Learning Efficient Diverse Communication for Cooperative Heterogeneous Teaming – Supplementary

Esmaeil Seraj[1,*], Zheyuan Wang[1,*], Rohan Paleja[1,*], Daniel Martin[1], Matthew Sklar[1], Anirudh Patel[2], Matthew Gombolay[1]
[1]Georgia Institute of Technology, [2]Sandia National Laboratory
[1]Atlanta, GA, USA, [2]Albuquerque, NM, USA
{eseraj3,pjohnwang,rpaleja3,dmartin1,msklar3}@gatech.edu,anipate@sandia.gov,matthew.gombolay@cc.gatech.edu

ACM Reference Format:
Esmaeil Seraj[1,*], Zheyuan Wang[1,*], Rohan Paleja[1,*], Daniel Martin[1], Matthew Sklar[1], Anirudh Patel[2], Matthew Gombolay[1]. 2022. Learning Efficient Diverse Communication for Cooperative Heterogeneous Teaming – Supplementary. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 2 pages.

1 EVALUATION ENVIRONMENT: ADDITIONAL DETAILS AND PARAMETERS

Here we provide additional details regarding the employed evaluation environments for training and testing InfoPG and cover the associated environment parameters related to our experiments. For reproducibility, we publicly provide our code at `github.com/HetNet`[1].

1.1 Predator-Prey (PP) [7]

The objective within this homogeneous environment is for N predator agents with limited vision to find a stationary prey and move to its location. The agents in this domain are homogeneous in their state, observation, and action spaces and thus, all agents are of the same *class*. All agents are able to sense/observe the environment and each agent's observation is a concatenated array of the state vectors of all grids within the agent's Field of View (FOV). The predator agents' action-space is of dimension five, including cardinal movements and a null action, and is the same for all agents. Each predator agent will receive a small penalty per timestep until it has discovered the prey. A higher-performing algorithm in this domain is defined as one that minimizes the average number of steps taken by agents to complete an episode.

Within our evaluation, we evaluate in a grid size of 5x5 with 3 predators. We set the maximum steps for an episode to be 80. For the reward, each agent receives -0.05 per time step before they find the prey. An episode is considered unsuccessful if the prey is not discovered within the maximum steps.

1.2 Predator-Capture-Prey (PCP)

In our second domain, we have two classes of agents: *predator* agents and *capture* agents. The first class of agent, called the *predator* agents, have the goal of discovering the prey and have an action-space of dimension five, including cardinal movements and a null (stay) action. *Predator* agents have an observation space similar to the agents in PP domain. The second class of agents, called the *capture* agents, have the objective of locating the prey *and* capturing it. Capture agents differ from the predator agents in both their observation and their action spaces. Capture agents do not receive any observation inputs from the environment (i.e., no scanning sensors) and have an additional action of *capture-prey* in their action-space. This additional action must be used at a prey's location to capture the prey. Note that this domain is an explicit example of the perception-action composite teams, as introduced in Section 6.1. An episode is deemed successful once all agents have completed their class-specific objectives. Each *predator* agent is penalized with -0.05 reward every timestep until it has discovered the prey. Each *capture* agent is also penalized with -0.05 every timestep until it has captured (i.e., find the prey and then capture it) the prey. Note the difference in reward scheme, a *capture* agent may have discovered the prey but will receive a negative reward until the *capture-action* is utilized.

We utilize PCP as a testbed for several test heterogeneous interactions. In our head-to-head evaluation against baselines, we utilize a problem with two *predator* agents and one *capture* agent within a 5x5 grid. We set the maximum steps for an episode to be 80.

1.3 FireCommander (FC) [6]

We also evaluate the performance both our HetNet variants, HetNet-Binary and HetNet-Real, in a new cooperative multi-agent environment with heterogeneous agents, called FireCommander (FC) [5, 6]. FireCommander can be categorized as a strategic game, in which a composite team of robots (i.e., UAVs) must collaboratively find hidden areas of propagating wildfire and extinguish the fire in such areas as fast as possible. The robot team in FC is composed of two classes of agents: (1) *perception* agents (class P), which can only sense the environment and detect areas of fire and, (2) *action* agents (class A), which can only manipulate the environment by extinguishing a firespot which has already been detected by class P agents. Neither class P, nor class A agents are capable of accomplishing the task on their own, and therefore must communicate and collaborate.

Under the notations in our problem formulation in Section **??**, we have $C = \{\mathrm{P}, \mathrm{A}\}$ where, $\mathcal{A}^{(\mathrm{P})} = \{1, 2, \cdots, 4\}$ representing the four primitive motions and $\mathcal{A}^{(\mathrm{A})} = \{1, \cdots, 5\}$, representing the four primitive motions and an extra action which corresponds to extinguishing fire by dousing water. Agents of class P are equipped with fire detection sensors and can observe the environment, receiving an input vector of length 29 for each grid within their FOV. Agents of class A, do not receive any observation from the environment. The reward scheme in this domain includes a small temporal penalty of -0.1 per timestep for all agents, a false water-drop penalty of -0.1 for

[1]Available online at: `https://github.com/CORE-Robotics-Lab/HetNet`

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

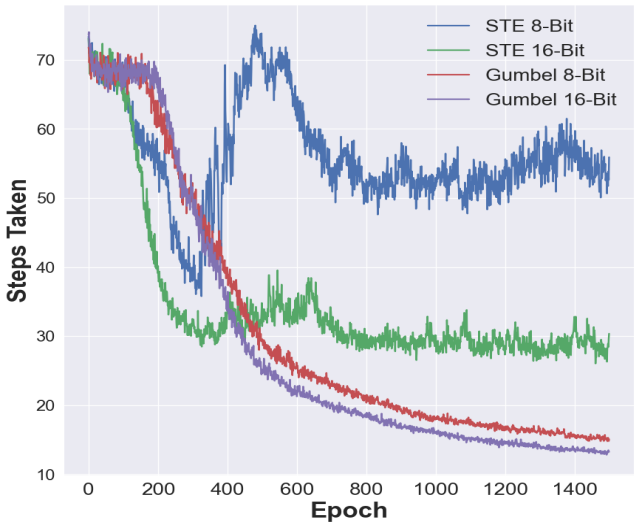

Figure 1: Performance comparison for two different binarization methods: (1) STE and (2) Gumbel-Softmax, for 8-bits and 16-bits message dimensions.

action agents, a -0.1 penalty per new firespot for all agents, and a positive reward of +10 for all agents per each extinguished firespot.

In our head-to-head evaluation against baselines, we utilize a problem with two *perception* agents and one *action* agent within a 5x5 grid and one initial firespot that propagates to a new location at each timestep, leaving the previous grid on fire. We set the maximum steps for an episode to be 300. An episode of the game is marked as successful only if all the active firespots within the map are discovered and extinguished. Please refer to the provided FireCommander supplementary document for further details.

2 SUPPLEMENTARY RESULTS AND ABLATION STUDIES

In this section, we provide our supplementary results. We first provide the implementation and model details for our experiments and then, present the results of an ablation study on the binarization process for digitizing the communication messages. For reproducibility, we publicly provide our code at `github.com/HetNet`[2].

2.1 Implementation and Model Details

For our empirical results, our HetNet implementation consists of three multi-head HetGAT layers stacked on top of the feature preprocessing modules. The first two multi-head layers use $L = 4$ attention heads computing 16 features each (for a total of 64 features merged by concatenation). The final layer also uses $K = 4$ attention heads, but the output dimension is set to the size of an agent's action-space and is merged by averaging. We used the Adam optimizer [3] through training with a learning rate of 10^{-3} for all our experiments and results presented here. We leverage the per-class critic architecture (chosen as a result of a sensitivity analyses detailed in 6.3.5) for all our experiments in Section 6. Algorithm 1 provides a pseudocode to train HetNet with the per-class critic architecture. The policy and critic network parameters are initialized per class (line 2) and the communication steps through HetGAT layers are performed at each step of an episode (line 8). The rest of training procedure implements an on-policy advantage AC procedure except that the gradient updates are class-specific (lines 15-20). We implement HetNet using PyTorch [4] and Deep Graph Library [8]. All of our experiments are performed across three random seeds (0, 1 and 2) and the presented results are averaged across all seeds.

[2]Available online at: `https://github.com/CORE-Robotics-Lab/HetNet`

2.2 Message Binarization Method: An Ablation Study

Learning a communication model that sends binary messages requires a differentiable approach to convert data from continuous-scale to a discrete (binarized) representation. We experiment with two different techniques that enable binarization: Straight-Through Estimators (STE) with a fixed threshold function [1] and Gumbel-Softmax [2]. The STE binarizes the activations of a layer during forward-pass and directly passes gradients similar to the identity function during backpropagation. The Gumbel-Softmax utilizes a differentiable sampling approach to produce binarized messages from a continuous distribution via categorical reparameterization. By analyzing the results of HetNet with each approach across multiple message dimension sizes (8 and 16 bit), we conclude that Gumbel-Softmax allows for better performance.

REFERENCES

[1] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR* abs/1308.3432 (2013). arXiv:1308.3432 http://arxiv.org/abs/1308.3432

[2] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[3] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[5] Esmaeil Seraj, Letian Chen, and Matthew C Gombolay. 2021. A hierarchical coordination framework for joint perception-action tasks in composite robot teams. *IEEE Transactions on Robotics* (2021).

[6] Esmaeil Seraj, Xiyang Wu, and Matthew Gombolay. 2020. FireCommander: An Interactive, Probabilistic Multi-agent Environment for Joint Perception-Action Tasks. *arXiv e-prints* (2020), arXiv–2011.

[7] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. 2018. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755* (2018).

[8] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).