Anonymous authors
Paper under double-blind review

ABSTRACT

In multi-agent reinforcement learning environments, exploration can be inefficient as the joint policy space is in general exponentially large. Hence, even if the dynamics are simple, the optimal policy can be combinatorially hard to discover. We introduce Multi-Agent Coordinated Exploration (MACE), a deep reinforcement learning approach to learn expressive multi-agent policies that are both statistically and computationally efficient. MACE implements a form of multi-agent coordination during exploration and execution, by using a hierarchical stochastic multi-agent policy class that encodes flexible structure between individual policies. In this way, MACE preserves expressivity along with low sample complexity. In order to make learning tractable, we derive a joint learning and exploration strategy by combining hierarchical variational inference with actor-critic learning. The benefits of our learning approach are that 1) it is principled, 2) simple to implement and 3) easily scalable to settings with many agents. We demonstrate empirically that MACE can more efficiently learn optimal policies in challenging multi-agent games with a large number (∼20) of agents, compared to conventional baselines. Moreover, we show that our hierarchical structure leads to meaningful agent coordination.

1 INTRODUCTION

In many real-life situations, intelligent agents have to solve (multiple) tasks collectively, rather than in isolation. For example, a fleet of $N$ autonomous delivery vehicles might need to deliver $M \geq N$ packages, while avoiding collisions and satisfying time and fuel constraints. In such multi-agent-task settings, agents must learn jointly to coordinate to optimize their (collective) rewards. Instances of this problem have been studied previously, but often make limiting assumptions, e.g. consider a static environment (the bandit setting) or discrete payoffs in matrix games (Bouzy & Métivier (2010)). For dynamic multi-agent-task problems, it is instead natural to use reinforcement learning to train a collective of agents. One important technical challenge is dealing with the combinatorial complexity of the multi-agent-task learning problem. For instance, agents can be assigned to tasks in an exponentially large number of ways (w.r.t. to the number of agents & tasks).

Coordination can take many forms: we focus on (long-term) spatiotemporal task assignment. As a motivating example, consider a variant of the Hare-Hunters problem (see Figure 1), with hyper-parameters $(N = 2, M = 2, H = 1, T)$. In this game, $N = 2$ identical hunters need to capture $M = 2$ identical prey within $T$ time-steps, and exactly $H = 1$ hunter is needed to capture each prey. $T$ is set such that no hunter can capture both preys. This problem has two equivalent solutions: hunter 1 captures prey 1 and hunter 2 captures prey 2, or vice versa. There are also two suboptimal choices: both hunters choose the same prey. Hence, the hunters must coordinate over a (large) number of time-steps to maximize their reward. This implies the solution space has low-dimensional structure that can be exploited to accelerate training.

In general, for $N \gg 1$ agents and $M \gg 1$ tasks, discovering such coordinated solutions using unstructured trial-and-error methods (i.e. agents explore state-actions independently) quickly becomes intractable, as the joint state-action space is exponentially large (e.g. $O(|A|^N)$). Hence, reinforcement learning agents can greatly benefit from employing structured exploration in order to achieve tractable multi-agent-task learning, e.g. by encoding coordination during learning.

Figure 1: Equivalent solutions in a 2-hunter 2-prey game.
In this work, we propose a hierarchical stochastic policy class to enable efficient structured exploration and learning. The key ideas of our approach are: 1) to utilize a shared stochastic latent variable model that defines the structured exploration policy, and 2) to employ a principled variational method to learn the posterior distribution over the latents jointly with the optimal policy. Our approach has several desirable properties. First we do not incorporate any form of prior domain knowledge, but rather discover the coordination structure purely from empirical experience during learning. Second, our variational learning method enables fully differentiable end-to-end training of the entire multi-agent policy class. Finally, by utilizing a hierarchical policy class, our approach can easily scale to large numbers of coordinating agents.

To summarize, our contributions in this work are as follows:

- We introduce a multi-agent policy class that uses a hierarchy of stochastic latent variables.
- We propose an efficient and principled algorithm using variational methods to train the policy end-to-end.
- To validate our learning framework, we introduce several synthetic multi-agent environments that explicitly require team coordination, and feature competitive pressures that are characteristic of many coordinated decision problems.
- We empirically verify that our approach improves sample complexity on coordination games with a large number \( (N \sim 20) \) of agents.
- We show that learned latent structures correlate with meaningful coordination patterns.

## 2 COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING

In multi-agent RL, agents sequentially interact within an environment defined by the tuple: \( E \equiv (S, A, r, f_P) \). Each agent \( i \) starts in an initial state \( s_i^0 \), and at each time \( t \) observes a state \( s_i \in S \) and executes an action \( a_i^t \) chosen by a (stochastic) policy \( a_i^t \sim P(a_i^t|s_i) \). Each agent then receives a reward \( r_i^t(\mathbf{s}_i, \mathbf{a}_i) \) and the environment transitions to a new state \( s_{i+1} \) with probability \( f_P(s_{i+1}|s_i, a_t) \). We define the joint state and actions as \( s_i = \{ s_i^t \in S \} \) and \( a_i = \{ a_i^t \in A \} \), where \( i \in I \) indexes the agents. Note that the rewards for each agent \( r^i \) can depend on the full joint state and actions.

In this work, we restrict to fully cooperative MDPs that are fully observable, deterministic and episodic. Each agent can see the full state \( s \), \( f_P \) is deterministic and each episode \( \tau = (s_i, a_i, r_i)_{0 \leq i \leq T} \) ends when the agent encounters a terminal state and the MDP resets.

In the fully cooperative case, the goal for each agent is to learn its optimal policy \( P^* (a_i|s_i, \theta) \) that maximizes the total reward \( R(\tau) = \sum_i R^i(\tau) = \sum_t r^i(s_i, a_i) \):

\[
\max_{\theta} J(\theta) = \max_{\theta} \sum_{i \in I} J^i(\theta), \quad J^i(\theta) = \mathbb{E} \left[ R^i(\tau) \right | a_t \sim P(a_t|s_t, \theta)]
\]  

(1)

To optimize, we can apply gradient descent with policy gradient estimators \( \hat{g}_\theta \) (Williams (1992))

\[
g_\theta = \nabla_\theta J(\theta) = \mathbb{E} [\nabla_\theta \log P(a_t|s_t, \theta) R(\tau) | a_t, s_t] \approx \frac{1}{M} \sum_{k=1}^M \sum_t \nabla_\theta \log P(a_t^k|s_t^k, \theta) R(\tau^k), \quad (2)
\]

where we sample \( M \) rollouts \( \tau^k \) by sampling actions from the policy that is being learned.

Compared to the single-agent RL setting, multi-agent RL poses unique difficulties. A central issue is the exploration-exploitation trade-off: agents must sample rollouts from an exponentially large state-action space, but discovering good coordinated policies when each agent samples independently becomes combinatorially intractable, e.g. as \( N \) grows or the problem requires long-term coordination.

### 2.1 MACE: Joint Coordination and Exploration

We now formulate our multi-agent objective (1) using a hierarchical policy class that enables \textit{structured exploration}. Our approach, MACE (“Multi-Agent Coordinated Exploration”), builds upon two complementary approaches:
where we denoted \( \lambda \) the joint policy for a single time-step as: \( \lambda \). Global variables \( R \) view the total reward so, we assume that the total reward \( R \) is intractable in general as computing the normalization \( P(s_t) \) involves marginalization over \( \lambda_t \). Hence, to make learning tractable, we will use a variational approach.

Hierarchical Latent Model. To encode coordination between agents, we assume the individual policies have shared structure, encoded by a latent variable \( \lambda_t \). This leads to a hierarchical policy model, as shown in Figure 2. Hence, our goal is to learn the joint distribution \( P(a_t, \lambda_t|s_t) \). We first write the joint policy for a single time-step as:

\[
P(a_t|s_t) = \int d\lambda_t P(a_t, \lambda_t|s_t) = \prod_{i=1}^N \int d\lambda_i P(a_t^i, \lambda_i, s_t) = \prod_{i=1}^N \int d\lambda_i P(a_t^i|\lambda_i, s_t)P(\lambda_i|s_t). \tag{3}
\]

Since the \( a_t \) are conditionally dependent via \( \lambda_t \), the policy is more flexible compared to standard fully factorized policies (Ranganath et al. 2015). Finding the true posteriors \( P(\lambda_t|s_t) \propto P(s_t|\lambda_t)P(\lambda_t) \) is intractable in general as computing the normalisation \( P(s_t) \) involves marginalization over \( \lambda_t \). Hence, to make learning tractable, we will use a variational approach.

Hierarchical Variational Lower Bound. We next derive a tractable learning algorithm using variational methods. Instead of directly optimizing (1), we cast it as a probabilistic inference problem, as in Levine & Koltun (2013); Vlassis et al. (2009), and instead optimize a lower bound. To do so, we assume that the total reward \( R^i(\tau) \) for each \( i \) to be non-negative and bounded. Hence, we can view the total reward \( R^i(\tau) \) as a random variable, whose unnormalized distribution is defined as \( P(R^i|\tau) = R^i \). We can then rewrite (1) as a maximum likelihood problem (omitting \( \phi \) for clarity):

\[
\max_{\theta} \mathbb{E}_{\pi(s_t,\theta)}[R] = \max_{\theta} \int d\tau R(\tau)P(\tau;\theta) \Leftrightarrow \max_{\theta} \int d\tau P(R|\tau)P(\tau;\theta), \tag{4}
\]

\[
P(\tau;\theta) = P(s_0) \prod_{t=0}^T P(s_{t+1}|s_t, a_t) \int d\lambda_t P(a_t, \lambda_t|s_t; \theta) \tag{5}
\]

Denoting time sequences as \( \lambda = (\lambda_t)_{0 \leq t \leq \tau} \) for clarity, we can use a variational posterior \( Q_R(\lambda|s_t; \phi) \) and Jensen’s inequality (Hoffman et al. 2013) to get a lower bound on the log-likelihood of (4):

\[
\log P(R; \theta) \geq \int d\tau d\lambda Q_R(\lambda|s_t; \phi) \log \left( \frac{P(R|\tau)P(s_0) \prod_{t=0}^T P(s_{t+1}|s_t, a_t)P(a_t, \lambda_t|s_t; \theta)}{Q_R(\lambda|s_t; \phi)} \right) \tag{6}
\]

where we denoted \( \int d\tau d\lambda = \sum_{t=0}^T \int ds_t da_t d\lambda_t \). The term \( L(Q_R, \theta, \phi) \) in Equation (6) is called the evidence lower bound (ELBO) which we can maximize as a proxy for (4). From the form of the ELBO, we see that the optimal \( Q_R \) factorizes similarly to \( P \):

\[
Q_R(\lambda|s_t; \phi) = P(R|\tau)P(s_0) \prod_{t=0}^T P(s_{t+1}|s_t, a_t)Q(\lambda_t|s_t; \phi). \tag{7}
\]
We can then simplify the ELBO as:

\[ L(Q_R, \theta, \phi) = \int d\tau d\lambda Q_R(\lambda|s; \phi) \sum_{t=0}^{T} \left( \log P(a_t|\lambda_t, s_t; \theta) + \log \frac{P(\lambda_t|s_t)}{Q(\lambda_t|s_t, \phi)} \right), \tag{8} \]

using \( P(a_t|\lambda_t, s_t; \theta) = P(a_t|\lambda_t, s_t; \theta) P(\lambda_t|s_t) \). For more details, see the Appendix. The standard choice is to use maximum-entropy standard-normal priors: \( P(\lambda_t|s_t) = N(0,1) \). We can then optimize \( L \) using e.g. stochastic gradient ascent. Formally, the gradient \( \nabla_\theta L \) of \( L \) is:

\[ \nabla_\theta L(Q_R, \theta, \phi) = \prod_{t=0}^{T} \int ds_t da_t d\lambda Q_R(\lambda|s_t; \phi) \sum_{t'=0}^{T} \nabla_\theta \log P(a_{t'}|\lambda_{t'}, s_{t'}; \theta), \tag{9} \]

with \( \nabla_\phi L(Q_R, \theta, \phi) \) calculated similarly. These gradients can be estimated using sampled roll-outs \( \tau^k \) of the policy \( P^\pi \). During a rollout \( \tau^k \), we sample \( \lambda \sim Q \), observe rewards \( R \sim P(R|\tau) \) and transitions \( s_{t+1} \sim P(s_{t+1}|.) \): the sample estimated \( \hat{g}_\theta \) approximates the policy gradient \( g^\pi \):

\[ g_\theta \approx \hat{g}_\theta, Q = \frac{1}{M} \sum_{k=1}^{M} \sum_{t=0}^{T} \nabla_\theta \log P(a^k_t|\lambda^k_t, s^k_t; \theta) R(\tau^k), \quad g_\phi \approx \hat{g}_\phi, Q = \nabla_\phi L(Q_R, \theta, \phi). \tag{10} \]

The second term in \( \hat{g} \) is the gradient for the variational posterior \( Q_R \), which learns to sample the optimal \( \lambda_t \). This shows that using an approximate posterior \( Q \) introduces a mismatch between \( \hat{g} \) and \( g \); if \( Q(\lambda|s_t) = P(\lambda_t|s_t) \), the second term in \( \hat{g} \) would be 0 and \( g^\pi \) would be equivalent.

**Actor-Critic and Bias-Variance.** Estimating policy gradients \( g_\theta \) using empirical rewards can suffer from high variance and instabilities. It is thus useful to consider more general objectives \( F^\eta \): such that the variance in \( \hat{g} \) is reduced. In practice, we find that using \( F^\eta \) with more general \( F \), such as generalized advantages (Schulman et al. (2015)), performs quite well.

### 3 Experimental Validation

#### 3.1 Multi-agent Environments

To validate our approach, we created two grid-world games, depicted in Figure 2, inspired by the classic Predator-Prey and Stag-Hunt games (Shoham & Leyton-Brown (2008)). In both games, the world is periodic and the initial positions of the hunters and prey are randomized. Also, we consider two instances for both games: either the prey are moving or fixed.

**Hare-Hunters.** Predator-Prey is a classic test environment for multi-agent learning, where 4 predators try to capture a prey by boxing it in. We consider a variation defined by the settings \((N, M, H, T)\): \(N\) hunters and \(M\) prey. Each prey can be captured by exactly \(H\) hunters: to capture the prey, a hunter gets next to it, after which the hunter is frozen. Once a prey has had \(H\) hunters next to it, it is frozen and cannot be captured by another hunter. The terminal rewards used are:

\[ R^i = \begin{cases} 1, & \text{if all prey are captured } H \text{ times before the time limit } T \\ 0, & \text{otherwise} \end{cases} \tag{12} \]

The challenge of the game is for the agents to inactivate all prey within a finite time \(T\). Due to the time limit, the optimal strategy is for the agents to distribute targets efficiently, which can be challenging due to the combinatorially large number of possible hunter-to-prey assignments.

**Stag-Hunters.** The Stag-Hunt is another classic multi-agent game designed to study coordination. In this game, hunters have a choice: either they capture a hare for low reward, or, together with another hunter, capture a stag for a high reward. We extend this to the multi-agent \((N, M, H, T)\)-setting: \(N\) hunters hunt \(M\) prey \((M/2\) stags and \(M/2\) hares\). Each stag has \(H\) hit-points, while hares and

---

Note that if the total reward \(R\) is bounded, we can define \(R\)-weighted probabilities by shifting and normalizing \(R\). In general, the derivation applies if \(F\) is similarly bounded.
hunters have 1 hit-point. Capturing is as in Hare-Hunters. The spatial domain is similar to the Hare-Hunters game and we also use a time limit \( T \). The terminal reward is now defined as:

\[
R^i = \begin{cases} 
1, & \text{if } i \text{ captured a live stag that became inactive before the time limit } T \\
0.1, & \text{if } i \text{ captured a live hare before the time limit } T \\
0, & \text{otherwise}
\end{cases}
\]  

(13)

The challenge for the agents here is to discover that choosing to capture the same prey can yield substantially higher reward, but this requires coordinating with another hunter.

### 3.2 Neural Coordination Model

For experiments, we instantiated our multi-agent policy class (as in Figure 2) with deep neural networks. For simplicity, we only used reactive policies without memory, although it is straightforward to apply MACE to policies with memory (e.g. LSTMs). The model takes a joint state \( s_i \) as input and computes features \( \phi(s) \) using a 2-layer convolutional neural network. To compute the latent variable \( \lambda \in \mathbb{R}^d \), we use the reparameterization trick (Kingma & Welling, 2013) to learn the variational distribution (e.g. \( Q(\lambda|s) \)), sampling \( \lambda \) via \( \epsilon \sim \mathcal{N}(0,1) \) and distribution parameters \( \mu, \sigma \) (omitting \( t \)):

\[
\mu(s) = W_0\phi(s) + b_\mu, \quad \log \sigma(s)^2 = W_\sigma\phi(s) + b_\sigma, \quad \lambda = \mu(s) + \sigma(s) \odot \epsilon.
\]  

(14)

Given \( \lambda \), the model then computes the policies \( P(a^i|\lambda,s) \) and value functions \( V^i(s) \) as (omitting \( t \)):

\[
P(a^i|\lambda,s) = \text{softmax} \left( W_\pi^i [\lambda \phi(s)] + b_\pi^i \right), \quad V^i(s) = W_v^i \phi(s) + b_v^i,
\]  

(15)

where \( \text{softmax}(x) = \exp x / \sum_j \exp x^j \). In this way, the model can be trained end-to-end.

**Training.** We used A3C (Mnih et al., 2016) with KL-controlled policy gradients (10), generalized advantage as \( F \) (Schulman et al., 2015), and policy-entropy regularization. For all experiments, we performed a hyper-parameter search and report the best 5 runs seen (see the Appendix for details).

**Baselines.** We compared MACE against two natural baselines:

- **Shared** (shared actor-critic): agents share a deterministic hidden layer, but maintain individual weights \( \theta^i \) for their (stochastic) policy \( P(a^i|\lambda,s;\theta^i) \) and value function \( V^i(s;\theta) \). The key difference is that this model does not sample from the shared hidden layer.
- **Cloned** (actor-critic): a model where each agent uses an identical policy and value function with shared weights. There is shared information between the agents, and actions are sampled according to the agents’ own policies.

### 4 Quantitative Analysis

Above, we defined a variational approach to train hierarchical multi-agent policies using structured exploration. We now validate the efficacy of our approach by showing our method scales to environments with a large number of agents. We ran experiments for both Hare-Hunters and Stag-Hunters for \( N = M = 10, 20 \) in a spatial domain of \( 50 \times 50 \) grid cells.

**Sample complexity.** In Table 1, we show the achieved rewards after a fixed number of training samples, and Figure 3 showcases the corresponding learning curves. We see that MACE achieves up to 10\times reward compared to the baselines. Figure 4 shows the corresponding distribution of training episode lengths. We see that MACE solves game instances more than 20\% faster than baselines in 50\% (10\%) of Hare-Hunters (Stag-Hunters) episodes. In particular, MACE learns to coordinate for higher reward more often: it achieves the highest average reward per-episode (e.g. for 10-10 Stag-Hunters with frozen prey, average rewards are 4.64 (Cloned), 6.22 (Shared), 6.61 (MACE)). Hence, MACE coordinates successfully more often to capture the stags. Together, these results show MACE enables more efficient learning.

**Using the ELBO.** A salient difference between (10) and (2) is the KL-regularization, which stems from the derivation of the ELBO. Since we use a more general objective \( F \), c.f. (11), we also investigated the impact of using the KL-regularized policy gradient (10) versus the standard (2). To this end, we ran several instances of the above experiments both with and without KL-regularization.
Figure 3: Train-time cumulative terminal reward for $N$ agents in $(N, M, 1, T)$ Hare-Hunters (upper, $T = 2000$) and Stag-Hunters (lower, $T = 1000$) on a $50 \times 50$ gridworld, for 10-vs-10 or 20-vs-20 agents; randomly moving or fixed preys. Average, minimal and maximal rewards for the best 5 runs for each model are shown. MACE accumulates increasingly higher rewards compared to the baselines, by 1) achieving higher terminal reward per episode and 2) finishing episodes faster (see Figure 4). For 10-10 Stag-Hunters with frozen prey, average reward per-episode is 4.64 (Cloned), 6.22 (Shared), 6.61 (MACE) after 1 million samples. For more, see the Appendix.

Figure 4: Train-time episode lengths during 1 million steps for 10-vs-10 Hare-Hunters (left) and Stag-Hunters (right), with fixed preys. MACE (orange) finishes an episode successfully before the time limit more often than the baselines (Cloned (blue) and Shared (yellow)).

We found that without KL-regularization, training is unstable and prone to mode collapse: the variance $\sigma$ of the variational distribution can go to 0. This reflects in essentially 0 achieved reward: the model does not solve the game for any reasonable hyperparameter settings.

**Impact of dynamics and $T$.** Inspecting training performance, we see the relative difficulty of capturing moving or randomly moving prey. Capturing moving prey is easier to learn than capturing fixed preys, as comparing rewards in Table 1 shows. This shows a feature of the game dynamics: the expected distance between a hunter and an uncaptured prey are lower when the preys are randomly moving, resulting in an easier game. Comparing Hare-Hunters and Stag-Hunters, we also see the impact of the time limit $T$. Since we use terminal rewards only, as $T$ gets larger, the reward becomes very sparse and models need more samples to discover good policies.

5 Model Inspection

Beyond training benefits, we now demonstrate empirical evidence that suggest efficacy and meaningfulness of our approach to structured exploration. We start by inspecting the behavior of the latent variable $\lambda$ for a simple $N = M = 2$ Hare-Hunters game, which enables semantic inspection of the learned policies, as in Figure 5. We make a number of observations. First, $\lambda$ is relevant: many components are statistically significantly correlated with the agents’ actions. This suggests the model
Preys are Frozen Moving Frozen Moving
Samples (x100k) 5 10 5 10 5 10 5 10

<table>
<thead>
<tr>
<th></th>
<th>10-vs-10</th>
<th>20-vs-20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloned Hare-Hunters</td>
<td>85.0 178.3 465.0 912.5 20.0 70.0 706.7 1401.7</td>
<td></td>
</tr>
<tr>
<td>Shared Hare-Hunters</td>
<td>65.0 155.0 457.5 923.8 65.0 105.0 491.4 962.9</td>
<td></td>
</tr>
<tr>
<td>MACE</td>
<td>240.0 580.0 662.7 1381.8 200.0 393.2 1260.0 2344.0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>10-vs-10</th>
<th>20-vs-20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloned Stag-Hunters</td>
<td>1229.2 2482.9 2079.4 4171.2 3224.5 6219.9 5934.5 11429.2</td>
<td></td>
</tr>
<tr>
<td>Shared Stag-Hunters</td>
<td>1214.5 2423.7 2005.7 4144.3 3150.7 6379.8 6344.4 12196.8</td>
<td></td>
</tr>
<tr>
<td>MACE</td>
<td>1515.2 3047.3 2275.7 4610.7 3799.3 7158.1 6880.7 13358.6</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Total terminal reward (averaged over 5 best runs) for $N$ agents for set # of training samples.

Figure 5: Predators (red) and prey (green) during training for 2v2 Hare-Hunters for 100 episodes. Arrows show where agents move to in the next frame. Top: at the start, predators explore via $\lambda$, but do not succeed before the time limit $T$ (red dot). Bottom: after convergence agents succeed consistently (green dot) before the time limit (purple dot) and $\lambda$ encodes the two strategies from Figure 1. Highlighted $\lambda$-components correlate with rollout under a 2-sided $t$-test at $\alpha = 0.1$ significance.

does indeed use the latent $\lambda$: it (partly) controls the coordination between agents. Second, the latent $\lambda$ shows strong correlation during all phases of training. This suggests that the model indeed is performing a form of structured exploration. Third, the components of $\lambda$ are correlated with semantic meaningful behavior. We show a salient example in the bottom 2 rows in Figure 5: the correlated components of $\lambda$ are disjoint and each component correlates with both agents. The executed policies are exactly the two equivalent ways to assign 2 hunters to 2 preys, as illustrated in Figure 1.

Coordination with a large $N$. In the large $N = M = 10$ case, the dynamics of the agent collective are a generalization of the $N = M = 2$ case. There are now redundancies in multi-agent hunter-prey assignments that are analogous to the $N = M = 2$ case that are prohibitively complex to analyze due to combinatorial complexity. However our experiments strongly suggest (see e.g. Figure 6) the latent code is again correlated with the agents’ behavior during all phases of training, showing that $\lambda$ induces meaningful multi-agent coordination.

6 Related Work

Deep Structured Inference. Recent works have focused on learning structured representations using expressive distributions, which enable more powerful probabilistic inference. For instance, Johnson et al. (2016) has proposed combining neural networks with graphical models, while

2This is parallel to the discussion in Chen et al. (2016), which investigates the effectiveness of latent codes $\lambda$. 6 Related Work

Deep Structured Inference. Recent works have focused on learning structured representations using expressive distributions, which enable more powerful probabilistic inference. For instance, Johnson et al. (2016) has proposed combining neural networks with graphical models, while
Figure 6: Visualization of MACE for a (10, 10, 1, 1000) Hare-Hunters game in a 30 × 30 world. Left: components of the latent code λ that significantly correlate with sampled actions (computed as in Figure 5). Right: Three episode snapshots: at the start, middle and end. Red: predators; green: prey. Arrows indicate where agents move to in the next snapshot. The hunters solve the game (green dot) before the time limit T = 1000 (purple dot), by distributing targets amongst themselves.

[Ranganath et al. (2015)] learn hierarchical latent distributions. Our work builds upon these approaches to learn structured policies in the reinforcement learning setting.

Variational methods in RL. [Neumann (2011); Furmston & Barber (2010)] discuss variational approaches for RL problems, but did not consider end-to-end trainable models. [Levine & Koltun (2013)] used variational methods for guided policy search. [Houthooft et al. (2016)] learned exploration policies via information gain using variational methods. However, these only consider 1 agent.

Coordination in RL. Multi-agent coordination has been studied in the RL community (e.g. [Kapetanakis & Kudenko (2002); Chalkiadakis & Boutilier (2003)]) as a method to reduce the instability of training multiple agents simultaneously using RL. The benefit of coordination was demonstrated in simple multi-agent settings in e.g. [Tan (1993)]. However, previous methods learned hand-crafted communication models and do not scale well to complex state spaces and many agents. In contrast, our method learns coordination end-to-end via on-policy methods, learns the multi-agent exploration policy and scales well to many agents via its simple hierarchical structure.

Communication Models. Recently, end-to-end learning of communication models has been studied for shared broadcast channels (Sukhbaatar et al. (2016)), sequential communication (Peng et al. (2017)), heuristic multi-agent exploration (Usunier et al. (2016)) and bit-channels (Foerster et al. (2016)). These works show that non-trivial communication protocols can be learned through back-propagation or heuristic stabilization methods, but often do not scale well to a large number of agents. Our hierarchical approach is complementary, learns via variational methods, and can scale to large N.

Multi-task learning. Hierarchical models have been studied for multi-task learning, e.g. [Daume III (2014)] learns latent hierarchies via EM in a supervised learning setting. Instead, we study flexible end-to-end trainable latent hierarchies in the reinforcement learning setting.

7 Discussion

In a sense, we studied the simplest setting that can benefit from structured exploration, in order to isolate the contribution of our work. Our hierarchical model and variational approach are a simple way to implement multi-agent coordination, and easily combine with existing actor-critic methods. Moving forward, there are many ways to expand on our work. Firstly, for complex (partial-information) environments, instead of using reactive policies with simple priors \( P \sim \mathcal{N}(0, 1) \), memory full policies with flexible priors (Chen et al. (2016)) may be needed. Secondly, our approach is complementary to richer forms of communication between agents. Our hierarchical structure can be interpreted as a broadcast channel, where agents are passive receivers of the message \( \lambda \). Richer communication protocols could be encoded by policies with more complex inter-agent structure. It would be interesting to investigate how to learn these richer structures. Taken together, these could enable using our method for efficient multi-agent RL in more complex coordination games.
REFERENCES


8 Supplementary Material

8.1 Hierarchical Variational Lower Bound.

We next derive a tractable learning algorithm using variational methods. To apply variational methods, we first cast (1) as a probabilistic inference problem, as in Levine & Koltun (2013); Vlassis et al. (2009). To do so, we assume that the total reward $R^t$ for each $i$ to be non-negative and bounded. Hence, we can view the total reward $R^t(\tau)$ as a random variable, whose unnormalized distribution is defined as $P(R^t(\tau)) = R^t$. We can then rewrite (1) as a maximum likelihood problem (omitting $i$ for clarity):

\[
\max_{\theta} \mathbb{E}_{\pi(s_t, a_t)}[R^t] = \max_{\theta} \int d\tau R^t(\tau) P(\tau; \theta) \Rightarrow \max_{\theta} \int d\tau P(R^t|\tau) P(\tau; \theta), \tag{16}
\]

\[
P(\tau; \theta) = P(s_0) \prod_{t=0}^T d\lambda_t P(s_{t+1}|s_t, a_t) P(a_t, \lambda_t|s_t; \theta). \tag{17}
\]

Denoting time sequences as $\lambda = (\lambda_t)_{0 \leq t < T}$ for clarity, we can use a variational posterior $Q_R(\lambda|s; \phi)$ and Jensen’s inequality (Hoffman et al., 2013) to get a lower bound on the log-likelihood of (16):

\[
\log P(R^t; \theta) = L(Q_R, \theta, \phi) + \kappa L(Q_R(\lambda|s; \phi)||P(\lambda|s)) \geq L(Q_R, \theta, \phi)
\]

\[
L(Q_R, \theta, \phi) = \int d\tau d\lambda Q_R(\lambda|s; \phi) \log \left( \frac{P(R^t|\tau) P(s_0) \prod_{t=0}^T P(s_{t+1}|s_t, a_t) P(a_t, \lambda_t|s_t; \theta)}{Q_R(\lambda|s; \phi)} \right), \tag{19}
\]

where we used $\int d\tau d\lambda = \prod_{t=0}^T \int ds_t da_t d\lambda_t$. The first term $L(Q_R, \theta, \phi)$ is called the evidence lower bound (ELBO) which we can maximize as a proxy for (16). From the form of the ELBO, it is clear that the optimal $Q_R$ factorizes similarly to $P^t$:

\[
Q_R(\lambda|s; \phi) = P(R^t|\tau) P(s_0) \prod_{t=0}^T P(s_{t+1}|s_t, a_t) Q(\lambda_t|s_t; \phi). \tag{20}
\]

Substituting this:

\[
L(Q_R, \theta, \phi) = \int d\tau d\lambda Q_R(\lambda|s; \phi) \log \left( \frac{P(R^t|\tau) P(s_0) \prod_{t=0}^T P(s_{t+1}|s_t, a_t) P(a_t, \lambda_t|s_t; \theta)}{P(R^t|\tau) P(s_0) \prod_{t=0}^T P(s_{t+1}|s_t, a_t) Q(\lambda_t|s_t; \phi)} \right)
\]

\[
= \int d\tau d\lambda Q_R(\lambda|s; \phi) \log \left( \prod_{t=0}^T \frac{P(a_t, \lambda_t|s_t; \theta)}{Q(\lambda_t|s_t; \phi)} \right)
\]

We can rewrite this as:

\[
L(Q_R, \theta, \phi) = \int d\tau d\lambda Q_R(\lambda|s; \phi) \sum_{t=0}^T \left( \log P(a_t|\lambda_t, s_t; \theta) + \log \frac{P(\lambda_t|s_t)}{Q(\lambda_t|s_t; \phi)} \right) \tag{21}
\]

\[
P(a_t, \lambda_t|s_t; \theta) = P(a_t|\lambda_t, s_t; \theta) P(\lambda_t|s_t). \tag{22}
\]

The standard choice is to set the priors to the maximum-entropy standard-normal: $P(\lambda_t|s_t) = \mathcal{N}(0, 1)$. With this choice, we can optimize (22) using standard optimization methods, e.g. stochastic gradient ascent. Formally, the gradient $\nabla_\theta$ of (22) is:

\[
\nabla_\theta L(Q_R, \theta, \phi) = \prod_{t=0}^T \int d\lambda_t Q_R(\lambda_t|s_t; \phi) \sum_{t'=0}^T \nabla_\theta \log P(a_t|\lambda_t, s_t; \theta), \tag{23}
\]

with $\nabla_\phi L(Q_R, \theta, \phi)$ calculated similarly. These gradients can be estimated using sampled roll-outs $\tau^k$ of the policy $P^\pi$. During a rollout $\tau^k$, we sample $\lambda \sim Q$, observe rewards $R \sim P(\tau^k)$ and transitions $s_{t+1} \sim P(s_{t+1}|s_t)$; the sample estimates (23) approximates the policy gradient (2).

\[
g_\theta \approx g_{\theta, Q} = \frac{1}{M} \sum_{k=1}^M \sum_{t=0}^T \nabla_\theta \log P(a_t^k|\lambda_t^k, s_t^k; \theta) R(\tau^k), \quad g_\phi \approx g_{\phi, Q} = \nabla_\phi L(Q_R, \theta, \phi). \tag{24}
\]

The second term in (24) provides the gradient for the variational posterior $Q_R$, which learns to sample the optimal $\lambda_t$. 
8.2 Training

**Training method.** We used the A3C method with 5-20 threads for all our experiments. Each thread performed SGD with \( \frac{24}{25} \). The loss for the value function at each state \( s_t \) is the standard \( L_2 \)-loss between the observed total rewards for each agent \( i \) and its value estimate:

\[
\alpha \sum_t \sum_i \left( V_i(s_t) - R^i(s_t, a_t) \right)^2.
\] (25)

In addition, in line with other work using actor-critic methods, we found that adding a small entropy regularization on the policy can sometimes positively influence performance, but this does not seem to be always required for our testbeds. The entropy regularization is:

\[
H(P) = -\beta \sum_t \sum_a P(a_t | s_t; \theta) \log P(a_t | s_t; \theta).
\] (26)

A3C additionally defines training minibatches in terms of a fixed number of environment steps \( L \): a smaller \( L \) gives faster training with higher variance and a higher \( L \) vice versa.

**Hyperparameter search.** For experiments, we performed a random search over hyperparameters:

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Search range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning-rate ( \eta )</td>
<td>([10^{-6}, 10^{-1}])</td>
</tr>
<tr>
<td>Discount ( \gamma )</td>
<td>([0.9, 0.99])</td>
</tr>
<tr>
<td>GAE discount ( \tau )</td>
<td>([0.9, 0.99])</td>
</tr>
<tr>
<td>Value loss ( \alpha )</td>
<td>([0.1, 0.5])</td>
</tr>
<tr>
<td>Policy entropy ( \beta )</td>
<td>([0, 0.01])</td>
</tr>
</tbody>
</table>

Table 2: Search range of hyper-parameters.

8.3 Results

**Figure 7:** Train-time cumulative terminal reward in \((N, M, 1, 1000)\) Hare-Hunters (upper) and Stag-Hunters (lower) on a 50 × 50 gridworld, for 10-vs-10 or 20-vs-20 agents; randomly moving or fixed preys. Average, minimal and maximal rewards for the best 5 runs for each model are shown. MACE accumulates increasingly higher rewards compared to the baselines.