of Computer Science
Stanford University
asattira@stanford.edu

Abstract

Long-horizon goal-conditioned reinforcement learning is hard because sparse rewards, compounding errors, and difficult credit assignment make it hard for a naive policy to map distant goals to low-level actions. We move long-horizon planning into a learned latent space: a masked, dual-input encoder embeds states and goals into a shared latent over which we build a graph from observed transitions, shortest-path search yields a sequence of latent subgoals, and a world model imagines waypoints across disconnected regions when no path exists. A goal-conditioned PPO policy then follows these subgoals while the planner replans as the agent progresses. On the multi-goal antmaze, the latent is accurate, decoding to state correctly, and imagined subgoals via inverse-imagination planning edge out the baseline (0.65 vs. 0.62 mean goals reached). However, the gains are small, and policies conditioned on the latent state fail to learn at all, an exploration failure under sparse reward rather than a broken representation. We find that an informative representation is not necessarily a useful one, and that the offline, frozen pipeline caps the benefit of added structure. These results argue for learning representation, planning, and control together rather than in sequence, with end-to-end training the most promising path forward.

1 Introduction

Goal-conditioned reinforcement learning (GCRL) is a form of RL where agents are conditioned on goals they must achieve, but its central difficulty grows sharply with the task horizon. To reach a distant goal an agent must execute a long sequence of low-level actions, yet the only learning signal is typically a sparse reward delivered at the goal itself. A naive policy must therefore simultaneously decide where to go and how to get there. The space of action sequences grows exponentially with the horizon, the reward is too weak and too delayed to shape early behavior, and small mistakes compound until the agent reaches states from which it has never seen a path to the goal. The result is that exploration collapses and credit assignment becomes unreliable, which is where standard GCRL methods break down.

A natural response is to shorten the effective horizon by planning over intermediate subgoals rather than primitive actions. Hierarchical and subgoal-based methods reduce the problem a policy faces, but they must still learn what the subgoal space is and which subgoals are reachable, and they tend to be unstable when the subgoal representation and the low-level policy are learned together from sparse reward. Graph-based planners instead summarize visited states as nodes and search for paths between them, which makes long-range planning explicit, but they typically operate in raw observation space and depend on a learned or hand-designed distance metric. In parallel, world models support imagination, rolling dynamics forward to reach states absent from the data, but compounding model error makes long latent rollouts unreliable.

We study a method that brings these threads together and asks a simple question: if we move all long-horizon planning into a learned latent space, can a policy learn to solve shorter-horizon subproblems which when combined solve the long-horizon problem? Our method utilizes a masked, dual-input encoder that embeds both full states and goals into a single shared latent space, so that states, goals, and subgoals lie in the same representation space. The explored latent space can be described as a graph whose nodes are cluster centroids and whose edges are observed latent transitions. We run shortest-path search to produce a sequence of latent subgoals between any start and goal. Where the graph leaves two regions disconnected because the offline data never linked them, a learned world model imagines the intermediate latent waypoints directly. A goal-conditioned policy, trained with PPO, is then responsible only for reaching the next nearby subgoal, while the planner advances and replans as the agent progresses. This design cleanly separates long-range planning, handled by the graph and world-model imagination, from local control, handled by the policy.

We make the following contributions:

- We present a method for long-horizon GCRL that unifies a contrastive latent representation, latent-graph planning, and world-model imagination, using a masked dual-input encoder that places states and goals in one shared latent space so that it can perform subgoal planning.
- We introduce an inverse world model that imagines multi-step latent subgoal sequences to bridge disjoint regions of the graph.
- We evaluate our method across 256 parallel environments to show our method outperforms the baselines, substantiating that latent subgoal decomposition with graph planning and inverse world modeling is a useful approach to long-horizon GCRL problems.

2 Related Work

2.1 GCRL and Contrastive Learning

Goal-conditioned reinforcement learning conditions a single policy and value function on a goal (Schaul et al., 2015), with Hindsight Experience Replay (Andrychowicz et al., 2017) relabeling trajectories so agents can learn from sparse reward. Prior work has applied representation learning to these value functions. Contrastive RL (Eysenbach et al., 2023) uses a contrastive loss to learn GC values, while Quasimetric RL (Wang et al., 2023) learns a quasimetric-parameterized distance function between states. However, both works are single-step and have no subgoal decomposition and thus value error propagates over extremely long horizons.

2.2 GCRL and Hierarchy

Research also addresses long horizons through hierarchy. HIQL (Park et al., 2024) uses a learned goal-conditioned value function to extract a high-level policy that proposes latent subgoals, while a low-level policy reaches them. However, the high-level policy proposes subgoals one at a time, with no intermediate planning to mitigate bad subgoal guesses. Director (Hafner et al., 2022) combines hierarchy with a world model by training a high-level policy that proposes subgoals in imagination, but subgoals come from a learned policy rather than a planning module. Landmark-based methods add explicit structure: HIGL (Kim et al., 2021) steers the high-level policy toward the first landmark on a shortest path through a landmark graph, DHRL (Lee et al., 2022) plans over the full graph to decouple the high- and low-level horizons, and LEAP (Nasiriany et al., 2019) optimizes a sequence of latent subgoals using goal-conditioned value functions. However, these methods still rely on a learned policy or value model to propose or score subgoals rather than searching an explicit reachability graph.

2.3 GCRL and Planning

The closest line of work is graph-based GCRL. SoRB (Eysenbach et al., 2019) creates a graph from states in the replay buffer and runs Dijkstra over a learned distance for planning, while HILL (Zhang et al., 2023) does the same but draws graph nodes from a contrastively learned latent space. However, in these works similar states are not grouped together and the graph does not extend beyond experienced data. LEXA (Mendonca et al., 2021) uses a world model to imagine candidate goal states, albeit for use in a single-step policy with no overall planning structure. Our project fills these gaps,

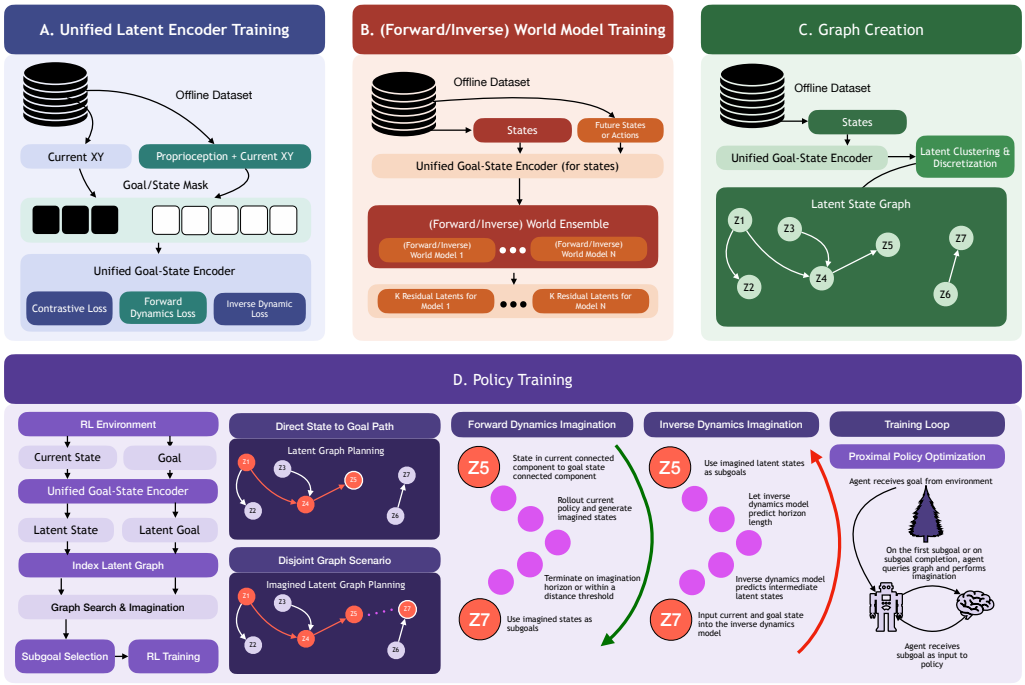

Figure 1: Overview of MIRAGE. A masked encoder maps states and goals into a shared latent space; k-means centroids form graph nodes connected by observed transitions; forward and inverse world models imagine edges that bridge disconnected components; and a PPO policy follows the resulting latent subgoals.

learning a discretized latent space with a forward dynamics model to create a graph of experienced and imagined states for subgoal decomposition and planning.

2.4 GCRL and World Models

World models learn environment dynamics so an agent can plan or train in imagination (Ha and Schmidhuber, 2018). The Dreamer series (Hafner et al., 2023) learns latent dynamics and rolls them forward to train an actor-critic, while TD-MPC2 (Hansen et al., 2024) learns a reward-predictive latent model for decision-time planning, and probabilistic dynamics ensembles (Chua et al., 2018) capture uncertainty to limit error accumulation. However, these models are used for short-horizon rollouts or policy learning, not to expand a planning graph; we use forward and inverse world models to imagine transitions that connect disjoint graph components and supply multi-step subgoals.

3 Method

We (i) learn a single encoder that embeds both states and goals into a shared latent, (ii) discretize that latent into a graph whose nodes are clusters and whose edges are observed transitions, (iii) train forward and inverse world models that imagine transitions absent from the data, and (iv) train a goal-conditioned policy to reach the next latent subgoal produced by graph search. Figure 1 summarizes the pipeline.

3.1 Preliminaries

We consider a goal-conditioned Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{G}, P, r, \gamma)$ with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, goals $g \in \mathcal{G}$, transition dynamics $P(s' \mid s, a)$, and a sparse goal-reaching reward $r(s, a, g)$.

The objective is a goal-conditioned policy that maximizes the expected discounted return,

$$\pi^*(a \mid s, g) = \arg\max_{\pi} \mathbb{E}_{\tau \sim \pi(\cdot|g)} \left[\sum_{t=0}^{T} \gamma^t \, r(s_t, a_t, g) \right]. \tag{1}$$

We assume access to an offline dataset $\mathcal{D} = \{(s_t, a_t, s_{t+1})\}$ of transitions organized into trajectories. We write $z = \phi(s)$ for the latent of a state under the encoder ϕ, and reserve $\mathrm{sg}(\cdot)$ for the stop-gradient operator.

3.2 Unified State-Goal Encoder

The core of our representation is a single encoder that maps *both* full states and goals into one shared latent space, so that states, goals, and subgoals are directly comparable. The encoder is a multilayer perceptron ϕ whose input is the concatenation of proprioception p, position xy, and a binary mask flag m. When $m = 0$ the encoder sees the full state; when $m = 1$ the proprioception is zeroed and only the position survives, which is exactly the information available for a goal. We L2-normalize the output so every latent lies on the unit hypersphere, $z = \phi(p, xy, m) \in \mathbb{S}^{d-1}$. During training we set $m = 1$ with a fixed probability, so the same network learns to encode states and goals consistently.

We train ϕ with a contrastive objective and four auxiliary losses. We sample a trajectory and draw two states within a temporal window as a positive pair (z_i, z_i^+), treating all other states in the batch as negatives, and apply a symmetric InfoNCE loss that pulls temporally near states together and pushes distant ones apart:

$$\mathcal{L}_{\text{NCE}} = -\frac{1}{2B} \sum_{i} \left[\log \frac{e^{z_i^\top z_i^+ / \tau}}{\sum_j e^{z_i^\top z_j^+ / \tau}} + \log \frac{e^{z_i^{+\top} z_i / \tau}}{\sum_j e^{z_i^{+\top} z_j / \tau}} \right]. \tag{2}$$

Four auxiliary terms shape the latent: forward- and inverse-dynamics losses make it action-aware (Pathak et al., 2017), a reconstruction loss keeps it decodable, and an alignment loss ties the masked goal encoding to the full-state encoding,

$$\begin{aligned} \mathcal{L}_{\text{fwd}} &= \|h_f(z_t, a_t) - \mathrm{sg}(z_{t+1})\|_2^2, & \mathcal{L}_{\text{inv}} &= \|h_g(z_t, z_{t+1}) - a_t\|_2^2, \\ \mathcal{L}_{\text{rec}} &= \|D(z_t) - \bar{s}_t\|_2^2, & \mathcal{L}_{\text{align}} &= \|\phi(s) - \phi_{\text{goal}}(s)\|_2^2, \end{aligned} \tag{3}$$

where $\bar{s}$ is the standardized state and ϕ_{goal} is the encoder evaluated with $m = 1$. The encoder minimizes a weighted sum of these terms; after training it is frozen and all downstream components operate on its latents.

3.3 Latent Graph Construction

To enable discrete planning we summarize the latent space as a graph $G = (V, E)$. We encode every state in the dataset, run k-means with K clusters over the latents, and take the centroids $\{c_1, \ldots, c_K\}$ as nodes. Each latent is assigned to its nearest centroid,

$$n(z) = \arg\min_{k \in \{1, \ldots, K\}} \|z - c_k\|_2, \tag{4}$$

and we add a directed edge between two distinct clusters whenever a transition between them appears in the data. Discretization groups similar states into a single node, bounding the graph size and denoising the planning problem relative to building a graph over raw states.

3.4 Forward and Inverse World Models

A graph built from observed transitions cannot connect regions the data never linked; we learn two world models that imagine such transitions in latent space.

Forward model. The forward model predicts the next latent from a latent action pair. To capture uncertainty we use an ensemble of M probabilistic networks, each outputting a diagonal Gaussian over a residual update, trained with a multi-step negative log-likelihood that unrolls H steps with geometric weighting ρ:

$$\mathcal{L}_{\text{F}} = -\frac{1}{M} \sum_{m=1}^{M} \sum_{h=1}^{H} \rho^{\,h-1} \log \mathcal{N}\Big(z_{t+h} \,;\, \mu_m(\hat{z}_{t+h-1}, a_{t+h-1}),\, \sigma_m^2(\cdot) \Big), \tag{5}$$

Algorithm 1 MIRAGE Rollout

$z_{\text{goal}} \leftarrow \phi_{\text{goal}}(g)$
$\Pi \leftarrow \textsc{Plan}(s, g, G)$; set z_{sub} to first subgoal of Π
for each environment step **do**
    $a \sim \pi(\cdot \mid s, z_{\text{sub}}, z_{\text{goal}})$; step environment
    if $\|\phi(s) - z_{\text{sub}}\| < \delta$ **then** advance z_{sub} along Π
    end if
    if goal changed or episode reset **then** $\Pi \leftarrow \textsc{Plan}(s, g, G)$
    end if
end for

with targets $z_{t+h} = \text{sg}(\phi(s_{t+h}))$ from the frozen encoder. Probabilistic ensembles follow standard model-based practice for limiting compounding error (Chua et al., 2018).

Inverse model. Given a current latent z_t and a target latent z_{t+k}, the inverse model predicts how to get from one to the other through a shared trunk with multiple heads,

$$H(z_t, z_{t+k}) \mapsto \big(\hat{a}_{t:t+k-1},\ \Delta\hat{z}_{1:k},\ \hat{k},\ \hat{r},\ \hat{d}\big), \qquad \hat{z}_{t+j} = z_t + \textstyle\sum_{l \le j} \Delta\hat{z}_l, \tag{6}$$

predicting the action sequence, intermediate latent deltas, horizon k, and auxiliary reward and goal-distance signals; it is trained with masked mean-squared error over each output. The reward and distance heads are used only as auxiliary training objectives and are not used directly during planning. Because the offline data is collected by an imperfect behavior policy, the actions linking two states are often indirect or inefficient. We therefore train the inverse model on graph-planned paths: for a pair of graph nodes we compute the shortest path between them over G using Dijkstra's algorithm and use the resulting sequence of centroid states, edge actions, and path length as targets. The model therefore learns to imitate trajectories induced by graph planning rather than the behavior policy observed in the dataset.

3.5 Graph Augmentation via Imagination

We use the world models to add edges not observed in the dataset. *Forward imagination* augments the graph with local connections: from a node latent we roll sampled actions through the forward model, bin each predicted next latent to a node, and add the resulting edges. *Inverse imagination* augments globally: for a disconnected pair, the inverse model predicts the intermediate latents, which we bin to nodes to form a chain of bridging edges. Forward imagination primarily adds local connectivity, while inverse imagination generates directed waypoint sequences that can bridge disconnected graph components.

3.6 Planning over the Latent Graph

At inference we encode the current state and goal, bin them to the nearest graph nodes, and run Dijkstra's algorithm to obtain a sequence of latent subgoals between them. When the start and goal fall in different components and no path exists, we invoke imagination to bridge them. The planner advances to the next subgoal once the agent is within a distance threshold of the current one, and replans when the goal changes or the episode resets.

3.7 Policy Training

The policy is responsible only for short-horizon control toward the next subgoal. We assemble its input from the current state, the next latent subgoal, and the latent goal, $\pi(a \mid s, z_{\text{sub}}, z_{\text{goal}})$, and train it with Proximal Policy Optimization (Schulman et al., 2017) and Generalized Advantage Estimation (Schulman et al., 2016). Algorithm 1 ties the components together: at each step the agent encodes its state, acts toward the current subgoal, advances the plan when the subgoal is reached, and replans as needed.

4 Experimental Setup

4.1 Environment

We evaluate on AntMaze (Fu et al., 2021), a goal-conditioned navigation task in which a quadruped ant must reach a target (x, y) location in a maze. The observation separates proprioception (joint positions and velocities) from the agent's planar position, and external forces acting on the ant. The goal is specified as a 2D coordinate, giving the distinct state and goal spaces our encoder is designed to share. The reward is sparse, granted only on reaching the goal. We use the continuing multi-goal variant of AntMaze: once a goal is reached a new target is resampled, so a single episode of up to 1000 steps can reach several goals in succession. Simulation is GPU-parallelized through MuJoCo Warp, which lets us run many environments concurrently.

4.2 Evaluation Setup

We evaluate each trained configuration on 256 episodes with deterministic actions under a fixed seed. We report the mean and standard deviation number of goals reached per episode and the fraction of episodes that reach at least one goal.

Our configurations vary along four axes:

- **Policy state input:** the raw state (RS) or the encoder's latent (LS)
- **Goal Conditioning**: a raw (x, y) goal (RG) or its latent embedding (LG)
- **Subgoal Conditioning:** a raw (RSG) or latent (LSG) subgoal
- **Subgoal Source:** none, motion planning, forward imagination, or inverse imagination

Table 1 reports all configurations.

5 Results

5.1 Quantitative Evaluation

Policy Inputs	Subgoal Type	Goals Reached	% with ≥ 1 goal
Baseline Methods			
RS, RG	—	0.617 ± 0.816	44.1
RS, LG	—	0.582 ± 0.786	41.4
LS, LG	—	0 ± 0	0
RS, RSG, RG	Motion Planning	0.547 ± 0.904	34.4
RS, LSG, LG	Motion Planning	0.512 ± 0.838	34.0
LS, LSG, LG	Motion Planning	0.008 ± 0.008	0.8
Imagination Methods			
LS, LSG, LG	Forward Imagination	0	0
LS, LSG, LG	Inverse Imagination	0	0
RS, LSG, LG	**Inverse Imagination**	0.652 ± 0.853	46.1

Table 1: Policy input configurations and results. RS = raw state, RG = raw goal, RSG = raw subgoal, LS = latent state, LG = latent goal, LSG = latent subgoal. **Note:** We do not include RS, LSG, LG for Forward Imagination because we hypothesize that compounding errors from successive decoding steps would result in poor performance

The single strongest determinant of performance is the policy state input. Every configuration that conditions the policy on the raw state reaches between 0.51 and 0.65 goals and clears at least one goal in 34 to 46% of episodes, whereas every configuration that conditions the policy on the latent state collapses to essentially zero. The collapse is not caused by a poor representation. When we use a linear probe to analyze the latent, it is able to recover the original state from the latent with high accuracy, so the information the policy needs is present in the input. Rather, we hypothesize that it is

likely because a densely packed latent coupled with sparse goal reward causes exploration to collapse before the policy ever discovers a goal. Latent-state control is therefore the central bottleneck. For the raw state configurations, subgoal structure helps only when it is high quality. Our best method, a raw-state policy following latent subgoals produced by inverse-imagination planning, achieves the highest performance overall at 0.652 goals and 46.1% of episodes reaching a goal, outperforming the raw-state baseline (0.617, 44.1%). The gain comes from the imagined subgoals: when the start and goal fall in different graph components, the inverse model fills in the missing latent waypoints, breaking a long, sparsely rewarded reach into shorter segments the policy can complete. By contrast, subgoals from the ground-truth motion-planning oracle do not help and in fact slightly hurt (0.547 and 0.512 with raw and latent subgoals respectively), suggesting that the learned latent subgoals provide guidance that is more useful to the policy than manually defined motion-planning subgoals.

5.2 Qualitative Analysis

We perform a qualitative analysis between the following runs: 1) RS, RG 2) LS, LSG, LG Inverse Imagination 3) RS, LSG, LG Inverse Imagination. We show frames from video rollouts of the policies in fig. 2

The baseline (top) moves but is undirected. It never commits to the corridor that leads to the goal. This is the behavior of a policy that has learned to move but has no signal telling it where to go over a long horizon. The latent-state agent (middle) barely moves at all and effectively freezes near its start. It reinforces our hypothesis of the collapse as an exploration failure. A representational failure would still encourage random exploration.

By contrast, our best method (bottom) climbs the corridor steadily and reaches the goal. The difference between it and the baseline is the imagined subgoals. Rather than being asked to reach a distant goal in one shot, the policy is sequentially handed nearby latent waypoints that trace a route up the corridor, turning the long-horizon problem into a series of short steps.

6 Discussion

Our three takeaways about learning, planning, and acting in a latent space for long-horizon GCRL are as follows:

Learning in latent space is hard. The first thing we notice is that swapping the raw state for a learned latent collapses RL performance. Every latent-state configuration reaches essentially zero goals, while every raw-state configuration learns. This is not because the latent discards the information the policy needs, since a linear probe recovers the state from it with high accuracy. Instead, the latent is simply harder to optimize against under sparse reward. We hypothesize the primary reason for this is that the low-dimensional bottleneck may pack distinct states too densely for the policy to separate them from a weak gradient. While this is less problematic in tabular RL methods, with function approximation, it may make it more difficult to optimize. We hope to conduct more experiments in the interpretability of our RL policy to determine where the shortcoming is for future work.

Manually defined subgoals hurt performance. A natural assumption is that handing the policy correct geometric waypoints should help, but we observe the opposite. Subgoals from the ground-truth motion planner consistently lower performance relative to the flat baseline (0.547 and 0.512 goals versus 0.617), whereas latent imagined subgoals raise it. We read this as evidence that manually specified subgoals may interfere with the behavior it would otherwise learn. Learned latent subgoals, by contrast, live in the same space the policy is conditioned on. They encode richer, more compatible structure than a hand-defined subgoal, which is why they help rather than hurt.

World modeling helps planning. Finally, moving subgoals from goal space into latent space, where they are produced by our world model, yields the best configuration overall (0.652 goals, 46.1%), ahead of all of the baselines. The fact that inverse-imagination subgoals improve performance at all indicates that the encoder and inverse world model learn useful structure. The model can bridge regions the data never connected and produce waypoints the policy can follow toward distant goals. The gain is modest, which we attribute to the encoder, graph, and world models being trained offline

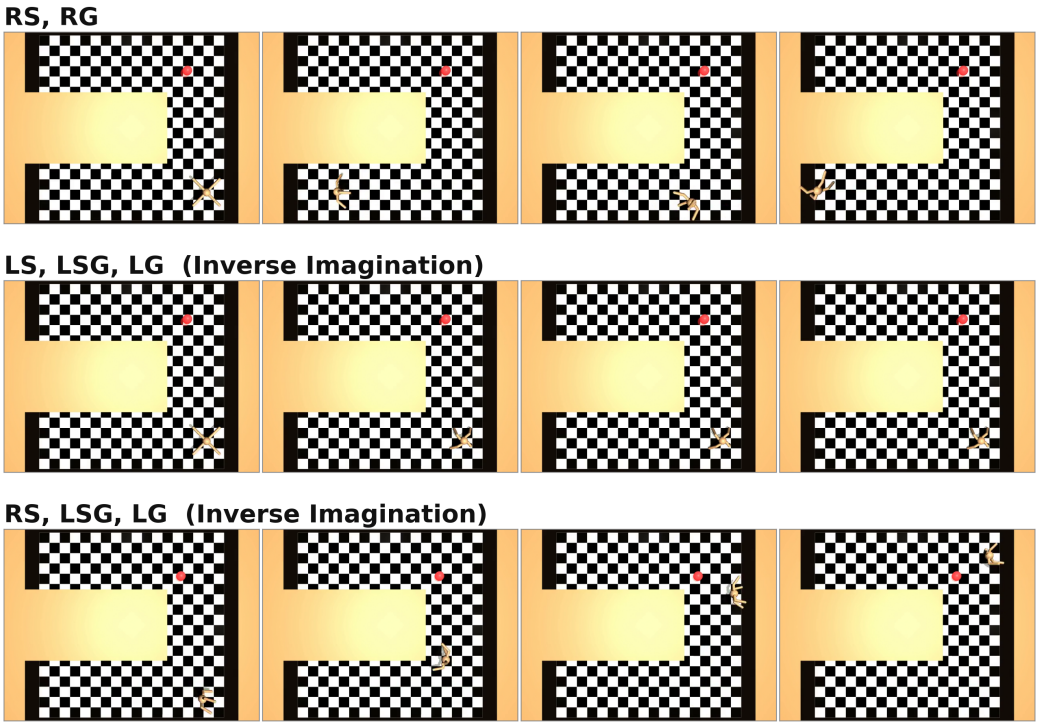

Figure 2: Red dot indicates the goal for the agent. Top shows the RS, RG baseline: starting from the right, the ant fails to climb the right corridor toward its goal and flails along the bottom wall without making progress. Middle shows the LS, LSG, LG, Inverse Imagination run, where the agent freezes, indicating an exploration problem. Bottom shows the RS, LSG, LG Inverse Imagination run, where the agent climbs steadily up the corridor, successfully reaching the goal

and frozen, so the plan is never adapted to what the policy can execute. Together these results point to a clear next step. Training the representation, world model, and policy jointly would let gradients from control shape the very latent space planning operates in.

6.1 Investigating Latent Space Collapse

To determine whether we could raise the performance floor of the latent state policies to above 0, we tested a few methods. Specifically, we trained policies that utilized exploration methods like random network distillation (Burda et al., 2018), applied curricula over the goal space, and asymmetric critics. These methods increased performance from 0 to 0.05 mean goals reached. While this does improve performance, due to the statistical insignificance of these results, we omit them from section 5 and the main codebase.

We believe that more advanced and rigorous methods of exploration, such as entropy-regularized RL (Haarnoja et al., 2018) or intrinsic curiosity modules, could yield further improvements (Pathak et al., 2017). We leave that as future work.

7 Conclusion

We studied whether long-horizon goal-conditioned RL can be made easier by moving planning into a learned latent space, pairing a masked state-goal encoder with a latent graph and forward and inverse world models that imagine subgoals across gaps the data leaves unconnected. On AntMaze, the learned latent is informative and its imagined subgoals give our best method a small but real edge over the baseline, while hand-defined motion-planning subgoals instead hurt, showing that learned latent structure is more compatible with the policy than manually specified waypoints. The central

limitation is that conditioning the policy on the latent state collapses learning entirely, an exploration failure under sparse reward rather than a loss of information, and the offline, frozen pipeline keeps the planning gains modest. These findings argue that representation, planning, and control should be learned together rather than in sequence, and we see end-to-end training, in which gradients from the policy shape the latent space that planning operates over, as the most promising path toward realizing the full benefit of latent-space planning.

8 Team Contributions

- **Samrat Sahoo:** PPO Implementation, Evaluation of Policies, Experiments on full pipeline, Repository Setup and Clean Up, Proposal / Poster / Paper Writing
- **Abhinav Sattiraju:** Encoder Setup and Training, World Model (Forward + Inverse) Setup and Training, Experiments on graph creation, Proposal / Poster / Paper Writing

Changes from Proposal The primary change from our initial proposal was how we used world modeling. Originally, we proposed augmenting the graph with novel states. During experimentation, however, we noticed there may be scenarios where the graph may be disjoint between two states of interest. Thus, we instead chose to utilize world modeling as a bridge between disjoint states.

9 AI Usage Disclosure

We used coding assistant tools for infrastructure for our project, such as data organization pipelines and evaluation scripts. Core PPO training logic, encoder and world model architecture, and graph creation logic were written by hand. We also utilized LLMs to revise grammar in the final report.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2018. Exploration by Random Network Distillation. arXiv:1810.12894 [cs.LG] `https://arxiv.org/abs/1810.12894`

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. 2018. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. 2019. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning. arXiv:1906.05253 [cs.AI] `https://arxiv.org/abs/1906.05253`

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. 2023. Contrastive Learning as Goal-Conditioned Reinforcement Learning. arXiv:2206.07568 [cs.LG] `https://arxiv.org/abs/2206.07568`

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2021. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. arXiv:2004.07219 [cs.LG] `https://arxiv.org/abs/2004.07219`

David Ha and Jürgen Schmidhuber. 2018. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. arXiv:1801.01290 [cs.LG] `https://arxiv.org/abs/1801.01290`

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. 2022. Deep Hierarchical Planning from Pixels. arXiv:2206.04114 [cs.AI] `https://arxiv.org/abs/2206.04114`

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. 2023. Mastering Diverse Domains through World Models. *arXiv preprint arXiv:2301.04104* (2023).

Nicklas Hansen, Hao Su, and Xiaolong Wang. 2024. TD-MPC2: Scalable, Robust World Models for Continuous Control. In *International Conference on Learning Representations (ICLR)*.

Junsu Kim, Younggyo Seo, and Jinwoo Shin. 2021. Landmark-Guided Subgoal Generation in Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Seungjae Lee, Jigang Kim, Inkyu Jang, and H. Jin Kim. 2022. DHRL: A Graph-Based Approach for Long-Horizon and Sparse Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. 2021. Discovering and Achieving Goals via World Models. arXiv:2110.09514 [cs.LG] `https://arxiv.org/abs/2110.09514`

Soroush Nasiriany, Vitchyr H. Pong, Steven Lin, and Sergey Levine. 2019. Planning with Goal-Conditioned Policies. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. 2024. HIQL: Offline Goal-Conditioned RL with Latent States as Actions. arXiv:2307.11949 [cs.LG] `https://arxiv.org/abs/2307.11949`

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. 2017. Curiosity-driven Exploration by Self-supervised Prediction. arXiv:1705.05363 [cs.LG] `https://arxiv.org/abs/1705.05363`

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. 2015. Universal Value Function Approximators. In *International Conference on Machine Learning (ICML)*.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations (ICLR)*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. 2023. Optimal Goal-Reaching Reinforcement Learning via Quasimetric Learning. arXiv:2304.01203 [cs.LG] `https://arxiv.org/abs/2304.01203`

Qingyang Zhang, Yiming Yang, Jingqing Ruan, Xuantang Xiong, Dengpeng Xing, and Bo Xu. 2023. Balancing Exploration and Exploitation in Hierarchical Reinforcement Learning via Latent Landmark Graphs. arXiv:2307.12063 [cs.LG] `https://arxiv.org/abs/2307.12063`