Transfer Learning via Temporal Contrastive Learning

Weihao Zeng

CMU-RI-TR-24-10

April 16

School of Computer Science
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee:
Katia Sycara, *chair*
Andrea Bajcsy
Sha Yi

Submitted in partial fulfillment of the requirements for the Degree of Master of Science in Robotics

To shiba-inus.

Abstract

This thesis introduces a novel transfer learning framework for deep reinforcement learning. The approach automatically combines goal-conditioned policies with temporal contrastive learning to discover meaningful sub-goals. The approach involves pre-training a goal-conditioned agent, finetuning it on the target domain, and using contrastive learning to construct a planning graph that guides the agent via sub-goals. Experiments on PointMaze and multi-agent coordination Overcooked tasks demonstrate improved sample efficiency, the ability to solve sparse-reward and long-horizon problems, and enhanced interpretability compared to baselines. The results highlight the effectiveness of integrating goal-conditioned policies with unsupervised temporal abstraction learning for complex multi-agent transfer learning. Compared to state-of-the-art baselines, our method achieves the same or better performances while requiring only 23.4% of the training samples.

Acknowledgments

I am extremely grateful to my advisor, Dr. Katia Sycara, who introduced me to the field of Explainable AI and multi-agent systems. Her invaluable support, vast experiences, and guidance have led to the development of this thesis. I am also thankful to my committee members, Dr. Andrea Bajcsy and Sha Yi, for their feedback and insightful comments on my work. I would also like to thank Dana Hughes, Ini Oguntola, Joseph Campbell, Renos Zabounidis, and Simon Stepputtis for their intellectual contributions and help during this project. To my labmates and friends, Fan Jiang, Guo Yue, Muhan Lin, Sarthak Bhagat, Seth Karten, Shuyang Shi, and Yuzhe Lu, thank you for all the stress-busting and stressing-together conversations.

Last, I must express heartfelt gratitude to my family for their unconditional love and for being my source of strength at every stage.

Funding

This project is sponsored by the Artificial Social Intelligence for Successful Teams (ASIST) program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Reinforcement Learning (RL) has emerged as a powerful framework for solving decision-making problems by enabling agents to learn optimal policies through interactions with the environments. It has achieved remarkable success in various challenging domains, such as robotics [2, 27, 28] and game-playing [44, 49]. Despite its impressive advancements, RL often struggles with sample inefficiency, requiring many interactions with the environment to learn effective policies [52]. This problem is exacerbated in multi-agent systems, where the size of the state and action space increase combinatorially with the number of agents. Moreover, sparse rewards and partial observability can further worsen the sample inefficiency of RL algorithms [52].
Transfer learning (TL) has emerged as a promising approach to address these challenges and improve the sample efficiency of RL. TL aims to leverage knowledge learned in a task to accelerate learning and boost performance in a target task [52]. The key idea behind TL is that related tasks often share common structures or features, which can be extracted and reused instead of learned from scratch. For example, skills learned in previous navigation tasks can transfer to new navigation goals or environment layouts [51].
This thesis introduces a novel transfer learning framework that integrates goal-conditioned reinforcement learning (GCRL) policies [38] with unsupervised temporal abstraction learning and graph-based planning to capture and exploit reusable knowledge across tasks. Our approach employs contrastive learning [10] to learn a compact representation of the temporal structure from agent trajectories and then transforms

this learned latent space into a graph through clustering. The resulting graph encodes abstract states as nodes representing clusters of similar states and temporal transitions between these clusters as edges. This graph structure enables efficient planning and sub-goal generation, guiding the GCRL policy in the target domain. Our approach consists of three main steps: 1. First, we train a GCRL agent by reaching diverse short-horizon goals in the source domain, enabling it to acquire diverse skills for reaching various goals. 2. Next, we finetune the GCRL agent on the target domain, learn a latent space of the temporal structure from the trajectories generated by the GCRL agent using contrastive learning [10], and construct a planning graph from the latent space. 3. Finally, we guide the GCRL agent using sub-goals generated from the planning graph to complete the task in the target domain.

We demonstrate the effectiveness of our proposed framework through extensive experiments across the PointMaze environment [33] and multiple multi-agent transfer scenarios on the Overcooked environment [9]. Our approach offers several key benefits, including: 1. improved sample efficiency when learning new tasks, 2. the ability to solve challenging sparse-reward or long-horizon tasks by leveraging the learned temporal abstractions and 3. enhanced the learning process's interpretability by discovering meaningful sub-goals and skills.

The main contributions of this thesis are:

1. We introduced a novel TL approach for RL that enables agents to learn new tasks efficiently by leveraging prior experience.
2. We combined goal-conditioned policies with unsupervised learning of temporal abstractions, enabling more sample-efficient and adaptable RL agents

Chapter 2

Background

This chapter provides the relevant technical backgrounds, related works, and problem definition.

2.1 Preliminaries

In this section, we introduce the technical backgrounds relevant to this thesis.

2.1.1 Reinforcement Learning

Reinforcement learning (RL) is a paradigm for agents to achieve pre-defined goals through interaction with the environment.

Markov Decision Process

Markov Decision Processes (MDPs) [5] offer a structured approach for modeling sequential decision-making in environments where outcomes are influenced by the environment's randomness and the agent's actions. An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p_0, \gamma)$. Here, $\mathcal{S}$ represents the set of states in the environment; $\mathcal{A}$ is the set of actions available to the agent; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function defining the probability of moving from one state to another given an action; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ assigns a reward to each transition between states under an

action; $p_0 : \mathcal{S} \to [0, 1]$ specifies the initial state distribution; and $\gamma \in [0, 1]$ is the discount factor for future rewards.

An extension to MDP is the Partially Observable Markov Decision Processes (POMDPs) [3], which are defined by $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, p_0, \gamma)$, where $\mathcal{O} : S \to \Omega$ maps states to a set of observations Ω. Instead of having access to the complete state information, the agent in POMDPs only has access to partial observations.

Policy Gradients

Policy gradient methods are a category within reinforcement learning that aims to optimize a policy directly to maximize the expected total discounted rewards, denoted as $E_{\pi_\theta}[\sum_{t=0}^{T} \gamma^t R_t]$, where T represents the horizon of the environment, π_θ denotes a policy parameterized by θ. Policy gradient methods compute an estimator of the policy gradients and optimize the policy parameters via the gradient ascent algorithm. The most common gradient estimator has the form $\hat{g} = E_t[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t]$, where $\hat{A}_t$ is an estimator of the advantage function at timestep t.

Various trade-offs between bias and variance exist in the choice of the advantage estimator. A standard option is the generalized advantage estimator $GAE(\gamma, \lambda)$ [41], where λ controls the bias-variance trade-off. Higher λ values result in lower bias but higher variance in the estimator, and vice versa.

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V, \text{ where } \delta_t^V = -V(s_t) + r_t + \gamma V(s_{t+1}) \tag{2.1}$$

Proximal Policy Optimization (PPO) [42] aims to improve standard policy gradient methods by employing a clipped objective function, preventing large policy updates, and ensuring more stable and efficient learning. This approach encourages moderate adjustments to the policy.

$$L^{CLIP}(\theta) = \mathbb{E}\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{2.2}$$

Here, $r_t = \frac{\pi_\theta(at_t|s_t)}{\pi_{\theta_{\text{old}}}(at_t|s_t)}$, and ϵ is a hyperparameter that controls the degree in which the updated policy can deviate from the old policy.

2.1.2 Contrastive Representation Learning

Contrastive representation learning is a self-supervised learning approach that aims to learn meaningful representations by contrasting positive pairs against negative pairs. The goal is to learn an embedding space where similar samples are pulled together, and dissimilar ones are pushed apart.

The basic contrastive loss, also known as the triplet loss [40], is defined for a triplet (x, x^+, x^-), where x is an anchor point, x^+ is a positive example (i.e., a point similar to the anchor), and x^- is a negative example (i.e., a point dissimilar to the anchor). The loss function encourages the anchor-positive distance to be smaller than the anchor-negative distance by a margin ϵ:

$$\mathcal{L}(x, x^+, x^-) = \sum_{x \in X} (0, d(x, x^+) - d(x, x^-) + \epsilon) \tag{2.3}$$

where $d(\cdot, \cdot)$ is a distance metric, such as Euclidean distance.

InfoNCE (Information Noise-Contrastive Estimation) [31] generalizes the contrastive loss to handle multiple positive and negative examples. Given a positive pair (i, j), the InfoNCE loss is defined as:

$$\mathcal{L}_{\text{InfoNCE}} = -\mathbb{E}\left[\log \frac{f(x, c)}{\sum_{x' \in X} f(x', c)}\right] \tag{2.4}$$

where $f(x, c) = \exp(x^T W c)$, W is a trainable weight matrix.

Large batch sizes and hard negative mining are crucial to successfully training contrastive models. SimCLR [10] demonstrates that larger batch sizes can significantly improve the quality of the learned representations by providing more negative examples. Hard negative mining techniques, such as the ones used in [36], focus on selecting the most informative negative examples to improve the contrastive learning process.

2.2 Related Works

In this section, we introduce the related works to this thesis.

2.2.1 Goal-Conditioned Reinforcement Learning

Goal-conditioned reinforcement learning (GCRL) is a framework where an agent learns to achieve a specified goal state instead of maximizing a scalar reward signal. Schaul et al. [38] introduced the concept of universal value function approximators (UVFA), which extends the standard value function to consider goal states. Andrychowicz et al. [1] proposed Hindsight Experience Replay (HER), a technique that allows the agent to learn from failures by treating the achieved state as the desired goal state. Recent works have extended GCRL to handle multi-goal scenarios [34] and hierarchical goal-setting [24, 30].

Exploration is crucial for GCRL, especially in sparse reward settings. Go-Explore [12] addresses this by building an archive of diverse, high-performing states during exploration and learning a policy to reach these states reliably. Skew-Fit [35] introduces a goal sampling scheme that favors goals of intermediate difficulty, encouraging exploration and learning. DISCERN [25] learns a goal-conditioned policy using an unsupervised reward function that promotes exploration and skill discovery. Plan2Explore [43], LEXA [29] and PEG [20] build on top DreamerV2 [18] and promote exploration during training.

2.2.2 Contrastive Representation Learning in Robotics

Contrastive learning has been successfully applied to robotics for learning state and reward representations. Srinivas et al. [22] proposed the Contrastive Unsupervised Representations for Reinforcement Learning (CURL) framework, which learns a contrastive representation of raw pixels to improve sample efficiency in robotic control tasks. Zhan et al. [50] introduced a framework for learning robotic manipulation skills using contrastive learning, demonstrating improved performance and generalization. Other works have utilized contrastive learning for various aspects of robotic learning. Singh et al. [45] employed contrastive learning to learn reward functions, while Laskin et al. [23] used it to learn invariant representations. Florence et al. [14] and Cao et al. [7] trained view-angle invariant contrastive representations to improve robotic manipulation tasks, enabling the agent to handle variations in object poses and camera viewpoints. Cao et al. [8] proposed a method for learning sim-to-real pixel-to-pixel consistent contrastive representations, which allows for zero-shot transfer of policies

learned in simulation to real-world robotic manipulation tasks. [32] and [32] used contrastive learning to learn a mapping from states to latent representation that preserves the temporal structure.

2.2.3 Hierarchical Reinforcement Learning

Hierarchical reinforcement learning (HRL) aims to learn a hierarchy of policies operating at different abstraction levels. The goal is to break down a complex task into simpler subtasks, which can be learned more efficiently. Sutton et al. [46] introduced the options framework, which extends the standard MDP to include temporally extended actions. Bacon et al. [4] proposed the Option-Critic architecture, which simultaneously learns the policy over options and the options themselves. Recent works have explored learning goal-conditioned hierarchical policies [24, 30] and combining HRL with meta-learning [15].

2.2.4 Transfer Learning in Reinforcement Learning

Transfer learning in RL aims to leverage knowledge learned from one task to improve learning efficiency and performance in another related task. Zhu [52] provides a comprehensive survey of transfer learning methods in RL. Rusu et al. [37] introduced the Progressive Neural Networks (PNN) architecture, which allows for transferring knowledge across a sequence of tasks while avoiding catastrophic forgetting. Other approaches include learning invariant feature spaces [17], meta-learning for fast adaptation [13], and learning transferable representations [19].

Recent advancements include Distilling Policy Distillation [11], which combines policy distillation with teacher-student curriculum learning for efficient knowledge transfer, and Kickstarting Deep Reinforcement Learning [39], which uses human demonstrations in a source task to initialize policies in a target task, reducing exploration and improving learning efficiency. JumpstartRL [47] uses a guidance policy to help a new policy to learn in a curriculum setting.

2.3 Problem Definition

This thesis addresses the challenge of transfer learning in the context of reinforcement learning (RL). Transfer learning in RL entails adapting knowledge gained from one scenario to enhance learning efficiency and effectiveness in a related but distinct scenario [52]. By leveraging information extracted from the source domain alongside the inherent characteristics of the target domain, the aim is to accelerate the learning process and enhance the agent's performance in achieving desired objectives within the target environment. Transfer learning involves leveraging knowledge from a source domain, denoted as $\mathcal{M}_s = \{M_s \in \mathcal{M}_s\}$, to facilitate faster learning in a target domain, $\mathcal{M}_t$. In the realm of RL, the objective is to enable an agent to acquire new skills or improve its performance in a target environment by harnessing both external information, $\mathcal{I}_s$, from the source domain $\mathcal{M}_s$, and internal information, $\mathcal{I}_t$, inherent to the target domain $\mathcal{M}_t$. This thesis assumes one expert demonstration is given for each environment.

Chapter 3

Method

Our transfer learning framework facilitates the efficient adaptation of trained agents to new environments through a three-stage approach, as shown in Figure 3.1: 1. training a GCRL agent on a source environment to acquire diverse skills that can be leveraged in target environments, as shown in section 3.2; 2. finetuning the GCRL agent on the target environment, learning a latent representation of the agent's behavior using contrastive learning to capture the temporal structure of the agent's trajectories, and constructing a planning graph based on the learned latent space, as shown in section 3.3; and 3. execute the task in the target environment by following sub-goals generated from the planning graph using the finetuned GCRL agent as shown in section 3.4. We assume access to a single demonstration of successful task completion in the target environment, which we utilize to guide agent finetuning and graph construction. During training the GCRL agent, by resetting to states in the expert trajectory, we allow the GCRL agent to focus on state regions related to completing the task rather than searching over a much larger space for finding the optimal policy, which improves sample efficiency [48].

3.1 Assumptions

This thesis makes the following assumptions in our transfer learning framework: 1. We utilize a GCRL setup, where the agent's observations are augmented with a goal state that the agent aims to achieve [38]. This enables the agent to learn general skills

for reaching various goals. 2. We assume access to a single expert demonstration trajectory τ_{expert} of successful task completion in the target environment. This demonstration guides agent finetuning and graph construction in the target domain. 3. We employ partially observable agents that receive incomplete observations of the true environment state [21]. Partial observability can improve the generalization of learned behaviors across related environments.

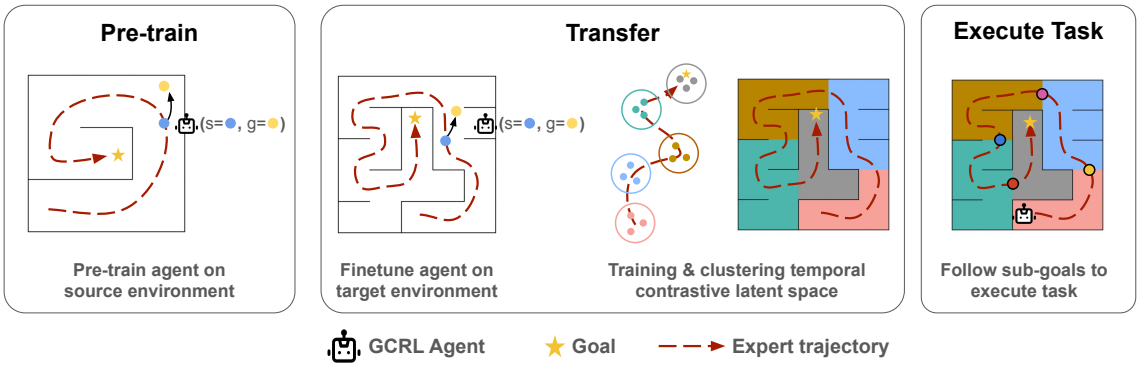


Figure 3.1: Our method follows three steps: 1) pre-train the GCRL agent to acquire diverse transferable skills by achieving short-horizon goals in the source environment; 2) finetune the GCRL agent on the target environment, learn a latent space to encapsulate the temporal structure of trajectories form rolling out the finetuned GCRL agent, and construct a planning graph, whose nodes are clusters from the latent space and edges are transitions between clusters observed in the expert trajectory; 3) and, execute task in the target environment by following sub-goals. We assume a single successful demonstration in the target environment is given, which we utilize to guide agent finetuning and graph construction.

3.2 Goal-Conditioned Reinforcement Learning Agent

To train the GCRL agents, we utilize the universal value approximator [38] and Proximal Policy Optimization [42]. We assume we can sample goal states for a given initial state. On each episode, we sample the initial state from the expert trajectory τ_{expert} and sample a goal state $g \sim P(g|s_0)$, where $P(g|s_0)$ is sampling a goal state by random walking from s_0. For a comprehensive algorithm description, we refer the

reader to [38] and [42]. Upon transferring, we first finetune the GCRL agent on the target environment and perform temporal contrastive learning and clustering.

3.3 Temporal Contrastive Learning and Clustering

Providing sub-goals guiding the GCRL agents to complete tasks in target environments is a promising avenue to efficiently transfer skills learned in the source environment to the target environment. This motivates the efficient construction of planning graphs grounded in agent behaviors. To achieve this, we utilize contrastive learning to distill a latent space representing temporal distances, specifically, the minimal steps required for an agent to transition from one state to another. However, obtaining the minimal temporal distance between state pairs is hard because this requires optimal control between every pair of states. Hence, we use state pairs and corresponding temporal distances from rollouts generated by the GCRL agent for approximation. The resulting temporal distances are noisy. Hence, we employ the InfoNCE [31] approach to learn a mapping f_w from the observational space to the embedding space, where geometric proximities in the embedding space mirror temporal distances in the trajectories. This relationship is encapsulated in Equation 3.1, with $d(\cdot, \cdot)$ representing a metric distance function. In this thesis, we choose $d(\cdot, \cdot)$ as the L2 distance. Adopting a metric space as $d(\cdot, \cdot)$ enables estimating temporal distances between unobserved state pairs using the triangular inequality. This contrastive learning and metric formulation, coupled with neural network modeling, empowers our system to process and generalize from noisy trajectory data. During training, we select state pairs within T timesteps in a trajectory to be positive samples and randomly sample states within the same batch to be negative samples. T is a hyper-parameter governing the maximum temporal threshold for positive sample pairs.

$$L_{\text{tc}}(x, x_{pos}, X) = -\,\mathbb{E}\left[\log \frac{exp(-d(f_w(x), f_w(x_{pos})))}{\sum_{x' \in X} exp(-d(f_w(x), f_w(x')))}\right] \tag{3.1}$$

Note that the learned latent space reflects the temporal distances of the underlying trajectories used for training. Thus, curating a dataset representative of the state and

transition distribution for the designated task is crucial. Collecting rollouts of states relevant to the desired task with temporal distances close to the minimal temporal distances is essential for learning latent space structures useful for the task.

In Algorithm 1, we sample initial states from an expert trajectory τ_{expert} to ensure we efficiently cover state regions relevant to the completing the task; we use the trained GCRL agent π_θ to collect rollouts; furthermore, we sample state pairs to balance the probabilities of sampling each state. After learning the temporal embeddings, we construct a graph to capture the essential temporal structure of the task. The graph is constructed as follows: first, we employ the K-means clustering algorithm to group the embeddings into distinct clusters and utilize the elbow method to determine the optimal number of clusters [6, 26]. Each cluster in the embedding space represents a node in the graph. Then, we create edges between nodes based on the observed transitions between clusters in the expert trajectory. Specifically, for each consecutive pair of states in the expert trajectory, we identify their corresponding clusters and add an edge between the associated nodes in the graph. It is crucial to note that the learned embeddings and the resulting graph are grounded in the original state space, enabling us to map each state to its corresponding embedding, cluster, and graph node. This property allows for seamless integration of the graph-based planning with the reinforcement learning agent. The constructed graph captures the essential temporal structure of the task, facilitating efficient planning and sub-goal generation for the agent during the transfer learning process.

Algorithm 1 Training Temporal Latent Space

Input: env, f_w, π_θ, τ_{expert}, $P(g|s_0)$
$s_0 \sim \tau_{\text{expert}}$
$g \sim P(g|s_0)$
Dataset $\leftarrow$ rollouts(π_θ, env, s_0, g)
while not converged **do**
    $x, x_{pos}, X \leftarrow$ BalancedSampling(Dataset)
    Optimize $L_{\text{tc}}(x, x_{pos}, X)$
end while
ClusterClassifier $\leftarrow$ Cluster f_w(Dataset)
PlanningGraph $\leftarrow$ construct_graph(Dataset, f_w, τ_{expert})

3.4 Task Execution

After finetuning on the target environment, we combine the GCRL agent π_τ, the temporal contrastive mapping f_w, the expert demonstration τ_{expert}, and the cluster classifier to execute tasks. As shown in Algorithm 2, on each step, we predict the current cluster and select the next sub-goals g as the state that transitions to the next cluster on the shortest path from the current cluster to the target cluster, or the target state if we are already in the target cluster, and execute the action sampled from $\pi_\theta(s, g)$.

Algorithm 2 Task Execution

Input: env, π_θ, τ_{expert}, f_w, ClusterClassifier
$s \leftarrow$ env.reset()
while not done **do**
    $c \leftarrow$ ClusterClassifier($f_w(s)$)
    $g \leftarrow$ GetSubGoal($f_w, c, \tau_{\text{expert}}$)
    action $\sim \pi_\theta(s, g)$
    s, done $\leftarrow$ env.step(action)
end while

Chapter 4

Experiments

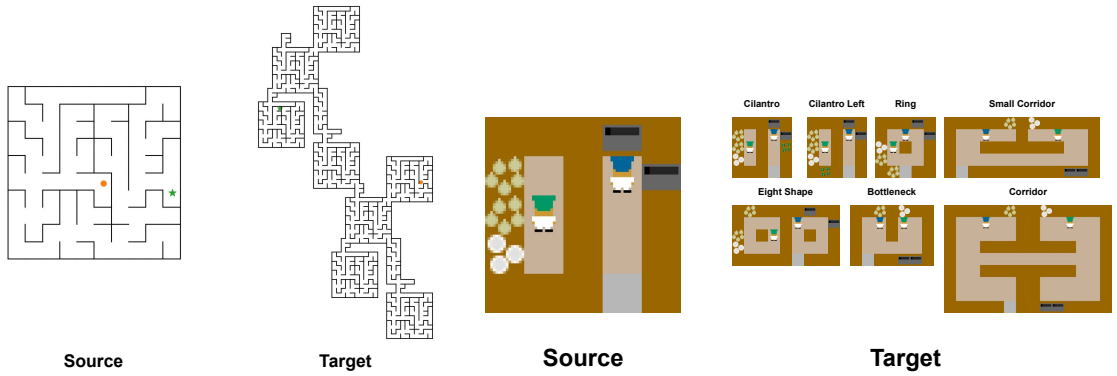
In this chapter, we provide qualitative and quantitative experiment results to demonstrate the effectiveness of our method. We aim to answer the following questions: 1) Does our method enable learning in a new environment for single-agent and multi-agents faster? 2) Are the generated sub-goals qualitatively interpretable?

4.1 Setup

We evaluated our methods and four other baselines on eight transfer learning experiments across two environments. For each experiment, we pre-train agents on the source environment env_s and transfer to the target environment env_t. We set the target environments as variants (different layouts or tasks) of the source environment. We report the learning curve and time-to-threshold for each experiment. We used partial observable agents in all experiments unless specified otherwise.

Point Maze is a continuous 2D maze environment. The agent's goals and initial positions are randomly initialized on the maze. The agent's observation includes distance measures of a simulated 2D lidar sensor and a delta to the goal position. The action space is the 2D planar velocity. The agent is considered to reach the goal if it is close to the goal position and not blocked by any wall. The task reward, not used in our method, is the shortest distance to the target position, approximated from a graph search over a discretized maze version. We evaluated each method on 500 episodes and reported the success rate in reaching the goal position. The source

and target tasks are shown in Figure 4.1a. Our implementation of Point Maze is adapted from [33].



(a) PointMaze (b) Overcooked

Figure 4.1: a) The source and target PointMaze [33] tasks. Agents must navigate from the initial position (the orange point) to the target position (the green star). b) The source and target Overcooked [9] tasks. The two chefs need to coordinate to make soup and deliver soups. In each environment, there are two chefs (the chef with the green hat and the chef with the blue hat), onion dispensers, plate dispensers, ovens (the grey box with a black top), a serving area (the plain light grey box), walls (brown box) and optionally cilantro dispensers.

Overcooked is a simplified version of the popular video game *Overcooked* [16], where 2-4 players control chefs cooking and serving dishes in a kitchen. We consider two-player scenarios where the chefs must coordinate to prepare and deliver soups. Each dish recipe contains several high-level steps, as shown in Figure 4.2. The task rewards are dish pickup (3), soup pickup (5), placing onion in the oven (3), adding cilantro to soup (5), delivering soup (20), and 0 otherwise. Note that task rewards are used in baselines but not in our method. The target environments were designed as variants of the source environment, differing in layout or task. The *cilantro* and *cilantro left* environments have different recipes and layouts, and the *ring*, *eight shape*, *bottleneck*, *small corridor* and *corridor* environments have different layouts. The source and target tasks are visualized in Figure 4.1b. We used partially observable agents in all experiments unless specified otherwise. Each episode consisted of 500 timesteps, and the performance was evaluated based on the number of soups delivered per episode. The Overcooked environment has a fixed initial configuration

and deterministic dynamics. We randomly rolled out the agent for ten steps before executing the policy to introduce randomness to the environment. We provide a single expert trajectory for each environment via hard-coded policies.

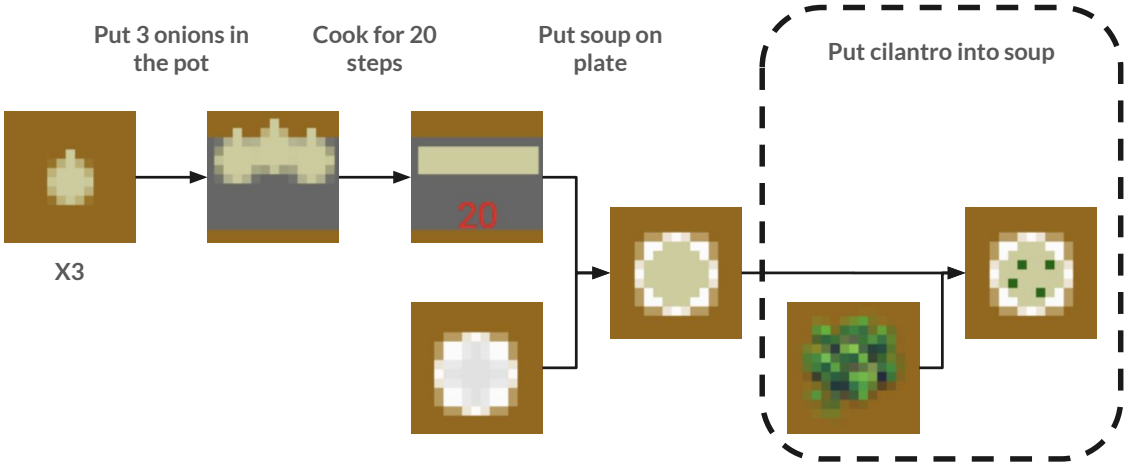


Figure 4.2: Overcooked recipes. To make one soup, the two chefs need to 1) fetch three onions from the onion dispenser and put them into the oven one by one, and 2) turn on the oven and wait for 20 steps, and 3) fetch a plate from the plate dispenser, take the soup from the oven to the plate, and 4) Optionally, to make a cilantro soup, fetch cilantro from the dispenser and put it on the soup plate.

We compare our method to the following five methods.

- **Vanilla RL**: Training an RL agent from scratch.
- **Fine-tuning**: Fine-tuning the agent trained on the source environment.
- **Policy Distillation (Loss)**: Policy distillation through an auxiliary cross-entropy loss between the action probabilities from the policy pre-trained in the source environment and the learning policy [39].
- **Policy Distillation (Reward)**: Policy distillation through a reward shaping term captures the difference of the pre-trained critic in the source environment between current the previous timesteps [11].
- **JumpStart RL**: JumpStart RL [47] uses a guiding policy to form a curriculum learning, where we gradually sample fewer actions from the guiding policy. In this thesis, we evaluated eight variants of JumpStart RL: 1) whether the curriculum schedule is random or curriculum, 2) whether the policy is initialized from

the agent trained on the source environment and 3) whether the guiding policy is trained on the source or target environment. Note that using policies trained on the target environment as guide policies might give it unfair advantages.

4.2 Transfer Learning Results

Point Maze. We show the learning curves for transferring from the source environment to the target environment for the point maze environment in Figure 4.3. To verify the GCRL agent could adapt local behaviors to the environments, we made the target environment by copying and pasting parts of the source environment. We augmented the planning graph from the source environment by copying and pasting the node/cluster label of each position and cluster connection according to how we copied and pasted it to create the target environment. We used the policy trained from the source environment without fine-tuning with the adapted planning graph. As a result, our method can transfer to the much bigger maze without fine-tuning. This demonstrated the effectiveness of adapting the planning graph to new environments and transferring the local skills of GCRL agents.

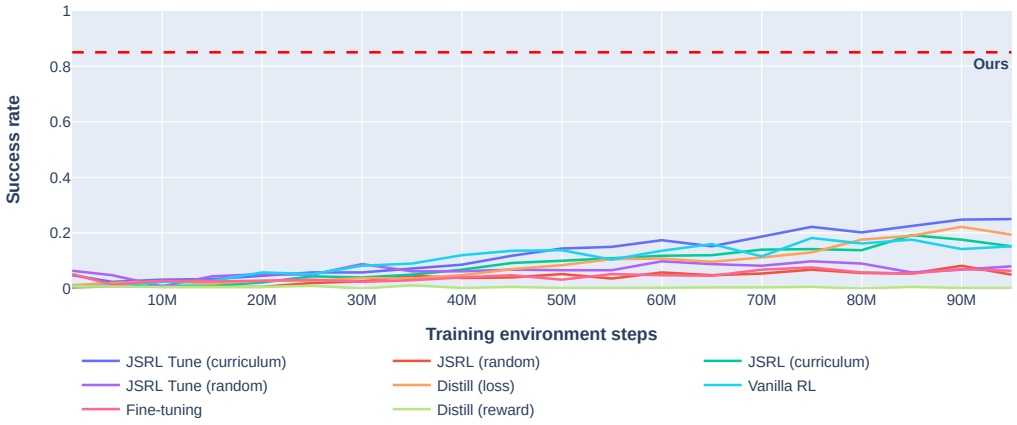


Figure 4.3: Point maze Transfer Learning Curve. The average success rate of reaching the goal is calculated over 500 episodes. Note that our method does not require training for this experiment.

Overcooked. We show the average soups delivered for each method throughout

training on each environment in Figure 4.6. We show the training steps taken to convergence in Table 4.1 and the maximum soups delivered per method for each environment in Table 4.2. The normalized performance and convergence speed is at Figure 4.4. Time to convergence is defined as reaching 90% of maximum performance per method. Each environment's performance and convergence comparison is at Figure 4.5. Our methods consistently learn 4.27 times faster on average than the fastest baselines across all experiments, reaching similar or better performances. On experiments transferring to environments with similar layouts but different tasks from the source environment, the *cilantro* and *cilantro left* environments, the transfer learning baselines perform poorly and, sometimes, even worse than the vanilla RL. This is because the guidance from the source environment policy can be biased toward the old behaviors, making it challenging to learn behaviors needed for the new environments. This is especially true in the environments with the cilantro recipes because delivering soups before putting cilantro in can significantly hinder the resulting performance. This also shows that our method can effectively transfer to environments with tasks where slight differences can result in significant performance degradation. On experiments transferring to environments with similar tasks but different layouts from the source environment, the *cilantro*, *cilantro left*, *eight shape*, *small corridor*, *corridor* and *bottleneck* environments, our method could effectively transfer to such environments. This demonstrates our method's ability to efficiently transfer skills learned from the source environment to target environments. Note on the *small corridor* and *corridor* environments, other methods struggle to deliver soups. This is because the narrow and long corridors require agents to coordinate so as not to block others from delivering soups successfully. This demonstrates our method's ability to perform long-horizon multi-agent planning and coordination.

Table 4.1: Overcooked training steps to convergence (reaching 90% of the max soups per method per environment) table. n/a means the method did not deliver any soup.

Environment	Cilantro	Cilantro Left	Eight Shape	Ring	Small Corridor	Corridor	Bottleneck
Ours	**680.5K**	**806.3K**	**1.1M**	**1.0M**	**1.1M**	**1.3M**	**1.0M**
Vanilla RL	5.0M	6.0M	7.0M	6.0M	n/a	n/a	8.0M
Fine-tuning	3.0M	1.0M	n/a	5.0M	n/a	n/a	8.0M
Distill	9.0M	3.0M	5.5M	6.5M	n/a	n/a	6.5M
JSRL	6.2M	5.8M	6.6M	5.8M	5.0M	n/a	7.4M

Table 4.2: Overcooked max soups delivered table.

Environment	Cilantro	Cilantro Left	Eight Shape	Ring	Small Corridor	Corridor	Bottleneck
Ours	**12.58**	**12.10**	10.32	10.36	**4.92**	**3.84**	6.60
Vanilla RL	9.72	10.54	9.00	11.06	0.00	0.00	**8.10**
Fine-tuning	11.22	0.02	0.00	12.32	0.00	0.00	8.00
Distill	9.58	0.03	**10.53**	10.91	0.00	0.00	7.90
JSRL	8.28	5.97	6.59	**12.38**	0.42	0.00	7.87

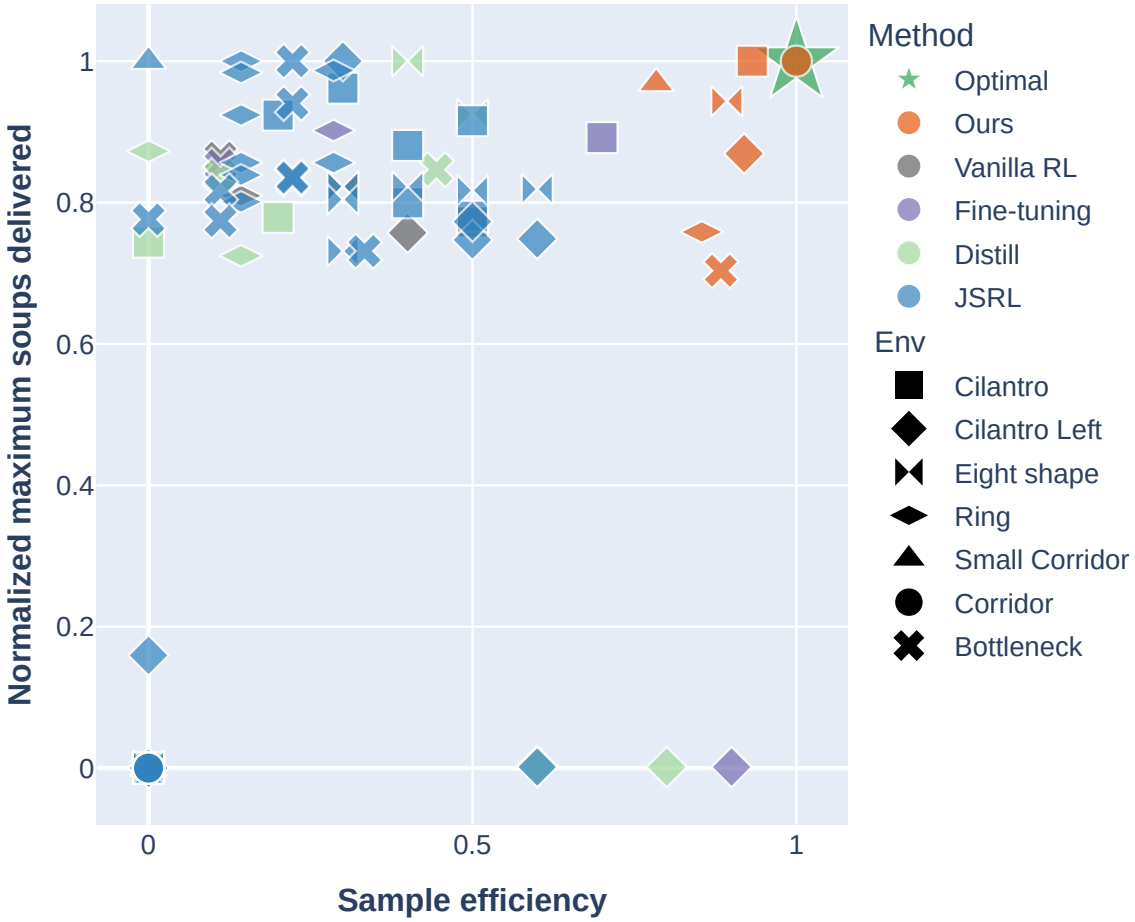
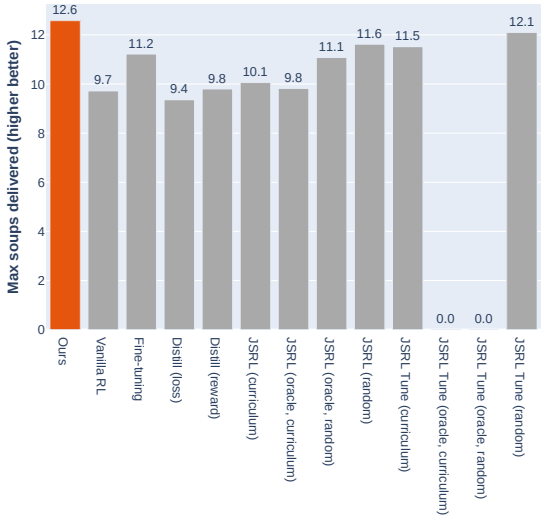
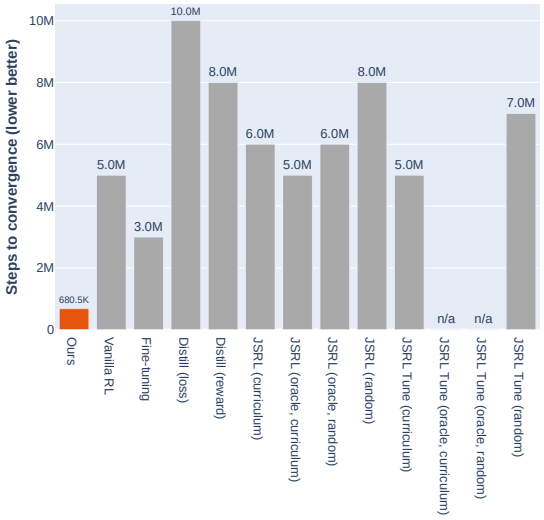


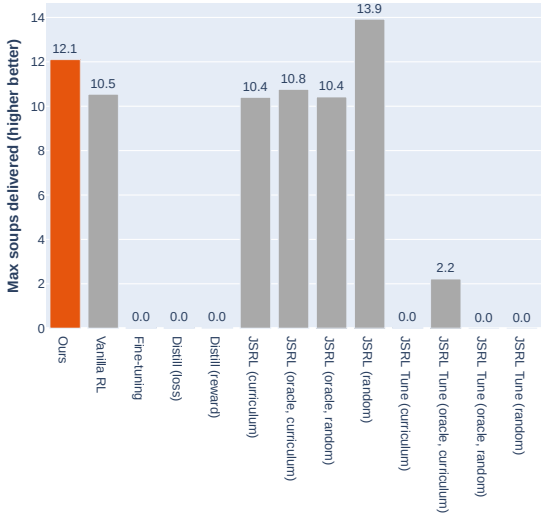
Figure 4.4: The scatter plot for normalized performance and sample efficiency in the Overcooked environment. The maximum number of soups delivered is normalized using the formula: maximum number of soups delivered for a given method / maximum number of soups delivered for all methods in an environment. The Sample efficiency is normalized using the formula: 1 - (steps to convergence for a given method / maximum steps to convergence in the environment). Steps to convergence are determined by the steps at which a method reaches 90% of its maximum performance. Variants of the same method are grouped under a single plot category.



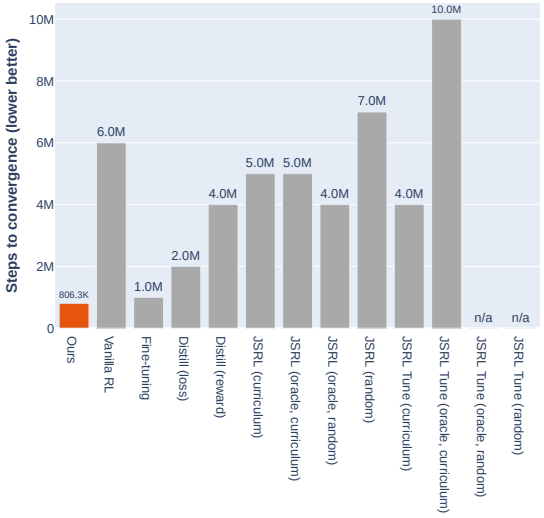
(a) *Cilantro* environment performance, max soups delivered (higher better).



(b) *Cilantro* environment steps taken to reach convergence (lower better).

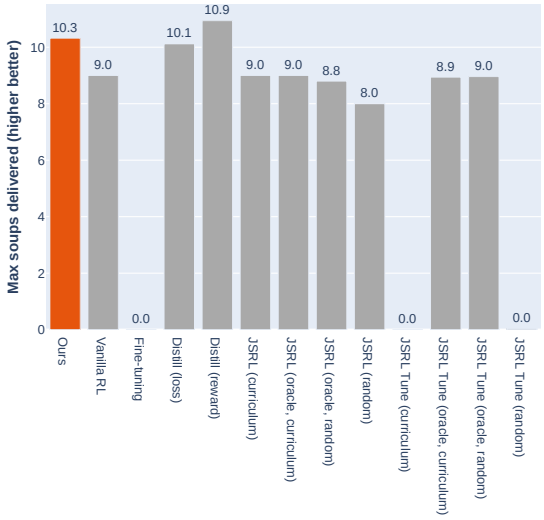


(c) *Cilantro left* environment performance, max soups delivered (higher better).

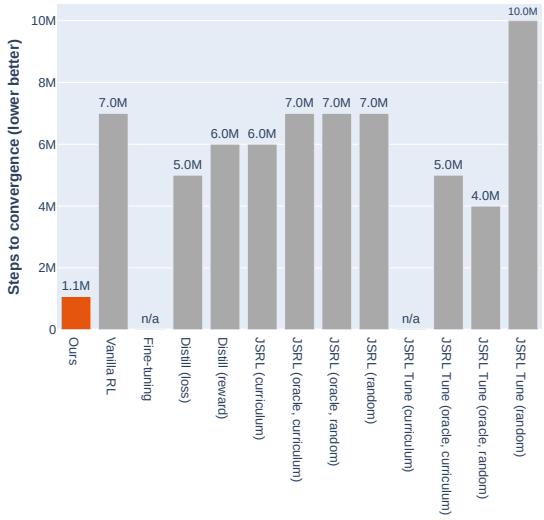


(d) *Cilantro left* environment steps taken to reach convergence (lower better).

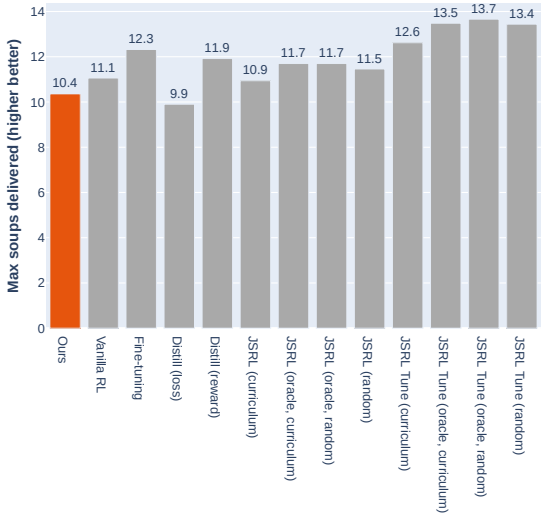
Figure 4.5



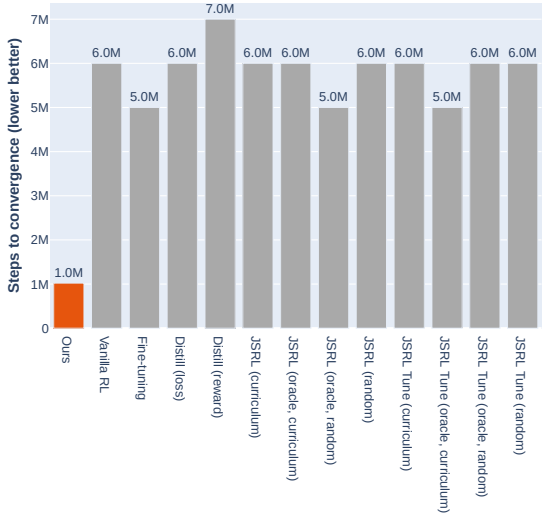
(e) *Eight shape* environment performance, max soups delivered (higher better).



(f) *Eight shape* environment steps taken to reach convergence (lower better).

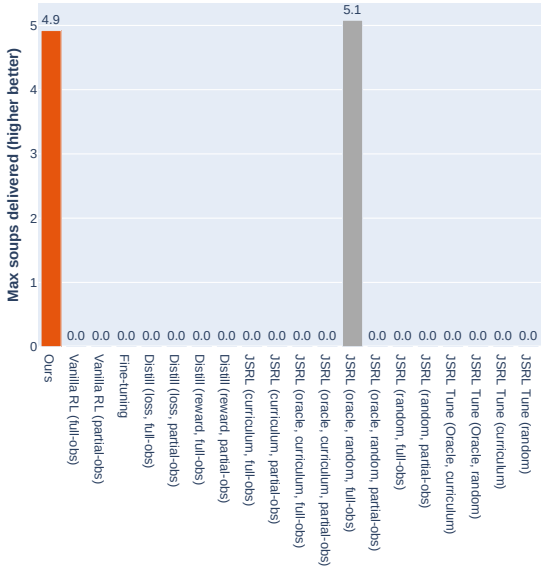


(g) *Ring* environment performance, max soups delivered (higher better).

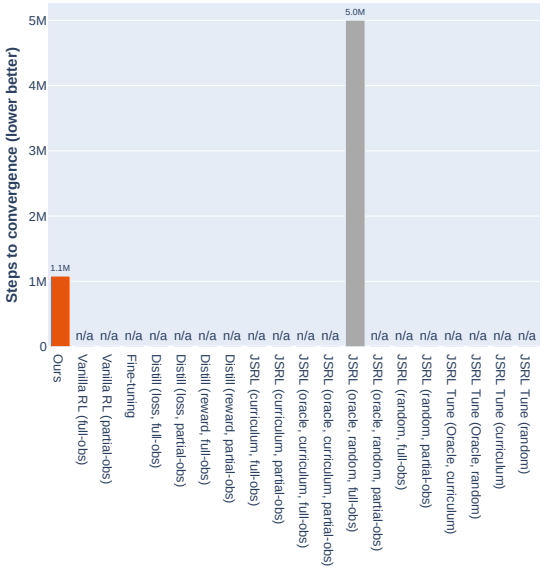


(h) *Ring* environment steps taken to reach convergence (lower better).

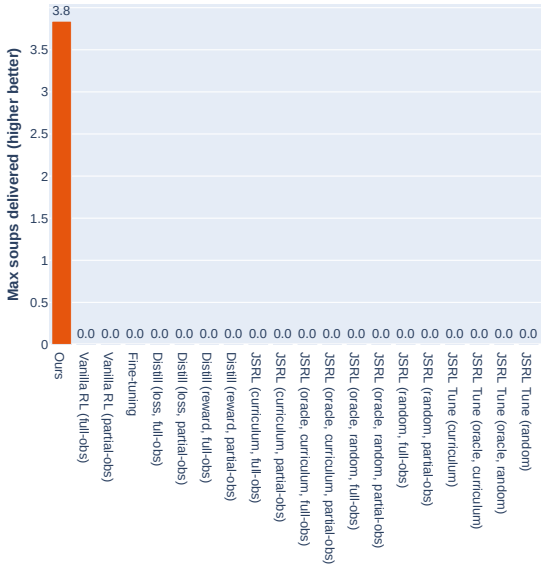
Figure 4.5



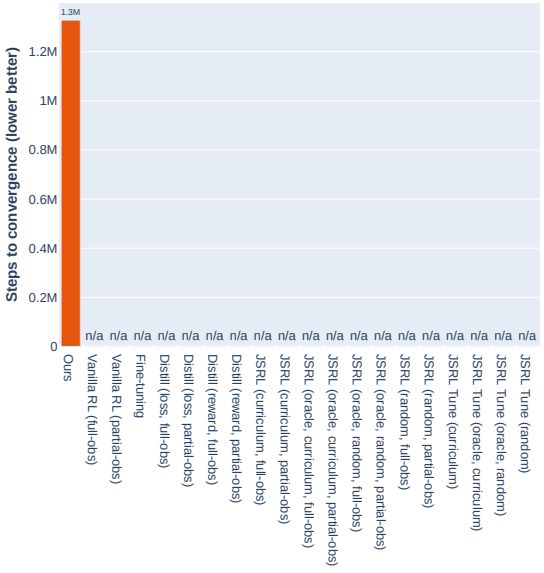
(i) *Small corridor* environment performance, max soups delivered (higher better).



(j) *Small corridor* environment steps taken to reach convergence (lower better).

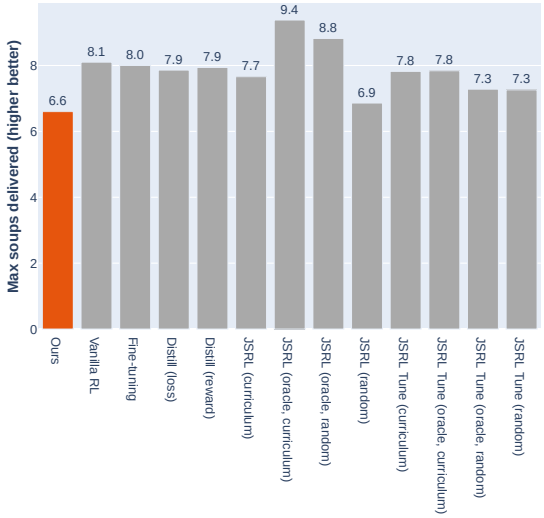


(k) *Corridor* environment performance, max soups delivered (higher better).

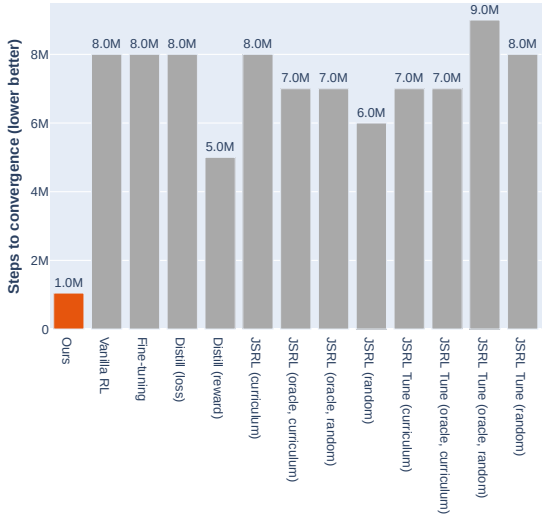


(l) *Corridor* environment steps taken to reach convergence (lower better).

Figure 4.5



(m) *Bottleneck* environment performance, max soups delivered (higher better).



(n) *Bottleneck* environment steps taken to reach convergence (lower better).

Figure 4.5: Overcooked performance and steps to convergence across various environments. The step to convergence is n/a when no soup is delivered.

4.3 Interpretable Sub-goals

The sub-goals generated from section 3.4 exhibit semantically meaningful breakdown of tasks, e.g., fetching onions, loading onion to the oven, and serving soups, as shown qualitatively in Figure 4.7. This empirically demonstrated that unsupervised temporal contrastive learning could discover semantically meaningful structures from rollouts. One intuitive explanation behind this is that the connection between clusters in the latent space tends to be the connection of a bottleneck structure, where the bottleneck transitions are a sequence of actions that enable the agent to reach states previously impossible to achieve. Such transitions often correspond to sub-goals for a task since the agent can advance to previously inaccessible states by following the next sub-goal. One example of such a bottleneck is fetching an onion when the agent has no onion. By fetching an onion, the agent can reach states of carrying onions around that were previously inaccessible.

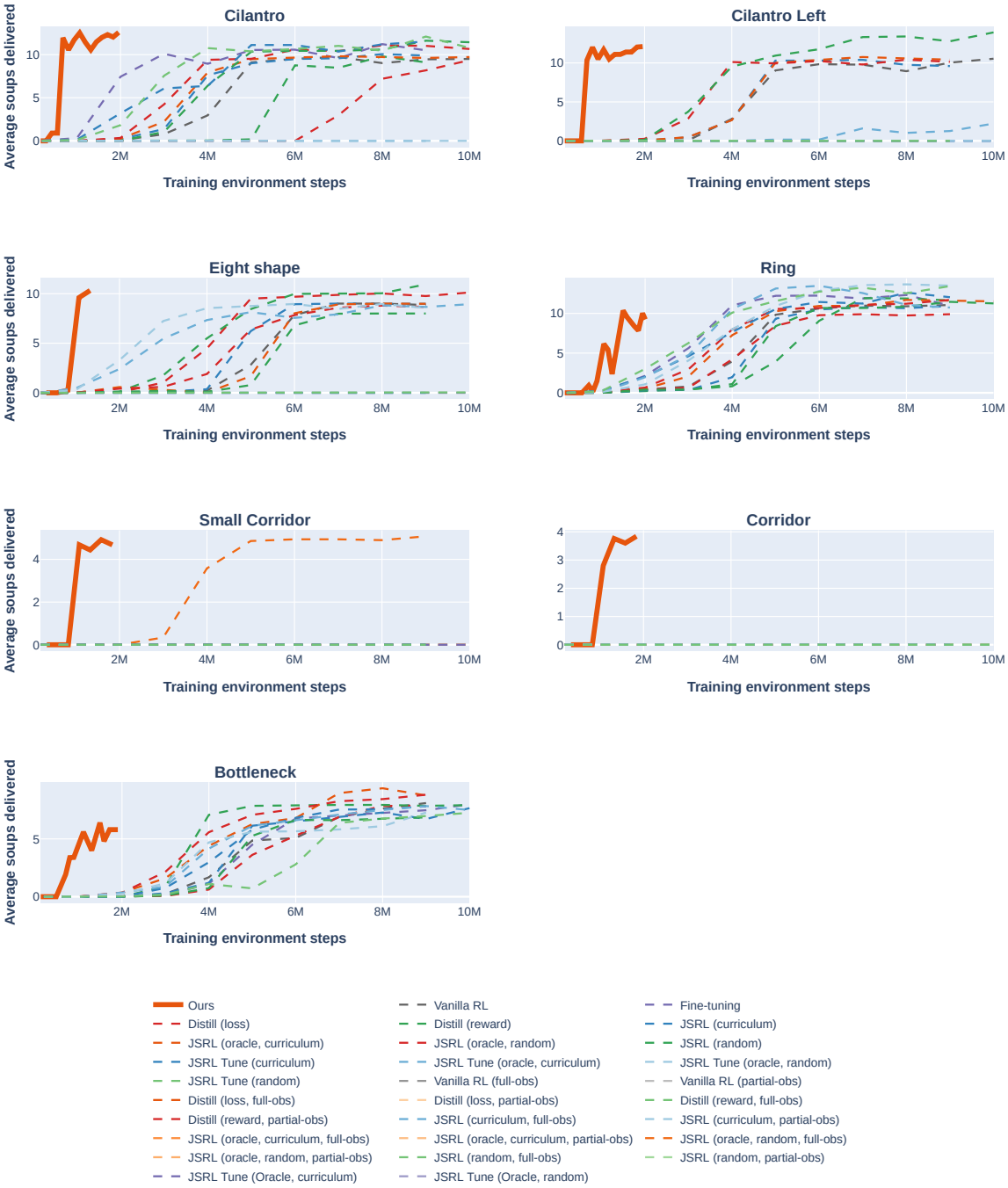
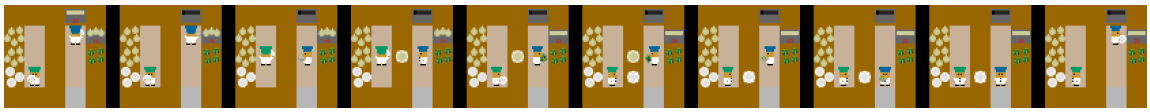
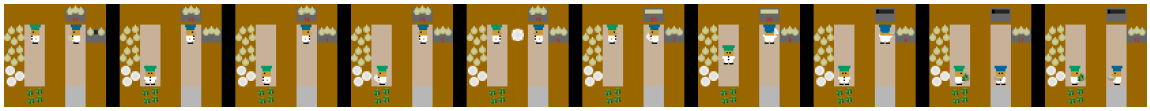


Figure 4.6: Overcooked Learning Curves. Average soups delivered over 50 episodes throughout training.

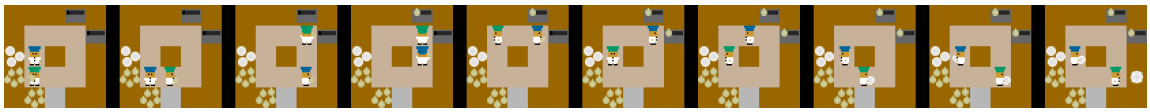


(a) Cilantro

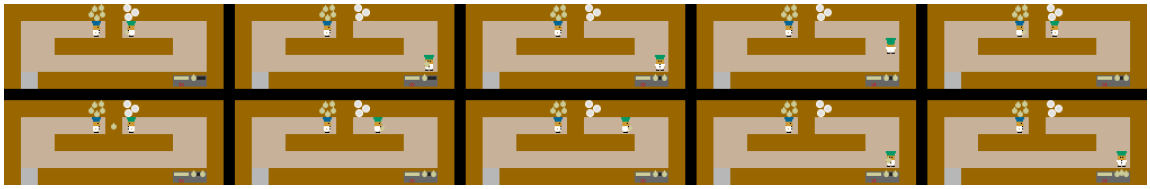


(b) Cilantro Left

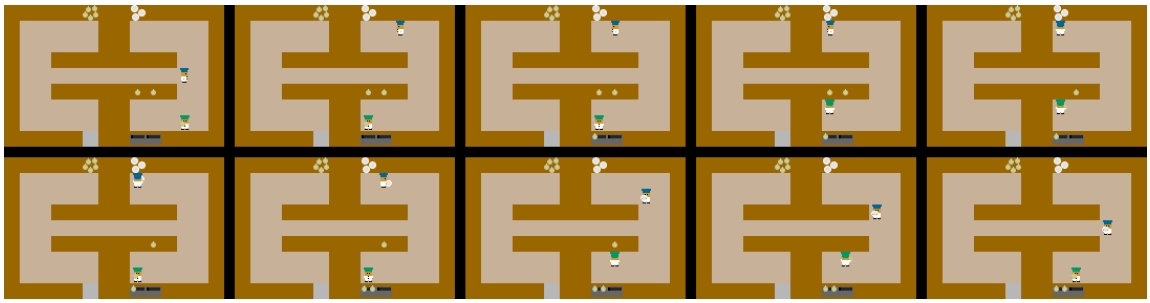
(c) Eight Shape



(d) Ring



(e) Small Corridor



(f) Corridor

(g) Bottleneck

Figure 4.7: Overcooked sub-goals. Samples of sub-goal sequences were generated for each overcooked environment. Semantically meaningful breakdown of the task emerges naturally from the temporal contrastive embedding clusters. For example, the sub-goals qualitatively demonstrate the intentions for handing over onions, fetching plates, putting onions into the oven, and taking soups out of the oven.

4.4 Experiment take-aways

The experimental results demonstrate several key takeaways of our transfer learning framework. First, our method successfully learns to construct planning graphs that capture meaningful subgoals for task completion in the target environments, without requiring manual specification or domain knowledge. Second, the experiments on the Overcooked environment showcase the method's ability to handle multi-agent scenarios by learning from joint agent trajectories and capturing the coordination between agents. Finally, the PointMaze experiments show that when the target environment is a permutation of the source environment, we can efficiently adapt to the new environment by augmenting the learned graph structure, enabling zero-shot transfer to related but unseen environments. These takeaways highlight the effectiveness of our approach in automatically discovering transferable subgoals, handling multi-agent settings, and efficiently adapting to new environments, opening up exciting possibilities for more flexible and sample-efficient transfer learning in reinforcement learning.

Chapter 5

Conclusions

This paper introduced a novel transfer learning framework for deep reinforcement learning that combines goal-conditioned policies with unsupervised learning of temporal abstractions. Our approach automatically abstracts tasks into subgoals using temporal proximity, eliminating the need for manual specification or domain knowledge. This makes transfer learning more flexible and applicable to various domains. Experiments on Overcooked multi-agent coordination tasks demonstrated the effectiveness of our framework in terms of improved sample efficiency, the ability to solve sparse-reward and long-horizon challenges, and enhanced interpretability through the automatic discovery of meaningful sub-goals. These findings highlight the advantages of integrating goal-conditioned RL with unsupervised temporal abstraction learning for successful transfer to complex target domains, demonstrating superior performance compared to baseline methods such as fine-tuning, policy distillations, and curriculum learning methods. Notably, our method achieves the same or better performance while requiring only 23.4% of the training samples compared to state-of-the-art baselines, showcasing its sample efficiency and potential for real-world applications where data collection and training can be expensive or time-consuming.

Our work opens up exciting directions for future research. One promising avenue is integrating language guidance or instructions into the subgoal discovery process, which could lead to more semantically meaningful and interpretable subgoals, enhancing the transparency and explainability of the learned policies. Another direction is to extend our framework to handle multiple tasks and environments simultaneously, potentially

by learning a shared representation space or a hierarchical structure that captures the commonalities and differences across tasks and environments. This could enable more efficient and generalizable transfer learning. Furthermore, exploring the application of our method to real-world problems, such as robotics, autonomous navigation, or game AI, could demonstrate the benefits of automatic subgoal discovery and efficient transfer learning in these domains. Finally, integrating our transfer-learning framework with other learning paradigms, such as imitation learning, meta-learning, or model-based reinforcement learning, could lead to developing more powerful and versatile transfer-learning algorithms. Pursuing these research directions can pave the way for more intelligent, adaptable, and collaborative AI systems that can efficiently learn and transfer knowledge across various tasks and environments.

Bibliography

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017. 2.2.1

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. 1

[3] Karl Johan Åström. Optimal control of markov processes with incomplete state information i. *Journal of mathematical analysis and applications*, 10:174–205, 1965. 2.1.1

[4] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017. 2.2.3

[5] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957. 2.1.1

[6] Benjamin Bengfort and Rebecca Bilbro. Yellowbrick: Visualizing the Scikit-Learn Model Selection Process. 4(35), 2019. doi: 10.21105/joss.01075. URL `http://joss.theoj.org/papers/10.21105/joss.01075`. 3.3

[7] Hoang-Giang Cao, Weihao Zeng, and I-Chen Wu. Reinforcement learning for picking cluttered general objects with dense object descriptors. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6358–6364. IEEE, 2022. 2.2.2

[8] Hoang-Giang Cao, Weihao Zeng, and I-Chen Wu. Learning sim-to-real dense object descriptors for robotic manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9501–9507. IEEE, 2023. 2.2.2

[9] Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

(document), 1, 4.1

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1, 2, 2.1.2

[11] Wojciech M Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *The 22nd international conference on artificial intelligence and statistics*, pages 1331–1340. PMLR, 2019. 2.2.4, 4.1

[12] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019. 2.2.1

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 2.2.4

[14] Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018. 2.2.2

[15] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017. 2.2.3

[16] Ghost Town Games. Overcooked. `https://store.steampowered.com/app/448510/Overcooked/`, 2016. 4.1

[17] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017. 2.2.4

[18] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020. 2.2.1

[19] Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pages 1480–1490. PMLR, 2017. 2.2.4

[20] Edward S Hu, Richard Chang, Oleh Rybkin, and Dinesh Jayaraman. Planning goals for exploration. *arXiv preprint arXiv:2303.13002*, 2023. 2.2.1

[21] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101 (1-2):99–134, 1998. 3

[22] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsu-

pervised representations for reinforcement learning. In *International conference on machine learning*, pages 5639–5650. PMLR, 2020. 2.2.2

[23] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020. 2.2.2

[24] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017. 2.2.1, 2.2.3

[25] Siyuan Li, Zicheng Liu, Zelin Zang, Di Wu, Zhiyuan Chen, and Stan Z Li. Genurl: A general framework for unsupervised representation learning. *arXiv preprint arXiv:2110.14553*, 2021. 2.2.1

[26] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. doi: 10.1109/TIT.1982.1056489. 3.3

[27] Yao Lu, Karol Hausman, Yevgen Chebotar, Mengyuan Yan, Eric Jang, Alexander Herzog, Ted Xiao, Alex Irpan, Mohi Khansari, Dmitry Kalashnikov, et al. Aw-opt: Learning robotic skills with imitation and reinforcement at scale. *arXiv preprint arXiv:2111.05424*, 2021. 1

[28] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023. 1

[29] Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391, 2021. 2.2.1

[30] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018. 2.2.1, 2.2.3

[31] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2.1.2, 3.3

[32] Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. *arXiv preprint arXiv:2402.15567*, 2024. 2.2.2

[33] Silviu Pitis, Harris Chan, and Stephen Zhao. mrl: modular rl. `https://github.com/spitis/mrl`, 2020. (document), 1, 4.1, 4.1

[34] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018. 2.2.1

[35] Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019. 2.2.1

[36] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020. 2.1.2

[37] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 2.2.4

[38] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015. 1, 2.2.1, 1, 3.2

[39] Simon Schmitt, Jonathan J Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M Czarnecki, Joel Z Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018. 2.2.4, 4.1

[40] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 2.1.2

[41] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 2.1.1

[42] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2.1.1, 3.2

[43] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International conference on machine learning*, pages 8583–8592. PMLR, 2020. 2.2.1

[44] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017. 1

[45] Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020. 2.2.2

[46] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999. 2.2.3

[47] Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, et al. Jump-start reinforcement learning. In *International Conference on Machine Learning*, pages 34556–34583. PMLR, 2023. 2.2.4, 4.1

[48] Anirudh Vemula, Yuda Song, Aarti Singh, Drew Bagnell, and Sanjiban Choudhury. The virtues of laziness in model-based rl: A unified objective and algorithms. In *International Conference on Machine Learning*, pages 34978–35005. PMLR, 2023. 3

[49] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017. 1

[50] Albert Zhan, Ruihan Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. Learning visual robotic control efficiently with contrastive pre-training and data augmentation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4040–4047. IEEE, 2022. 2.2.2

[51] Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2371–2378. IEEE, 2017. 1

[52] Zhuangdi Zhu, Kaixiang Lin, Anil K Jain, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 1, 2.2.4, 2.3