Appendix

Role-aware Architecture

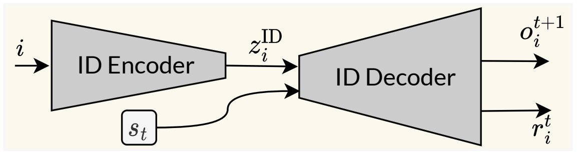


Figure 1: The Identity encoder-decoder architecture. The encoder learns to encode the identity of an agent in an embedding space while the decoder predicts the reward and next observation.

The encoder-decoder model is designed to learn from the experiences of all agents. It is trained using samples collected from every agent and is intended to capture the aggregate of agent-specific transition $P_i(s'|s, a_i)$ and reward functions $R_i(s, a_i)$ for all $i \in n$. Given the inputs to the decoder, the identity of each agent can only be conveyed through the latent variable $z^{\mathtt{ID}}$. Minimizing the model loss in Equation 3 can be performed before the reinforcement learning phase. During this stage, we sample actions $a_i \sim A$ and store the resulting trajectories in a shared experience replay buffer that is accessible to all agents. Empirically, we find that the amount of data required for this modeling phase is significantly lower by orders of magnitude than what is typically necessary for reinforcement learning. Moreover, this data can also be reused for policy training, meaning it does not contribute additional sample complexity.

The final stage of the pre-training process involves applying a clustering algorithm to the encoder outputs $f_E^{\mathtt{ID}}(\cdot)$ for all $i \in n$, and using the resulting clusters to define the agent grouping μ. In our experiments, we use k-means clustering for simplicity, and the number of clusters is chosen based on the number of roles/missions provided Google Research Football (GRF)[1], StarCraft Multi-Agent Challenge (SMAC)[2], and SMACv2[3] documentations. Once the agents are partitioned, a static computational graph can be constructed for automatic differentiation, enabling efficient training of the policies with considerable computational speed-up.

Identity-aware Diversity

Estimating $p(\mathtt{ID} \mid \tau_t)$ precisely can make the optimization process intractable. However, as we will illustrate in this section, a high-quality, though not necessarily precise, approximation of $p(\mathtt{ID} \mid \tau_t)$ is often sufficient to ensure effective learning. We now introduce two approaches for approximating this distribution.

The first method, inspired by prior work (Sharma et al. 2019), assumes that $p(\mathtt{ID} \mid \tau_t) \approx p(\mathtt{ID})$, where $p(\mathtt{ID})$ is modeled as a uniform distribution. Under this assumption, the conditional probability of selecting an action given the trajectory can be estimated as:

$$p(a_t \mid \tau_t) \approx \frac{1}{n} \sum_{\mathtt{ID}} \pi(a_t \mid \tau_t, \mathtt{ID}), \tag{1}$$

where n denotes the number of agents. However, since our framework encourages agents to act distinctly when needed, the actual conditional distribution $p(\mathtt{ID} \mid \tau_t)$ may diverge from the uniform prior, meaning that this assumption may not always be valid.

To address this, the second method lifts the uniformity assumption and instead applies Monte Carlo (MC) sampling to estimate $p(\mathtt{ID} \mid \tau_t)$. In tabular or discrete settings, this can be computed through empirical visitation frequencies:

$$p(\mathtt{ID} \mid \tau_t) = \frac{N(\mathtt{ID}, \tau_t)}{N(\tau_t)}, \tag{2}$$

where $N(\cdot)$ is the time of visitation. Nevertheless, MC becomes infeasible in complex scenarios like GRF and SMAC, which feature high-dimensional, continuous spaces and long decision horizons.

In such cases, following (Wang et al. 2019), we adopt a variational inference strategy to learn an approximation $q_\xi(\mathtt{ID} \mid \tau_t)$, parameterized by a neural network with parameters ξ. This network is trained to approximate $p(\mathtt{ID} \mid \tau_t)$ by maximizing the ELBO.

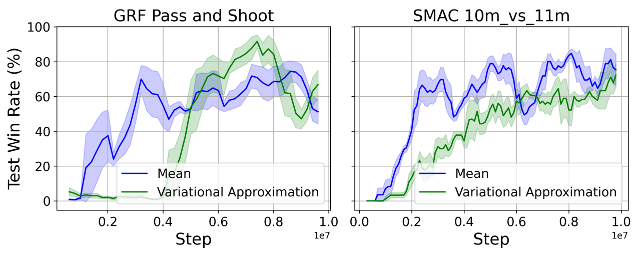


Figure 2: Comparison in SDE-HARL using mean estimation and variation approximately.

We evaluate both estimation techniques on the GRF `Academy_Pass_and_Shoot` and SMAC `10m_vs_11m`. As shown in Figure 2, these findings indicate that while both approaches yield comparable performance, the uniform prior assumption yields better average return and lower variance across random seeds. We attribute this to the instability caused by estimation errors in the variational method. Consequently, throughout this paper, we employ the uniform-based estimation from Equation 1 to compute $p(a_t \mid \tau_t)$.

Experiment Details

Hyperparameters. We use the PKU-MARL[4] library to implement all methods, applying identical hyperparameter values based on its default settings to ensure fair comparisons. Our method introduces three key control parameters, β, β_1, and β_2, which govern the intrinsic rewards. For both GRF and SMAC,we tune with grid search

[1]https://github.com/google-research/football/blob/master/gfootball/doc/scenarios.md

[2]https://github.com/oxwhirl/smac/blob/master/docs/smac.md

[3]https://github.com/oxwhirl/smacv2

[4]https://github.com/PKU-MARL/MARLlib

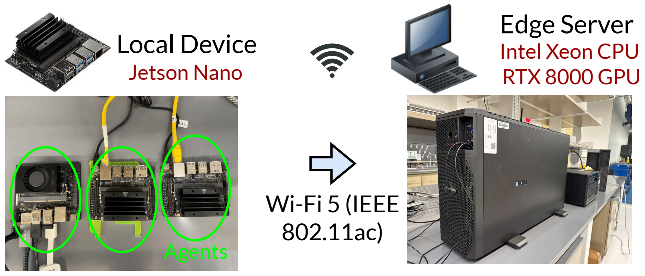
Dataset	Environment	β	β_1	β_2
GRF	Run Pass and Shoot	0.1	0.5	1.0
	3 vs 1 with Keeper	0.1	0.5	1.0
	Counterattach Hard	0.1	0.5	1.0
	Academy Corner	0.1	0.5	1.0
SMAC	3s5z	0.1	2.0	1.0
	8m_vs_9m	0.07	2.0	0.5
	Corridor	0.03	0.5	1.0
	27m_vs_30m	0.04	0.5	0.5
SMACv2	Protoss 5 vs 5	0.3	0.8	1.0
	Terran 5 vs 5	0.3	0.8	1.0
	Zerg 5 vs 5	0.3	0.8	1.0

Table 1: identity-aware diversity (IAD) Hyperparameters.

$\beta \in (0.05, 0.1, 0.2)$, $\beta_1 \in (0.05, 0.1, 0.2)$, and $\beta_2 \in (0.05, 0.1, 0.2)$. Specifically, tuning on GRF is conducted using the `academy_3_vs_1_with keeper` scenario. In the case of SMAC, we observe that `corridor` differs fundamentally from `3s5z` and `8m_vs_9m` due to the greater ratio of enemy agents to our agents, which demands more sophisticated cooperative strategies. For SMACv2, hyperparameter tuning is performed on `protoss_5_vs_5`. As a result, our model adopts different hyperparameter configurations across tasks to adaptively align with scenario-specific dynamics.

Environments. In Table 1, we use GRF scenarios—`Run Pass and Shoot` and `Academy Corner` to represent settings with 2 and 11 agents, respectively. For SMAC/SMACv2, we adopt `protoss_5_vs_5` and `27m_vs_30m` to configure environments with 5 and 27 agents, respectively.

Infrastructure. The setup in Figure 4 is configured as follows. We used three Raspberry Pi 5 units, each powered by a 2.4GHz quad-core Arm Cortex-A76 CPU and equipped with 16GB of LPDDR4X-4267 SDRAM, to serve as three agents. The edge server was implemented on a Linux-based machine featuring an Intel® Xeon® Gold 6246R CPU and an NVIDIA Quadro RTX 8000 GPU. Communication between devices utilized the Wi-Fi 5 (IEEE 802.11ac) standard, operating at 2.4 GHz with a 20 MHz bandwidth. We consider this configuration as **Scenario 1**. To highlight resource efficiency, we construct **Scenario 2** by replacing three Raspberry Pi 5 with three Jetson Nano, powered by a 1.43GHz quad-core ARM Cortex-A57 CPU and a 128-core NVIDIA Maxwell GPU, equipped with 4GB of LPDDR4 memory to serve as a low-power edge AI device. The setup is illustrated in the figure below.



Extension Results

In this section, we evaluate the inference time of our proposed framework under **Scenario 2**. Specifically, we measure how inference latency varies with the number of deep neural network (DNN) blocks allocated to local devices, which in this setting are equipped with more powerful hardware.

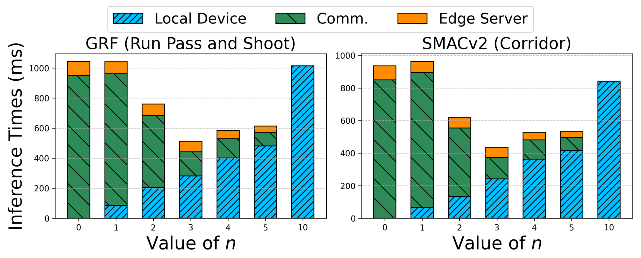


Figure 3: Inference time of SDE-HARL on **Scenarios 2** when changing the number local DNN blocks n.

As illustrated in Figure 3, leveraging more powerful edge devices ($i.e$, Jetson Nano) significantly reduces local inference time, decreasing from 140ms to 96ms when allocating $n = 3$ DNN blocks locally. Notably, distributing exactly three blocks across the local devices yields the lowest overall inference latency, nearly halving the runtime compared to both the fully local setup ($n = 10$) and the fully server-based configuration ($n = 3$). These findings underscore the necessity of distributed computation for deploying Multi-agent reinforcement learning (MARL) systems, even when local devices are equipped with substantial processing capabilities, especially in large-scale settings with constrained resource environments.

References

Sharma, A.; Gu, S.; Levine, S.; Kumar, V.; and Hausman, K. 2019. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*.

Wang, T.; Wang, J.; Wu, Y.; and Zhang, C. 2019. Influence-Based Multi-Agent Exploration. arXiv:1910.05512.