# Semantic Heterogeneous-Agent Reinforcement Learning Systems

Mohammad Abdi[1], Francesca Meneghello[1], Nathaniel D. Bastian[2], and Francesco Restuccia[1]

[1]Northeastern University [2]United States Military Academy

Corresponding author e-mail: abdi.mo@northeastern.edu

**Abstract.** Heterogeneous-agent reinforcement learning (HARL) enables cooperative agents with different observations and action roles to learn specialized policies. However, state-of-the-art deep neural network (DNN)-based policies are difficult to execute continuously on resource-constrained mobile agents. Edge-assisted policy execution mitigates this computation bottleneck. Yet, existing systems either offload raw observations or transfer intermediate latents through conventional network protocols, making communication a dominant source of delay. We present *Semantic HARL* (S-HARL), a dynamic data-driven framework that instantiates the dynamic data driven applications systems (DDDAS) model–instrumentation feedback loop for cooperative mobile agents: the edge-assisted policy dynamically incorporates live channel measurements to adapt its communication interface at execution time, while the actions it produces govern agent behavior and shape future observations. Each agent processes its observation with a local policy DNN, produces a stochastic semantic latent, and maps it through a network-conditioned precoder into a policy waveform. The edge decodes the received signals, aggregates the recovered latents with a shared global policy module, and produces actions through agent-specific heads. Instead of requiring bit-exact feature delivery, S-HARL learns representations that preserve the task semantics needed for cooperative control under current network conditions. We train S-HARL end-to-end with a unified heterogeneous-agent proximal policy optimization (HAPPO) objective and a multi-agent information-bottleneck regularizer that controls the information density of each transmitted policy waveform. We prototype S-HARL using Raspberry Pi agents, Universal Software Radio Peripheral (USRP) software-defined radios (SDRs), and an edge server. In Google Research Football (GRF), S-HARL reduces the latency of policy execution by 2.7× and the total agent-side energy by 70.7%, while matching the score rate of full-stack distributed policy execution within 0.6 percentage points.

**Keywords:** Heterogeneous-Agent Reinforcement Learning · Dynamic Data-Driven Multi-Agent Systems · Semantic Communications · Distributed Edge Intelligence · Efficient Machine Learning.

## 1 Introduction

Teams of mobile agents are increasingly expected to make decisions from high-dimensional observations while coordinating under partial observability, limited computation, and heterogeneous behavior requirements. Examples include ground robots, drones, and autonomous cars that must act toward a shared objective while each agent may observe only a local view of the environment/task, experience different dynamics, or need agent-specific action distributions. Such systems are a natural instance of the dynamic data driven applications systems (DDDAS) paradigm [3, 4]: a cooperative policy is an executing model that must dynamically incorporate sensory data from distributed instrumentation (agents) while the model's own outputs (actions) in turn govern how that instrumentation is deployed and what it observes next. Realizing this feedback loop efficiently over a physical network is the core of our investigation.

While heterogeneous-agent reinforcement learning (HARL) provides a foundation for cooperative decision making in this setting [6, 12, 7], deploying cooperative policies on mobile agents remains difficult. Modern HARL commonly relies on executing computationally complex deep neural networks (DNNs) that process rich observations, such as image frames. Running the entire policy locally on embedded processors forces the architecture to be simplified, often reducing cooperative performance. A natural alternative is to offload policy execution to an edge server: agents stream observations, the edge executes the policy, and actions are returned. This solves the device computation bottleneck, but creates a networking bottleneck because raw observations contain far more information than is needed for one policy decision.

Split policy execution reduces this cost by placing a lightweight encoder on each agent and executing the remaining policy layers at the edge. Recent distributed HARL systems follow this direction by transferring compressed policy latents instead of raw observations [5, 11]. This is well matched to heterogeneous teams because the edge can share a global module while retaining agent-specific action heads. Yet the communication interface is still a conventional network path: the latent is serialized, packetized, acknowledged, and reconstructed before the edge policy can use it. This breaks the DDDAS feedback loop: the executing model at the edge cannot adapt to current network conditions without retraining, and the network itself is treated as a passive, fixed-overhead channel rather than a dynamic component of the instrumentation architecture. Ultimately, this introduces unbearable network delays, especially for time-critical applications.

In this paper, we propose S-HARL, a semantic networking framework for low-latency HARL that instantiates the DDDAS model–instrumentation feedback loop at the physical layer. The key observation is that the edge does not need a bit-exact copy of the intermediate features; it only needs the necessary information to choose effective cooperative actions. Hence, full-stack network overhead can be avoided by directly transmitting learned representations as waveforms [1]. As shown in Fig. 1, S-HARL has an edge-assisted distributed policy structure, while waveforms are used as the interface. Each agent maps its observation to a learned waveform, which is transmitted through a physical-layer

frame, instead of a full-stack packetized latent. The edge consumes the received, channel-distorted latents directly, aggregates them through a shared global policy and then uses agent-specific heads to output actions. These actions direct agent behavior, which determines the future sensing positions and observations of the team, closing the model–instrumentation DDDAS loop. The agent–edge communication interface is a trainable component of the end-to-end policy.

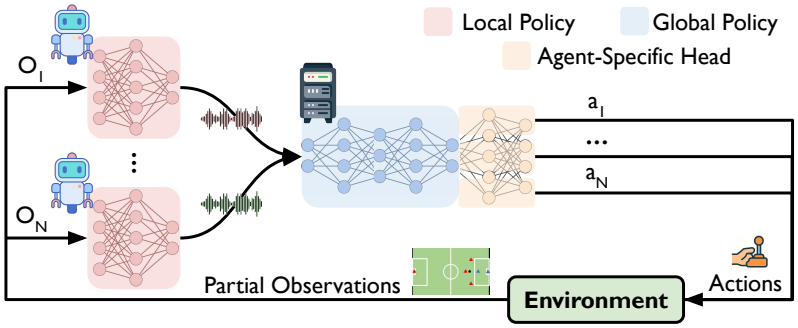



**Fig. 1.** Overview of S-HARL. Agents encode local observations into semantic policy waveforms conditioned on channel measurements for dynamic data-driven adaptation; the edge aggregates the received latents through a global policy module followed by agent-specific action heads. The actions govern agent behavior and future observations.

**Challenges.** The main challenge is that the agent–edge communication interface must be trained without reducing the problem to feature reconstruction. S-HARL addresses this by optimizing the policy and network interface jointly. The heterogeneous-agent proximal policy optimization (HAPPO) objective rewards latents that support cooperative action selection, while a multi-agent information bottleneck (IB) regularizer ensures that each transmitted latent only carries the task-relevant information [10]. S-HARL leverages precoder and postcoder modules conditioned on the channel state information (CSI) to enable dynamic data-driven adaptation to the changing network conditions at execution time without the need for runtime policy fine-tuning.

**Summary of Contributions.** • We introduce a semantic policy-execution architecture for HARL grounded in the DDDAS paradigm, where agents directly transmit learned waveforms instead of raw observations or full-stack latent packets. • We formulate end-to-end training by combining the HAPPO objective with a multi-agent IB regularizer over Gaussian policy latents. • We prototype S-HARL on Raspberry Pi agents, Universal Software Radio Peripheral (USRP) software-defined radios (SDRs), and an edge workstation, and show on Google Research Football (GRF) that S-HARL reduces policy execution latency 2.7× and total agent-side energy by 70.7% while matching the score rate of full-stack distributed policy execution within 0.6 percentage points.

## 2 S-HARL Design and Problem Formulation

S-HARL considers $N$ resource-constrained agents that execute heterogeneous policies with help from an edge server. We model the task as a cooperative

decentralized partially observable Markov decision process (Dec-POMDP) [9]. At policy step $t$, agent $i$ observes a local frame $o_{i,t} \in \mathcal{O}_i$, the team selects a joint action $\mathbf{a}_t = (a_{1,t}, \ldots, a_{N,t})$, and the environment returns a shared reward $r_t$. Without loss of generality, we write policies as functions of the current observation $o_i$, while recurrent policies can replace $o_i$ with the local history. The objective is to maximize $J(\Theta) = \mathbb{E}_{\pi_\Theta}[\sum_{t\geq 0} \gamma^t r_t]$, where $\Theta$ is the DDDAS executing model that entails the local policy, the network-interface, and the edge policy parameters. In the following, we omit the time index $t$ for clarity of exposition.

### 2.1 Semantic Policy Interface

Conventional split execution treats an intermediate representation as data to be delivered reliably. S-HARL instead makes the agent-edge interface part of the differentiable policy graph. Figure 2 walks through this interface: each agent maps its observation to a stochastic semantic latent, conditions a precoder on the current network state, and directly outputs and transmits a policy signal, while the edge postcodes the received symbols before computing actions.

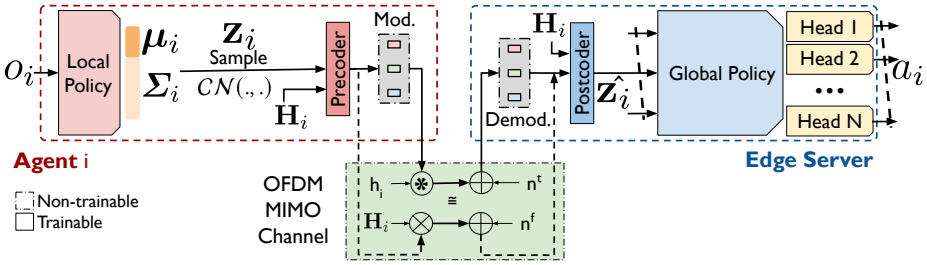



**Fig. 2.** End-to-end training of S-HARL. The agent–edge interface is inside the policy graph, allowing the local encoders, channel-conditioned precoders/postcoders, global policy, and action heads to be optimized jointly.

**Local Semantic Encoder.** Agent $i$ processes $o_i$ with a local policy DNN and outputs a complex Gaussian latent,

$$p_{\theta_i}(\mathbf{z}_i \mid o_i) = \mathcal{CN}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \qquad (\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = f_{\theta_i}(o_i). \tag{1}$$

We state the model for a general covariance and use $\boldsymbol{\Sigma}_i = \mathrm{diag}(\boldsymbol{\sigma}_i^2)$ in the implementation. The latent is sampled using the reparameterization $\mathbf{z}_i = \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \odot \boldsymbol{\epsilon}_i$, with $\boldsymbol{\epsilon}_i \sim \mathcal{CN}(\mathbf{0}, \mathbf{I})$, so the policy objective can update the local encoder.

**Network-Conditioned Transmission.** A learnable precoder ($\mathrm{Pre}_{\alpha_i}$) dynamically maps the latent to an orthogonal frequency-division multiplexing (OFDM) multiple-input multiple-output (MIMO) policy waveform by incorporating, as a conditioning variable, the estimated CSI $\mathbf{H}_i$ which comes from the runtime DDDAS instrumentation measurements and drives adaptation at execution time without the need for retraining:

$$\mathbf{X}_i = \mathrm{Pre}_{\alpha_i}(\mathbf{z}_i, \mathbf{H}_i)\ , \qquad \frac{1}{K}\sum_{k=1}^{K} \|\mathbf{x}_{i,k}\|_2^2 \leq P_i\ , \tag{2}$$

where $\mathbf{X}_i = [\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,K}]$, $\mathbf{x}_{i,k} \in \mathbb{C}^{N_t}$, and the inequality bounds the average transmit power of agent $i$ across the $K$ OFDM subcarriers within the radio power budget $P_i$. The edge receives, for subcarrier $k$,

$$\mathbf{y}_{i,k} = \mathbf{H}_{i,k} \cdot \mathbf{x}_{i,k} + \mathbf{n}_{i,k} , \qquad \mathbf{n}_{i,k} \sim \mathcal{CN}(\mathbf{0}, \sigma_n^2 \mathbf{I}) , \tag{3}$$

with $\mathbf{H}_{i,k} \in \mathbb{C}^{N_r \times N_t}$. The edge postcoder recovers a distorted latent $\hat{\mathbf{z}}_i = \mathrm{Post}_{\delta_i}(\mathbf{Y}_i, \mathbf{H}_i)$ from $\mathbf{Y}_i = [\mathbf{y}_{i,1}, \ldots, \mathbf{y}_{i,K}]$. By conditioning the postcoder $\mathrm{Post}_{\delta_i}$ on the runtime CSI $\mathbf{H}_i$ as it is done for the precoder, the entire system is adapted dynamical and in a data-driven fashion to the changing channel conditions.

**Edge Policy.** The edge concatenates the recovered latents, extracts a shared cooperative representation $\mathbf{h}$, and applies agent-specific heads $\pi_{\omega_i}$:

$$\mathbf{h} = g_\psi([\hat{\mathbf{z}}_1; \ldots; \hat{\mathbf{z}}_N]) , \qquad \pi_\Theta(\mathbf{a} \mid \mathbf{o}, \mathbf{H}) = \prod_{i=1}^{N} \pi_{\omega_i}(a_i \mid \mathbf{h}) . \tag{4}$$

This preserves a shared edge-side policy while allowing heterogeneous action behavior through the final heads. The selected actions are small control messages and are returned over a low-rate downlink. These actions constitute the DDDAS reverse channel: the executing edge model directs agent behavior, which governs the agents future sensing positions and thereby shapes the observations that will drive the next policy step.

### 2.2 End-to-End Training

The key training question is how to make the policy signal useful for long-term return without requiring supervised reconstruction. S-HARL replaces supervised inference loss with the HAPPO actor-critic objective [12, 7]. Let $\mathcal{L}_{\mathrm{HAPPO}}(\Theta)$ denote the HAPPO loss, including the clipped policy surrogate, value loss, and entropy regularization. Since actions are sampled after Equations (1)–(4), this loss jointly updates the local encoders, precoders, postcoders, global module, and action heads. The CSI $\mathbf{H}_i$ enters the graph as a non-trainable runtime-varying input, serving as the dynamic data that the model assimilates at each step without the need for weight update. To prevent the local encoder from embedding arbitrary observation details, S-HARL adds a multi-agent IB penalty that keeps each Gaussian latent close to $p(\mathbf{z}) = \mathcal{CN}(\mathbf{0}, \mathbf{I})$:

$$\mathcal{L}_{\text{S-HARL}}(\Theta) = \mathcal{L}_{\mathrm{HAPPO}}(\Theta) + \beta \sum_{i=1}^{N} \mathbb{E}_{o_i}\left[ D_{\mathrm{KL}}\big(p_{\theta_i}(\mathbf{z}_i \mid o_i) \| \mathcal{CN}(\mathbf{0}, \mathbf{I})\big)\right] . \tag{5}$$

For the diagonal covariance, the regularizer becomes

$$D_{\mathrm{KL}}\big(\mathcal{CN}(\boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2)) \| \mathcal{CN}(\mathbf{0}, \mathbf{I})\big) = \sum_{m=1}^{d} \left( |\mu_{i,m}|^2 + \sigma_{i,m}^2 - \log \sigma_{i,m}^2 - 1 \right) . \tag{6}$$

The scalar $\beta$ controls the information contained in the policy waveform. From a DDDAS perspective, it controls the instrumentation fidelity: higher $\beta$ forces the signal closer to the prior, discouraging memorization and making the representation more robust to network distortion.

## 3 Experimental Evaluation

Our evaluation asks whether S-HARL preserves cooperative policy quality, removes full-stack networking delay, reduces agent-side energy, and remains robust to changing network conditions.

**Task & Training.** We use `academy_3_vs_1_with_keeper` in GRF [8], with three controlled agents and standard discrete football actions. Each agent receives a $72 \times 96 \times 3$ RGB observation processed by the local policy DNN. We report score rate, i.e., the fraction of evaluation episodes in which the team scores before termination. All methods use the same HAPPO backbone, rollout budget, optimizer, and policy structure. S-HARL is trained from scratch using Equation (5). During training, the non-trainable network layer samples the CSI associated with independent $2 \times 2$ OFDM MIMO links with $K = 64$ subcarriers and signal-to-noise ratio (SNR) drawn from $[10, 30]$ dB. At test time, the policy weights are frozen and only the CSI input to the precoder/postcoder changes for dynamic data-driven adaptation to varying channel conditions. The results are averaged over 100 evaluation episodes.

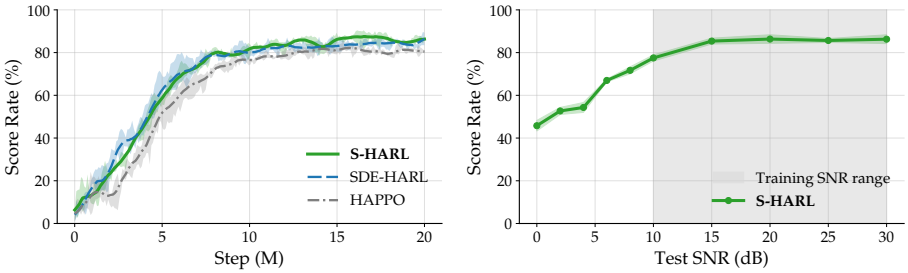
**Prototype.** The local policy runs on Raspberry Pi 4 agents while the edge policy runs on an Intel Xeon workstation with NVIDIA A100 graphics processing units (GPUs). Each agent communicates with the edge through an independent USRP B210 link, which supports $2 \times 2$ MIMO operations. For the full-stack baselines, raw observations or latents are carried by Transmission Control Protocol (TCP) over the `gr-ieee802-11` SDR Wi-Fi stack, an IEEE 802.11a/g/p OFDM transceiver for GNU Radio [2]. For S-HARL, the frame contains the learned policy signal plus a short physical layer (PHY) preamble/header. We evaluated local execution, agent–edge communication, and edge computation; the reported policy-step latency is the slowest agent-side path plus shared edge execution.

**Baselines & Metrics.** We compare our proposed strategy against: (i) *Local HAPPO*, which executes the full policy on the Raspberry Pi agents; (ii) *Edge HAPPO*, which performs full edge offloading by sending raw observations to the edge over Wi-Fi; and (iii) *Full-stack distributed policy execution*, which follows the SDE-HARL split-policy structure [5] and transfers policy latents reliably over Wi-Fi. We report score rate, policy-step latency, total agent-side energy per policy step, and score rate under different SNR values.

### 3.1 Experimental Results

**Policy Performance.** Figure 3(a) shows that S-HARL reaches a final score rate of 86.7%, matching full-stack distributed policy execution (87.3%) within 0.6 percentage points and outperforming local HAPPO (80.4%) by 6.3 percentage points. The results support the main semantic-networking premise: bit-exact latent reconstruction is unnecessary when the received message preserves the information needed for action selection.

**Latency.** Figure 4(a) breaks down policy-step latency. Edge HAPPO is dominated by raw-frame transfer. Full-stack distributed execution reduces payload

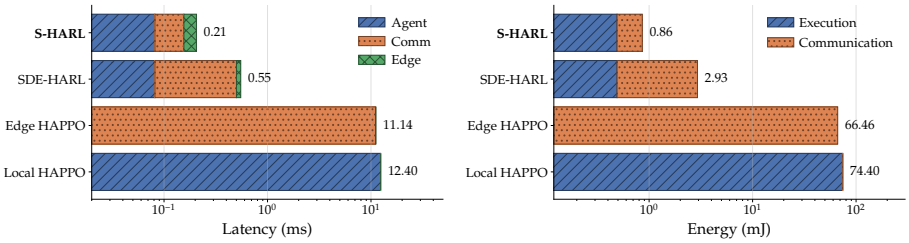



(a) Score rate during training (b) Score rate under different SNR values

**Fig. 3.** Policy performance of S-HARL. S-HARL tracks the SDE-HARL baseline during training while using a learned semantic policy signal, and remains robust across the SNR range used during training.

size but still incurs transfer delay before the edge can act. S-HARL replaces this path with a policy signal and a short PHY preamble/header, reducing latency from 0.55 ms to 0.21 ms, i.e., by 2.7× over full-stack distributed execution, and by 54.1× over edge HAPPO. The remaining time is dominated by local policy computation and edge execution, meaning communication is no longer the critical bottleneck. This advantage is made possible as S-HARL treat the model as a dynamic data-driven component conditioned on the changing network interface.

**Energy & Robustness.** Figure 4(b) shows that S-HARL reduces total agent-side energy from 2.93 mJ to 0.86 mJ, corresponding to a 70.7% reduction relative to full-stack distributed execution by avoiding packetized latent transfer while keeping local computation lightweight. Figure 3(b) shows that S-HARL maintains score rate above 85% for SNR values of at least 15 dB and degrades gracefully at lower SNR. This indicates that channel-conditioned precoding/postcoding enables the dynamic adaptation of the semantic policy signal without runtime fine-tuning.




(a) Policy-step latency breakdown (b) Agent-side energy breakdown

**Fig. 4.** System cost of S-HARL. By replacing packetized latent delivery with a learned policy message and a short PHY frame, S-HARL removes most of the communication delay and energy incurred by full-stack distributed policy execution.

Overall, S-HARL preserves the policy quality of edge-assisted HARL while replacing reliable latent transfer with a learned policy waveform, yielding lower latency and energy under dynamic network conditions. Our results show that dynamically incorporating live channel measurements into the executing policy

produces a system that is more efficient and more adaptive than approaches that treat communication as a static, passive layer.

## 4 Conclusion

We introduced S-HARL, a semantic dynamic data-driven networking framework for distributed HARL. By making the agent-edge interface part of the policy graph, S-HARL avoids bit-exact latent delivery and instead learns policy signals that preserve action-relevant information under ongoing network conditions. Our experimental results show the potential of semantic policy execution to reduce latency and energy while preserving cooperative performance.

## References

1. Abdi, M., Haque, K.F., Meneghello, F., Ashdown, J., Restuccia, F.: PhyDNNs: Bringing Deep Neural Networks to the Physical Layer. In: IEEE INFOCOM 2025-IEEE Conference on Computer Communications. pp. 1–10. IEEE (2025)
2. Bloessl, B., Segata, M., Sommer, C., Dressler, F.: An IEEE 802.11a/g/p OFDM Receiver for GNU Radio. In: Proceedings of the ACM SIGCOMM 2013 Workshop on Software Radio Implementation Forum. pp. 9–16 (2013)
3. Darema, F.: Dynamic Data Driven Applications Systems: A New Paradigm for Application Simulations and Measurements. In: International conference on computational science. pp. 662–669. Springer (2004)
4. Darema, F., Blasch, E.P., Ravela, S., Aved, A.J.: The Dynamic Data Driven Applications Systems (DDDAS) Paradigm and Emerging Directions. Handbook of Dynamic Data Driven Applications Systems: Volume 2 pp. 1–51 (2023)
5. Gian, T.D., Abdi, M., Bastian, N.D., Restuccia, F.: SDE-HARL: Scalable Distributed Policy Execution for Heterogeneous-Agent Reinforcement Learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 40, pp. 21287–21295 (2026)
6. Gronauer, S., Diepold, K.: Multi-agent deep reinforcement learning: a survey. Artificial Intelligence Review **55**(2), 895–943 (2022)
7. Kuba, J.G., Chen, R., Wen, M., Wen, Y., Sun, F., Wang, J., Yang, Y.: Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning. arXiv preprint arXiv:2109.11251 (2021)
8. Kurach, K., Raichuk, A., Stańczyk, P., Zając, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., Gelly, S.: Google Research Football: A Novel Reinforcement Learning Environment. Proceedings of the AAAI Conference on Artificial Intelligence **34**(04), 4501–4510 (2020)
9. Oliehoek, F.A., Amato, C.: A Concise Introduction to Decentralized POMDPs. Springer, 1st edn. (2016)
10. Tishby, N., Pereira, F.C., Bialek, W.: The Information Bottleneck Method. arXiv preprint physics/0004057 (2000)
11. Zhang, M., Abdi, M., Dasari, V.R., Restuccia, F.: Semantic Edge Computing and Semantic Communications in 6G Networks: A Unifying Survey and Research Challenges. Computer Networks **270**, 111531 (2025)
12. Zhong, Y., Kuba, J.G., Feng, X., Hu, S., Ji, J., Yang, Y.: Heterogeneous-Agent Reinforcement Learning. Journal of Machine Learning Research **25**(32), 1–67 (2024)