Learning World Graphs to Accelerate Hierarchical Reinforcement Learning

Wenling Shang* **Alex Trott†** **Stephan Zheng†** **Caiming Xiong** **Richard Socher**
Salesforce Research

Abstract

In many real-world scenarios, an autonomous agent often encounters various tasks within a single complex environment. We propose to build a graph abstraction over the environment structure to accelerate the learning of these tasks. Here, nodes are important points of interest (*pivotal states*) and edges represent feasible traversals between them. Our approach has two stages. First, we jointly train a latent pivotal state model and a curiosity-driven goal-conditioned policy in a task-agnostic manner. Second, provided with the information from the world graph, a high-level Manager quickly finds solution to new tasks and expresses subgoals in reference to pivotal states to a low-level Worker. The Worker can then also leverage the graph to easily traverse to the pivotal states of interest, even across long distance, and explore non-locally. We perform a thorough ablation study to evaluate our approach on a suite of challenging maze tasks, demonstrating significant advantages from the proposed framework over baselines that lack world graph knowledge in terms of performance and efficiency.

1 Introduction

Many real world scenarios require an autonomous agent to play different roles within a single complex environment. For example, Mars rovers carry out scientific objectives ranging from searching for rocks to calibrating orbiting instruments [70]. Intuitively, a good understanding of the high-level structure of its operational environment would help an agent accomplish its downstream tasks. In reality, however, both acquiring such world knowledge and effectively applying it to solve tasks are often challenging. To address these challenges, we propose a generic two-stage framework that first learns *high-level world structure* in the form of *a simple directed weighted graph* [11] and then integrates it into a hierarchical policy model.

In the initial stage, we alternate between exploring the world and updating a descriptor of the world in a graph format [11], referred to as *the world graph* (Figure 1), in an unsupervised fashion. The nodes, termed *pivotal states*, are the most critical states in recovering action trajectories [17, 46, 32]. In particular, given a set of trajectories, we optimize a fully differentiable recurrent variational auto-encoder [19, 34, 50] with binary latent variables [69]. Each binary latent variable is designated to a state and the prior distribution learned conditioning on that state indicates whether it belongs to the set of pivotal states. Wide-ranging and meaningful training trajectories are therefore essential ingredients to the success of the latent model. Existing world descriptor learning frameworks often use random [37] or curiosity-driven trajectories [4]. Our exploring agent collects trajectories from both random walks and a simultaneously learned curiosity-driven goal-conditioned policy [32, 68]. During training, exploration is also initiated from the current set of pivotal states, similar to the "cells" in Go-Explore [25], except that ours are learned by the latent model instead of using heuristics. The edges of the graph, extrapolated from both the trajectories and the goal-conditioned policy, correspond to the actionable transitions between close-by pivotal states. Finally, the goal-conditioned policy can be used to further promote transfer learning on downstream tasks [85].

Preprint. Under review.

*: also University of Amsterdam, corresponding at w.shang@uva.nl. †: indicates equal contributions.
See project page (link) for selected video demos and the Appendix.

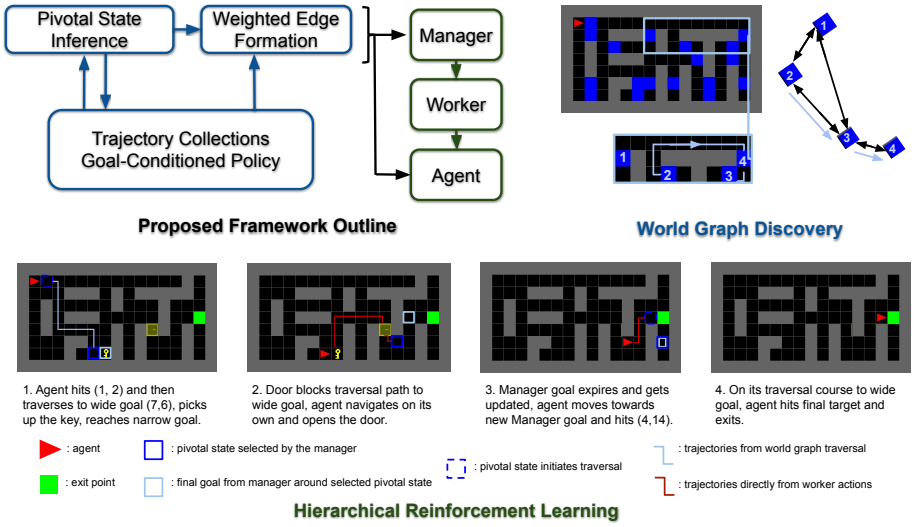

Figure 1: Top Left: Overall pipeline of our proposed 2-stage framework. Top Right (*world graph discovery*): a subgraph exemplifies how to forge edges and traverse between pivotal states (in blue). Bottom (*Hierarhical RL*): an example rollout from our proposed HRL policy with Wide-then-Narrow Manager instructions and world graph traversals, solving a challenging *Door-Key* task.

At first glimpse, the world graph seems suitable for model-based RL [58, 49], but our method emphasizes the connections among neighboring pivotal states rather than transitions over any arbitrary pair, which is usually considered a much harder problem [35]. Therefore, in the next stage, we propose a hierarchical reinforcement learning [53, 62] (HRL) approach to incorporate the world graph for solving specific downstream tasks. Concretely, within the paradigm of goal-conditioned HRL [21, 67, 89, 56], our approach innovates how the high-level *Manager* provides goals and how the low-level *Worker* navigates. Instead of sending out a single objective, the Manager first selects a pivotal state from the world graph and then specifies a final goal within a nearby neighborhood of the selected pivotal state. We refer to this sequential selection as the *Wide-then-Narrow* (WN) instruction. This construction allows us to utilize the information from the learned graph descriptor and to form passages between pivotal states through application of graph traversal techniques [9], thanks to which the Worker can now focus on local objectives. Lastly, as previously mentioned, the goal-conditioned policy derived from learning the world graph can serve as an initialization to the Manager and Worker, allowing fast skill transfer to new tasks as demonstrated by our experiments.

In summary, our main contributions are:

- A complete two-stage framework for 1) unsupervised world graph discovery and 2) accelerated HRL by integrating the graph.
- The first stage proposes an unsupervised module to learn world graphs, including a novel recurrent differentiable binary latent model and a curiosity-driven goal-conditioned policy.
- The second stage proposes a general HRL scheme with novel components such as the Wide-then-Narrow instruction, navigation via world graph traversal and skill transfer from goal-conditioned policy models.
- Quantitative and qualitative empirical findings over a complex 2D maze domain show that our proposed framework 1) produces a graph descriptor representative of the world and 2) improves both sample efficiency and final performance in solving downstream tasks by a large margin over baselines that lack the descriptor.

2 Environment

For ease of clear exposition and scientific control, we choose finite, fully observable yet complex 2D mazes [18] as our testbeds, i.e. for each state-action pair and their transitions $(s_t, a_t) \to s_{t+1}$, $s_t, s_{t+1} \in \mathcal{S}, a_t \in \mathcal{A}$ are finite. More involved environments can introduce interfering factors, shadowing the effects from the proposed method, e.g. the need of a well-calibrated latent goal space [43, 24, 66]. Section 7 briefly speculates on extensions of our framework to other environments as future directions. We employ 3 mazes of small, medium and large sizes with varying compositions (see the Appendix for visualization). Despite being finite and fully observable, these mazes still pose

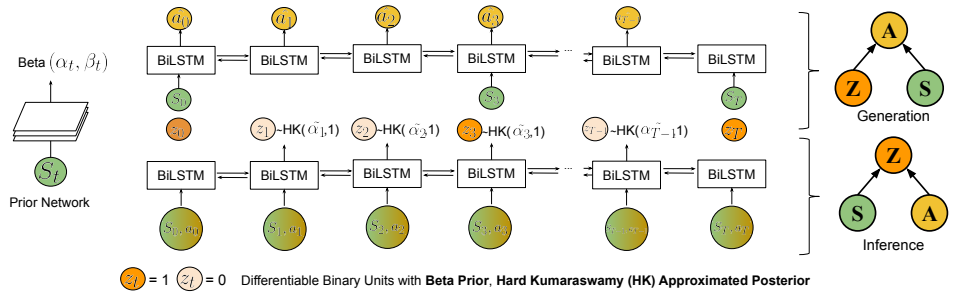

Figure 2: Our recurrent latent model with differentiable binary latent units to discover pivotal states. A prior network (left) learns the state-conditioned prior in Beta distribution, $p_\psi(z_t|s_t) = \text{Beta}(\alpha_t, \beta_t)$. An inference encoder learns an approximate posterior in HardKuma distribution [8] inferred from (s_t, a_t)'s, $q_\phi(z_t|a_t, s_t) = \text{HardKuma}(\tilde{\alpha}_t, 1)$. A generation decoder reconstructs the action sequence from $\{s_t|z_t = 1\}$. During training, we sample from $\text{HardKuma}(\tilde{\alpha}_t, 1)$ using the reparametrization trick [50].

much challenge, especially when the environment becomes large, engages stochasticity, provides only sparse reward or requires more complicated logic. The maze states received by the agent are in the form of bird-eye view matrix representations. More details on preprocessing are available in the Appendix.

3 World Graph Discovery

We envision *a simple directed weighted graph* [11] $\mathcal{G}_w$ to capture the high-level structure of the world. Its nodes are a set of points of interest, termed *pivotal states* ($s_p \in \mathcal{V}_p$), and edges represent transitions between nodes. Drawing intuition from unsupervised sequence segmentation [17, 46] and imitation learning [1, 45], we define $\mathcal{V}_p$ as the most critical states in recovering action sequences generated by some agent, indicating that these states lead to the most information gain [4]. In other words, given a trajectory $\tau = \{(s_t, a_t)\}_0^T$, we learn to identify the most critical subset of states $\{s_t|s_t \in \mathcal{V}_p\}$ for accurately inferring the action sequences taken in the full trajectory τ.

Supposing the state-action trajectories are available, we formulate a recurrent variational inference model (see Section 3.1) [12, 19, 34, 50], treating the action sequences as evidence and, for each state s_t in the sequence, inferring a binary latent variable z_t that controls whether to keep the state for action reconstruction. We learn a prior over each latent z_t conditioned on its designated state s_t, as opposed to using a fixed prior or conditioning on the surrounding trajectory, and use the prior mean as the criterion for including s_t in $\mathcal{V}_p$.

Meaningful $\mathcal{V}_p$ are learned from meaningful trajectories; hence, we develop a procedure to alternately update the latent model and improve trajectory collections. When collecting training trajectories, we place the agent at a state from the current iteration's set of $\mathcal{V}_p$—this is possible since the agent can straightforwardly document and reuse the paths from its initial position to states in $\mathcal{V}_p$. This way naturally allows the exploration starting points to expand as the agent discovers more of its environment. Random walk trajectories tend to be noisy thus perhaps irrelevant to real tasks. We instead take inspiration from prior work on actionable representations [32] and learn a goal-conditioned policy π_g for navigating between close-by states, reusing observed trajectories for unsupervised learning (Section 3.2). To ensure broad state coverage and diverse trajectories, we add a curiosity reward from the unsupervised action reconstruction error to learn π_g. The latent model is then updated with new trajectories. This cycle repeats until the action reconstruction accuracy plateaus. To form the edges of $\mathcal{G}_w$ and, we again use both random trajectories and π_g (Section 3.3). Lastly, the implicit knowledge of the world embedded in π_g can be further transferred to downstream tasks through weight initialization, which will be discussed later on (Section 4.4).

The pseudo-code summarization of world graph discovery, implementation details, a visualization of how $\mathcal{V}_p$ progresses over training and the final $\mathcal{V}_p$ from different rollout policies are provided in the Appendix. The following sections concretely describe each component of our proposed process.

3.1 Recurrent Variational Model with Differentiable Binary Latent Variables

We propose a recurrent variational model with differentiable binary latent variables to discover $\mathcal{V}_p$ (Figure 2). Given a trajectory $\tau = \{(s_t, a_t)\}_0^T$, we treat the action sequence $\{a_t\}_0^{T-1}$ as evidence in

order to infer a sequence of binary latent variables z_t's. The evidence lower bound is

$$\text{ELBO} = \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{A},\mathbf{S})}\left[\log p_\theta(\mathbf{A}|\mathbf{S},\mathbf{Z})\right] + D_{\text{KL}}\left(q_\phi(\mathbf{Z}|\mathbf{A},\mathbf{S})|p_\psi(\mathbf{Z}|\mathbf{S})\right). \tag{1}$$

The objective $\mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{A},\mathbf{S})}\left[\log p_\theta(\mathbf{A}|\mathbf{S},\mathbf{Z})\right]$ is to reconstruct the action sequence given only the states s_t where $z_t = 1$, with the boundary states always given $s_0 = s_T = 1$. To ensure differentiablity, we opt to use a continuous relaxation of discrete binary latent variables by learning a Beta distribution as the priors for z_t's [81]. Moreover, we learn the prior for each z_t conditioned on its associated state s_t (Figure 2). The prior mean for each z_t signifies *on average* how necessary s_t is for action reconstruction. Also, the KL-divergence term in Equation 1 between the approximated posterior and the learned prior encourages similar trajectories to pick the same states for action reconstruction. We define $\mathcal{V}_p$ as the top 20% states ranked by the learned prior means.

The approximate posteriors follow the Hard Kumaraswamy distribution [8] [$\text{HardKuma}(\tilde{\alpha}_t, \tilde{\beta}_t)$] which resemble the Beta distribution but is outside the exponential family. This choice allows us to sample 0's and 1's without sacrificing differentiability, accomplished via the stretch-and-rectify procedure [8, 59]. The simple CDF of Kuma also makes the reparameterization trick easily applicble [50, 78, 61]. Lastly, KL-divergence between Kuma and Beta distribution can be approximated in closed form [69]. We fix $\tilde{\beta}_t = 1$ to ease optimization since the Kuma and Beta distributions coincide when $\alpha_i{=}\tilde{\alpha_i}, \beta_i{=}\tilde{\beta}_i{=}1$.

There is not yet any constraint to prevent the model from selecting all states to reconstruct $\{a_t\}_0^{T-1}$. To introduce a selection bottleneck, we impose a regularization on the expected L_0 norm of $\mathbf{Z} = (z_1 \cdots z_{T-1})$ to promote sparsity at a targeted value μ_0 [59, 8]. In other words, this objective constraints that there should be μ_0 of activated $z_t = 1$ given a sequence of length T. Another similarly constructed transition regularization encourages isolated activation of z_t, meaning the number of transition between 0 and 1 among z_t's should roughly be $2\mu_0$. Note that both expectations in Equation 2 have closed forms for HardKuma.

$$\mathcal{L}_0 = \left\|\mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{S},\mathbf{A})}[\|\mathbf{Z}\|_0] - \mu_0\right\|^2, \mathcal{L}_T = \left\|\mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{S},\mathbf{A})}\Sigma_{t=0}^{T}\mathbb{1}_{z_t \neq z_{t+1}} - 2\mu_0\right\|^2 \tag{2}$$

Lagrangian Relaxation. The overall optimization objective consists of action sequence reconstruction, KL-divergence, $\mathcal{L}_0$ and $\mathcal{L}_T$ (Equation 3). We tune the objective weights λ_i using Lagrangian relaxation [42, 8, 10], treating λ_i's as learnable parameters and performing alternative optimization between λ_i's and the model parameters. We observe that as long as their initialization is within a reasonable range, λ_i's converge to local optimum autonomously,

$$\max_{\{\lambda_{1,2,3}\}} \min_{\{\theta,\phi,\psi\}} \mathbb{E}_{q_\psi(\mathbf{Z}|\mathbf{A},\mathbf{S})}\left[\log p_\theta(\mathbf{A}|\mathbf{S},\mathbf{Z})\right] + \lambda_1 D_{\text{KL}}\left(q_\phi(\mathbf{Z}|\mathbf{A},\mathbf{S})|p_\psi(\mathbf{Z}|\mathbf{S})\right) + \lambda_2\mathcal{L}_0 + \lambda_3\mathcal{L}_T. \tag{3}$$

Our finalized latent model allows efficient and stable mini-batch training. Alternative designs, such as Poisson prior [51] for latent space and Transformer [88] for sequence modeling, are also possibilities for future investigation. More details related to the latent model can be found in the Appendix.

3.2 Curiosity-Driven Goal-Conditioned Agent

A goal-conditioned policy, $\pi(a_t|s_t, g)$, or π_g, is trained to reach a goal state $g \in \mathcal{S}$ given current state s_t [32]. For large state spaces, training a goal-conditioned policy to navigate between any two states is non-trivial. However, our use-cases, including trajectory generation for unsupervised learning and navigation between nearby pivot states in downstream tasks, only require π_g to reach goals over a short range. We train such an A2C-based goal-conditioned policy by sampling goals using the end points of random walks with reasonable length from a given starting state. Inspired by the success of intrinsic motivation methods—in particular, curiosity [14, 2, 74, 4]—we leverage the readily available action reconstruction errors from the generative decoder as intrinsic reward signals to boost exploration when training π_g. The pseudo-code describing this method is in the Appendix.

3.3 Edge Connections

The last crucial step towards the world graph completion is building the edge connections. After finalizing $\mathcal{V}_p$, we perform random walks from $s_p \in \mathcal{V}_p$ to discover the underlying adjacency matrix [11] connecting individual s_p's. More precisely, we claim a directed edge $s_p \to s_q$ if there exist a random walk trajectory from s_p to s_q that does not intersect a third pivotal state. We then collect

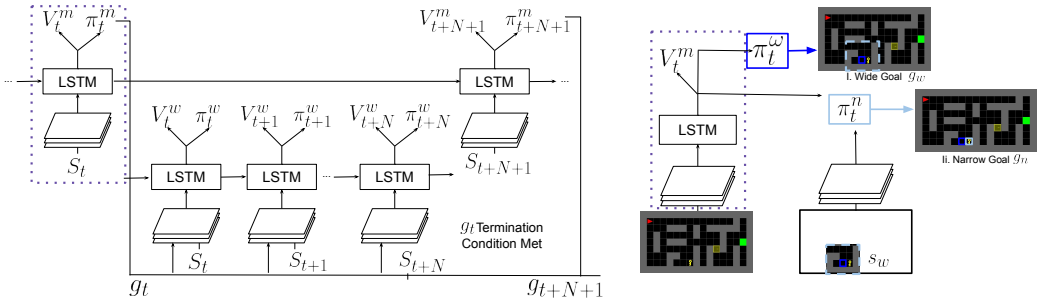

Figure 3: Left: a general configuration of Feudal Netowrk; Manager and Worker are both A2C-LSTMs operating at different temporal resolutions. Right: proposed Wide-then-Narrow Manager instruction, where Manager first outputs a wide goal g_w from a pre-defined set of candidate states $\mathcal{V}$, e.g. $\mathcal{V}_p$, and then zooms its attention to a closer up area around g_w to narrow down the final subgoal g_n.

the shortest such actionable paths as the edge paths. Each path is further refined by π_g if feasible. The action sequence length of the edge path between adjacent pivotal states defines the *weight* of the edge. Traversal between pivotal states are planned basing on the weight information using dynamic programming [84, 27]. For deterministic environments, the agents can simply follow the action sequence from the edge to transit between pivotal states. When the environment is stochastic, the agent traverses following the goal-conditioned policy (see Section 4.3). The planing in this case can potentially be improved by probabilistically or functionally encoding the edge weights [93, 79], which is left for future work.

4 Accelerated Hierarchical Reinforcement Learning

We now introduce a hierarchical reinforcement learning [53, 62] (HRL) framework that leverages the world graph $\mathcal{G}_w$ to accelerate learning downstream tasks. This framework has three core features:

- the Manager uses two-step "Wide-then-Narrow" goal descriptions (Section 4.2),
- the Worker traverses the learned world graph $\mathcal{G}_w$ at appropriate time(Section 4.3),
- the goal-conditioned policy π_g learned in the graph discovery stage is used for weight initialization for the Worker and Manager (Section 4.4).

We show that our method learns to solve new tasks significantly faster and better compared to related baselines (Section 4.1). For implementation details, see the Appendix.

4.1 Preliminaries and Hierarchical Reinforcement Learning

Formally, we consider a Markov Decision Process, where at time t an agent in a state s_t executes an action a_t via a policy $\pi(a_t|s_t)$ and receives rewards r_t. The agent's goal is to maximize its cumulative expected return $R = \mathbb{E}_{(s_t,a_t)\sim\pi,P,P_0}[r_t]$, where $p(s_{t+1}|s_t,a_t), p_0(s_0)$ are the transition and initial state distributions. To solve this problem, we consider a model-free, on-policy learning baseline, the advantage actor-critic (A2C) [92, 73, 65] and its hierarchical extension called Feudal Network (FN) [21, 89] (Figure 4). At the core, A2C models both a value function $V(s_t)$ by regressing over the estimated $t_{\max}$-step discounted returns with discount rate $\gamma \in (0,1)$, $\Sigma_{t'=t}^{t_{\max}} \gamma^{t'-t} r_{t'}$, and a policy π guided by advantage-based policy gradient [82]. The policy entropy is regularized to encourage exploration. A2C's hierarchical extension FN consists of a high-level controller ("Manager"), which learns to propose subgoals to the low-level controller ("Worker"), which learns to complete the subgoals. The Manager receives rewards from the environment and the Worker receives rewards from the Manager by reaching its subgoals. The high and low-level policy models are distinct and operate at different temporal resolutions: the Manager only outputs a new subgoal if either Worker completes its current one or the subgoal horizon c is exceeded. In this work, we mainly consider finite, discrete and fully observable mazes. As such, FN can use any state as a subgoal and the Manager policy can emit a probability vector of dimension $|\mathcal{S}|$, although our framework supports more general subgoal definitions. More implementation details and pseudo-codes on our baselines are in the Appendix.

4.2 World Graph Nodes for Wide-then-Narrow Manager Instructions

To connect the learned graph $\mathcal{G}_w$ to the HRL framework, the Manager needs the ability to designate any state s as a subgoal while using the abstraction provided by $\mathcal{G}_w$. To that end, we structure the Manager's output using a *Wide-then-Narrow* (WN) format:

1. Given a pre-defined set of candidate states $\mathcal{V}$, the Manager uses a "wide" policy $\pi^{\omega}(s_t)$ that outputs a "wide" subgoal $g_w \in \mathcal{V}$. This work proposes the "wide" goals come from the learned *pivotal states* $\mathcal{V} = \mathcal{V}_p$.
2. Next, the Manager zooms its attention to an $N \times N$ local area $s_{w,t}{=}s_w(g_{w,t})$ around g_w. A "narrow-goal" policy π^n then selects a final "narrow" goal $g_n \in s_{w,t}$, using both global s_t and local $s_{w,t}$ information. Both goals are passed to the Worker, who is rewarded if reaching g_w or g_n within the horizon c.

Using the WN goal format, the policy gradient for the Manager policy π_m becomes:

$$\mathbb{E}_{(s_t,a_t)\sim\pi,p,p_0}\left[A_{m,t}\nabla\log\left(\pi^{\omega}\left(g_{w,t}|s_t\right)\pi^{n}\left(g_{n,t}|s_t,g_{w,t},s_{w,t}\right)\right)\right]+\nabla\left[\mathcal{H}\left(\pi^{\omega}\right)+\mathcal{H}\left(\pi^{n}(\cdot|g_{w,t})\right)\right],$$

where $A_{m,t}$ is the Manager's advantage at time t. Since the size of the action space scales linearly with $|\mathcal{S}|$, the exact entropy for the π^m can easily become intractable (see the Appendix). Thus in practice we resort to an effective alternative $\mathcal{H}\left(\pi^{\omega}\right)+\mathcal{H}\left(\pi^{n}(\cdot|g_{w,t})\right)$.

4.3 Using World Graph Edges for Traversal

With pivotal states serving as wide-goals, we can effectively take advantage of the edges in the world graph $\mathcal{G}_w$ through graph traversals:

1. **When to Traverse:** When a Worker is given a goal pair (g_w, g_n), it can traverse the world via $\mathcal{G}_w$ if it encounters a pivotal state g'_w that has a feasible connection to g_w in $\mathcal{G}_w$.
2. **Planning:** We estimate the optimal traversal route from g'_w to g_w based on the $\mathcal{G}_w$ edge weights. Here we use the classic dynamic programming planning methods [84, 27], although other (learned) methods can be applied.
3. **Execution:** once the route is planned, for deterministic environments, the agent simply follows the action sequence from the edge paths. For stochastic environments, we either disallow the agent to follow a route that is newly blocked and expect the Manager to adapt accordingly (e.g. in *Door-Key*) or rely on π_g to navigate between pivotal states (e.g. in *MultiGoal-Stochastic*).

During learning, π_g can be simultaneously fine-tuned to adapt task-specific environment stochasticity. When traversing under π_g, if the agent fails to reach the next target pivotal state within a certain time limit, it would simply stop its current pursuit. The benefit of world graph traversal is to allow the Manager to assign more task-relevant goals that can be far away from an agent's position yet easily reachable by leveraging the connectivity knowledge of the world. In this way, we can speed up learning by focusing the low-level exploration on the relevant parts of the world only, i.e., around those highly task-relevant pivotal states g_w.

4.4 Knowledge Transfer through Goal-Conditioned Policy Initialization

Lastly, we leverage implicit knowledge of the world acquired by π_g during world graph discovery to the subsequent HRL training. Transferring and generalizing skills between RL tasks often leads to performance gains [85, 7] and goal-conditioned policies have been shown to capture the underlying structure of the environment well [32]. Additionally, optimizing a neural network system like HRL [20] is sensitive to weight initialization [63, 54], due to its complexity and lack of clear supervision. Therefore, taking inspiration from the prevailing pre-training procedures in computer vision [80, 23] and NLP [22, 77], we achieve implicit skill transfer and improved optimization by initializing the task-specific Worker and the Manager with the weights from π_g. Our empirical results put forward strong evidence that such initialization serves as an essential basis to solve challenging RL tasks later on, analogously to similar practices in the other domains.

5 Experiments

We validate the effectiveness and assess the impact of each proposed component in a thorough ablation study on a set of 4 challenging maze tasks with different reward structures, levels of stochasticity and logic. Furthermore, we evaluate each task in three different mazes of increasing sizes (small, medium and large). Implementation details, snippets of the tasks and mazes are in the Appendix.

In all tasks, every action taken by the agent receives a negative reward penalty -0.01. The other specifics of each task are:

- In *MultiGoal*, the agent needs to collect 5 randomly spawned balls and exit from a designated exit point. Reaching each ball or the exit point gives reward $+1$.

Task	**MultiGoal Dense Reward**			**MultiGoal Sparse Reward**			**Stochastic MultiGoal**		
Maze size	Small	Medium	Large	Small	Medium	Large	Small	Medium	Large
Baselines									
A2C	2.04	Fail	Fail	-0.10	Fail	Fail	1.38	Fail	Fail
FN	Fail	Fail	Fail	0.19	Fail	Fail	Fail	Fail	Fail
FN + π_g init	2.93	Fail	Fail	Fail	Fail	Fail	1.93	Fail	Fail
Wide-Narrow									
+ π_g**-init**									
$\mathcal{V}_{\text{all}}$	4.73	4.71	Fail	0.32	Fail	Fail	2.59	1.62	Fail
$\mathcal{V}_{\text{rand}}$	3.67	4.72	Fail	0.36	Fail	Fail	2.82	Fail	Fail
$\mathcal{V}_p$	**5.25**	**5.15**	Fail	0.39	Fail	Fail	**3.06**	**2.99**	Fail
+ $\mathcal{G}_w$**-traversal**									
$\mathcal{V}_{\text{rand}}$	3.85	2.59	1.65	0.17	0.19	0.20	Fail	1.67	Fail
$\mathcal{V}_p$	3.92	2.56	2.18	0.24	0.20	0.16	Fail	2.42	1.05
+ π_g**-init**									
+ $\mathcal{G}_w$**-traversal**									
$\mathcal{V}_{\text{rand}}$	4.16	3.29	2.30	0.25	0.24	0.19	2.72	2.42	1.93
$\mathcal{V}_p$	5.05	3.00	**2.72**	**0.42**	**0.25**	**0.26**	2.92	2.62	**1.79**

Door-Key	+ π_g **init**			+ $\mathcal{G}_w$ **traversal**		+ $\mathcal{G}_w$ **init** + **traversal**	
Wide-Narrow	$\mathcal{V}_{\text{all}}$	$\mathcal{V}_{\text{rand}}$	$\mathcal{V}_p$	$\mathcal{V}_{\text{rand}}$	$\mathcal{V}_p$	$\mathcal{V}_{\text{rand}}$	$\mathcal{V}_p$
Small	94±5	97±2	**99**±0	Fail	37±15	76±14	92±2
Medium	25±15	1±1	56±2	Fail	Fail	**79**±11	76±6
Large	Fail	Fail	Fail	Fail	Fail	**27**±40	**26**±19

Table 1: Top: results over *MultiGoal* and its sparse/stochastic versions (average reward after 100k training iterations). Bottom: results over *Door-Key* (average success rate in % ± std), omitting suboptimal models for clearer presentation. Note that WN with neither π_g init nor $\mathcal{G}_w$ traversal fail across tasks thus not displayed here. **For full results, including standard-deviations, see the Appendix.**

- Its sparse version, *MultiGoal-Sparse*, only gives a single reward $r \leq 1$ proportional to the number of balls collected upon exiting.
- Its stochastic version, *MultiGoal-Stochastic*, spawns a lava block at a random location each time step that immediately terminates the episode with a negative reward of -0.5 if stepped on.
- *Door-Key* is a much more difficult task that adds new actions ("pick" and "open") and new objects to the environment (additional walls, doors, keys). The agent needs to pick up the key, open the door (reward $+1$) and reach the exit point on the other side (reward $+1$).

Control Experiments We ablate each proposed components and compare against the non-hierarchical and hierarchical baselines, A2C and FN. The proposed components always augment on top of FN.

- First, we test initializing the Manager and Worker with the weights of π_g.
- Next, we evaluate WN with 3 different sets of $\mathcal{V}$'s for the Manager to pick g_w from: $\mathcal{V}_{\text{all}}$ includes all valid states, $\mathcal{V}_{\text{rand}}$ are uniformly sampled states, $\mathcal{V}_p$ are learned pivotal states. $\mathcal{V}_p$ and $\mathcal{V}_{\text{rand}}$ are of the same size and their edge connections are obtained in the same way (Section 3.3)[2].
- Finally, we enable $\mathcal{G}_w$ traversal on top of WN. If traversal is done through π_g, along side with HRL training, we also refine π_g (if given for initialization) or learn one from scratch (if not given for initialization).

We inherit most hyperparameters from the training of π_g in the world graph discovery stage, as the Manager and the Worker both share similar architecture as π_g. The hyperparameters of π_g in turn follow those from [83]. For details, see the Appendix. We follow a rigorous evaluation protocol acknowledging the variability in outcomes in deep reinforcement learning [41]: each experiment is repeated with 3 seeds [91, 72], 10 additional validation seeds are used to pick the best model which is then tested on 100 testing seeds. Mean of selected testing results are in Table 1. We omit results of those experimental setups that consistently fail, meaning training is either not initiated or validation

[2]The edge connection of $\mathcal{V}_{\text{all}}$ is a trivial case excluded here as every state is 1 step away from its adjacent states. Also, note neither π_g nor guaranteed state access is available to $\mathcal{V}_{\text{rand}}$ when forming edge connections, but we grant all pre-requisites for the fairest comparison possible.

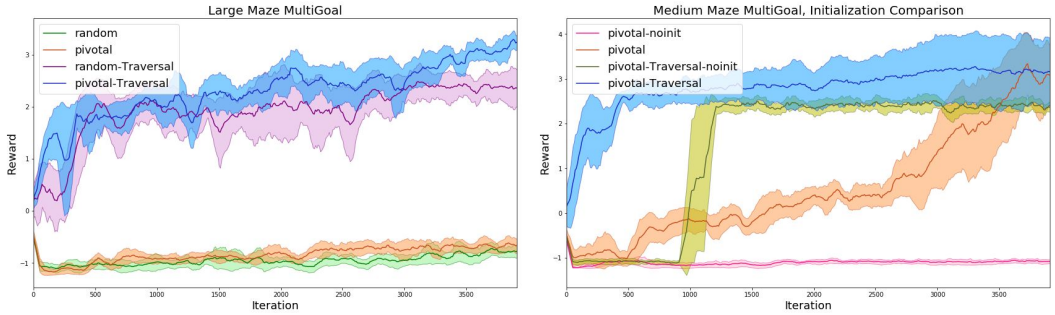

Figure 4: Validation curves during training (mean and standard-deviation of reward, 3 seeds) for *MultiGoal*. Left: Compare between $\mathcal{V}_p$ and $\mathcal{V}_{\text{rand}}$, with or without traversal, all models here use WN and π_g initialization. Observe that (1) traversal evidently speeds up convergence (2) $\mathcal{V}_{\text{rand}}$ carries higher variance and slightly inferior performance than $\mathcal{V}_p$. Right: compare with or without π_g initialization on $\mathcal{V}_p$, all models use WN; initialization shows clear advantage.

rewards are never above 0, on all tasks. See the Appendix for the full report, including standard deviations over the 3 seeds.

5.1 Empirical Analysis

Initialization with π_g Table 1 and Figure 4 show initialization with π_g is crucial across all tasks, especially for the hierarchical models—e.g. a randomly initialized A2C outperforms a randomly initialized FN on small-maze *MultiGoal*. Models starting from scratch fail on almost all tasks within the maximal number of training iterations, unless coupled with $\mathcal{G}_w$ traversal, which is still inferior to using π_g-initialization.

Wide-then-Narrow Comparing A2C, FN and $\mathcal{V}_{\text{all}}$ suggests WN is a highly effective way to structure Manager subgoals. For example, in small *MultiGoal*, $\mathcal{V}_{\text{all}}$ (4.73 ± 0.5) surpasses FN (2.93 ± 0.74) by a large margin. We posit that the Manager tends to select g_w from a certain smaller subset of $\mathcal{V}$, simplifying the learning of transitions between g_w's for the Worker. As a result, the Worker can focus on solving local objectives. The same reasoning conceivably explains why $\mathcal{G}_w$ traversal does not yield performance gains on small and medium *MultiGoal*. For instance, $\mathcal{V}_p$ on small *MultiGoal* scores 5.25 ± 0.13, slightly higher than with traversal (5.05 ± 0.13). However once mazes become large, the Worker struggles to master traversals on its own and thus starts to fail the tasks.

World Graph Traversal In the case described above, the addition of world graph traversal plays an essential role, e.g. for large *MultiGoal*. As we conjectured in Section 4.3, this phenomenon can be explained by the much expanded exploration range and a lift of responsibility off the Worker to learn long distance transitions as a result of using $\mathcal{G}_w$ traversal. Moreover, Figure 4 confirms another conjecture from Section 4.3: $\mathcal{G}_w$ traversal speeds up convergence, more evidently with larger mazes. Lastly, in *Door-Key*, the agent needs to plan and execute a particular combination of actions. The huge discrepancy on medium *Door-Key* between using traversal or not, 75 ± 6 vs 56 ± 2, suggests $\mathcal{G}_w$ traversal indeed improves long-horizon planning.

Benefit of Learned Pivotal States Comparing $\mathcal{V}_p$ to $\mathcal{V}_{\text{rand}}$ reveals the quality of pivotal states identified by the latent model. Overall, $\mathcal{V}_p$ either performs better (particularly for non-deterministic environments) or similarly as $\mathcal{V}_{\text{rand}}$, but with much less variance between different seeds. If one luckily picks a set of random states suitable for a task, it can deliver great results but the opposite is equally possible. In addition, edge formation over $\mathcal{V}_{\text{rand}}$ still depends on the products from learning world graph, hence using pivotal states with its coupled $\mathcal{G}_w$ is more favorable over $\mathcal{V}_{\text{rand}}$.

6 Related Works

Pivotal state discovery is related to unsupervised sequence segmentation [17, 46, 75, 13, 16] and option or sub-task discovery in the context of RL [47, 6, 71, 30, 55, 52, 51]. Among them, both [51] and [75] employ sequential variational models to infer either task boundaries or key frames of demonstrations in an unsupervised manner, followed by applications to hierarchical RL or planning.

Besides technical differences, instead of turning points for individual sequences, our module aims to identify a set of landmark states that all together can represent the world well. Also, our training examples come from carefully orchestrated exploration rather than human demonstrations.

Understanding the world is central to planning, control and (model-based) RL. In robotics, one often needs to locate or navigate itself by interpreting a map of the world [60, 86, 3]. Our exploration strategy borrows the high-level insight from robotics active localization, where robots are actively guided to investigate unfamiliar regions by humans [29, 57]. Another direction in this area is to learn a world model [4, 37, 36] that generates latent states [87, 38, 76]. If the dynamics of the world are also learned, then it can be applied to planning [64, 39] or model-based RL [33, 49]. Although involving generative modeling, our framework differentiates itself through the functionality of our binary latent variables—indicators of whether a state, regardless of its representation, is a pivotal state.

The policy-learning phase in our framework uses the paradigm of goal-conditioned HRL [56, 21, 67, 89]. In addition, th WN mechanism borrows ideas from attentive object understanding [31, 5, 94] in vision. World graph traversal is inspired by classic optimal planning in Markov Decision Processes with dynamic programming [9, 27, 90, 84]. Lastly, initialization with goal-conditioned policies to transfer knowledge is inspired by transfer learning [23, 85] and skill generalization in RL [7, 40, 32].

Concurrently, [26] also proposes to plan a sequence of subgoals leading to a final destination according to a graph abstraction of the world that is obtained via goal-conditioned policy. However, under a different problem setup and use-case assumptions, the nodes in [26] are not learned pivotal states, but directly come from a replay buffer; similarly, their graph is task-specific whereas ours is designed to assist a variety of downstream tasks.

7 Conclusion and Future Work

We propose a general two-stage framework to learn a concise world abstraction as a simple directed graph with weighted edges that facilitates HRL for a diversity of downstream tasks. Our thorough ablation studies on several challenging maze tasks show clear advantage of each proposed innovative component in our framework.

The framework can be extended to other types of environments, such as partially observable and high-dimensional ones, through, e.g., probabilistic or differentiable planning [48, 93, 79], latent embedding of (belief) states [36] and goals [66]. Other directions are to adapt our framework to evolving or constantly changing environments, through, e.g., meta-learning [28], and/or to include off-policy methods to achieve better sample efficiency. Finally, the learned world graphs can potentially be applied beyond HRL, e.g., structured exploration by using pivotal state as checkouts [25] or in multiagent settings [15, 44].

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] J. Achiam and S. Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

[3] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, pages 1027–1037, 2008.

[4] M. G. Azar, B. Piot, B. A. Pires, J.-B. Gril, F. Altche, and R. Munos. World discovery model. *arXiv*, 2019.

[5] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[6] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[7] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

[8] J. Bastings, W. Aziz, and I. Titov. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 2019 Conference of the Association for Computational Linguistics, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2019.

[9] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. 1995.

[10] D. P. Bertsekas. *Nonlinear Programming*. 1999.

[11] N. Biggs. *Algebraic Graph Theory*. 1993.

[12] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[13] D. M. Blei and P. J. Moreno. Topic segmentation with an aspect hidden markov model. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 343–348. ACM, 2001.

[14] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[15] L. Buşoniu, R. Babuška, and B. De Schutter. Multi-agent reinforcement learning: An overview. In *Innovations in multi-agent systems and applications-1*, pages 183–221. Springer, 2010.

[16] W. Chan, Y. Zhang, Q. Le, and N. Jaitly. Latent sequence decompositions. *arXiv preprint arXiv:1610.03035*, 2016.

[17] M. Chatzigiorgaki and A. N. Skodras. Real-time keyframe extraction towards video content identification. In *2009 16th International conference on digital signal processing*, pages 1–6. IEEE, 2009.

[18] M. Chevalier-Boisvert and L. Willems. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

[19] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.

[20] J. D. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. *arXiv preprint arXiv:1806.02813*, 2018.

[21] P. Dayan and G. E. Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–278, 1993.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, 2018.

[23] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[24] Z. Dwiel, M. Candadi, M. Phielipp, and A. Bansal. Hierarchical policy learning is sensitive to goal space design. *arXiv preprint*, (2), 2019.

[25] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[26] B. Eysenbach, R. Salakhutdinov, and S. Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *arXiv preprint arXiv:1903.00606*, 2019.

[27] Z. Feng, R. Dearden, N. Meuleau, and R. Washington. Dynamic programming for structured continuous markov decision problems. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 154–161. AUAI Press, 2004.

[28] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135, 2017.

[29] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3-4):195–207, 1998.

[30] R. Fox, S. Krishnan, I. Stoica, and K. Goldberg. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*, 2017.

[31] G. Fritz, C. Seifert, L. Paletta, and H. Bischof. Attentive object detection using an information theoretic saliency measure. In *International workshop on attention and performance in computational vision*, pages 29–41. Springer, 2004.

[32] D. Ghosh, A. Gupta, and S. Levine. Learning actionable representations with goal-conditioned policies. *arXiv preprint arXiv:1811.07819*, 2018.

[33] K. Gregor and F. Besse. Temporal difference variational auto-encoder. *arXiv preprint arXiv:1806.03107*, 2018.

[34] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. In *ICML*, 2015.

[35] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838, 2016.

[36] Z. D. Guo, M. G. Azar, B. Piot, B. A. Pires, T. Pohlen, and R. Munos. Neural predictive belief representations. *arXiv preprint arXiv:1811.06407*, 2018.

[37] D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[38] T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine. Latent space policies for hierarchical reinforcement learning. *arXiv preprint arXiv:1804.02808*, 2018.

[39] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

[40] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[41] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[42] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR)*, 2017.

[43] I. Higgins, N. Sonnerat, L. Matthey, A. Pal, C. P. Burgess, M. Bosnjak, M. Shanahan, M. Botvinick, D. Hassabis, and A. Lerchner. Scan: Learning hierarchical compositional visual concepts. *arXiv preprint arXiv:1707.03389*, 2017.

[44] J. Hu, M. P. Wellman, et al. Multiagent reinforcement learning: theoretical framework and an algorithm. Citeseer, 1998.

[45] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):21, 2017.

[46] D. Jayaraman, F. Ebert, A. A. Efros, and S. Levine. Time-agnostic prediction: Predicting predictable video frames. *arXiv preprint arXiv:1808.07784*, 2018.

[47] Y. Jinnai, J. W. Park, D. Abel, and G. Konidaris. Discovering options for exploration by minimizing cover time. *arXiv preprint arXiv:1903.00606*, 2019.

[48] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[49] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[50] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2013.

[51] T. Kipf, Y. Li, H. Dai, V. Zambaldi, E. Grefenstette, P. Kohli, and P. Battaglia. Compositional imitation learning: Explaining and executing one task at a time. *arXiv preprint arXiv:1812.01483*, 2018.

[52] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1503–1510. IEEE, 2015.

[53] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[54] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[55] A. Léon and L. Denoyer. Options discovery with budgeted reinforcement learning. *arXiv preprint arXiv:1611.06824*, 2016.

[56] A. Levy, R. Platt, and K. Saenko. Hierarchical actor-critic. *arXiv preprint arXiv:1712.00948*, 2017.

[57] A. Q. Li, M. Xanthidis, J. M. O'Kane, and I. Rekleitis. Active localization with dynamic obstacles. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1902–1909. IEEE, 2016.

[58] M. L. Littman. Algorithms for sequential decision making. 1996.

[59] C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

[60] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2015.

[61] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[62] B. Marthi and C. Guestrin. Concurrent hierarchical reinforcement learning. 2005.

[63] D. Mishkin and J. Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[64] V. Mnih, J. Agapiou, S. Osindero, A. Graves, O. Vinyals, K. Kavukcuoglu, et al. Strategic attentive writer for learning macro-actions. *arXiv preprint arXiv:1606.04695*, 2016.

[65] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[66] O. Nachum, S. Gu, H. Lee, and S. Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018.

[67] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

[68] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.

[69] E. Nalisnick and P. Smyth. Stick-breaking variational autoencoders. *arXiv preprint arXiv:1605.06197*, 2016.

[70] NASA. The scientific objectives of the mars exploration rover. 2015.

[71] S. Niekum and S. Chitta. Incremental semantically grounded learning from demonstration. 2013.

[72] G. Ostrovski, M. G. Bellemare, A. v. d. Oord, and R. Munos. Count-based exploration with neural density models. *ICML*, 2017.

[73] Y. P. Pane, S. P. Nageshrao, and R. Babuška. Actor-critic reinforcement learning for tracking control in robotics. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 5819–5826. IEEE, 2016.

[74] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[75] K. Pertsch, O. Rybkin, J. Yang, K. Derpanis, J. Lim, K. Daniilidis, and A. Jeable. Keyin: Discovering subgoal structure with keyframe-based video prediction. *arXiv*, 2019.

[76] S. Racanière, T. Weber, D. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, et al. Imagination-augmented agents for deep reinforcement learning. In *Advances in neural information processing systems*, pages 5690–5701, 2017.

[77] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[78] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[79] S. M. Ross. *Introduction to stochastic dynamic programming*. Academic press, 2014.

[80] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[81] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, Z. Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

[82] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[83] W. Shang, H. van Hoof, and M. Welling. Stochastic activation actor-critic methods. In *ECML-PKDD*, 2019.

[84] R. S. Sutton. *Introduction to reinforcement learning*, volume 135. 1998.

[85] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[86] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[87] Y. Tian and Q. Gong. Latent forward model for real-time strategy game planning with incomplete information. In *NIPS Symposium*, 2017.

[88] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[89] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3540–3549. JMLR. org, 2017.

[90] G. Weiss. Dynamic programming and markov processes., 1960.

[91] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *NIPS*, 2017.

[92] Y. Wu and Y. Tian. Training agent for first-person shooter game with actor-critic curriculum learning. 2016.

[93] A. Yamaguchi and C. G. Atkeson. Neural networks and differential dynamic programming for reinforcement learning problems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5434–5441. IEEE, 2016.

[94] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659, 2016.