# Implicit Ensemble Training for Efficient and Robust Multiagent Reinforcement Learning

**Macheng Shen** [1] **Jonathan P. How** [2]

## Abstract

An important issue in competitive multiagent scenarios is the distribution mismatch between training and testing caused by variations in other agents' policies. As a result, policies optimized during training are typically sub-optimal (possibly very poor) in testing. Ensemble training is an effective approach for learning robust policies that avoid significant performance degradation when competing against previously unseen opponents, but this approach can lead to significantly increased computation. This paper proposes a novel implicit ensemble training (IET) approach that combines deep latent variable learning and multi-task reinforcement learning to address this computational issue while improving the robustness of ensemble training. We present the IET approach and apply it to three competitive board games to show that it learns strong policies that are more robust to unseen adversarial opponent policies.

## 1. Introduction

In competitive multiagent scenarios, the learned policy of each agent depends on the joint policy of all the other learning agents. However, since all agents are learning concurrently, this leads to a non-stationary environment (Lowe et al., 2017). One challenge resulting from such multiagent learning is that the policy learned from training might not perform well in the testing environment where the opponents' policies could be significantly different from those of the training opponents (distribution shift between training and testing). Even worse, the testing opponents could be trained to exploit the weakness of the policy learned from the training (Gleave et al., 2019).

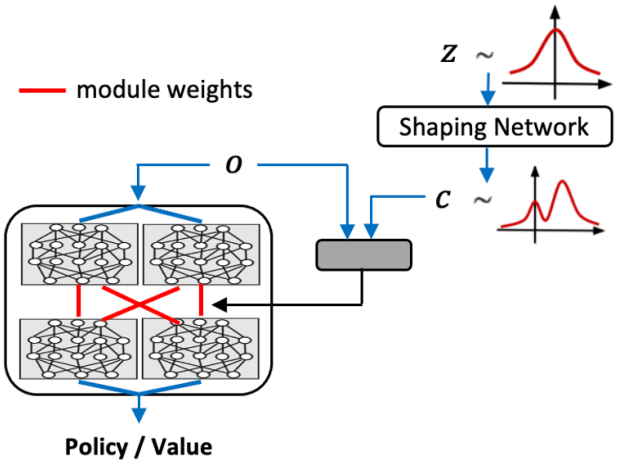



*Figure 1.* Architecture of our IET network, consisting of a shaping network for learning the distribution of the latent condition variable, and a modular conditional policy network for learning the conditional policy/value.

One effective approach to mitigate the performance degradation from training to testing is ensemble training, which has been applied in many previous works (Bansal et al., 2017; Lowe et al., 2017; Silver et al., 2016; Jaderberg et al., 2019; Vinyals et al., 2019). In ensemble training, each agent has multiple policies (as in (Lowe et al., 2017; Jaderberg et al., 2019; Vinyals et al., 2019)) or keeps multiple copies of previous policies (as in (Silver et al., 2016; Bansal et al., 2017)), from which one policy for each agent is sampled to play against each other. As a result, each policy is optimized against many different combinations of the other agents' policies, which effectively reduces the distribution shift between training and testing. However, one drawback of applying this ensemble training technique is the significant increase in computation and memory consumption due to the learning and storage of multiple policies.

In this paper, we propose a novel implicit ensemble training (IET) approach[1] by formulating ensemble training as a multitask learning problem. Instead of maintaining multiple policies explicitly with independent neural networks

[1]Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA [2]Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA. Correspondence to: Macheng Shen <macshen@mit.edu>.

*Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning.*, Copyright 2021 by the author(s).

[1]github.com/MachengShen/Implicit_ensemble_training

(NN), our IET approach uses a unified hierarchical modular network architecture (Yang et al., 2020) with a learnable conditional latent variable distribution. The unified network architecture increases the knowledge sharing within the modules for improved learning efficiency, and the conditional latent variable captures the diversity of the policy, which is necessary for robustness.

Our contributions are: 1) we identify the cause of inefficiency in standard ensemble training and suggest a multitask solution to improving the efficiency of ensemble training; 2) we propose a novel algorithm that extends ensemble training with latent variable and multitask network; and 3) we show that the new algorithm improves both learning efficiency and robustness.

## 2. Preliminaries

### 2.1. Markov games

We use Markov Games as the basis for our decision-making framework. A Markov game for $N$ agents is defined by a set of states $\mathcal{S}$ describing the possible configurations of all agents, a set of actions $\mathcal{A}_1, \ldots, \mathcal{A}_N$, and a set of observations $\mathcal{O}_1, \ldots, \mathcal{O}_N$ for each agent. Each agent has a stochastic policy $\pi_i : \mathcal{O}_i \times \mathcal{A}_i \mapsto [0, 1]$, and a reward function $r_i : \mathcal{S} \times \mathcal{A}_i \mapsto \mathbb{R}$. The objective of each agent is to maximize its own cumulative reward $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$ with discount factor $\gamma$ and time horizon $T$ (Lowe et al., 2017).

### 2.2. Ensemble training

Ensemble training is a standard approach for improving the robustness of policies in multiagent reinforcement learning (MARL). In ensemble training, each agent's policy $\pi_i$ is an ensemble of $K$ sub-policies, with each denoted as $\pi_{\theta_i^{(k)}}$ or $\pi_i^{(k)}$ and parameterized by separate NN parameters $\theta_i^{(k)}$. For each agent, the RL objective is to maximize the ensemble weighted objective $\mathbb{E}_{k\sim\text{unif}(1,K), s\sim p^\pi, a\sim\pi_i^{(k)}}[r_i(s, a)]$.

### 2.3. Generalization of ensemble training as latent-conditioned policy

The generative process for the ensemble training policy $\pi_i$ is

$$\begin{aligned} k &\sim \text{unif(1, K)} = \frac{1}{K}\sum_{j=1}^{K}\delta(k-j), \\ \theta_i|k &\sim \delta(\theta_i - \theta_i^{(k)}), \\ a_i|\theta_i &\sim f_{\theta_i}(o_i), \end{aligned} \tag{1}$$

where $\delta$ is the Dirac measure. One interpretation of Eq. 1 is that $\pi_i$ is a conditional policy on a uniform discrete latent random variable $k$,

$$a_i|k \sim \bar{f}(o_i, \theta_i^{(1)}, \theta_i^{(2)}, \ldots, \theta_i^{(K)}; k) = f_{\theta_i^{(k)}}(o_i), \tag{2}$$

Eq. 2 suggests that ensemble training is an inefficient over-parameterized form of a latent-conditioned policy (Haarnoja et al., 2018; Petangoda et al., 2019; Lee et al., 2019) because only one out of the $K$ sets of sub-policy parameters $[\theta_i^{(1)}, \theta_i^{(2)}, \ldots, \theta_i^{(K)}]$ is activated per sampling of $a_i$. As a result, the rollout from executing one set of sub-policy parameter cannot be used to optimize the rest $K-1$ sets of sub-policy parameters, which undesirably reduces the sample efficiency by a factor of $K$.

## 3. Approach

### 3.1. Implicit ensemble training

Our IET generalizes ensemble training via two steps: 1) relax the discrete random variable $k \in \{1, 2, \ldots, K\}$ into a continuous latent variable $c \in \mathbb{R}^L$ with a learned distribution that adaptively captures the diversity of the policy ensemble; 2) replace the $K$ independent NNs with a unified modular network architecture with parameter $\phi$ that improves sample-efficiency by knowledge sharing between sub-modules within the network. The continuous relaxation also makes the policy differentiable with respect to the latent variable, thus making it possible to synthesize new policies from the learned skills represented by the sub-modules within the modular network by perturbing the latent variable distribution.

The generative process of this implicit ensemble is

$$\begin{aligned} z &\sim \mathcal{N}(\mathbf{0}, \mathbb{I}_{L\times L}), \\ c|z &= g_\psi(z), \\ a_i|c &\sim h_\phi(o_i; c), \end{aligned} \tag{3}$$

In Eq. 3, a random Gaussian noise vector $z$ is sampled from the standard multivariate Gaussian distribution (sampled once at the beginning of each episode and remains fixed during the episode), then passed through a shaping network parameterized by $\psi$ to output a latent condition variable $c$. Finally, the action is sampled from a policy which is parameterized by $\phi$ and conditioned on the latent condition variable $c$. Both $\psi$ and $\phi$ are learnable parameters that are optimized end-to-end with respect to the reinforcement learning objective.

The implicit ensemble Eq. 3 is a more flexible parameterization than the ensemble training Eq. 1. The latter is a special case of the former, where the distribution of the latent variable $c$ collapses into a discrete distribution (corresponding to the uniform discrete distribution of $k$), and the shared parameter $\phi$ is divided into $K$ disjoint sets of parameters each of which is activated in the policy network when only one of the discrete values of $c$ is drawn.

In contrast to the ensemble training formulation Eq.1, where the ensemble size $K$ is a hyperparameter to control the di-

versity of the (ensemble) policy, Eq. 3 does not contain this explicit hyperparameter. Instead, the diversity of the policy is captured adaptively by the learned latent distribution parameterized by $\psi$: a complicated multi-modal distribution corresponds to high diversity, while a simple uni-modal distribution corresponds to low diversity.

### 3.2. Model architecture

Fig. 1 shows the model architecture for implementing the IET. There are two major components within this architecture: 1) a shaping network, corresponding to the mapping $g_\psi(\cdot)$ in Eq. 3 that transforms the Gaussian noise vector $z$ into the latent condition variable $c$; and 2) a modular conditional policy network, corresponding to $h_\phi$ that parameterizes the conditional policy.

The shaping network is responsible for adding diversity to the conditional policy for improving the robustness of the learned policy, and the modular conditional policy network improves sample-efficiency via knowledge sharing between sub-modules. We include the detailed design of these two networks in Appendix A.

## 4. Evaluation

### 4.1. Scenarios

We evaluate our approach on three 2-player (which we call the blue agent and the red agent hereafter) turn-based board-games implemented in the PettingZoo multiagent environment (Terry et al., 2020) and the RLCard toolkit (Zha et al., 2019): **Connect Four**, **Leduc Hold'em**, and **Texas Hold'em (Limit)**. The detail about these scenarios is included in Appendix B. We chose board games for evaluation because we found that the distribution mismatching between training and testing is most prominent on these games, which we speculate is because of the wide range of available tactics in this type of games.

### 4.2. Baselines

We compare our approach with three baselines:

1. **Single Policy Training** (SPT): This is the standard multiagent training approach, where each agent optimizes only one policy.
2. **Simple Ensemble Training** (SET): This is the standard ensemble training approach, where each agent optimizes an ensemble of policies against each other. We use an ensemble size of 3 for each agent, which increases the amount of computation by a factor of 3.
3. **Fixed Latent Variable Training** (FLVT): This is the ablated version of our implicit ensemble approach, where the impact of the shaping network is removed by fixing the latent condition variable to a constant vector.

We include the detailed implementation of these training settings in Appendix C.

For evaluating the robustness of the learned policies, we adopted a similar approach as in (Vinyals et al., 2019; Gleave et al., 2019) by training an independent exploiter agent. Specifically, we launched two concurrent threads, one for the training, the other for the testing, and repeated the following steps:

1. Train the blue agent and the red agent in the training thread for one training epoch.
2. Copy the blue agent's policy to the testing thread and freeze it.
3. Train the red exploiter agent in the testing thread against the fixed blue agent.

As a result, the red exploiter agent learns to exploit the weakness of the blue policy, and the corresponding reward is an informative indicator of the adversarial robustness of the blue policy.

### 4.3. Results and Comparisons

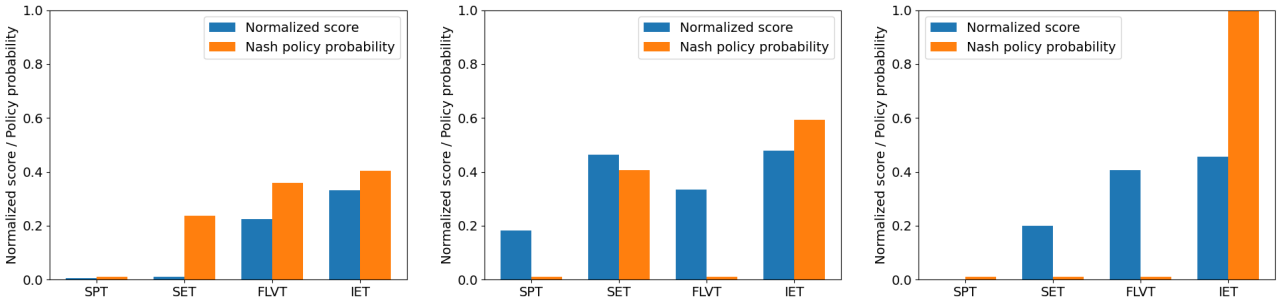
This section discusses experimental results and compares baseline approaches with our proposed IET approach.

We include rewards of the blue agent (in both training and testing), the red agent (only in training), and the red exploiter (only in testing) in Appendix D (Fig. 3, 4, and 5). The results show that our approach effectively reduces the performance degradation between the training and the testing, which indicates that our approach has better adversarial robustness.

To evaluate the out-of-distribution robustness, we also include in Appendix E the competition scores between the blue agent and the red exploiter agent for each pair of training settings (Tables 1).

However, drawing conclusions directly from Appendix E about the relative strength of the policies learned by the four training settings is difficult, because the competition scores show that there is no single policy that can dominate all the rest policies. This non-transitive nature of real-world games has also been observed in previous work (Czarnecki et al., 2020).

To quantitatively evaluate the robustness of the learned blue policy, we provide two metrics: (1) Normalized score: we find the worst-case reward gap between blue and red $\min_{\pi_r \in \Pi}[r_{\text{b}}(\pi_b, \pi_r) - r_{\text{red}}(\pi_b, \pi_r)]$, $\Pi = [\text{SPT}, \text{SET}, \text{FLVT}, \text{IET}]$ (higher is better) and normalize it to $[0, 1]$, (2) Nash policy probability: we follow the approach proposed in (Balduzzi et al., 2018) by solving for the Nash-Equilibrium (NE) (Holt & Roth, 2004) of the meta-game (Tuyls et al., 2020). There are a row-player and a column-player in the meta-game, whose actions are selecting which row/column policy to execute from the four




(a) Connect Four (b) Leduc Hold'em (c) Texas Hold'em

*Figure 2.* Normalized score measures the adversarial robustness, and Nash policy probability of the meta-player measures the overall strength of the policy, our approach (IET) consistently outperforms the other approaches.

available policies. The NE of the meta-game is a pair of stochastic policies. The probability of a policy being selected by the meta-player measures the strength (robustness) of this policy.

We show these two metrics in Fig. 2. This result shows that the blue policy learned by our IET approach is the most preferred one to a NE meta-player, which suggests that this policy is the most robust one.

### 4.4. Visualization of latent variable distribution

We compare the 2D t-distributed stochastic neighbor embedding (t-SNE) of the latent condition variable in a single-agent scenario with that in multiagent scenarios in Appendix F (Fig. 6).

The t-SNE plots verify our conjecture that IET is an adaptive way of introducing diversity during training. In the single-agent scenario, there is only one optimal policy, and no diversity is needed. As a result, the shaping network maps the Gaussian noise vector into a uni-modal distribution that minimizes the variation in the conditional policy. In contrast, in the multiagent scenarios, diversified policies in training are a necessity for learning robust policies. As a result, the shaping network learns a non-trivial latent variable distribution to facilitate this diversity requirement.

## 5. Related works

This work is at the intersection of robust MARL and efficient MARL. Ensemble training for robust MARL has been studied in many previous works such as (Lowe et al., 2017; Bansal et al., 2017; Jaderberg et al., 2017; 2019; Vinyals et al., 2019). These approaches focus on improving the robustness of the learned policies, regardless of the associated increase of computational complexity. In contrast, our work focuses on improving the efficiency of ensemble training without sacrificing robustness. Another approach for robust MARL is minimax policy optimization, in which each agent optimizes its policy assuming all the other agents and the environment dynamics are adversarial (see (Zhang et al., 2020; Li et al., 2019)). One difficulty with this approach is the requirement of solving the nested minimax optimization, which is typically approximated by optimizing the inner minimization for one step only (Li et al., 2019). In addition, the minimax formulation tends to result in overly-conservative policies because of the pessimistic assumption about the other agents.

Our work is also closely related to multi-task reinforcement learning (MTRL). MTRL aims to improve the learning efficiency by knowledge sharing across tasks (D'Eramo et al., 2019). Recent works (Huang et al., 2020; Yang et al., 2020; Devin et al., 2017) found that the modular policy network is an effective architecture for improving sample-efficiency via sharing of learned modules. However, that MTRL work focuses on solving the single-agent multi-task problem. In contrast, our work leverages the modular network architecture combined with latent conditional policy learning (Haarnoja et al., 2018; Lee et al., 2019) to improve the efficiency of ensemble-based robust MARL. The innovation of our approach is re-formulating ensemble training as multi-task learning while using a latent variable to preserve policy diversity which is essential for robust MARL.

## 6. Conclusions

This paper proposed an implicit ensemble training approach that effectively reduces the computational complexity of ensemble training while maintaining the policy diversity for learning robust multiagent policies. Numerical results show that, in addition to higher learning efficiency, our approach also improves robustness of the learned policy. For future works, we would like to investigate the extension of our approach to cooperative scenarios and adversarial scenarios where each team has more than one agent.

## References


Balduzzi, D., Tuyls, K., Pérolat, J., and Graepel, T. Re-evaluating evaluation. *CoRR*, abs/1806.02643, 2018. URL http://arxiv.org/abs/1806.02643.

Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., and Mordatch, I. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

Czarnecki, W. M., Gidel, G., Tracey, B., Tuyls, K., Omidshafiei, S., Balduzzi, D., and Jaderberg, M. Real world games look like spinning tops. *arXiv preprint arXiv:2004.09468*, 2020.

D'Eramo, C., Tateo, D., Bonarini, A., Restelli, M., and Peters, J. Sharing knowledge in multi-task deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

Devin, C., Gupta, A., Darrell, T., Abbeel, P., and Levine, S. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2169–2176. IEEE, 2017.

Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., and Russell, S. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

Haarnoja, T., Hartikainen, K., Abbeel, P., and Levine, S. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 1851–1860. PMLR, 2018.

Holt, C. A. and Roth, A. E. The nash equilibrium: A perspective. *Proceedings of the National Academy of Sciences*, 101(12):3999–4002, 2004.

Huang, W., Mordatch, I., and Pathak, D. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464. PMLR, 2020.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Jaderberg, M., Czarnecki, W., Dunning, I., Marris, L., Lever, G., Castaneda, A., Beattie, C., Rabinowitz, N., Morcos, A., Ruderman, A., et al. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

Li, S., Wu, Y., Cui, X., Dong, H., Fang, F., and Russell, S. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4213–4220, 2019.

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J., Jordan, M., and Stoica, I. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pp. 3053–3062. PMLR, 2018.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

Petangoda, J. C., Pascual-Diaz, S., Adam, V., Vrancx, P., and Grau-Moya, J. Disentangled skill embeddings for reinforcement learning. *arXiv preprint arXiv:1906.09223*, 2019.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Terry, J. K., Black, B., Hari, A., Santos, L., Dieffendahl, C., Williams, N. L., Lokesh, Y., Horsch, C., and Ravi, P. Pettingzoo: Gym for multi-agent reinforcement learning. *arXiv preprint arXiv:2009.14471*, 2020.

Tuyls, K., Perolat, J., Lanctot, M., Hughes, E., Everett, R., Leibo, J. Z., Szepesvári, C., and Graepel, T. Bounds and dynamics for empirical game theoretic analysis. *Autonomous Agents and Multi-Agent Systems*, 34(1):1–30, 2020.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Yang, R., Xu, H., Wu, Y., and Wang, X. Multi-task reinforcement learning with soft modularization. *arXiv preprint arXiv:2003.13661*, 2020.

Zha, D., Lai, K.-H., Cao, Y., Huang, S., Wei, R., Guo, J., and Hu, X. Rlcard: A toolkit for reinforcement learning in card games. *arXiv preprint arXiv:1910.04376*, 2019.

Zhang, K., Sun, T., Tao, Y., Genc, S., Mallya, S., and Basar, T. Robust multi-agent reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 33, 2020.

# A. Network design

## A.1. Shaping network

The shaping network takes a Gaussian noise vector $z \in \mathbb{R}^L$ as input and transforms $z$ into a latent condition variable $c$, which modifies the standard multi-variate Gaussian distribution into a learned (complicated) distribution. We use a feed-forward network with 2 fully-connected layers followed by a Softmax layer to parameterize this shaping network.

## A.2. Modular network for multi-tasking

Recent studies (Yang et al., 2020; Devin et al., 2017; Huang et al., 2020) found that modular architecture is a sample-efficient way of learning multi-tasking policies. In ensemble training, the multi-tasking requirement naturally arises from the fact that each policy needs to be optimized against the ensemble of policies of the other agents.

To improve the efficiency of ensemble training, we use the modular architecture proposed in (Yang et al., 2020) as our conditional policy network for multi-tasking. The modular network is composed of a base network and a routing network. The base network is a modular network that has $n$ layers, and each layer has $m$ modules. Each module is a feed-forward network, and the input and the output are d-dimension vectors. The routing network takes the observation and the latent condition variable as inputs and outputs $n$ normalized weight vectors for weighting the modules of the base policy network. The weight vectors $w^j \in \mathbb{R}^{m^2}$ are calculated as

$$\begin{aligned} p^{j=1} &= W_d^{j=1}\left(\sigma\left(F\left(o\right) \cdot H\left(c\right)\right)\right), \\ p^{j+1} &= W_d^j\left(\sigma\left(W_u^j p^j \cdot \left(F\left(o\right) \cdot H\left(c\right)\right)\right)\right), \\ w^j &= \text{Softmax}(p^j), \end{aligned} \tag{4}$$

where $F$ and $H$ are the embedding layers, which maps the observation vector and the latent condition variable into $D$-dimension embeddings. $\sigma$ is the activation function for which we use the ReLU activation. $W_u$ and $W_d$ are fully-connected layers of size $\mathbb{R}^{D \times m^2}$ and $\mathbb{R}^{m^2 \times D}$, respectively.

The base network takes the observation vector as input and outputs policy logits / value, with the following relationship between the $i$-th module in the $j+1$ layer's input and the $k$-th module in the $j$-th layer's output,

$$\hat{f}_i^{j+1} = \sum_{l=1}^{m} w_{i,l}^j \left(\sigma\left(W_l^j \hat{f}_l^j\right)\right), \tag{5}$$

where $W_l^j \in \mathbb{R}^{d \times d}$ is the learnable module parameters.

# B. Evaluation scenarios

1. **Connect Four**: Players must connect four of their tokens vertically, horizontally, or diagonally on a board with 6 rows and 7 columns. The players drop their respective tokens in a column of a standing grid, where each token will fall until it reaches the bottom of the column or reaches an existing token. Players cannot place a token in a full column, and the game ends when either a player has made a sequence of 4 tokens, or when all 7 columns have been filled. This game has an observation space of size (6, 7, 2) and 7 actions.
2. **Texas Hold'em (Limit)**: A poker game with a 52 cards deck. In the beginning, both players get two cards. After betting, three community cards are shown and another round follows. At any time, a player could fold and the game will end. The winner will receive +1 as a reward and the loser will get -1. This game has an observation space of size 72 and 4 actions.
3. **Leduc Hold'em**: A poker game with 2 rounds and a deck of six cards (Jack, Queen, and King in 2 suits). At the beginning of the game, each player receives one card and, after betting, one public card is revealed. Another round follows. In the end, the player with the best hand wins and receives a reward +1 and the loser receives -1. At any time, any player can fold. This game has an observation space of size 36 and 4 actions.

# C. Implementation Detail

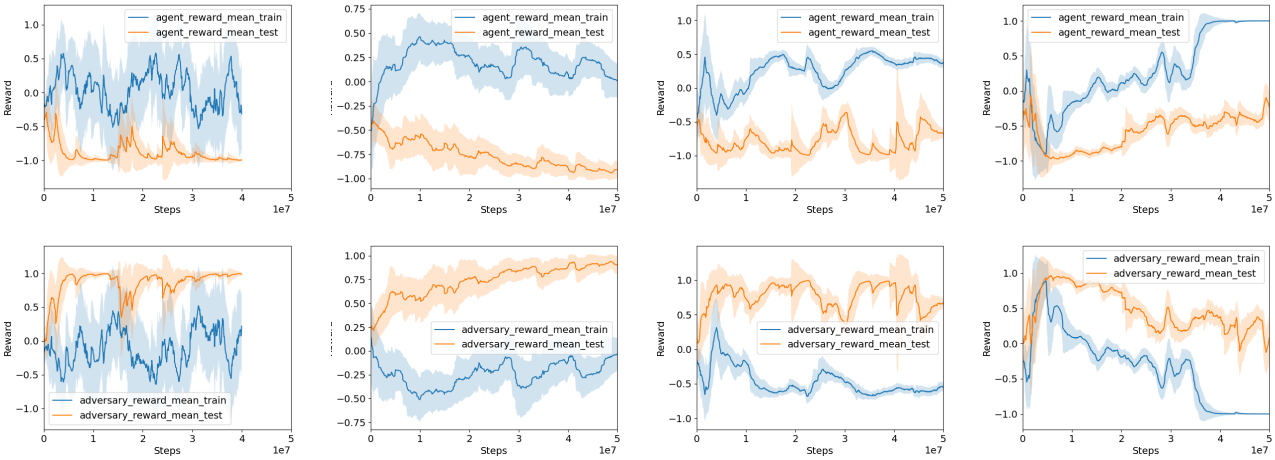
We used the RLlib (Liang et al., 2018) implementation of Proximal Policy Optimization (PPO) with a mini-batch size of 256, and a learning rate of $5 \times 10^{-5}$. We use independent networks for the policy and the value function approximations, and the following hyperparameters: $L = 10$ for the latent condition variable dimension; $H = 64$ for the hidden layer dimension in the shaping network; $n = 2$ and $m = 2$ for the number of layers and number of modules of the modular network; $D = 64$ and $d = 64$ for the embedding and module hidden dimension.

In the simple ensemble training setting, each sub-policy within an ensemble only has a probability of $1/3$ to be selected for execution. If the same number of environment rollouts is used to train the simple ensemble as used for the other training settings, the simple ensemble policy will perform poorly due to insufficient training. Therefore, we roll out two additional environment simulations but counted as one training step when training the simple ensemble, for a fair comparison, at the cost of additional computational overhead.

# D. Results

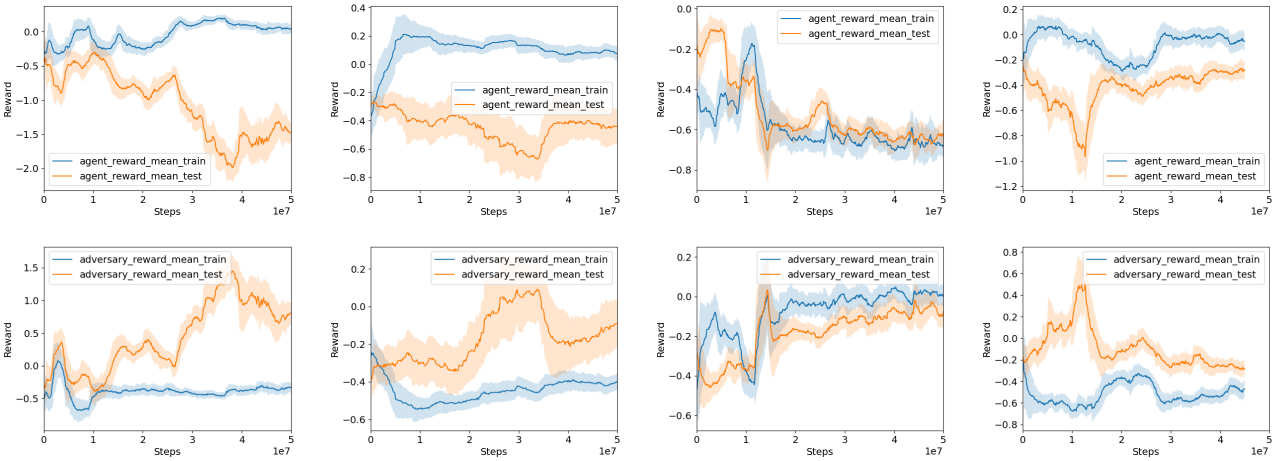
### D.0.1. Adversarial robustness

We investigate the adversarial robustness of the learned blue policy by training a red exploiter agent against the frozen



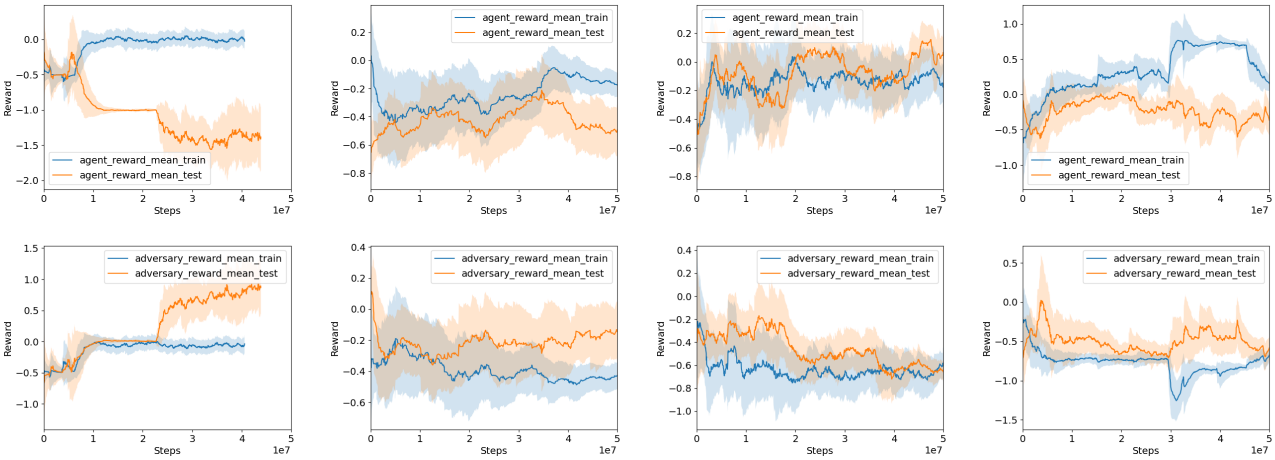

(a) Single Policy Training (b) Simple Ensemble Training (c) Fixed Latent Variable Training (d) Implicit Ensemble Training

*Figure 3.* Training and testing reward of the blue (agent) and the red (adversary) in the board game **Connect Four**




(a) Single Policy Training (b) Simple Ensemble Training (c) Fixed Latent Variable Training (d) Implicit Ensemble Training

*Figure 4.* Training and testing reward of the blue (agent) and the red (adversary) in the board game **Leduc Hold'em**




(a) Single Policy Training (b) Simple Ensemble Training (c) Fixed Latent Variable Training (d) Implicit Ensemble Training

*Figure 5.* Training and testing reward of the blue (agent) and the red (adversary) in the board game **Texas Hold'em**

blue agent policy.

Fig. 3, 4, and 5 show the training and testing rewards of both agents corresponding to the three baseline approaches and our proposed IET approach. Since the red exploiter agent is trained against the fixed blue policy during the testing, these plots reflect the adversarial robustness (assuming the red exploiter agent can successfully learn the optimal exploiting policy against the blue policy, which is not always true, as will be discussed in later sections) of these four training settings.

We can see that there is always a gap between the training reward and the testing reward. The training blue reward is always higher than the corresponding testing blue reward, while the testing red reward is always higher than the training red reward, which manifests the robustness issue of policies learned through MARL. Moreover, sometimes the training reward and the testing reward are not positively correlated, for example, in Fig. 4a and 5a, which indicates that the single policy training approach suffers the most from the adversarial exploitation. The simple ensemble training mitigates this robustness issue, reflected by the smaller gaps, higher testing blue reward, and lower testing red reward, by adding policy diversity into the training. Our proposed IET approach achieves high blue agent testing rewards, low red agent testing rewards, as well as relatively small gaps between the training and the testing rewards, outperforming the other three baselines in both the Connect Four and the Leduc Hold'em scenarios. While Fig. 5c, 5d, show that in Texas Hold'em, FLVT achieves better results than IET, our discussion in the next section shows that this is likely because the red exploiter policy learned by FLVT is too weak.

## E. Competition scores between training settings

Tables 1 shows the competition scores between training settings, where the blue agent uses the policy learned in the training thread, while the red agent uses the exploiter policy learned in the testing thread. As a result, the diagonal scores are in favor of the red agent, because the red exploiter policy is trained against the corresponding blue policy, but not the other way around. The diagonal scores measure the adversarial robustness of the learned blue policy. In contrast, the off-diagonal scores measure the out-of-distribution robustness of the blue policy, because neither the blue policy nor the red policy has been trained against each other. It should be noted that sometimes the diagonal score may fail to measure adversarial robustness because of the difficulty of training the red exploiter to optimality. For example, the FLVT red exploiter in Table **Texas Hold'em** does not successfully exploit the corresponding blue policy, which makes the diagonal score misleading. We speculate that this failure is because of the limited fitting capability of the FLVT network.

## F. Visualization of latent variable distribution

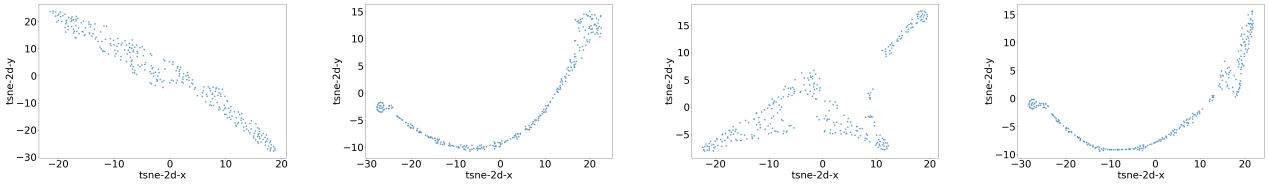
Fig. 6 shows the 2D t-distributed stochastic neighbor embedding (t-SNE) visualization of the latent condition variable. Besides the three board games, we also run our IET approach on the **Simple** scenario in the Multiagent Particle Environment (MPE), which is a simple single-agent scenario where the agent is rewarded based on its distance to its goal at a random location. The figure shows two different patterns of the latent condition variable distribution. The latent condition variable lives on a flat hyper-plane in the single-agent scenario (MPE: Simple). In constrast, in the multiagent scenarios (Connect Four, Leduc Hold'em, and Texas Hold'em), the latent condition variables live on more complicated manifolds with curvatures and clusters, which indicates that the corresponding conditional policies have more diversity.

*Table 1.* Competition scores between different training settings. IET achieves the best adversarial robustness in **Connect Four** and **Leduc Hold'em**. In **Texas Hold'em**, although FLVT achieves the highest diagonal blue score, the off-diagonal blue scores of FLVT are much lower than the diagonal score, which suggests that the blue FLVT policy is not robust to out-of-distribution red policy and the FLVT red exploiter fails to minimize the blue reward. In contrast, the IET blue policy achieves the second highest diagonal score as well as the highest off-diagonal blue scores across all the first three columns, suggesting that the IET blue policy is the most robust one.

| CONNECT FOUR | SPT | SET | FLVT | IET |
|---|---|---|---|---|
| SPT | -1.0±0.0/1.0±0.0 | -0.28±0.04/0.03±0.04 | 0.18±0.03/-0.72±0.03 | 0.21±0.04/-0.38±0.04 |
| SET | 0.32±0.04/-0.39±0.04 | -0.98±0.01/0.98±0.01 | 0.62±0.03/-0.76±0.03 | 0.49±0.03/-0.81±0.03 |
| FLVT | 0.95±0.01/0.95±0.01 | 0.47±0.04/-0.52±0.04 | -0.55±0.04/0.55±0.04 | 0.81±0.03/-0.85±0.02 |
| IET | 1.0±0.0/-1.0±0.0 | 1.0±0.0/-1.0±0.0 | 1.0±0.0/-1.0±0.0 | **-0.42±0.04/0.25±0.04** |

| LEDUC HOLD'EM | SPT | SET | FLVT | IET |
|---|---|---|---|---|
| SPT | -1.0±0.12/0.27±0.12 | -0.38±0.08/-0.30±0.09 | -0.13±0.1/-0.59±0.1 | -0.16±0.08/-0.41±0.07 |
| SET | 0.27±0.15/-0.83±0.14 | -0.35±0.09/-0.28±0.1 | -0.01±0.11/-0.73±0.1 | -0.35±0.09/-0.2±0.09 |
| FLVT | -0.49±0.06/-0.22±0.05 | -0.38±0.06/-0.24±0.06 | -0.65±0.06/-0.06±0.06 | -0.62±0.06/0.04±0.06 |
| IET | 0.15±0.12/-0.80±0.12 | -0.22±0.09/-0.45±0.09 | 0.17±0.11/-0.83±0.1 | **-0.33±0.09/-0.24±0.09** |

| TEXAS HOLD'EM | SPT | SET | FLVT | IET |
|---|---|---|---|---|
| SPT | -1.77±0.32/1.17±0.32 | -0.56±0.11/-0.31±0.11 | -0.04±0.08/-0.53±0.08 | -0.47±0.05/-0.48±0.05 |
| SET | -0.82±0.27/0.38±0.27 | -0.51±0.18/-0.18±0.18 | -0.04±0.1/-0.43±0.1 | -0.40±0.1/-0.51±0.10 |
| FLVT | -0.54±0.25/-0.17±0.25 | -0.64±0.18/-0.28±0.18 | **0.03±0.14/-0.77±0.14** | -0.21±0.13/-0.67±0.14 |
| IET | -0.09±0.24/-0.78±0.24 | -0.44±0.16/-0.26 ±0.16 | 0.11±0.12/-0.56±0.12 | -0.25±0.12/-0.52±0.12 |




(a) MPE: Simple (b) Connect Four (c) Leduc Hold'em (d) Texas Hold'em

*Figure 6.* t-SNE of the learned latent condition variable distributions, where the distribution in the single-agent (MPE: Simple) scenario is flat, while the distributions in the multiagent scenarios are complicated that involves multiple modes and clusters.