Learning World Graphs to Accelerate Hierarchical Reinforcement Learning

Wenling Shang [1 2] **Alex Trott** [* 2] **Stephan Zheng** [* 2] **Caiming Xiong** [2] **Richard Socher** [2]

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, a high-level Manager uses the world graph to quickly find solutions to new tasks and expresses subgoals in reference to their nearby pivotal states to a low-level Worker. The Worker can then also use the graph to traverse and explore in long range. 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, a Mars rover carries out scientific objectives ranging from searching for rocks to calibrating orbiting instruments (NASA, 2015). 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 enables agents to learn *high-level world structure* in the form of *a simple graph* (Biggs, 1993) and integrate this into a hierarchical policy model.

In the initial stage, we alternate between exploring and updating a descriptor of the world in a graph format (Biggs, 1993), 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 (Chatzigiorgaki & Skodras, 2009; Jayaraman et al., 2018; Ghosh et al., 2018). In particular, given a set of trajectories, we optimize a fully differentiable recurrent variational auto-encoder (Chung et al., 2015; Gregor et al., 2015; Kingma & Welling, 2013) with binary latent variables (Nalisnick & Smyth, 2016), each designated to a state whose prior distribution (conditioning on the state) is learned and 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 (Ha & Schmidhuber, 2018) or curiosity-driven ones (Azar et al., 2019). Our trajectory-collecting agent uses both random walks and a simultaneously learned curiosity-driven goal-conditioned policy (Ghosh et al., 2018; Nair et al., 2018). The agent also initiates exploration from the current set of pivotal states, similar to the "cells" in Go-Explore (Ecoffet et al., 2019), 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 in the next stage (Taylor & Stone, 2009).

At first glimpse, the world graph seems suitable for model-based RL (Littman, 1996; Kaiser et al., 2019), but our method emphasizes the connections among neighboring pivotal states rather than transitions over any arbitrary pair, which is usually deemed as a much harder problem (Gu et al., 2016). Therefore, in the next stage, we propose a hierarchical reinforcement learning (Kulkarni et al., 2016; Marthi & Guestrin) (HRL) approach to incorporate the world graph for solving specific downstream tasks. Concretely, within the paradigm of goal-conditioned HRL (Dayan & Hinton, 1993; Nachum et al., 2018b; Vezhnevets et al., 2017; Levy et al., 2017), 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

[*]Equal contribution [1]University of Amsterdam [2]Salesforce Research. Correspondence to: Wendy Shang <w.shang@uva.nl>.

Real-world Sequential Decision Making workshop at ICML 2019.

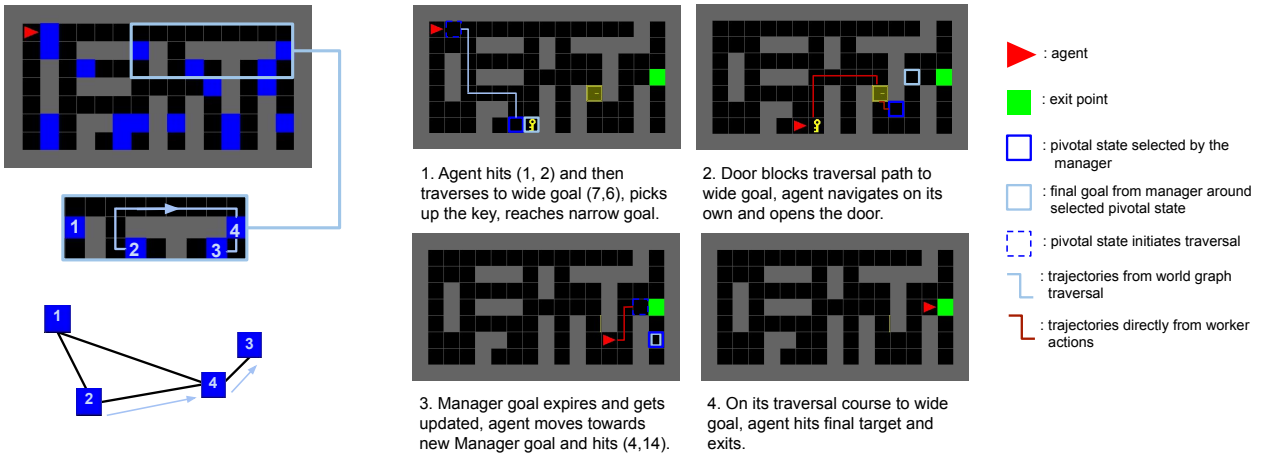

Figure 1: Left: a subgraph exemplifies how to forge edges and traverse between pivotal states (in blue). Right: An example rollout from our proposed HRL policy with Wide-then-Narrow Manager instructions and world graph traversals, solving a challenging Door-Key task.

pivotal state. Such sequential selection is referred to as the *Wide-then-Narrow* (WN) instruction. In this way, as navigating from its nearby pivotal state to the desired one is greatly simplified thanks to applying graph traversal techniques (Bertsekas, a) on the world graph, the Worker can focus more on achieving 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 and navigation via world graph traversal.
- 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 complex 2D mazes (Chevalier-Boisvert & Willems, 2018) that are finite, fully observable and deterministic (over each episode) as our test-bed, i.e. for each state-action pair, the transition $(s_t, a_t) \rightarrow s_{t+1}$ is deterministic, where $s_t \in \mathcal{S}, a_t \in \mathcal{A}$ 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 (Higgins et al., 2017; Dwiel et al., 2019; Nachum et al., 2018a); Section 6 briefly speculates on extensions of our framework to other environments as future directions. We design 3 mazes of small, medium and large sizes with varying compositions (see Appendix for visualization). Despite their finite, fully observable and deterministic nature, these mazes—especially the larger ones—still pose much challenge, especially when the downstream task is of sparse reward or 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 Supplementary Materials.

3. World Graph Discovery

We envision *a directed simple graph* (Biggs, 1993) $\mathcal{G}_w$ 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 feasible traversals among the nodes. Drawing intuition from unsupervised sequence segmentation (Chatzigiorgaki & Skodras, 2009; Jayaraman et al., 2018) and imitation learning (Abbeel & Ng, 2004; Hussein et al., 2017), 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 (Azar et al., 2019). In other words, given a trajectory $\tau = \{(s_t, a_t)\}_0^T$, we learn to identify the most needed state subset $\{s_t | s_t \in \mathcal{V}_p\}$ to infer the action sequence taken in τ with a decent accuracy.

Supposing the state-action trajectories are available, we formulate a recurrent variational inference model (see Section 3.1) (Blei et al., 2017; Chung et al., 2015; Gregor et al., 2015; Kingma & Welling, 2013), treating the action se-

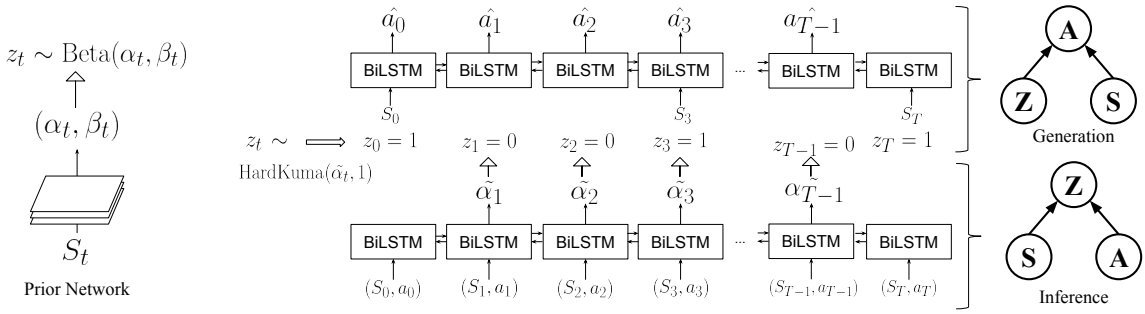

Figure 2: The pivotal state discovery model. A prior network (left) learns the state-conditioned prior in Beta distribution, $p_\psi(z_t|s_t) = \text{Beta}(\alpha_t, \beta_t)$. An inference encoder learns an approximate posterior in HardKuma distribution (Basting & et al, 2019) inferred from (s_t, a_t)'s, $q_\phi(z_t|a_t, s_t) = \text{HardKuma}(\tilde{\alpha_t}, 1)$. 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 (Kingma & Welling, 2013).

quences as evidence and inferring whether to keep a state for action recovery in a binary latent variable. We learn a prior over each latent z_t conditioned on its associated 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 the policy used by the agent to collect training trajectories. When collecting training trajectories, we place the agent at a state from the current iteration's set of $\mathcal{V}_p$—it 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. While straightforward to implement, a random walk rollout policy can result in noisy trajectories that are perhaps irrelevant to real tasks. We instead take inspiration from prior work on actionable representations (Ghosh et al., 2018) and learn a goal-conditioned policy π_g for navigating between close-by states, reusing its 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 is repeated until the action reconstruction accuracy plateaus. To complete $\mathcal{G}_w$ and form the edges, 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.

The pseudo-code summarization of world graph discovery, implementation details, a visualization of how $\mathcal{V}_p$ progresses over training, the final $\mathcal{V}_p$ from different rollout policies and the model architecture are provided in the Appendix. The following sections are used to 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. The evidence lower bound to optimize is

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

The reconstruction 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's (Russo et al., 2018). 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 useful s_t is for action reconstruction. In this way, regularizing the approximated posterior with the learned prior (KL-divergence term) encourages similar trajectories to use the same states for action reconstruction. We define $\mathcal{V}_p$ as the top 20% states with the largest learned prior means.

We model the approximate posteriors using the Hard Kumaraswamy distribution (Basting & et al, 2019) $[\text{HardKuma}(\tilde{\alpha}_t, \tilde{\beta}_t)]$ which resembles 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 (Basting & et al, 2019; Louizos et al., 2017) and applicability of the reparameterization trick thanks to its simple form of CDF (Kingma & Welling, 2013; Rezende et al., 2014; Maddison et al., 2016). Lastly, the KL-divergence between Kuma distribution and Beta distribution can be approximated in closed form (Nalisnick & Smyth, 2016). 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 (Louizos et al., 2017; Basting & et al, 2019). In other words, this objective constraints there should be μ_0 of activated $z_t = 1$. 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 $\mathcal{L}_0$ and $\mathcal{L}_T$ have closed forms for HardKuma.

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

Lagrangian Relaxation. The overall optimization objective consists of action sequence reconstruction, KL-divergence, $\mathcal{L}_0$ and $\mathcal{L}_T$. We tune the objective weights λ_i using Lagrangian relaxation (Higgins et al.; Basting & et al, 2019; Bertsekas, b), 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,

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

Our finalized latent model allows efficient and stable mini-batch training. Alternative designs, such as Poisson prior (Kipf et al., 2018) for latent space and Transformer (Vaswani et al., 2017) for sequential modeling, are also possibilities for future investigation. More mathematical 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 (Ghosh et al., 2018). 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 requires π_g to reach goals over a short range. We train such a policy by sampling goals using a random walk from a given starting state. Inspired by the success of intrinsic motivation methods (in particular, curiosity (Burda et al., 2018; Achiam & Sastry, 2017; Pathak et al., 2017; Azar et al., 2019)), 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 found 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 (Biggs, 1993) 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 collect the shortest such paths as the graph edges. Each path is further refined by π_g, using trajectories collected from the policy when substituting s_p and s_q for the starting state and goal state. For stochastic or partially observable environments, we may entirely count on π_g rather than path memorization.

4. Accelerated Hierarchical Reinforcement Learning

We now introduce a hierarchical reinforcement learning (Kulkarni et al., 2016; Marthi & Guestrin) (HRL) module that leverages the world graph $\mathcal{G}_w$ to accelerate learning downstream tasks. There are three core innovations, namely Wide-then-Narrow Manager instruction (Section 4.2), $\mathcal{G}_w$ traversal (Section 4.3), and knowledge transfer via initialization from π_g (Section 4.4). Each is generally applicable to many different HRL algorithms. We compare our method to an A2C baseline and its hierarchical extension, Feudal Network (Section 4.1). We provide the model architecture and all implementation details in the Appendix.

4.1. Preliminaries and Baselines

We consider the standard discrete time step, discounted RL setup. An agent, controlled by policy $\pi(a_t|s_{t-1})$ and receiving reward r_t at time t, is trained to maximize its cumulative expected return over time $R = \mathbb{E}_{(s_t,a_t)\sim\pi}[r_t]$. Advantage Actor-Critic (A2C), is a popular and relatively simple-to-use, model-free, on-policy RL algorithm (Wu & Tian, 2016; Pane et al., 2016; Mnih et al., 2016). We use the recurrent A2C-LSTM variant as a non-hierarchical baseline.As an on-policy algorithm, A2C learns a value function V for estimating future cumulative discounted reward given current state s_t and adjusts the probability given by the policy to actions based on the advantage of the observed reward compared to that predicted by V. Typically (and as we do here), policy optimization includes entropy regularization ($\mathcal{H}$) to prevent premature convergence. Feudal Network (FN) (Dayan & Hinton, 1993; Vezhnevets et al., 2017) is a hierarchical extension of A2C (Figure 4). It defines a high-level controller, i.e. Manager, which learns to propose subgoals to the low-level controller, i.e. Worker, which learns to complete the subgoals. The Manager receives rewards from the environment and the Worker receives rewards for completing the subgoals provided by the Manager. The high- and low-level

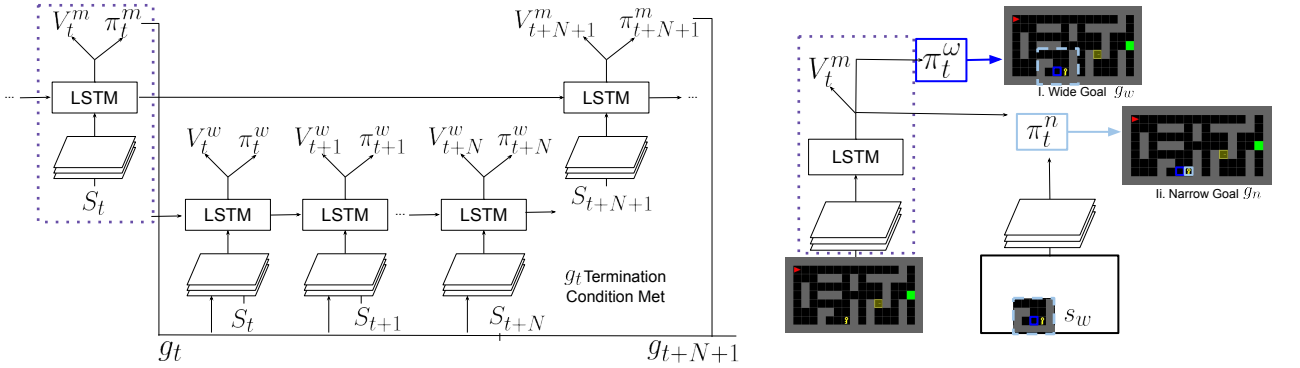

Figure 3: Left: a general configuration of Feudal Netowrk; Manager and Worker are both A2C-LSTMs operating at different temporal resolutions. Right: Our proposed Wide-then-Narrow Manager instruction, where Manager first outputs a wide goal g_w from a pre-defined set of candidate states $\mathcal{V}$,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

components learn distinct networks that operate at different temporal resolutions, such that the Manager only outputs a new subgoal if either Worker completes its current one or the subgoal horizon c is exceeded. Because the mazes in our setting are finite and fully observable, we can precisely characterize the set of subgoals. The baseline FN can select any well-defined state as a subgoal, i.e. the Manager policy network emits a probability vector of dimension $|\mathcal{S}|$.

4.2. Wide-then-Narrow Manager Instruction

To adapt the graph $\mathcal{G}_w$ from our unsupervised learning stage to the HRL framework, we similarly need a way to express any state as a subgoal while still constraining the Manager output according to the abstraction provided by the graph. To that end, we propose a *Wide-then-Narrow* (WN) mechanism for modeling Manager outputs. Given a pre-defined set of candidate states denoted $\mathcal{V}$, the Manager follows a "wide-goal" policy π^ω derived from global context s_t and outputs a "wide" subgoal $g_w \in \mathcal{V}$. We propose to use the learned *pivotal states* $\mathcal{V}_p$ as this set $\mathcal{V}$. After selecting this "wide-goal" g_w, the Manager zooms its attention to an $N \times N$ local area s_w around g_w. Taking into account both global s_t and local s_w information, a "narrow-goal" policy π^n selects a final, "narrow" goal $g_n \in s_w$, which is then together with the wide goal passed to the Worker as its next subgoal *pair* (g_w, g_n). The Worker is rewarded if it reaches g_w for the first time during current horizon or g_n. The policy gradient is straightforward to modify but the entropy regularization $\mathcal{H}$ easily becomes intractable when the state spaces grow large (see Appendix). In practice we resort to a functional approximation of $\mathcal{H}$ and obtain the final Manager policy network update, where $A_{m,t}$ is the Manager's advantage at time t:

$$\nabla_\theta A_{m,t} \log \pi^\omega\left(g_{w,t}|s_t\right) \pi^n\left(g_{n,t}|s_t, g_{w,t}, s_{w,t}\right) + \nabla_\theta \mathcal{H}\left(\pi^\omega\right) + \nabla_\theta \mathcal{H}\left(\pi^n(\cdot|g_{w,t})\right).$$

4.3. World Graph Traversal

By limiting the set of wide-goal options to pivotal states, i.e. set $\mathcal{V} = \mathcal{V}_p$, we can take advantage of the edge connections in the world graph. We hereby set forward an example on how. When the Worker is in pursuit of g_n in the neighborhood of g_w, we allow it to essentially re-use the traversal edges stored in $\mathcal{G}_w$ when it encounters a state that is part of the graph. Specifically, if the agent encounters a pivotal state $s_p \in \mathcal{V}_p$ such that there is a path on $\mathcal{G}_w$ to the wide-goal g_w, it can navigate from s_p to g_w along the path as if leveraging a documented repertoire of behaviors. The optimal traversal route can be estimated basing on edge information via e.g., in our case, dynamic programming (Sutton; Feng et al., 2004). If a new blockage in the environment (i.e. a door) makes the traversal unviable, we do not allow the Worker to hop onto the blocked path and expect the Manager to learn to plan according to this limitation. We demonstrate this behavior in our experiments. World graph traversal potentially allows the Manager to assign more task-relevant goals that are far away, speeding up training of high-level control by outsourcing basic planning of transportation. For the same reason, the Worker may also concentrate on learning to operate towards the localized g_n after arriving at g_w. Another foreseeable benefit is the enhancement of exploration, as the agent is no longer restricted to lingering around its current position.

4.4. Transfer from Goal-Conditioned Policy via Initialization

Lastly, we suggest further leveraging implicit knowledge of the world acquired by π_g in the subsequent HRL training. Transferring and generalizing skills between tasks in the context of RL is an important practice often leading to performance gain (Taylor & Stone, 2009; Barreto et al., 2017). In (Ghosh et al., 2018), the authors show how a goal-conditioned policy captures the underlying structure of the environment and actionable representations derived from such policy are beneficial for other tasks. Additionally,

Task	MultiGoal Dense Reward			MultiGoal Sparse Reward		
Maze	Small	Medium	Large	Small	Medium	Large
Baselines						
A2C	2.04±0.05	Fail	Fail	-0.10±0.34	Fail	Fail
FN	Fail	Fail	Fail	0.19±0.02	Fail	Fail
with π_g init	2.93±0.74	Fail	Fail	Fail	Fail	Fail
Wide-Narrow						
$\mathcal{V}_{\text{all}}$	Fail	Fail	Fail	Fail	Fail	Fail
$\mathcal{V}_{\text{rand}}$	Fail	Fail	Fail	Fail	Fail	Fail
$\mathcal{V}_p$	Fail	Fail	Fail	Fail	Fail	Fail
with π_g init						
$\mathcal{V}_{\text{all}}$	4.73±0.50	4.71±0.39	Fail	0.32±0.02	Fail	Fail
$\mathcal{V}_{\text{rand}}$	3.67±1.07	4.72±0.73	Fail	0.36±0.02	Fail	Fail
$\mathcal{V}_p$	**5.25**±0.13	**5.15**±0.11	Fail	0.39±0.09	Fail	Fail
with $\mathcal{G}_w$ traversal						
$\mathcal{V}_{\text{rand}}$	3.85±0.83	2.59±0.07	1.65±0.42	0.17±0.03	0.19±0.04	0.20±0.12
$\mathcal{V}_p$	3.92±0.22	2.56±0.09	2.18±0.12	0.20±0.04	0.20±0.04	0.16±0.02
init+traversal						
$\mathcal{V}_{\text{rand}}$	4.16±1.06	3.29±0.93	2.30±0.49	0.25±0.06	0.24±0.06	0.19±0.02
$\mathcal{V}_p$	5.05±0.03	3.00±0.90	**2.72**±0.50	**0.42**±0.07	**0.25**±0.03	**0.26**±0.11

Door-Key	π_g init			$\mathcal{G}_w$ traversal		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: Experimental results over MultiGoal and MultiGoal-Sparse on small, medium, and large mazes (average reward ± std). Bottom: Experimental results over Door-Key task on small and medium mazes (average success rate in % ± std). "Fail" means training is either not initiated or validation rewards are never above 0. We omit reporting some suboptimal models on Door-Key for clearer presentation.

optimization of deep neural networks is sensitive to weight initialization (Mishkin & Matas, 2015; Le et al., 2015), especially for a system (Co-Reyes et al., 2018) like HRL due to its complexity and lack of clear supervision. Therefore, we attempt to achieve both implicit skill transfer and improved optimization by using the weights from π_g to initialize the Worker and the Manager.

5. Experiments

We validate the effectiveness and assess the impact of each component of our framework in a thorough ablation study on three challenging maze tasks with different reward structures and logic. All implementation details, e.g., hyperparameters, are in the Appendix.

Experimental Setup We use 3 challenging maze tasks for ablation studies. 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 $r_t = +1$. Its sparse version, *MultiGoal-Sparse*, only gives a single reward $r \leq 1$ proportional to the number of balls collected upon exiting. *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 $+1$), see Figure 4. Lastly, every action taken by the agent receives a negative reward -0.01.

A2C and FN are our non-hierarchical and hierarchical baselines. We then augment WN on top of FN with 1 of 3 possible 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. For $\mathcal{V}_{\text{rand}}$ and $\mathcal{V}_p$[1], we compute their edge connections as illustrated in Section 3.3 and add $\mathcal{G}_w$ traversal on top of WN. Notice neither π_g nor guaranteed state access is available to $\mathcal{V}_{\text{rand}}$, but we grant all pre-requisites to the random case for the fairest comparison possible. Finally, we repeat all experiments with π_g initialization. Again, π_g is given to $\mathcal{V}_{\text{all}}$ and $\mathcal{V}_{\text{rand}}$ for free.

We inherit most hyperparameters from the training of π_g, as the Manager and the Worker both share similar architecture as π_g. The hyperparameters of π_g in turn follow those from (Shang et al., 2018). Because these tasks are more difficult than goal-orientation, we increase the maximal number of training iterations from 36K to 100K and the rollout steps for each iteration from 25 to 60. Hyperparameters specific to HRL are the horizon $c = 20$ and the size of s_w, $N = 5$ for small and medium, $N = 7$ for large. We follow a rigorous evaluation protocol acknowledging the variability in Deep RL (Henderson et al., 2018): each experiment is repeated with 3 seeds (Wu et al., 2017; Ostrovski et al., 2017), 10 additional validation seeds are used to pick the best model which is then tested on 100 testing seeds. Mean and variance of testing results are summarized in Table 1.

5.1. Result Discussion

Transfer via Initialization 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

[1] $\mathcal{V}_{\text{all}}$ is a trivial case excluded here as every state is 1 step away from its adjacent states.

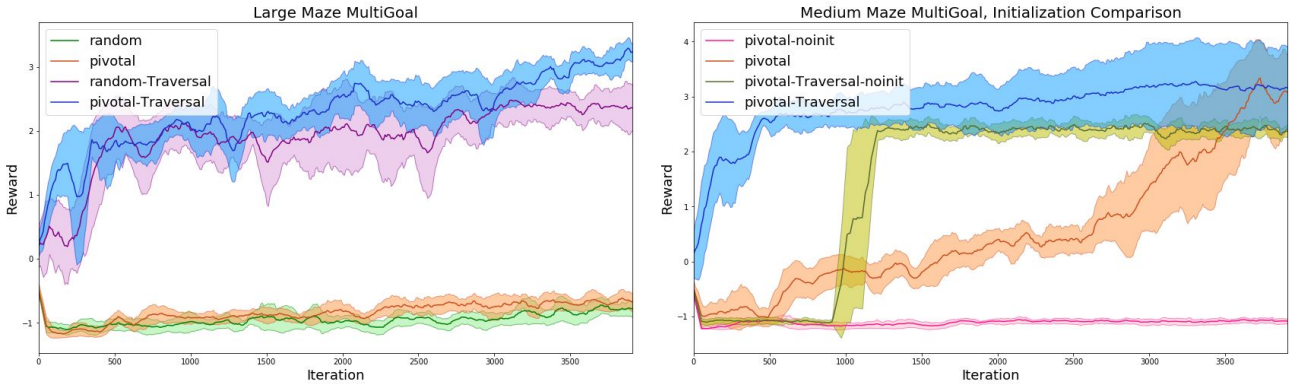

Figure 4: Validation curves during training (average reward over 3 seed ± std) for MultiGoal task with dense reward. Left: Compare between $\mathcal{V}_p$ and $\mathcal{V}_{\text{rand}}$, with or without traversal, all models here use WN and π_g initialization. Observe that (1) traversal evidently speeds up convergence (2) $\mathcal{V}_{\text{rand}}$ tends to carry 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.

MultiGoal. Models starting from scratch fail on almost all tasks unless coupled with $\mathcal{G}_w$ traversal, which is still inferior to their π_g initialized counterparts. These results also corroborate the claim from (Ghosh et al., 2018) that goal-conditioned policies are a promising venue for task transfer.

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 transition learning becomes more difficult with larger mazes, the Worker starts to fail discovering these transitions and at the end also the task, e.g., on large MultiGoal.

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 that $\mathcal{G}_w$ 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.

Role of $\mathcal{V}_p$ Comparing $\mathcal{V}_p$ to $\mathcal{V}_{\text{rand}}$ intuits the quality of pivotal states identified by the latent model. Over all, $\mathcal{V}_p$ either exhibits better or comparable results 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. Besides, edge formation between the random states is still procured from the learning of $\mathcal{G}_w$. Therefore the favorable performance of $\mathcal{V}_{\text{rand}}$ does not undermine the value of world graph discovery.

6. Future Work and Conclusions

We propose a two-stage framework to 1) learn a concise world abstraction as a graph and 2) apply it to accelerate HRL for specific downstream tasks. Our thorough ablation studies on several challenging finite state fully observable 2D mazes show clear advantage of each proposed innovative component.

Future work will extend the framework to other types of environments such as partially observable, stochastic and high dimensional environments, through e.g. probabilistic planning (Kaelbling et al., 1998), latent embedding of (belief) states (Guo et al., 2018) and goals (Nachum et al., 2018a). It is also worth investigating a principled way to adapt our framework to evolving or constantly changing environments through for instance meta-learning (Finn et al., 2017). Another direction is to engage off-policies training to achieve better sample efficiency. Finally, we'd like to test applying the learned world graph beyond HRL, such as in structured exploration by using pivotal state as checkouts (Ecoffet et al., 2019) or in a multiagent setting (Buşoniu et al., 2010; Hu et al.).

References

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

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

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

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

Basting, J. and et al. Interpretable neural predictions with differentiable binary variables. 2019.

Bertsekas, D. P. *Dynamic programming and optimal control*, volume 1. a.

Bertsekas, D. P. *Nonlinear Programming*. b.

Biggs, N. *Algebraic Graph Theory*. 1993.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework.

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

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

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

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

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

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

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

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

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

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

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

Littman, M. L. Algorithms for sequential decision making. 1996.

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

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

Marthi, B. and Guestrin, C. Concurrent hierarchical reinforcement learning.

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

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

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

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

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

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

NASA. The scientific objectives of the mars exploration rover. 2015.

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

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

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

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

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

Shang, w., van Hoof, h., and Welling, m. Stochastic activation actor-critic methods. 2018.

Sutton, R. S. *Introduction to reinforcement learning*, volume 135.

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

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

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

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

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