Extended Abstract

Motivation Designing multi-agent systems that scale to dynamic team sizes and maintain coordination under communication constraints is a critical challenge in real-world swarm robotics. Existing approaches often struggle with rigid architectures that either fail to generalize across agent counts, or that fail to allow communication among team members. Message pooling techniques inspired by Graph Neural Networks show promise in addressing these challenges, and we investigate their efficacy under a novel framework.

Method We propose HIVE, a novel message-pooling framework that combines shared-policy neural networks, attention-based message aggregation, and a centralized "cloud brain." This architecture allows decentralized agents to communicate efficiently and generalize to different team sizes at test time. The framework is modular, allowing for flexibility in different environments. In our environment, a team of "ants," influenced by a "hive" cloud brain, collaborate to move a "MacGuffin" to a goal location. We test two implementations of our framework: a non-recurrent version and a recurrent version using an LSTM.

Implementation Our system was implemented in the GPU-accelerated Madrona Engine, enabling training at over 500,000 timesteps per second. Agents receive LiDAR-based observations and are trained with Proximal Policy Optimization (PPO). The reward function combines normalized distance progress, a penalty for leaving the MacGuffin stationary, and a sparse success bonus to encourage coordinated movement through cluttered environments.

Results Compared to a baseline joint-state MLP that demonsrtated some coordination but did not successfully complete the task, HIVE achieved an 89% success rate with 5 agents. Additionally, the same model generalized to an 86% success rate with 8 agents without further training, demonstrating robustness to dynamic agent count. Attention-based message pooling is demonstrably effective in coordinating multiple agents.

Discussion and Conclusion HIVE demonstrates that scalable, message-pooling-based coordination enables strong generalization and robustness in dynamic multi-agent systems. The framework lays the foundation for real-world deployable swarms and offers promising directions for future work, including dropout resilience and runtime agent variability.

HIVE: A Multi-Agent Message Pooling Framework

Ty Toney
Department of Computer Science
Stanford University
tytoney@stanford.edu

Julian Allchin
Department of Computer Science
Stanford University
jallchin@stanford.edu

Diego Bustamante
Department of Computer Science
Stanford University
diegobus@stanford.edu

Abstract

Multi-agent systems hold great promise for complex tasks requiring distributed coordination, but designing architectures that scale to dynamic team sizes and communication constraints remains challenging. We introduce HIVE, a novel message-pooling framework that combines shared-policy neural networks, attention-based aggregation, and a centralized LSTM memory module (“cloud brain”) to enable efficient, scalable collaboration among homogeneous agents. In our 2D transport environment, identical “ants” must cooperatively maneuver a single payload through randomly generated obstacle courses to a fixed goal. HIVE’s per-agent MLP policy heads send and receive compressed messages via self-attention, while a global LSTM maintains temporal context and broadcasts strategic guidance back to each agent. We train all agents with Proximal Policy Optimization (PPO) using the GPU-accelerated Madrona Engine, collecting over 500,000 timesteps per second. Compared to a monolithic joint-state MLP baseline—which fails on all test environments—HIVE achieves success rates above 89% across 1000 evaluation maps, demonstrating robust obstacle negotiation and sustained cooperative transport. Ablations confirm that attention is essential for performance. By supporting variable team sizes at inference time without retraining, HIVE advances resource-efficient multi-agent coordination and lays the groundwork for deployable swarm-like systems in robotics and beyond.

1 Introduction

Multi-agent systems in the real world face challenges including connection-loss, malfunction, or bandwidth constraints. Moreover, some systems must be able to react dynamically to new command inputs.

We propose a novel framework to address these common challenges. We apply attention-based message-aggregation inspired by Graph Neural Networks to enable a dynamic number of agents controlled by a single hive mind, with model size independent of the number of agents.

We implemented two instances of our framework and investigated their efficacy in simulated environments. In our scenario, multiple decentralized agents ("ants") learn to collaborate and communicate under bandwidth constraints, guided by a single shared "hive mind" that aggregates agent messages and broadcasts a single message to all ants.

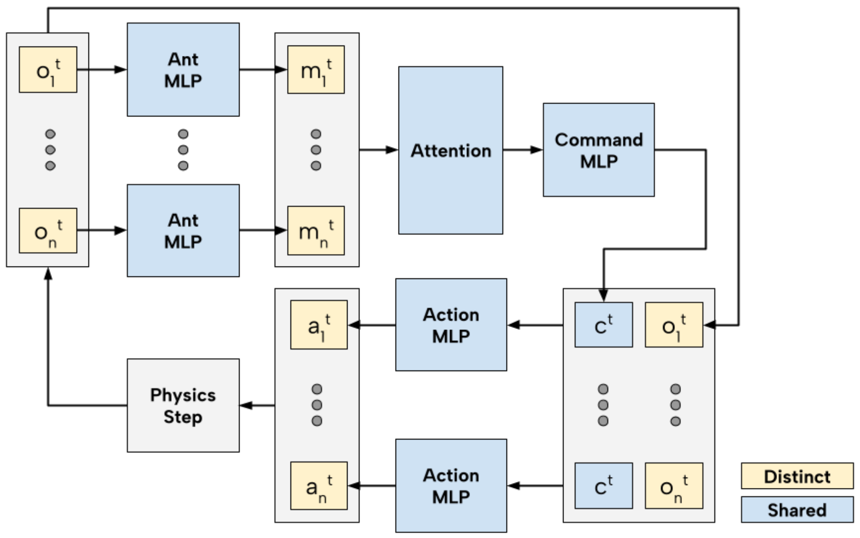

Figure 1: Non-recurrent version of proposed model architecture which utilizes attention to aggregate individual ant messages but does not feature an LSTM.

To do this, we designed a 2D, multi-agent reinforcement learning environment in the high-speed vectorized Madrona Engine in which ants cooperate to navigate a target object ("MacGuffin") through obstacles to a goal location.

Real-world swarms must tackle complex tasks with minimal communication. By handling dynamic team sizes and channel constraints, HIVE pushes resource-efficient coordination forward, addresses challenges faced by previous works, and opens the door to a future with deployable swarm-like assistants. Moreover, our framework is easily adaptable to general environments by substituting different ant and hive mind networks while maintaining fundamental message-pooling operations.

2 Related Work

Our work builds on several threads of multi-agent reinforcement learning (MARL), particularly models that emphasize communication and coordination. CommNet (Sukhbaatar et al., 2016) introduced differentiable communication by averaging message vectors, but this approach does not scale well to large, noisy agent populations. More recent works such as TarMAC (Das et al., 2019) and DIAL (Foerster et al., 2016) have introduced selective attention and message gating mechanisms to prioritize information. However, many of these systems assume fixed numbers of agents or require explicit receiver modeling. We instead adopt a shared-policy and cloud memory approach inspired by transformer style attention and collective memory, enabling the policy to generalize across a dynamically changing population. Our architecture also draws influence from “hivemind” models like Actor-Attention-Critic (Iqbal and Sha, 2019), though our use of a single LSTM to manage global temporal strategy is unique.

3 Method

3.1 Model Architecture

We designed two custom attention-based models. They draw inspiration from Graph Neural Networks (GNNs), which must aggregate information from a variable number of neighbors. Foremost among these techniques is attention-based message pooling. We apply this framework to agents, aggregating observations from a variable number of agents.

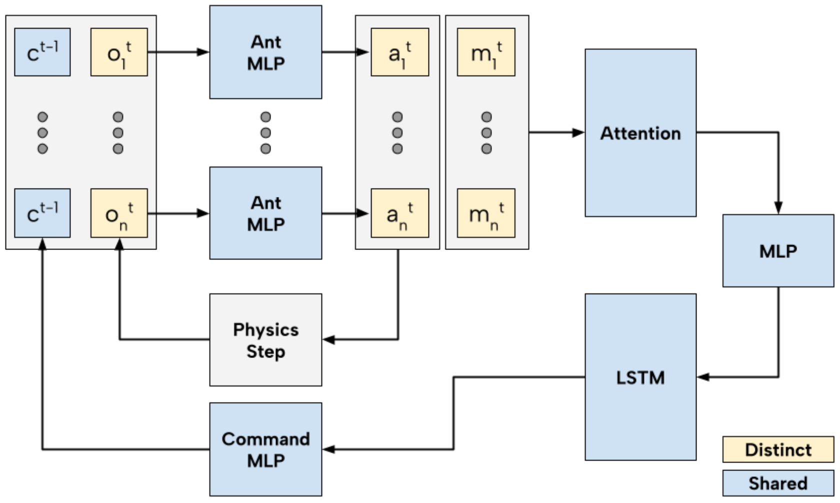

Figure 2: Full, recurrent network framework for training multi-agent models which allows for inter agent communication and dynamic number of agents.

Our first model, the "non-recurrent" model, has each ant convert observations o^t into a message m^t using an MLP with shared weights. Messages are combined using attention-based message pooling with a learned query parameter to create a single "command" c^t. This command is broadcast to all ants and combined with observations to create actions a_i^t, which are inputted to the simulation. Note that the model size is completely independent of the number of ants. See **Fig.**1 for a diagram. Because we train using PPO, we also design a critic. The critic is identical to the actor described here, except the command is transformed directly into the scalar value prediction using an MLP; no actions are produced.

Our second model is our "recurrent" model. It varies from the non-recurrent model in two ways. The first is the addition of an LSTM block after message pooling. The second difference is that ants use the command of the previous time step c^{t-1}, not the current command c^t, to generate their actions and messages. This enables the hivemind and the ants to run in parallel; ants interact with the environment while the hivemind processes their observations. Note that the model size is once again completely independent of the number of ants. See **Fig.** 2 for a diagram. The critic in this scenario is identical to the non-recurrent critic, with the exception of an LSTM block after message pooling.

We also implemented a simple MLP network to act as a baseline. It receives the concatenated observations of ants as inputs. The actor outputs actions, and the critic outputs a value estimation. Note that all agents are controlled together by a single model, and the model depends on the number of ants in the simulation.

3.2 Simulation Design

To assess whether the proposed model architecture offers improvements to existing multi-agent training methods, we designed a simulation that could demonstrate whether a dynamically sized hive of agents could perform a collective task (**Fig.** 3). In our simulation, a variable number of homogeneous “ant” agents cooperate to transport a single target object—the MacGuffin—to a fixed goal location. Each agent follows simple planar dynamics and perceives its surroundings through a simulated LiDAR sensor, returning distances to nearby obstacles and the MacGuffin. At the beginning of each episode, obstacles of varying shapes and sizes are scattered at random throughout the arena, creating a cluttered environment for the agents to navigate while manipulating the MacGuffin.

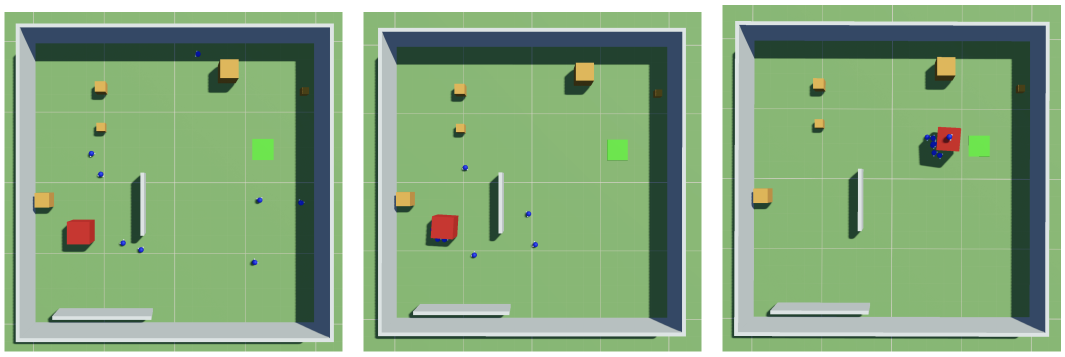
(a) Initialized HIVE environment **(b)** Ants retrieving MacGuffin **(c)** MacGuffin moving to goal

Figure 3: Snapshots from the HIVE simulation environment with 8 ants controlled by the full, recurrent model trained on 5 ants. The agents successfully coordinate to move the Macguffin (red) around a randomly placed wall and toward the target (green).

The collective (or "Hive") reward at each timestep t consists of three components: (1) a normalized distance-reduction reward

$$r_t^{\text{dist}} = \alpha \frac{d_{t-1} - d_t}{d_0} \tag{1}$$

where d_t is the Euclidean distance from the MacGuffin to the goal at time t and d_0 is the initial distance—ensuring that if the MacGuffin is carried all the way to the goal, $\sum_t r_t^{\text{dist}} = \alpha$; (2) a stationary penalty

$$r_t^{\text{stat}} = -\beta \mathbf{1}\left[v_t < v_{\text{th}}\right] \tag{2}$$

where v_t is the magnitude of the MacGuffin's planar velocity at timestep t, v_{th} is a small velocity threshold below which the MacGuffin is considered "stuck," and β is chosen so that if the MacGuffin remains below threshold for all $T_{\max}$ steps, $\sum_t |r_t^{\text{stat}}| = \beta$; and (3) a sparse success bonus

$$r^{\text{goal}} = \gamma \mathbf{1}\left[d_T \leq \epsilon\right] \tag{3}$$

awarded once at the end of the episode if the MacGuffin finishes within a small radius ϵ of the goal. Formally, the total episode reward is

$$R = \sum_{t=1}^{T} \left[r_t^{\text{dist}} + r_t^{\text{stat}}\right] + r^{\text{goal}} \tag{4}$$

Here, α, β, and γ control the relative weighting of progress, motion maintenance, and task completion, respectively, with $\alpha = 1$ and $\beta = 1$ chosen so that each term can contribute at most 1 over the course of $T_{\max}$ steps. In contrast to our earlier design—which used a constant existential penalty to encourage fast completion—we replaced that term with a velocity-based stationary penalty. This change ensures agents must keep the MacGuffin moving around obstacles, rather than simply rushing directly toward the goal, thereby promoting sustained cooperative transport behaviors in complex environments.

3.3 Simulation Implementation

Given the complexity of our proposed simulation and model, simulation and training was implemented using the Madrona Engine, a 3D research game engine which can run simulations in parallel on a GPU allowing us to run well over 500,000 concurrent timesteps per second (Shacklett et al., 2023). Instead of fully developing a new Madrona based system, the simulation was developed starting from Shacklett et al.'s Madrona Escape Room environment, which has similar agents and models, ultimately adapting the reward function, environment objects, agent observations, models, and training algorithm.

3.4 Training Algorithm

We adopted Proximal Policy Optimization (PPO) as our training algorithm due to its robustness, ease of implementation, and strong empirical performance in continuous control and multi-agent reinforcement learning settings. PPO strikes a balance between sample efficiency and training stability by clipping policy updates to prevent excessively large gradient steps, which is especially important in our scenario where the agents share a policy and interact through a centralized message-pooling architecture.

In our setup, all ants share a single policy network, which receives individualized inputs but contributes to a joint message-pooling mechanism and shared global LSTM memory. PPO's ability to handle shared policy learning with high variance input distributions makes it well-suited for this architecture. Additionally, PPO's compatibility with vectorized environments allowed us to leverage Madrona's massive parallelism, collecting thousands of episodes in parallel to quickly train robust cooperative behaviors across dynamic team sizes and randomized obstacle layouts.

4 Experimental Setup

To evaluate the impact of architectural choices on multi-agent cooperation, we conducted a series of controlled experiments across multiple environment seeds. We compared the performance of our non-recurrent and recurrent models to the baseline, simple MLP network, by training all three models in randomized environments with 5 ants. Models were trained until average rewards appeared to plateau.

Once models were trained, we evaluated their performance in 1000 episodes for 1000 timesteps each, recording (1) the percentage of successful episodes where the MacGuffin is moved to the goal and (2) the number of remaining timesteps in each of these successful episodes.

First, the MLP, non-recurrent, and recurrent models were evaluated with 5 ants, the same number of agents with which the models were trained with, and then the non-recurrent and recurrent models were evaluated with 8 agents, assessing how generalizable the models are to a number of agents not seen in the training data. Note that due to the fundamental structure of the baseline MLP network, the model trained on 5 agents could not be extended to any other number of agents, pointing to an inherent challenge in building multi-agent RL systems which our model seeks to address.

Table 1: Evaluation atrix for model architectures

Model	**Training**	**Evaluation**	**Metrics**
Baseline (MLP)	5 ants	5 ants	(1) Success rate (2) Remaining timesteps
Non-recurrent	5 ants	5, 8 ants	(1) & (2)
Recurrent	5 ants	5, 8 ants	(1) & (2)

5 Results

5.1 Quantitative Evaluation

Table 2 and Fig. 5 summarize the performance of our three architectures. The attention-based models dramatically outperform the baseline MLP, which fails on all 1000 test environments. The non-recurrent model achieves the highest peak reward and learns marginally faster than the recurrent model. Both attention-based models converge to similar long-run rewards (around 0.0005–0.0006), demonstrating that recurrence alone does not raise the final performance under the current reward scaling.

In contrast to the baseline, both attention-based models enabled all agents to participate effectively. The non-recurrent variant achieved the highest success rate (89%) on 5 ants, with strong generalization to 8 ants (86%), a situation completely outside the training data. The recurrent model, while slightly less successful (73%), still demonstrated strong coordination and generalized to 8 ants (68 %). These results confirm that attention-based message pooling leads to significantly more effective cooperation, enabling scalable, generalizable behaviors across varying agent counts.

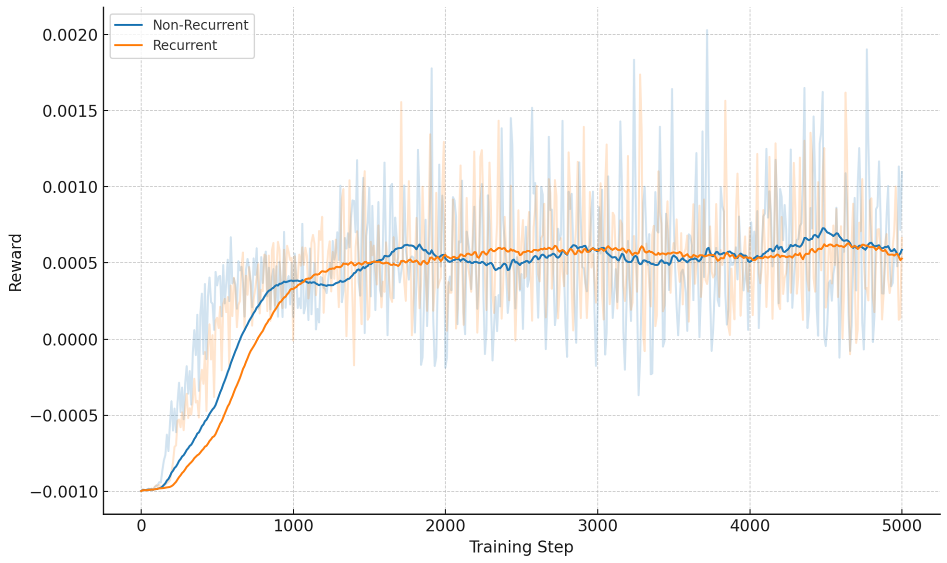

Figure 4: Smoothed training reward per step for the non-recurrent and recurrent HIVE models.

Table 2: Evaluation results for final models (1000 test environments)

Model	Success Rate	Worlds Completed	Avg. Time Left
Baseline MLP (5 ants)	0.00	0 / 1000	0.0000
Non-recurrent (5 ants)	0.89	892 / 1000	0.6961
Non-recurrent (8 ants)	0.86	863 / 1000	0.7101
Recurrent (5 ants)	0.73	731 / 1000	0.5896
Recurrent (8 ants)	0.68	676 / 1000	0.6120

5.2 Qualitative Evaluation

We visualized rollouts of each model to qualitatively understand their strategies and pitfalls. For the baseline MLP model, qualitative inspection revealed that only a subset of the ants contributed to pushing the MacGuffin, while the rest remained idle or counterproductive. This inefficiency led to slow or no progress, preventing completion within the time limit. We hypothesize that this inefficiency is partially because an MLP is inherently less suited to inter-agent coordination, as it fails to leverage the identical observation and command structure of each agent.

In contrast, we observed the recurrent and non-recurrent HIVE models successfully using all agents to cooperate and move the MacGuffin 5, even with 8 ants, a situation completely outside the training data. All agents contributed to the task's success.

However, no model completely learned to navigate all obstacles. We observed worlds where the ants pushed the MacGuffin into a barrier blocking the way to the goal, instead of moving the MacGuffin around the barrier. We did observe some qualitative improvement as training continued, despite a plateau of rewards, as they began to push the MacGuffin at an angle along the wall, not fully stuck. We hypothesize that a larger model or further training would overcome this obstacle.

6 Discussion

Generally, our results demonstrate the success of a novel framework for architecture design in multi-agent scenarios that is more robust to real-world challenges. We hope that this work serves as a launching point for further research into the messages-command framework, testing the framework in new environments and with different models for "ants" and the "hivemind".

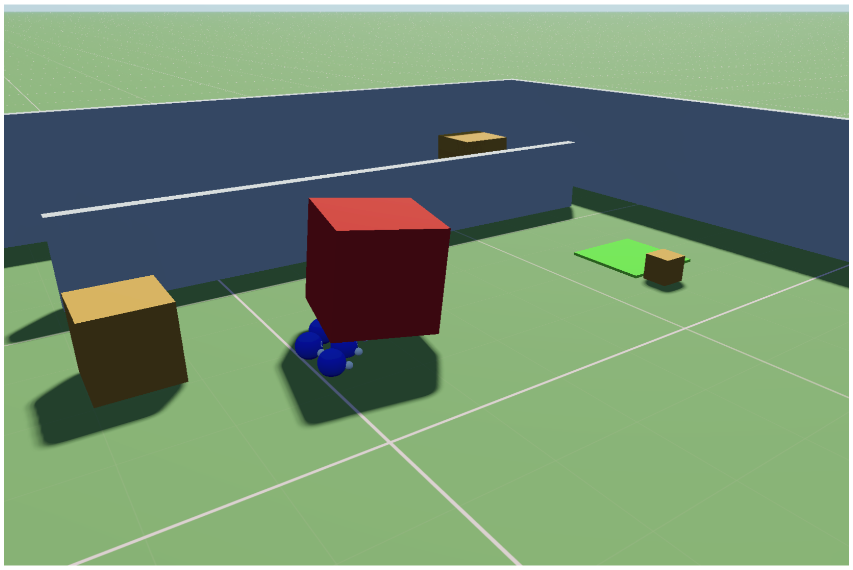

Figure 5: Agents in the non-recurrent model demonstrating an efficient strategy by scooping under the MacGuffin to reduce its friction.

This framework was designed to possess many characteristics making it uniquely suited to the real world. First, we explicitly test that models trained under the framework generalize well to scenarios with a new number of agents. When agents can lose connection to the controlling server or be destroyed (e.g. nano-bots or drones), the architecture must be robust to a dynamic number of agents. Additionally, by removing the command structure, or by repeatedly applying recent commands, we hypothesize that agents could be able to operate independently for short periods, without further input from the cloud. Finally, we note that the command structure is uniquely suited for dynamic input (e.g. natural language commands from a human) by inserting them into the command or message.

While we are hopeful that this framework could have substantial real-world impact in the future, there are current limitations. In our test environment, agents coordinated, but generally failed to navigate the MacGuffin around walls. We hypothesize this could be due to model size or training time.

Future work begins with experimentation with more dynamic factors is needed, testing how varying the number of agents within a single rollout affects performance. Additional testing includes "signal dropout," where commands are not transmitted for a short period of time. More broadly, we hope this framework inspires novel architectures in the messages-command framework, substituting alternative architectures for the "ants" and "hivemind."

7 Conclusion

In this work, we introduced HIVE, a flexible multi-agent message-pooling framework that combines shared-policy attention with a centralized LSTM memory module to enable dynamic team sizes and efficient inter-agent communication. Throughout experiments in a 2D cooperative transport environment, HIVE consistently outperformed a standard joint-state MLP baseline—achieving high success rates both at training scale (5 agents) and in zero-shot generalization to larger teams (8 agents). Our comparative analysis showed that both attention-based message aggregation and global memory are essential for sustaining coordinated behaviors in cluttered arenas. By replacing a simple existential time penalty with normalized progress and velocity-based stationary rewards, we encouraged agents to navigate around obstacles rather than merely rush toward the goal, resulting in more robust obstacle negotiation. While our current implementation did not explore signal dropout or varying agent counts within single rollouts, these remain promising directions for future research. Overall, HIVE represents a scalable, resource-efficient approach to multi-agent reinforcement learning, offering a clear path toward deployable swarm-like systems in robotics and beyond.

8 Team Contributions

All authors collaborated on the high-level problem formulation, model design, and the writing of all deliverables. In addition:

Ty Toney

- Adapted and implemented the Madrona simulation environment and training infrastructure.
- Conducted extensive debugging to ensure system stability.

Julian Allchin

- Set up and managed GPU-accelerated training infrastructure.
- Led hyperparameter tuning for model and algorithm configurations.

Diego Bustamante

- Designed and tuned the new reward function (normalized distance + stationary penalty).
- Built and integrated the logging pipeline for episode-level metrics and performance tracking.

Changes from Proposal While our core architectural plan (shared-policy attention with a centralized LSTM "hivemind") remained unchanged, we made two major departures from the original proposal. First, we redesigned the reward structure: instead of the proposed constant existential penalty, we adopted a normalized distance-reduction term and a velocity-based stationary penalty that each sum to one over the episode horizon, plus a sparse success bonus. This adjustment better incentivized agents to maintain motion around obstacles rather than simply rush toward the goal. Second, we extended our evaluation beyond the training-condition (5 ants) to test generalization to 8-agent teams—a scenario the baseline MLP cannot handle.

However, several planned experiments were deferred. We did not implement "signal-dropout" trials (randomly silencing the hivemind for brief intervals) nor blended varying agent counts within single rollouts, both of which remain avenues for future work. Additionally, while we intended a more extensive hyperparameter sweep over LSTM sizes and message-compression ratios, time and compute constraints limited us to a single configuration per model.

References

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. 2019. TarMAC: Targeted Multi-Agent Communication. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 1538–1546. `https://proceedings.mlr.press/v97/das19a.html`

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. `https://proceedings.neurips.cc/paper_files/paper/2016/file/c7635bfd99248a2cdef8249ef7bfbef4-Paper.pdf`

Shariq Iqbal and Fei Sha. 2019. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 2961–2970. `https://proceedings.mlr.press/v97/iqbal19a.html`

Brennan Shacklett, Luc Guy Rosenzweig, Zhiqiang Xie, Bidipta Sarkar, Andrew Szot, Erik Wijmans, Vladlen Koltun, Dhruv Batra, and Kayvon Fatahalian. 2023. An Extensible, Data-Oriented Architecture for High-Performance, Many-World Simulation. *ACM Trans. Graph.* 42, 4 (2023).

Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. 2016. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2016/file/55b1927fdafef39c48e5b73b5d61ea60-Paper.pdf