Hierarchical Cooperative Multi-Agent Reinforcement Learning with Skill Discovery

Jiachen Yang*[1], Igor Borovikov†[2], and Hongyuan Zha[1]

[1]School of Computational Science and Engineering, Georgia Institute of Technology
[2]EA Digital Platform, Data & AI, Electronic Arts

Abstract

Human players in professional team sports achieve high level coordination by dynamically choosing complementary skills and executing primitive actions to perform these skills. As a step toward creating intelligent agents with this capability for fully cooperative multi-agent settings, we propose a two-level hierarchical multi-agent reinforcement learning (MARL) algorithm with unsupervised skill discovery. Agents learn useful and distinct skills at the low level via independent Q-learning, while they learn to select complementary latent skill variables at the high level via centralized multi-agent training with an extrinsic team reward. The set of low-level skills emerges from an intrinsic reward that solely promotes the decodability of latent skill variables from the trajectory of a low-level skill, without the need for hand-crafted rewards for each skill. For scalable decentralized execution, each agent independently chooses latent skill variables and primitive actions based on local observations. Our overall method enables the use of general cooperative MARL algorithms for training high level policies and single-agent RL for training low level skills. Experiments on a stochastic high dimensional team game show the emergence of useful skills and cooperative team play. The interpretability of the learned skills show the promise of the proposed method for achieving human-AI cooperation in team sports games.

1 Introduction

Fully cooperative multi-agent reinforcement learning (MARL) is an active area of research [24, 14] with a diverse set of real-world application, which include autonomous navigation [6], game AI micromanagement [10, 26], and traffic network optimization [43], among many others. A unique challenge is the need for centralized training for agents to find global optimal cooperative policies, while ensuring scalable decentralized execution whereby agents choose actions independently. In this paradigm of centralized training with decentralized execution [5], a common approach [34, 10, 26, 42, 30] is to conduct centralized training at the level of *primitive* actions, which are the actions used in the transition function of the Markov game [19]. However, the design of hierarchical agents who can cooperate at a higher level of abstraction using temporally-extended *skills* in high-dimensional multi-agent environments is still an open area for investigation. A skill is a policy that is conditioned on a latent skill variable, executed for an extended duration, and generates behavior from which the latent variable can be decoded [9, 1]. It is also not yet clear how multiple agents can *discover* skills without the need to hand-craft reward functions for each skill, and how to construct such hierarchical policies to allow human interpretation of skills for potential human-AI cooperation.

In this paper, we take a hierarchical approach to fully cooperative MARL and address these questions by drawing inspiration from team sports games. At the team level, coaches train human players to execute complementary skills in parallel, such as moving to different field positions in an offensive formation, as well as effective sequences of skills over time, such as switching from preparation for a shot attempt to defensive maneuvers when an opponent gains ball possession. At the individual level, each player learns a sequence of low-level primitive actions to execute a chosen skill. Hierarchical approaches inspired from such real-world practices have several benefits for fully cooperative MARL. From an algorithmic viewpoint, a hierarchical decomposition in two key dimensions—over agents, and across

*jiachen.yang@gatech.edu (Work done at Electronic Arts)
†iborovikov@ea.com

time—simultaneously addresses both the difficulty of learning multi-agent cooperation at the level of noisy low-level actions in stochastic environments and the difficulty of long-term credit assignment due to highly-delayed rewards (e.g., scoring a goal in football) [12, 39]. Hierarchical approaches may also reduce computational complexity [36] to address the exponential increase in sample complexity with number of agents in MARL. From the viewpoint of human-AI cooperation, which has near-term application to video game AI to improve human players' experiences [44], hierarchical policies trained with explicit skills is a key step toward interpretable and more modular policies. In this work, we take interpretability to mean the decodability of a latent skill from an agent's observed behavior—i.e., a policy is interpretable if it produces events and actions in a consistent or distinguishable manner. We gain interpretability by explicitly conditioning the lower level of a hierarchical policy on latent variables. While a flat policy is a black-box, since the action output is purely determined by the agent's observation input, the modularity of hierarchical models also provides an entry point for external control over the skills executed by AI teammates (e.g., execute the offense skill when it observes a human teammate doing so).

However, decomposing a global team objective such as "scoring a goal" into many sub-objectives for training a collection of skills is extremely difficult without expert knowledge, which may be hard to access for complex settings such as competitive team sports. Manually crafting reward functions for each skill in high-dimensional state spaces involving numerous agents is also prone to misspecification and could result in unintended behavior [3]. Instead, we investigate a method for hierarchical agents in MARL to discover and learn a set of high-level latent skills. Agents should learn to cooperate by choosing effective combinations of skills with their teammates, and also dynamically choose skills in response to the state of the game. In contrast to prior work on single-agent skill discovery, in which an intrinsic reward that is unrelated to any task reward is used to discover motion skills [1, 9], MARL poses significant new challenges for skill discovery. Merely discovering distinguishable individual motion in an open-ended multi-agent environment may be useless for a team objective. While increasing the number of learned skills increases the chance that some skills are useful for tasks [9], doing so in the hierarchical multi-agent setting means exponentially increasing the size of a joint high-level action space and will exacerbate the difficulty of learning.

We present a method for training hierarchical policies with unsupervised skill discovery in cooperative MARL, with the following key technical and experimental contributions. 1) We construct a two-level hierarchical agent for MARL by defining a high-level action space as a set of latent variables. Each agent consists of a high-level policy that chooses and sustains a latent variable for many time steps, and a low-level policy that uses both its observation and the selected latent variable to take primitive actions. 2) We use an extrinsic team reward to conduct centralized training of high-level policies for cooperation, while we use a combination of an intrinsic reward and the team reward to conduct decentralized training of low-level policies with independent reinforcement learning (RL). This allows the use of powerful and general algorithms for cooperative MARL and single-agent RL to train high- and low-level policies, respectively. 3) We define the intrinsic reward as the performance of a decoder that predicts the ground truth latent variable from trajectories generated by low-level policies that were conditioned on the latent variables. By dynamically weighting the intrinsic versus extrinsic reward, each low-level policy is trained to reach a balance between decodability and usefulness—it executes a skill, without the need for hand-designed skill-specific reward functions. 4) We applied this algorithm to a highly stochastic continuous state simulation of team sports and performed a detailed quantitative investigation of the learned behaviors. Agents discover useful skills, that affect game events and determine low-level actions in distinct and interpretable ways, such as grouping together to steal possession from an opponent. They learn to choose complementary skills among the team, such as when one agent camps near the opponent goal to get a rebound when its teammate makes a long-range shot attempt. 5) Our hierarchical agents perform higher than flat methods in ad-hoc cooperation when matched with teammates who follow policies that were not encountered in training. This is an encouraging result for the possibility of human-AI cooperation.

2 Related works

Building on the framework of options, temporally-extended actions, and hierarchical single-agent RL [8, 36, 25, 31], early work on hierarchical MARL in discrete state spaces with hand-crafted subtasks [21, 12] showed that learning cooperation at the level of subtasks significantly speeds up learning over flat methods [37, 19, 33]. Recent work built on the effectiveness of deep reinforcement learning [23, 29] to demonstrate hierarchical single-agent RL in high-dimensional continuous state spaces, using predefined subgoals [16], end-to-end learning of options [4], and latent

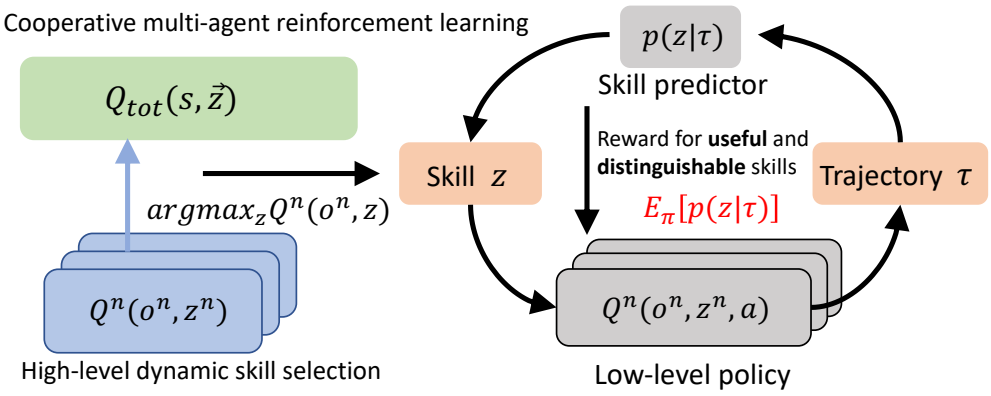


Figure 1: Hierarchical MARL with unsupervised skill discovery. At the high level (left), the extrinsic team reward is used to train a centralized action-value function $Q_{\text{tot}}(s, \mathbf{z})$ that decomposes into individual utility functions $Q^n(o^n, z^n)$ for decentralized skill selection. At the low level (right), skill-conditioned action-value functions $Q^n(o^n, z^n, a^n)$ act independently to execute primitive actions. Trajectories τ generated under each z are collected into a dataset $\mathcal{D} = \{(z, \tau)\}$. This is input to a skill decoder $p(z|\tau)$ that is trained to predict the latent skill z that corresponds to each trajectory. The probability of selected skills under $p(z|\tau)$ is the intrinsic reward for low-level Q^n.

directional subgoals [39] in a two-level hierarchy. In hierarchical MARL, different subtasks are chosen concurrently by all agents and sequentially in time, whereas only a single subtask is chosen for each segment in single-agent hierarchical RL [4, 39].

Progress in hierarchical learning benefits from a complementary line of work on automatic subgoal discovery [22]. Our work draws inspiration from variational option discovery [1, 13, 9], which—in formal analogy with variational auto-encoders [15]—trains a maximum-entropy policy encoder to map latent context vectors into trajectories from which the context can be recovered by a supervised decoder. In contrast to prior work on single-agent skill discovery that focus on finding distinguishable behavior in simulated robotics environments, option discovery in cooperative MARL poses significant new challenges: 1) merely discovering individually distinguishable behaviors has no direct guarantee that such behaviors are useful for the team objective; 2) increasing the number of latent skills in the hopes of discovering useful skills by chance is impractical for the exponentially larger action space of MARL; and 3) skills must be discovered in the actual multi-agent environment rather than in an isolated single-agent setting.

The key difference from the closest works in hierarchical MARL [2, 38] is that we discover skills with an intrinsic reward instead of hand-crafting subtask-specific rewards. FMH [2] specifies one agent to be a "Manager" while all other agents are "Workers", while there is no hierarchy *among* agents in our general case. A complementary line of work learns *role*-specific parameters and assignment of roles to agents with unique features, where each role is sustained for an entire episode [41]. This is a special case of our agents who dynamically choose skills at multiple time points within an episode. We build on QMIX [26] and independent DQN [37, 23] for our hierarchical agents; however, other algorithms for decentralized cooperative MARL [14] and single-agent RL [35] are equally applicable.

3 Methods

We present a method for fully-cooperative hierarchical MARL, whereby independently-acting agents learn to cooperate using latent skills that emerge from a combination of intrinsic and extrinsic rewards. Inspired by training practices of real world professional sports teams, we create our method within the paradigm of centralized training with decentralized execution [5]. For ease of exposition and intuition, we assume all agents have the same observation space and action space; nevertheless they take individual actions based on individual observations. In the rest of this section, we define the objective of hierarchical MARL with skill discovery, describe our method to solve the optimization problem, and discuss practical implementation techniques for effective learning.

3.1 Combining centralized and decentralized training in hierarchical MARL

We describe a two-level hierarchical MARL setup for training N agents, labeled by $n \in [N]$, as follows. Let $\mathcal{Z}$ denote a set of latent variables z, each of which corresponds to a *skill*. In this work, we use a finite set of latent variables with one-hot encoding; it is possible to generalize $\mathcal{Z}$ to be a learned continuous embedding space [1]. We treat $\mathcal{Z}$ as the *action space* for high-level policies[1] $\mu^n \colon \mathcal{O} \mapsto \mathcal{Z}, \forall n \in [N]$, each of which maps from an agent's observation $o^n \in \mathcal{O}$ to a choice of skill $z^n \in \mathcal{Z}$. Each choice of z^n is sustained for t_{seg} time steps: letting $T = Kt_{\text{seg}}$ denote the length of an episode, there are K time points at which a high-level skill selection is made (explained in Section 3.4). Conditioned on a chosen latent skill and given an agent's observation, a low-level policy $\pi^n \colon \mathcal{O} \times \mathcal{Z} \mapsto \mathcal{A}$ outputs a primitive action a^n in a low-level action space $\mathcal{A}$. Each $z \in \mathcal{Z}$ and the latent-conditioned policy $\pi^n(\cdot; z^n)$ is a skill, in accord with terminology in the literature [13, 9, 1]. Let boldface $\boldsymbol{\mu}$, $\boldsymbol{\pi}$, and $\mathbf{a}$ denote the joint high-level policy, joint low-level policy, and joint action, respectively. Let $(\cdot)^{-n}$ denote a joint quantity for all agents except agent n. At the high level, the learning problem is to train $\boldsymbol{\mu}$ to select skills to optimize an extrinsic team reward function $R \colon \mathcal{S} \times \{\mathcal{A}\}_{n=1}^N \mapsto \mathbb{R}$ that maps global state and joint action to a scalar reward. At the low level, $\{\pi^n\}_{n=1}^N$ are trained to choose primitive actions to produce useful and decodable behavior by optimizing a low-level reward function R_L. Combining the learning problem at high and low levels, we can view hierarchical MARL as a bilevel optimization problem [7]:

$$\max_{\boldsymbol{\mu}, \boldsymbol{\pi}} \mathbb{E}_{\mathbf{z} \sim \boldsymbol{\mu}, P} \left[\mathbb{E}_{s_t, a_t \sim \boldsymbol{\pi}, P} \left[\sum_{t=1}^{T} \gamma^t R(s_t, \mathbf{a}_t) \right] \right] \tag{1}$$

$$\pi^n \in \operatorname*{argmax}_{\pi} \mathbb{E}_{\mathbf{z} \sim \boldsymbol{\mu}, P} \left[\sum_{k=1}^{K} \mathbb{E}_{\tau_k^n \sim \boldsymbol{\pi}, P} \left[R_L(z_k^n, \tau_k^n) \right] \right], \forall n \in [N] \tag{2}$$

where τ_k^n is the k-th trajectory segment that consists of a sequence of observations by agent n, P denotes the environment transition probability $P(s_{t+1}|s_t, \mathbf{a})$, and $R_L(z^n, \tau^n) := \sum_{(s_t, a_t) \in \tau^n} R_L(z^n, s_t, a_t)$ denotes the sum of agent n's low-level rewards along trajectory τ^n. This may also be viewed as a general-sum meta-game between a $\boldsymbol{\mu}$-player and another $\boldsymbol{\pi}$-player. When R_L is the extrinsic team reward, we have a fully-cooperative meta-game, while the other extreme is where R_L solely promotes decodability. Our approach, explained in Section 3.2, lies in between these extremes to strike a balance between usefulness and decodability.

It is difficult to solve (1)-(2) exactly in high-dimensional continuous state spaces. Furthermore, in our work, R_L changes dynamically to promote skill predictability (see Section 3.2). Instead, we approach it using powerful algorithms for MARL and RL. First, we can use centralized MARL algorithms to train high-level policies $\boldsymbol{\mu}$ for cooperative high-level skill selection. While training flat policies using a team reward may lead to the emergence of cooperative behavior [20], explicitly training high-level skill-selection policies allows external control over the skills performed by an agent (i.e., by fixing a latent variable), and subsequently analysis of the behavior for each skill. Second, we apply independent reinforcement learning to train low-level policies $\{\pi^n\}_{n=1}^N$, each conditioned on a skill selected by the agent's corresponding high-level policy, to take primitive actions to optimize $R_L(z^n, \tau^n)$ (defined immediately below in Section 3.2). This reflects the fact that human players in team sports can master skills individually outside of team practice.

3.2 Skill discovery via dynamically weighted decoder-based intrinsic rewards

We define the low-level reward by first introducing a skill decoder $p_\psi(z^n|\tau^n)$ that predicts the ground truth latent skill z^n that was used in the low-level policy $\pi(\cdot; z^n)$ that generated the trajectory τ^n. The decoder is trained using a dataset $\mathcal{D} = \{(z, \tau)\}$ of skill-trajectory pairs, where each consists of the z chosen by a high level policy and the corresponding trajectory τ generated by the low level policy given z, over all agents. $\mathcal{D}$ is accumulated in an online manner during training. Hence, training p_ψ alone can be viewed as a supervised learning problem where we have access to the ground truth "label" z associated with each "datapoint" τ.

We define the intrinsic reward $R_I(z_k^n, \tau_k^n)$ for agent n's k-th trajectory segment τ_k^n via the prediction performance of the skill decoder on the tuple (z_k^n, τ_k^n). Agent n receives this scalar reward upon generating the segment τ_k^n. The key

[1] Without loss of generality, and for consistency with our algorithm implementation below, we use the notation for deterministic policies in this paper.

intuition is that a skill in many complex fully-cooperative team games can be inferred from the trajectory generated by a player who performs primitive actions that implement the skill [18, 14]. For example, any agent who executes a defensive subtask in soccer will move toward opponents in a consistent way that mainly depends on its own observations along a trajectory, while it is only weakly affected by the behavior of other physically distant agents[2]. This intrinsic reward encourages the generation of distinguishable behavior for different skills, since only by doing so can the low-level policy produce sufficiently distinct "classes" in the dataset $\mathcal{D}$ for the decoder to achieve high prediction performance. Hence we define the low-level reward R_L as a combination of team reward R and intrinsic reward R_I:

$$R_L(z^n, \tau^n) := \alpha \sum_{s_t, \mathbf{a}_t \in \tau^n} \gamma^t R(s_t, \mathbf{a}_t) + (1-\alpha) R_I(z^n, \tau^n) \tag{3}$$

$$\text{where } R_I := p_\psi(z^n | \tau^n) \tag{4}$$

In Equation (3), $\alpha \in \mathbb{R}$ is a dynamic weight, chosen via an automated curriculum defined below, which determines the amount of intrinsic versus environment reward. In contrast to prior work on single-agent option discovery that do not use an extrinsic reward [13, 9, 1], we take advantage of the team reward in MARL to guarantee that skills are useful for team performance, and rely on the intrinsic reward only to promote the association of each latent variable with predictable behavior. The reward (3) ensures that the low-level policies, when conditioned on different latent variables, produce trajectories that are 1) sufficiently different, such as "attack opponent net" versus "move to defend own net", to allow decoding of the latent variable, and 2) useful for attaining the true game reward. We decrease α from 1.0 to α_{end} via an automatic curriculum in which α decreases by α_{step} only when the win rate in evaluation episodes, conducted periodically during training, exceeds a threshold $\alpha_{\text{threshold}}$. At high α, low-level policies learn independently to maximize the team reward by taking useful actions, some of which can be composed into interpretable behavior such as making shot attempts and defending the goal. As α decreases and the skill decoder learns to associate trajectories with latent variables, the low-level policy is increasingly rewarded for generating more easily decodable modes of behavior when conditioned on different z, while still optimizing the team reward.

3.3 Algorithm

Algorithm 1 is our concrete approach to the optimization problem eqs. (1) and (2), with skill discovery based on eq. (3). We initialize replay buffers $\mathcal{B}_H, \mathcal{B}_L$ for both levels of the hierarchy, for off-policy updates in similar style to DQN [23], and initialize a dataset $\mathcal{D}$ for the decoder (line 2). At the k-th high-level step, which occurs once for every t_{seg} primitive time steps (line 6), we compute the SMDP reward $\tilde{R}_t := \sum_{i=0}^{t_{\text{seg}}-1} \gamma^i R(s_{t-i}, \mathbf{a}_{t-i})$ for the high-level policy (line 8) [36]. Each agent computes its reward and independently selects a new skill to execute for the next high-level step (lines 12-13). We periodically take gradient steps to optimize the high level cooperative skill-selection objective (1) (lines 15-17), by using the QMIX algorithm [26] to train a centralized Q-function $Q_\phi^{\text{tot}}(s_t, \mathbf{z})$ via minimizing the loss:

$$\mathcal{L}(\phi) := \mathbb{E}_{\boldsymbol{\mu}, \boldsymbol{\pi}} \left[\frac{1}{2} \left(y_k - Q_\phi^{\text{tot}}(s_k, \mathbf{z}_k) \right)^2 \right] \tag{5}$$

$$y_k := \tilde{R}_k + \gamma Q_\phi^{\text{tot}}(s_{k+1}, \mathbf{z}') |_{\{z'^n = \text{argmax}_{z^n} Q_\phi^n(o_{k+1}^n, z^n)\}_{n=1}^N} \tag{6}$$

Q_ϕ^{tot} is a non-linear function (e.g., neural network) that is monotonic in individual utility functions $Q_\phi^n, n \in [N]$, and we denote $\boldsymbol{\mu}$ as the collection of greedy policies induced by Q_ϕ^n. The hypernetwork of QMIX enforces $\frac{\partial Q_\phi^{\text{tot}}}{\partial Q_\phi^n} > 0$, which is a sufficient condition for a global argmax to be achieved via decentralized argmax, i.e., $\text{argmax}_{\mathbf{z}} Q_\phi^{\text{tot}}(\cdot, \mathbf{z}) = \{\text{argmax}_{z^n} Q_\phi^n(\cdot, z^n)\}_{n=1}^N$. This allows centralized training with decentralized skill selection. In general, one can choose from a diverse set of cooperative MARL algorithms with decentralized execution [10, 34, 30, 42].

Conditioned on the choices of skills, each agent independently executes primitive actions at every low-level time step (lines 19-20), using the greedy policy π^n induced by low-level Q-functions $Q_\theta^n(o_t^n, z_t^n, a^n)$. We periodically take gradient steps to optimize the low level objective (2) (lines 23-25), by using independent DQN [37, 40, 23] to optimize

[2] As a first step, we do not include higher-order skills that may only be identified from coordinated behavior that involves trajectories of two or more agents. Our method can be extended to higher-order skills by associating multiple agents' concurrent trajectories with a single skill.

Algorithm 1 Hierarchical MARL with unsupervised skill discovery

1: **procedure** ALGORITHM
2: Initialize high-level Q_ϕ, low-level Q_θ, decoder p_ψ, high-level replay buffer $\mathcal{B}_H$, low-level replay buffer $\mathcal{B}_L$, and trajectory-skill dataset $\mathcal{D}$
3: **for** each episode **do**
4: $s_t, \mathbf{o}_t =$ env.reset()
5: Initialize trajectory storage $\{\tau^n\}_{n=1}^N$ of max length t_{seg}
6: **for** each step $t = 1, \ldots, T$ in episode **do**
7: **if** $t \mod t_{\text{seg}} = 0$ **then**
8: Compute $\tilde{R}_t := \gamma^{t_{\text{seg}}} * \sum_{k=0}^{t_{\text{seg}}} R_{t-k}$
9: Store $(s_{t-t_{\text{seg}}}, \mathbf{o}_{t-t_{\text{seg}}}, \mathbf{z}, \tilde{R}_t, s_t, \mathbf{o}_t)$ into $\mathcal{B}_H$
10: **for** each agent n **do**
11: Store (z^n, τ^n) into $\mathcal{D}$
12: Compute intrinsic reward R_I^n using (4)
13: Select new skill z^n from ϵ-greedy$(Q_\phi^n(o^n, z))$
14: **end for**
15: **if** # (high level steps) $\mod t_{\text{train}} = 0$ **then**
16: Update $Q_\phi(s, \mathbf{z})$ using $\mathcal{B}_H$ and (5)
17: **end if**
18: **end if**
19: Get a_t^n from ϵ-greedy$(Q(o_t^n, z_t^n, a))$ for each agent
20: $s_{t+1}, \mathbf{o}_{t+1}, R_t =$ env.step$(\mathbf{a}_t)$
21: Compute $R_L^n := \alpha R_t + (1-\alpha) R_I^n$ for each agent
22: For all agents, store $(o_t^n, a_t^n, R_L^n, o_{t+1}^n, z^n)$ into low-level replay buffer $\mathcal{B}_L$, and append o_t^n to trajectory τ^n
23: **if** # (low-level steps) $\mod t_{\text{train}} = 0$ **then**
24: Update $Q_\theta(o^n, z^n, a^n)$ using $\mathcal{B}_L$ and (7)
25: **end if**
26: **end for**
27: **if** size of $\mathcal{D} \geq N_{\text{batch}}$ **then**
28: Update decoder $p_\psi(z|\tau)$ using $\mathcal{D}$, then empty $\mathcal{D}$
29: **end if**
30: **if** evaluation win rate exceeds $\alpha_{\text{threshold}}$ **then**
31: $\alpha \leftarrow \max(\alpha_{\text{end}}, \alpha - \alpha_{\text{step}})$
32: **end if**
33: **end for**
34: **end procedure**

Q_θ^n via minimizing the loss:

$$\mathcal{L}(\theta) := \mathbb{E}_{\boldsymbol{\mu}, \boldsymbol{\pi}} \left[\frac{1}{2} \left(y_t^n - Q_\theta^n(o_t^n, z^n, a_t^n) \right)^2 \right] \tag{7}$$

$$y_t^n := R_L(z^n, \tau^n) + \gamma \max_{a^n} \hat{Q}_\theta^n(o_{t+1}^n, z^n, a^n), \forall n \in [N] \tag{8}$$

$\boldsymbol{\pi}$ denotes the collection of greedy policies induced by all Q_θ^n. The low level reward R_L includes the contribution of the intrinsic reward R_I only at the final time step of each length-t_{seg} trajectory segment, i.e., at every high-level step. $\hat{Q}$ is a target network [23].

Once N_{batch} number of (z^n, τ^n) are collected into the dataset $\mathcal{D}$ (lines 11, 27-29), the skill decoder $p_\psi(z|\tau)$ is trained to predict z given τ via supervised learning on $\mathcal{D}$ by minimizing a standard cross-entropy loss. Each chosen z^n acts as the class label for the corresponding trajectory τ^n. Periodically, we evaluate the agents' performance (e.g., win rate) in seperate evaluation episodes; if performance exceeds $\alpha_{\text{threshold}}$, we decrease the weight α by α_{step} with lower bound α_{end} (Section 3.2). While it is extremely challenging to provide theoretical guarantees for hierarchical methods, especially

due to the need for nonlinear function approximation to tackle high-dimensional continuous state spaces, simultaneous optimization in hierarchical RL has shown promising practical results [4, 39].

3.4 Trajectory segmentation and compression

Hierarchical MARL requires agents to change their choice of skills dynamically at multiple times within an episode, such as in response to a change of ball possession in soccer. This means we use partial segments instead of full episode trajectories for skill discovery, in contrast to the single-agent case [13, 9, 1]. At first glance, using a fixed time discretization hyperparameter t_{seg} for segmentation may pose difficulties for the skill decoder, such as when a segment contains qualitatively different behavior that should correspond to different skills. We address this issue by using the time points at which the high-level policy chooses a new set of skill assignments as the segmentation. Hence, $\boldsymbol{\pi}$ learns to generate trajectory segments in between the time points, and p_ψ learns to associate these segments with the chosen latent variables. We synchronize the time points of all agents' high-level skill choice, and all skills are sustained for t_{seg} low-level steps. This corresponds to a special case of the "any" termination scheme, which is dominant over other termination schemes considered in [27]. A practical approach is to define a range of values based on domain knowledge (e.g., empirical average duration of a player's ball possession) and include it in hyperparameter search. We find that agents can still learn skills that requires more than t_{seg} steps to complete, by sustaining the same skill for multiple high-level steps.

Building on techniques in [1], we preprocess each trajectory before using it as input to the decoder. We downsample by retaining every k_{skip} steps, which filters out low-level noise in stochastic environments. We use the element-wise difference between the downsampled observation vectors. This prevents low-level policies from performing degenerate behaviors, such as staying in different regions of the field but taking no actions, as the difference will be indistinguishable for the decoder and result in low intrinsic reward. We reduce the dimension of observation vectors for the decoder by removing entries corresponding to all other agents, while retaining game-specific information (e.g., ball possession). Hence an agent's own trajectory must contain enough information for decoding the latent skill variable.

4 Experimental Setup

The primary purpose of our experiments is to demonstrate that our method discovers interpretable skills that are useful for high-level strategies and has potential for human-AI cooperation in team sports games. Secondarily, we contribute evidence that hierarchical MARL with unsupervised skill discovery can meet or exceed the performance of non-hierarchical methods in high-dimensional environments with only a global team reward. First we describe the simulation setup in Section 4.1. We provide full implementation details of our method and baselines in Section 4.2.

4.1 Simple Team Sports Simulator

The Simple Team Sports Simulator (STS2) captures the high-level rules and physical dynamics of general N versus N team sports while abstracting away fine-grained details that do not significantly impact strategic team play [45]. Stochasticity of ball possession and goals makes STS2 a challenging environment for MARL. Complementary to 3D simulations such as Kurach et al. [17] that require massively parallelized training, STS2 is a lightweight benchmark where MARL agents can outperform the scripted opponent team within hours on a single CPU. We train in 3v3 mode against the scripted opponent team for 50k episodes. Each episode terminates either upon a goal or a tie at 500 time steps.

State. We define a state representation that is invariant under 180 degree rotation of the playing field and switch of team perspective. For one team, the state vector has the following components, making up total dimension 34: normalized position of the player with possession relative to the goal, and its velocity; a 1-hot vector indicating which team or opponent player has possession; for each team and opponent player, its normalized position and velocity.

Observation. Each agent on the home team has its own egocentric observation vector with the following components, making up total dimension 31: normalized position of the player with possession relative to this agent, and its relative

velocity; a binary indicator of whether this agent has possession; a binary indicator of whether its team has possession; its normalized position and its velocity; relative normalized position of each teammate, and their relative velocities; a binary indicator of whether the opponent team has possession; relative normalized position of each opponent player, and their relative velocities.

Action. The low-level discrete set of actions consists of: do-nothing, shoot, pass-1, ... , pass-N, down, up, right, left. Movement and shoot directions are relative to the team's field side. If the agent does not have possession and attempts to shoot or pass, or if it has possession and passes to itself, it is forced to do nothing.

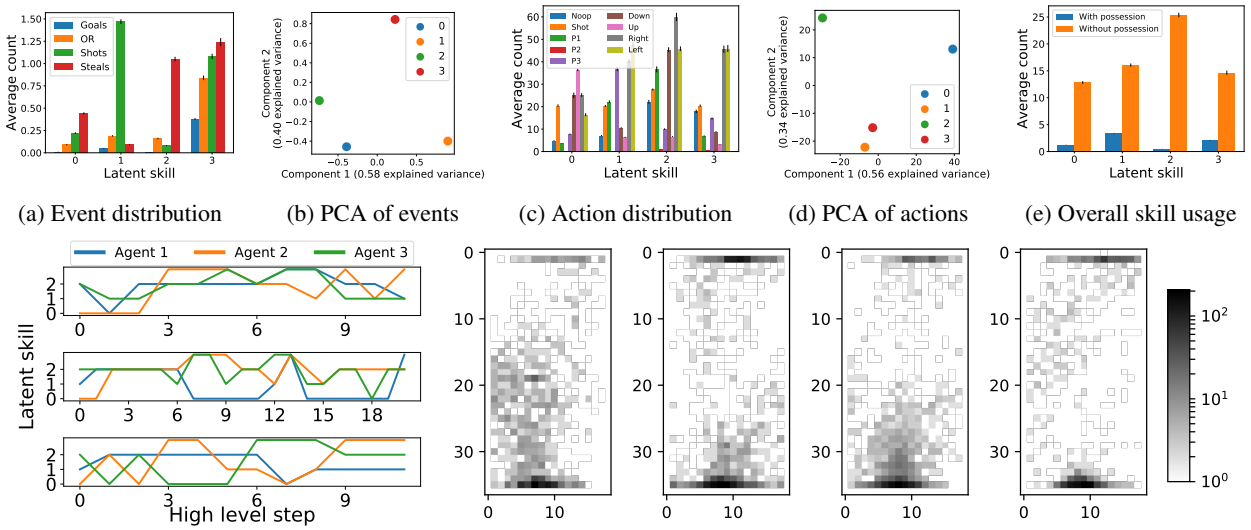
Reward. The team receives reward $+1$ for scoring, -1 when the opponent scores, ± 0.1 on the single step when it regains possession from, or loses possession to, the opponent. We include a reward of $\pm 1/(2 * \text{max steps per episode})$ for having or not having possession.

Game events. We define a set of game events, which are frequently used for analyzing team sports [11], to quantify the effect of skills. Goals: agent scored a goal, upon which an episode ends. Offensive rebound: agent's team made a shot attempt, which missed, and the agent retrieved possession. Shot attempts: agent attempted to score a goal. Made or received pass: agent made (received) a successful pass to (from) a teammate. Steals: agent retrieved possession from an opponent by direct physical contact.

4.2 Implementation and baselines

We use parameter-sharing among all agents, as is standard for homogeneous agents in cooperative MARL [14]. For function approximation, we use fully-connected neural networks without recurrent units since the game is fully observable. Each component is depicted in Figure 1. The low-level Q-function has two hidden layers, each with 64 units, and one output node per action. The high-level Q-function is a QMIX architecture: the individual utility function has two layers with 128 units per layer, and one output node per skill. Utility values of all agents are passed into a mixer network, whose non-negative weights in two hidden layers are generated by hypernetworks of output dimension 64, and whose final output is a single global Q value (see [26]). The skill decoder is a bidirectional LSTM [28] with 128 hidden units in both forward and backward cells, whose outputs are mean-pooled over time and passed through a softmax output layer to produce probabilities over skills. We use batch size $N_{\text{batch}} = 1000$ to train the decoder; ϵ-greedy exploration at both high and low levels with ϵ decaying linearly from 0.5 to 0.05 in 1e3 episodes; replay buffers $\mathcal{B}_H$ and $\mathcal{B}_L$ of size 1e5; learning rate 1e-4; and discount $\gamma = 0.99$. High and low level action-value functions are trained using minibatches of 256 transitions every 10 steps at the high and low levels, respectively. Target networks [23] are updated after each training step with update factor 0.01. We conduct 20 episodes of evaluation once every 100 training episodes. We experimented with 4 and 8 latent skills, $t_{\text{seg}} = 10$, and let α decay from 1.0 to a minimum of 0.6 by $\alpha_{\text{step}} = 0.01$ whenever average win rate during evaluation exceeds $\alpha_{\text{threshold}} = 70\%$. We process trajectory segments as described in Section 3.4 with $k_{\text{skip}} = 2$.

As we instantiate our general method using QMIX [26] at the high level and independent Q-learning (IQL) [37, 23] at the low level, we compare performance with these two baselines to demonstrate that the new hierarchical architecture maintains performance while gaining interpretability. QMIX uses the same neural architecture as our method, except that the individual utility function outputs action-values for primitive actions instead of action values for high-level skills. IQL uses a two-layer Q-network with 128 units per layer. We first performed a coarse manual search for hyperparameters of QMIX and IQL, and used the same same values for the corresponding subset of hyperparameters in our method. Additional hyperparameters ($\alpha_{\text{threshold}}$, α_{step}, and t_{seg}) in our method were chosen from a coarse manual search, and we show results on hyperparameter sensitivity. We also compared with an "Expert" variant of HSD that uses two expert-designed subtask reward functions with the same hierarchical architecture. An agent with subtask 1 gets reward +1 for making a goal when having possession; an agent with subtask 2 gets +1 for stealing possession from an opponent. These *individual* rewards mitigate the difficult problem of multi-agent credit assignment, and so this variant gives a rough indication of maximum possible win rate against the scripted opponent team.



(a) Event distribution (b) PCA of events (c) Action distribution (d) PCA of actions (e) Overall skill usage

(f) Time series of latent skill usage (g) Latent skill usage (skills 0-3, left to right) over the spatial playing field

Figure 2: (a-c): Behavioral investigation of one HSD policy, showing average and standard error over 100 test episodes. (a) Distribution of special game events for each latent skill. (b) Projection of each skill's event distribution via PCA. (c) Distribution of primitive actions for each latent skill, where "Px" denotes "pass to teammate x". (d) Projection of each skill's action distribution via PCA. (e) Count of overall skill usage, when agent team has or does not have possession. (f) Time series of skills selected high-level steps, each consisting of $t_{\text{seg}} = 10$ primitive steps; each subplot shows one independent test episode; (g) Count of skill usage over the full continuous playing field, discretized to a 36x18 grid.

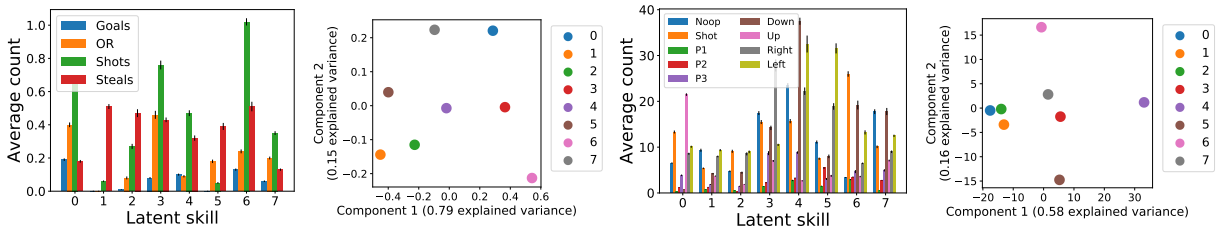
5 Results

Our method for Hierarchical learning with Skill Discovery, labeled "HSD", learns interpretable skills that are useful for high-level cooperation. HSD meets the performance of QMIX and IQL, exceeds them in ad-hoc cooperation, and enables deeper policy analysis due to its hierarchical structure. First, Section 5.1 provides a detailed quantitative behavioral analysis of learned skills. Section 5.2 discusses performance, the effect of hyperparameters, and the result of ad-hoc cooperation.

5.1 Quantitative behavioral analysis

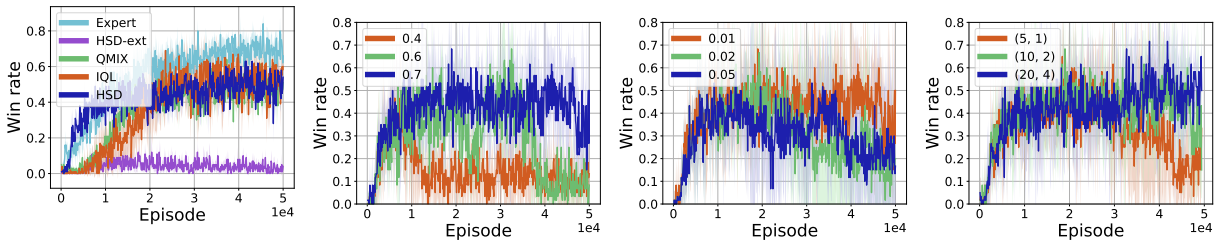
We conducted a quantitative analysis of the discovered skills by measuring the impact of skills on occurrence of game events and primitive actions, agents' choices of skills over an episode, and the spatial occurrence of skills. Figure 2 shows results for the case of four latent skills, which we describe immediately below. We describe the case of eight latent skills later in Figure 3.

Analysis of game events. Figure 2a shows the counts of each game event under each skill, summed over any agent who was assigned to execute the skill, and averaged over 100 test episodes. Skill 1 makes the most shot attempts, Skill 2 provides defense by focusing on steals, while Skill 3 contributes to the most number of successful goals. This difference in game impact, which emerged without any skill-specific reward functions, is also reflected by the large separation of principal components in Figure 2b that result from applying PCA to the vector of event counts of Figure 2a. Figure 2b suggests that component 1 corresponds to tendency to make offensive shots, while component 2 (vertical) corresponds to tendency to make steals. Figure 2c shows the distribution of primitive actions taken by the low-level policy when conditioned on each latent skill. Skill 0 predominantly moves up towards the opponent net to begin offense, Skill 1 is more biased toward the left field, while Skill 2 moves down to defend the home net more than other skills. Figure 2e shows the usage of each skill by the high-level policy, under the cases when agent team has possession and when the opponent team has possession. Skill 2 is strongly associated with lack of possession since it is a defensive skill for regaining possession.



(a) Event distribution (b) PCA of events (c) Action distribution (d) PCA of actions

Figure 3: Behavioral analysis of HSD policies with 8 latent skills. (a) Skill 0 makes the most goals, skill 1 focuses on defensive steals, skill 6 makes the most shot attempts. (b) Differences between skills, especially skills 0, 1 and 6, are reflected by the PCA reduction of events. (c) Skill 0 predominantly moves up, which explains its high goal rate, while skill 4 moves down the most (d) These distinguishable characteristics of skills are reflected by their large separation after PCA reduction.



(a) Win rate against scripted bots (b) Varying $\alpha_{\text{threshold}}$ (c) Varying α_{step} (d) Varying t_{seg} and k_{skip}

Figure 4: Win rate against scripted opponent team over training episodes. Each curve is the mean over random seeds (5 for (a) and 3 for (b-d)) with shaded region representing the 95% confidence interval. (a) HSD is within margin of error with QMIX and IQL. Expert has the same hierarchical architecture as HSD but is trained with expert hand-scripted subtask rewards. (b-d) Learning is more stable when using a high $\alpha_{\text{threshold}}$, small α_{step}, and when skills are sustained for longer t_{seg}.

Time series of skill usage. Figure 2f shows a time series of skill usage over high-level steps by each agent during three different episodes (from top to bottom). Importantly, agents learned to choose complementary skills, such as in Episode 3 when Agent 3 stays for defense while Agents 1 and 2 execute offense via Skills 1 and 3, at step 9. Each individual agent also dynamically switches between skills, such as in Episode 1 when Agents 1 and 3 switch from the defensive Skill 2 to the offensive Skill 3 at step 6. As shown by the extended periods in all episodes when all agents play the defensive Skill 2, agents are able to sustain the same skill over multiple consecutive high-level steps, which mitigates the concern over choosing a fixed t_{seg}. Note that at any given time in the game, the defensive Skill 2 is almost always used by some agent either to make steals or cover the home net.

Spatial occurrence of skills. Figure 2g is a heatmap of skill usage over the playing field. Consistent with the previous analysis, Skill 0 is used for moving up for offense, Skills 1 and 3 tend to camp near the opponent net (top of field) to make shot attempts, while Skill 2 is concentrated near the home net (bottom of field) to make defensive steals.

Increasing number of skills. The number of latent skills is also a key design choice to make based on domain knowledge. Figure 3 analyzes HSD when trained with eight skills. Skills 0, 3, and 6 focus on shot attempts and offensive rebounds (Figure 3a), and they have high values of the first principal component (Figure 3b). Skills 1 and 2 focus on defensive steals. Figure 3c shows that Skill 0 moves up for offense the most, while Skill 4 moves down to play defense. This is reflected by their large separation in the first principal component (Figure 3d).

5.2 Performance and parameter sensitivity

Figure 4 shows win rate against the scripted opponent team over training episodes for HSD and baselines, each with 5 independent runs, and for varying hyperparameter settings of HSD, each with 3 independent runs. HSD agents learn faster than QMIX and IQL, while their final performance are within the margin of error (Figure 4a). HSD-ext does not have access to extrinsic rewards and underperforms the rest. This supports our hypothesis that the extrinsic team reward is needed in combination with the intrinsic reward to promote useful behavior. The "Expert" variant of HSD outperformed other methods, showing that using cooperative learning at the high-level and independent learning at the low level is a strong approach, and improvement to skill discovery is possible.

We investigated the effect of varying the key hyperparameters of HSD. Figure 4b shows that larger values of $\alpha_{\text{threshold}}$ gives higher performance and lower variance. A small $\alpha_{\text{threshold}}$ increases the likelihood that a spuriously high evaluation performance crosses the threshold, which would cause a re-weighting of the extrinsic versus intrinsic reward even when the agents have not yet adapted to the current reward. Likewise, Figure 4c shows that a smaller value of α_{step} performs better, because each adjustment of the low-level reward is smaller and hence the automatic curriculum is easier for learning. Figure 4d shows that agents who sustain high-level skills for 10 or 20 time steps perform better than agents who sustain only for 5 steps. A smaller t_{seg} means that agents make more frequent decisions to sustain or switch their choice of skill, which allows for more flexible policies but increases the difficulty of learning.

Table 1: Win/lose percentage of final policies over 100 test episodes and 5 seeds, matched with different teammates.

	HSD		QMIX		IQL	
Teammate	Win	Lose	Win	Lose	Win	Lose
Training	46 (4)	39 (4)	55 (3)	23 (3)	36 (7)	46 (4)
1 scripted	49 (4)	45 (3)	48 (4)	44 (4)	32 (3)	54 (4)
2 scripted	52 (3)	45 (1)	45 (2)	51 (2)	37 (2)	58 (1)
1 defensive	43 (5)	42 (4)	-	-	-	-
1 offensive	45 (2)	41 (1)	-	-	-	-

Ad Hoc cooperation. We investigated the test performance of agents in ad-hoc cooperation, by giving them teammate(s) with whom they never previously trained [32]. This mimics the setting where AI agents must cooperate with a human player in team sports games. Table 1 shows the win and lose percentage of HSD, QMIX, and IQL (an episode may end in a draw). HSD agents perform as well or better when one or two of their teammates are replaced by the scripted bots. The fact that HSD agents have independently-trained low-level policies may be a contributing factor. However, QMIX agents performed significantly worse when paired with scripted bots, likely because the out-of-training behavior of bots pose difficulties for QMIX agents who underwent fully-centralized training. IQL agents also lost significantly more often with scripted teammates. For HSD, we can also fix one agent to always play a defensive or offensive skill for the entire episode. Based on Figure 2a, we chose Skill 1 for offense and Skill 2 for defense. HSD agents are able to maintain their performance within the margin of error.

6 Conclusion and discussion

We presented a method for hierarchical multi-agent reinforcement learning that discovers useful skills for strategic teamwork. We train cooperative decentralized policies for high-level skill selection and train independent low-level policies to execute chosen skills, which emerge from a dynamically weighted combination of intrinsic and extrinsic rewards. We demonstrated the emergence of quantifiable, distinct and useful skills in stochastic team sports simulations without assigning a reward to each skill. These findings are a step toward multi-agent game AI that execute realistic high-level strategies and can cooperate with human players.

There are many interesting avenues for future work. One may condition high-level policies on unique agent features, such that agents play different roles [41] that affect their choice of skills. Asynchronous termination [4] of subtasks is a natural generalization of this work to learn in a larger space of policies. The question of how to optimize the number of skills is also open. One can also consider curriculum-learning approaches that initialize the skill-conditioned low-level

policies from pretraining in an induced single-agent setting [42] or using expert data, analogous to the way professional players practice skills individually outside of team matches. This may speed up training since low-level policies can already generate goal-directed trajectories that are conducive for segmenting into distinguishable skills.

Acknowledgements

We are grateful to Ahmad Beirami (Facebook AI) for significant contributions to extensive and insightful discussions throughout the course of this work, and for detailed feedback that helped to improve the clarity and precision of the paper. We also thank the following individuals from Electronic Arts: Maziar Sanjabi and Yunqi Zhao for helpful discussions; Jason Rupert and Caedmon Somers for STS2 development; and Mohsen Sardari and Kazi Zaman for their support.

References

[1] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. 2018. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299* (2018).

[2] Sanjeevan Ahilan and Peter Dayan. 2019. Feudal multi-agent hierarchies for cooperative reinforcement learning. *arXiv preprint arXiv:1901.08492* (2019).

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[4] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[5] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research* 27, 4 (2002), 819–840.

[6] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2013. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics* 9, 1 (2013), 427–438.

[7] Benoît Colson, Patrice Marcotte, and Gilles Savard. 2007. An overview of bilevel optimization. *Annals of operations research* 153, 1 (2007), 235–256.

[8] Peter Dayan and Geoffrey E Hinton. 1993. Feudal reinforcement learning. In *Advances in neural information processing systems*. 271–278.

[9] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2019. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*.

[10] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[11] Alexander Franks, Andrew Miller, Luke Bornn, Kirk Goldsberry, et al. 2015. Characterizing the spatial structure of defensive skill in professional basketball. *The Annals of Applied Statistics* 9, 1 (2015), 94–121.

[12] Mohammad Ghavamzadeh, Sridhar Mahadevan, and Rajbala Makar. 2006. Hierarchical multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 13, 2 (2006), 197–229.

[13] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. 2016. Variational intrinsic control. *arXiv preprint arXiv:1611.07507* (2016).

[14] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. 2018. Is multiagent deep reinforcement learning the answer or the question? A brief survey. *arXiv preprint arXiv:1810.05587* (2018).

[15] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

[16] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*. 3675–3683.

[17] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. 2019. Google Research Football: A Novel Reinforcement Learning Environment. *arXiv preprint arXiv:1907.11180* (2019).

[18] Hoang M Le, Yisong Yue, Peter Carr, and Patrick Lucey. 2017. Coordinated multi-agent imitation learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1995–2003.

[19] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*. Elsevier, 157–163.

[20] Siqi Liu, Guy Lever, Josh Merel, Saran Tunyasuvunakool, Nicolas Heess, and Thore Graepel. 2019. Emergent coordination through competition. In *International Conference on Learning Representations*.

[21] Rajbala Makar, Sridhar Mahadevan, and Mohammad Ghavamzadeh. 2001. Hierarchical multi-agent reinforcement learning. In *Proceedings of the fifth international conference on Autonomous agents*. ACM, 246–253.

[22] Amy McGovern and Andrew G Barto. 2001. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. In *Proceedings of the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 361–368.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[24] Liviu Panait and Sean Luke. 2005. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems* 11, 3 (2005), 387–434.

[25] Doina Precup. 2000. Temporal Abstraction in Reinforcement Learning. *Ph. D. thesis, University of Massachusetts* (2000).

[26] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*. 4295–4304.

[27] Khashayar Rohanimanesh and Sridhar Mahadevan. 2003. Learning to take concurrent actions. In *Advances in neural information processing systems*. 1651–1658.

[28] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[29] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[30] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement learning. In *International Conference on Machine Learning*.

[31] Martin Stolle and Doina Precup. 2002. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*. Springer, 212–223.

[32] Peter Stone, Gal A Kaminka, Sarit Kraus, and Jeffrey S Rosenschein. 2010. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

[33] Peter Stone and Manuela Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots* 8, 3 (2000), 345–383.

[34] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2018. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2085–2087.

[35] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

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

[37] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*. 330–337.

[38] Hongyao Tang, Jianye Hao, Tangjie Lv, Yingfeng Chen, Zongzhang Zhang, Hangtian Jia, Chunxu Ren, Yan Zheng, Changjie Fan, and Li Wang. 2018. Hierarchical deep multiagent reinforcement learning. *arXiv preprint arXiv:1809.09332* (2018).

[39] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 3540–3549.

[40] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.

[41] Aaron Wilson, Alan Fern, and Prasad Tadepalli. 2010. Bayesian policy search for multi-agent role discovery. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

[42] Jiachen Yang, Alireza Nakhaei, David Isele, Hongyuan Zha, and Kikuo Fujimura. 2018. CM3: Cooperative Multi-goal Multi-stage Multi-agent Reinforcement Learning. *arXiv preprint arXiv:1809.05188* (2018).

[43] Zhi Zhang, Jiachen Yang, and Hongyuan Zha. 2019. Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization. *arXiv preprint arXiv:1909.10651* (2019).

[44] Yunqi Zhao, Igor Borovikov, Ahmad Beirami, Jason Rupert, Caedmon Somers, Jesse Harder, Fernando de Mesentier Silva, John Kolen, Jervis Pinto, Reza Pourabolghasem, et al. 2019. Winning Isn't Everything: Training Human-Like Agents for Playtesting and Game AI. *arXiv preprint arXiv:1903.10545* (2019).

[45] Yunqi Zhao, Igor Borovikov, Jason Rupert, Caedmon Somers, and Ahmad Beirami. 2019. On Multi-Agent Learning in Team Sports Games. *arXiv preprint arXiv:1906.10124* (2019).