Learning Emergent Discrete Message Communication for Cooperative Reinforcement Learning

Sheng Li[1] **Yutai Zhou**[2] **Ross Allen**[2] **Mykel J. Kochenderfer**[1]

[1]Aero/Astro Engineering, Stanford University, Stanford, CA, USA
[2]MIT Lincoln Lab, Lexington, MA, USA,

Abstract

Communication is a important factor that enables agents work cooperatively in multi-agent reinforcement learning (MARL). Most previous work uses continuous message communication whose high representational capacity comes at the expense of interpretability. Allowing agents to learn their own discrete message communication protocol emerged from a variety of domains can increase the interpretability for human designers and other agents.This paper proposes a method to generate discrete messages analogous to human languages, and achieve communication by a broadcast-and-listen mechanism based on self-attention. We show that discrete message communication has performance comparable to continuous message communication but with much a much smaller vocabulary size.Furthermore, we propose an approach that allows humans to interactively send discrete messages to agents.

1 INTRODUCTION

Communication allows agents to share information so that they can perform tasks cooperatively. There has been existing work on using deep reinforcement learning (RL) to produce communication protocols. For example, CommNet [Sukhbaatar and Fergus, 2016], a recurrent communication model, averages the hidden states for centralized communication. IC3Net [Singh et al., 2019], an extension on CommNet, adopts a more complicated but similar centralized aggregation approach to communication. Instead of centralized aggregation and averaging, TarMAC [Das et al., 2019] use multi-headed attention to distribute information to other agents. BiCNet [Peng et al., 2017] and ATOC [Jiang and Lu, 2018] both use a bidirectional recurrent network as a communication channel. They fix the positions of agents in the bidirectional recurrent network to specify their roles. DICG [Li et al., 2021] uses graph convolution to implicitly pass information between agents.

However, typically, existing multi-agent communication approaches use continuous messages to communicate. They use real-valued vectors to encode messages. Human languages, however, use discrete characters and words. An advantage of continuous messaging is its representational capacity, but it can be at the expense of interpretability from the perspective of human designers or other agents.

We propose a deep RL model for agents to learn to generate their own discrete message protocols. Our model produces discrete messages by identifying the maximum element in message vectors, resulting in greater stability than sampling. The model adopts a broadcast-and-listen procedure to send and receive messages. It uses self-attention mechanism [Cheng et al., 2016] to aggregate messages sent by other agents. The model is differentiable and therefore can be learned end-to-end. Evtimova et al. [2018] use bit-string messaging to learn emergent communication in referential games for two agents. Our approach is applicable to any number of agents.

We compare the performance of discrete message communication with continuous message communication in a variety of domains, showing that discrete message communication has comparable performance to continuous message communication with a much smaller vocabulary size. We also study the effects of communication bandwidth and vocabulary size on discrete message communication, using the metrics positive listening and positive signaling, where positive listening indicates received messages are influencing agents' behaviors in some way, and positive signaling indicates an agent is sending messages that are related in some way with its own observations or actions [Lowe et al., 2019, Jaques et al., 2019]. Furthermore, we propose an approach for human-agent interaction using discrete message communication, demonstrating its interpretability.

2 BACKGROUND

We represent the problem as a Dec-POMDP [Oliehoek and Amato, 2016] defined by the tuple $\langle \mathcal{I}, \mathcal{S}, \{\mathcal{A}^i\}_{i=1}^n, \mathcal{V}, \mathcal{T}, \mathcal{Z}, R, \mathcal{O}, \gamma \rangle$, where $\mathcal{I} = \{1, \ldots, n\}$ is the set of agents, $\mathcal{S}$ is the global state space, $\mathcal{A}^i$ is the action space of the ith agent, and $\mathcal{Z}$ is the observation space for an agent. The discrete communication vocabulary set is defined by $\mathcal{V} = \{0,1\}^b$, where b is the band width of communication. A message from $\mathcal{V}$ is therefore a binary vector. The transition function defining the next state distribution is given by $\mathcal{T} : \mathcal{S} \times \prod_i \mathcal{A}^i \times \mathcal{S} \to [0,1]$. The reward function is $R : \mathcal{S} \times \prod_i \mathcal{A}^i \to \mathbb{R}$, and the discount factor is $\gamma \in [0,1)$. The observation model defining the observation distribution from the current state is $\mathcal{O} : \mathcal{S} \times \mathcal{Z} \to [0,1]$. Each agent i has a stochastic policy π^i conditioned on its observations o_i. The discounted return is $G_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$, where r_t is the joint reward at step t. The joint policy π induces a value function $V^\pi(s_t) = \mathbb{E}[G_t \mid s_t]$ and an action-value function $Q^\pi(s_t, \mathbf{a}_t) = \mathbb{E}[G_t \mid s_t, \mathbf{a}_t]$, where $\mathbf{a}_t$ is the joint action. The advantage function is $A^\pi(s_t, \mathbf{a}_t) = Q^\pi(s_t, \mathbf{a}_t) - V^\pi(s_t)$.

2.1 POLICY OPTIMIZATION

We use policy optimization to maximize the expected discounted return. Given policy π_θ parameterized by θ, the surrogate policy optimization objective is [Schulman et al., 2017]:

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right], \tag{1}$$

where we use the generalized advantage estimation (GAE) [Schulman et al., 2016] to estimate $\hat{A}_t$ at time step t, and the expectation $\hat{\mathbb{E}}_t[\cdot]$ indicates the empirical average over a finite batch of samples. In practice, we use the clipped PPO objective [Schulman et al., 2017] to limit the step size for stable updates. The entropy bonus is also added to encourage exploration. The objective to maximize becomes

$$\begin{aligned} L_t^{\text{PPO}}(\theta) = \hat{\mathbb{E}}_t \Big[& \min\big(r_t(\theta)\hat{A}_t, \operatorname{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\big) \\ & + \beta H[\pi_\theta](s_t) \Big], \end{aligned} \tag{2}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, and $H[\pi_\theta](s_t)$ is the entropy of the policy given state s_t, the clipping parameter ε and the entropy coefficient β are hyperparameters. In the context of centralized training but decentralized execution, we employ a *parameter sharing* strategy whereby each agent in a homogeneous team uses identical copies of policy parameters [Gupta et al., 2017].

2.2 SELF-ATTENTION

Self-attention mechanism [Cheng et al., 2016] emerged from the natural language processing community. It is used to relate different positions of a single sequence. The difference between self-attention and standard attention is that self-attention uses a single sequence as both its source and target sequence. It has been shown to be useful in image caption generation [Liu et al., 2018, Yu et al., 2020] and machine reading comprehension [Cheng et al., 2016, Yu et al., 2018].

The attention mechanism has also been adopted recently for multi-agent reinforcement learning. The relations between a group of agents can be learned through attention. Iqbal and Sha [2019] use attention to extract relevant information of each agent from the other agents. Jiang and Lu [2018] use self-attention to learn when to communicate with neighboring agents. Wright and Horowitz [2019] use self-attention on the policy level to differentiate different types of connections between agents. Jiang et al. [2020] use multi-head dot product attention to compute interactions between neighboring agents for the purpose of enlarging agents' receptive fields and extracting latent features of observations. Li et al. [2021] use multiplicative self-attention to implicitly build coordination graphs by weighting the graph edges with attention weights.

We use self-attention to learn the attention weights between agents. The attention weights are used to differentiate the importance of messages in the public communication channel for each agent.

3 APPROACH

We use a broadcast-and-listen mechanism to achieve agent communication. Instead of building agent-pair specific communication channels, our model has a public 'chat room' to allow agents to share information. To selectively receive information, each agent differentiates the importance of messages from other agents by weighting them using attention weights. Then, agents aggregate the publicly broadcast messages using a weighted sum. The aggregated messages are concatenated with other agent-specific vectors to selection actions for agents. Fig. 1 shows the network architecture. It demonstrates the information flow between an agent pair, which can be easily vectorized for any number of agents.

In detail, we first pass n observations $\{o_i\}_{i=1}^n$ of n agents through a parameter sharing observation encoder f_o parameterized by θ_o. The observation encoder outputs observation embeddings $\{e_i\}_{i=1}^n$:

$$e_i = f_o(o_i; \theta_o), \text{ for } i = 1, \ldots, n. \tag{3}$$

We then compute the attention logits $\{c_i\}_{i=1}^n$ and the message logits $\{\mu_i\}_{i=1}^n$ with the parameter sharing attention encoder f_a parameterized by θ_a and the parameter sharing message encoder f_m parameterized by θ_m for all observation embeddings $\{e_i\}_{i=1}^n$:

$$c_i = f_a(e_i; \theta_a), \quad \mu_i = f_m(e_i; \theta_m) \text{ for } i = 1, \ldots, n. \tag{4}$$

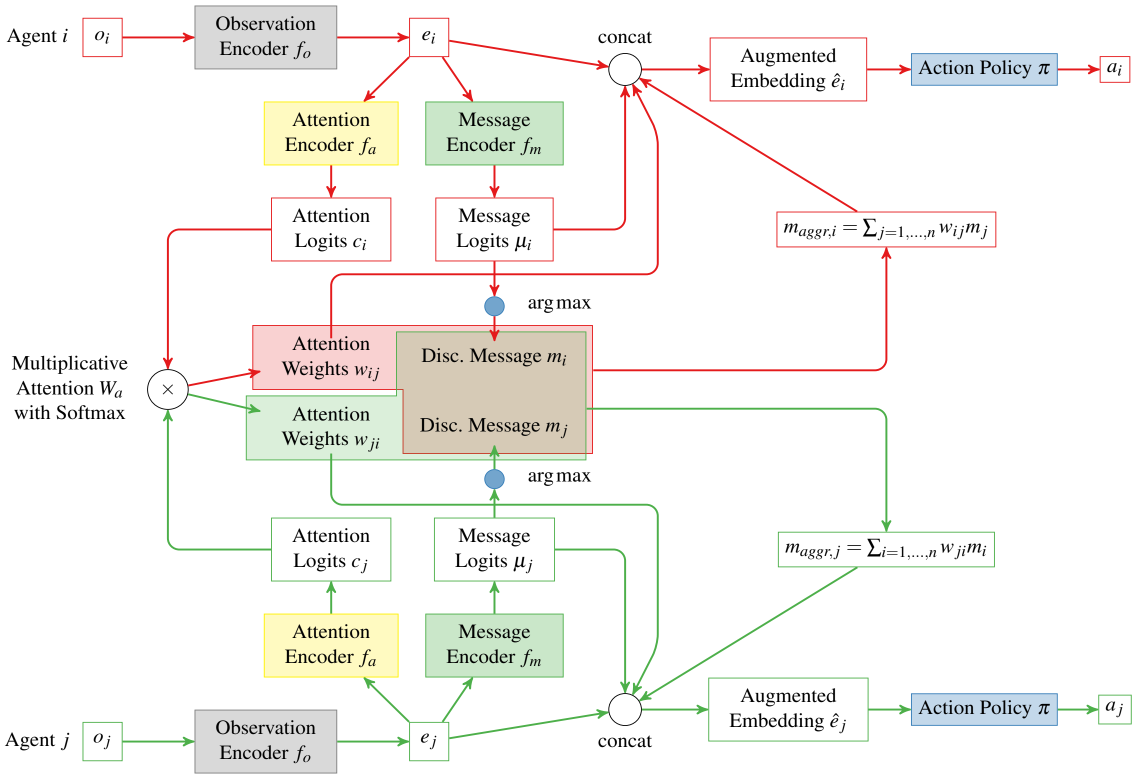

Figure 1: Network architecture

We use multiplicative attention [Luong et al., 2015] to compute the attention scores and then softmax to obtain the attention weights. For an arbitrary pair of agents i and j, the attention score s_{ij} and the attention weight w_{ij} of agent i towards agent j are:

$$s_{ij} = c_j^\top W_a c_i \qquad w_{ij} = \frac{\exp(s_{ij})}{\sum_{k=1}^{n} \exp(s_{ik})}. \tag{5}$$

The multiplicative attention operation is parameterized by a square matrix W_a. The attention scores and weights can be efficiently computed for all i, j pairs (including $i = j$, i.e. self-attention weights).

Instead of sampling from distributions, we use the arg max operation to extract discrete messages from message logits for the agents to broadcast. We adopt two approaches to formulate discrete messages: one-hot and bit-string. For a given message bandwidth b:

- *One-hot*: with a message logit vector $\mu_i \in \mathbb{R}^b$, the kth entry of $m_{\text{one-hot},i} \in \{0,1\}^b$ is given by

  $$m_{\text{one-hot},ik} = \begin{cases} 1 & \text{if } k = \arg\max_{l=1,\dots,b} \mu_{il}, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

  The resulting vocabulary size $|\mathcal{V}|$ of one-hot encoded messages is b. An example of an one-hot encoded message with bandwidth $b = 5$ can be $m_{\text{one-hot}} = [0,0,1,0,0]$.

- *Bit-string*: we first need a message logit vector $\mu_i \in \mathbb{R}^{2b}$, the kth entry of $m_{\text{bit-string},i} \in \{0,1\}^b$ is given by

  $$m_{\text{bit-string},ik} = \begin{cases} 1 & \text{if } k = \arg\max_{l \in \{k,k+b\}} \mu_{il}, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

  The resulting vocabulary size $|\mathcal{V}|$ of bit-string encoded messages is 2^b. An example of a bit-string encoded message with bandwidth $b = 5$ can be $m_{\text{bit-string}} = [1,0,0,1,1]$.

With the messages m_i and attention weights w_{ij}, agents can aggregate the messages: $m_{aggr,i} = \sum_{j=1}^{n} w_{ij} m_j$.

For agent i, we then concatenate the observation embedding e_i, the attention weights $w_i = \{w_{ij}\}_{j=1}^{n}$, the message logits μ_i and the aggregated message $m_{aggr,i}$ to form an augmented embedding $\hat{e}_i = [e_i; w_i; \mu_i; m_{aggr,i}]$.

The concatenation creates several skip connections in the computation graph. They compensate for the gradient cut-off at the arg max operation and boost the gradients of the network components closer to the input head. The augmented embedding $\hat{e}_i$ is then passed through a parameter sharing action policy π parameterized by θ_π to infer action

$$a_i = \pi(\hat{e}_i; \theta_\pi). \tag{8}$$

Continuous message communication can be achieved by circumventing the arg max operation in the network architecture and use message logits μ_i as the message to broadcast.

In summary, our model uses one round of communication, making the representational capacity of the communication protocols especially important.

4 EXPERIMENTS

We perform experiments and analysis for discrete message communication by (1) analyzing the effects of bandwidth and vocabulary size and comparing with continuous message communication; (2) analyzing the importance of self-attention for discrete message communication; (3) analyzing positive listening and signaling; and (4) introducing human interaction with agents by using discrete message communication.

We show the results obtained from three environments: Pulling Levers [Sukhbaatar and Fergus, 2016], Predator-Prey [Böhmer et al., 2020, Li et al., 2021], and Multi-Walker [Gupta et al., 2017, Terry et al., 2020]. These environments have challenging tasks that must be performed cooperatively to achieve high returns. Using only local information cannot achieve high returns.

We use PPO [Schulman et al., 2017] for policy optimization (see Section 2.1). A multi-layer perceptron (MLP) baseline (value function) with global state is used for reducing the variance of advantage estimation with GAE [Schulman et al., 2016] during training.

4.1 ANALYSIS FOR BANDWIDTH AND VOCABULARY SIZE

We compare the performance of communication protocols with various bandwidth and vocabulary size. The metric used is the average return. Average return indirectly measures the effects of communication on behaviors (i.e. positive listening). Continuous message communication is used as a baseline.

4.1.1 Pulling Levers

Pulling Levers [Sukhbaatar and Fergus, 2016] is a simple multi-agent multi-armed bandit task that requires communication to achieve high a score. There are n levers in the task. In each round, n agents are randomly sampled from a total of N participants. The action of each agent is to select a lever to pull. The reward of one round is given by the ratio between the number of unique levers pulled by the agents and the number of levers n. The maximum reward per round is therefore 1, meaning each pulled lever is unique in that step. The theoretical expected reward of randomly pulling levers is $1-(\frac{n-1}{n})^n$. The observation o_i for each agent is the one-hot encoding of its ID number. Choosing levers without communication in this game will be similar to acting randomly.

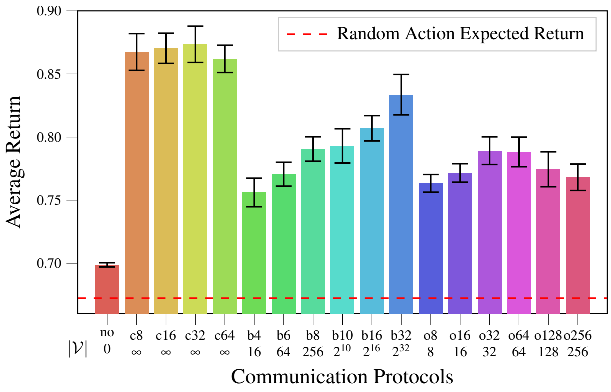

Figure 2: Returns of Pulling Levers (5 levers, 20 total participants) with various communication protocols.

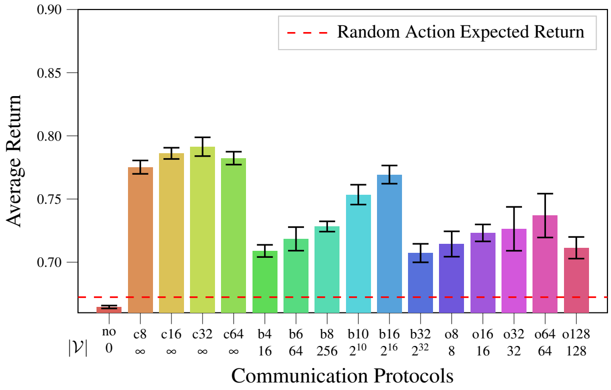

Figure 3: Returns of Pulling Levers (5 levers, 100 total participants) with various communication protocols.

A *zero baseline* (value function) is used for training the pulling levers environment, since the global state (observation) is not related to return. Each episode consists of 50 rounds, and reward for one episode is summed over rewards from each round. Results are averaged over 5 random seeds.

Fig. 2 shows the normalized average returns of various communication protocols with $n=5$ and $N=20$. The communication protocols and their vocabulary sizes $|\mathcal{V}|$ are labeled on the horizontal axis: 'no' stands for no communication, 'c' means continuous message, 'b' means bit-string, 'o' means one-hot, and the number following the letter is the bandwidth b of communication. A red dashed line is over-plotted to indicate the expected return with random action.

From Fig. 2, No communication shows the performance close to acting randomly. Continuous message works the best. The performance differences between continuous message bandwidths are not significant, since continuous message with any bandwidth can effectively encode an infi-

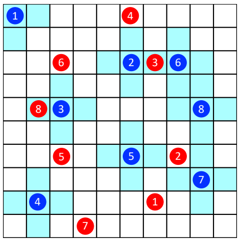

Figure 4: Predators are marked in blue, and prey are marked in red. The cyan grids are the capture range of predators. An example of successful capture is predator 2 and 6 capturing prey 3. An example of a single-agent capture attempt that results in a penalty is predator 3 capturing prey 8 alone.

nite number of meanings. Discrete message communication shows inferior performance due to its limited representational capacity. Generally, the performance of discrete message communication increases when the vocabulary size increases. This is particularly true for bit-string encoding. The 32-bit-string works the best among discrete communication, since its vocabulary size is $2^{32} \approx 4 \times 10^9$. This size is beyond human's cognitive range. The average vocabulary size of native English speakers is around 20000, while 6000 to 7000 are sufficient for understanding most communication [Rosenberg and Tunney, 2008]. The vocabulary sizes of the one-hot messages are within a human's cognitive range, and their performance are not too far off comparing with continuous and bit-string messaging. The performance of one-hot messaging drops as the bandwidth increases beyond 128. This result indicates that overly large bandwidths are harmful for learning emergent communication protocols.

Fig. 3 shows the normalized average returns for $n = 5$ and $N = 100$. The trend of the results is consistent with $N = 20$.

4.1.2 Predatory-Prey

We use a Predatory-Prey environment similar to that described by Böhmer et al. [2020] and Li et al. [2021]. The environment consists of a 7×7 grid world with 4 predators and 4 prey. We control the movement of predators to capture prey. The prey move by hard-coded and randomized rules to avoid predators. If a prey is captured, the agents receives a reward of 10. However, the environment penalizes any single-agent attempt to capture prey with a negative reward -0.5; at least two agents are required to be present in the neighboring grid cells of a prey for a successful capture. Fig. 4 illustrates the environment and the reward mechanism. The agents have a vision range of 2 grids from itself. We engineer the agent's observation so that we can remove other agents positions away from an agent's field of view. That means an agent can now only see the prey but not the other agents. The combination of agent invisibility and the single-agent capture-attempt penalty makes the task even harder. Cooperation is necessary to achieve a high return in this environment. We set the episode length to 50 steps, and impose a step cost of -0.1.

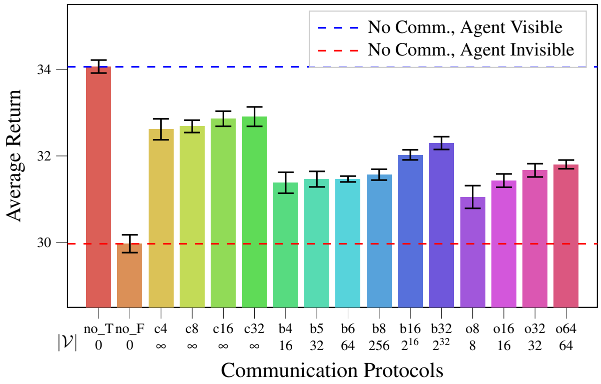

Figure 5: Returns of the 4-agent Predator-Prey with various communication protocols.

The results are shown in Fig. 5. Results are averaged over 5 random seeds. We ran experiments for various communication protocols. In particular, no communication **with** agent visibility (`no_T`) and no communication **without** agent visibility (`no_F`) form the performance upper-bound and lower-bound respectively. In the former setting, each agent has the richest information about the other agents, whereas in the latter setting, each agent has the scarcest information. Agents are set to invisible for all the other communication protocols. The dashed lines mark the upper and lower bounds. (The labels on the horizontal axis follow the convention defined in Section 4.1.1 and Fig. 2.)

The performance of all the communicative methods falls within the upper and lower bounds. We can see similar increasing patterns for increasing bandwidth as in Section 4.1.1 for Pulling Levers, which indicates that richer information transmission comes with higher bandwidth (up to a certain limit). Continuous messaging performs the best, followed by bit-string messaging. Bit-string messaging matches continuous messaging performance as its vocabulary size increases to 2^{32}. One-hot messaging is slightly better than bit-string messaging when their vocabulary sizes are same. All the discrete communication protocols outperform no communication by a large margin.

4.1.3 Multi-Walker

The previous two domains consist of discrete observation and action spaces. We would like to examine the consistency of discrete message communication in a more challenging continuous observation and action space task. In the Multi-Walker environment [Gupta et al., 2017, Terry et al., 2020], n bipedal robot agents try to carry a bar-shaped package and move forward as far as possible as illustrated in Fig. 6. The agents receive positive rewards for moving forward and negative reward for moving backward. Large negative

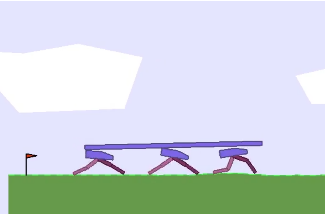

Figure 6: An illustration of the Multi-Walker environment with 3 agents.

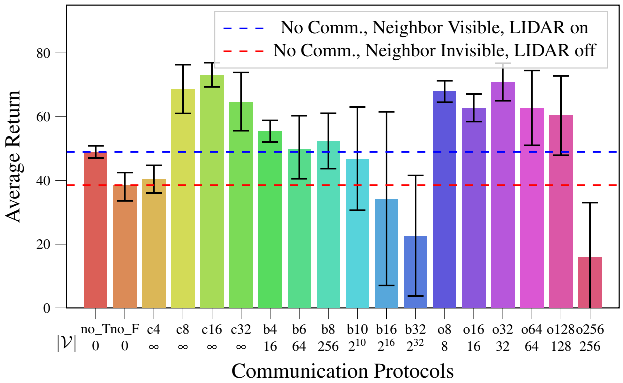

Figure 7: Returns of the 3-agent Multi-Walker with various communication protocol.

rewards are given if the an agent falls or the package falls. The task is a combination of learning robotic locomotion and inter-agent cooperation.

In the original observation design, each agent receives an observation composed of physical properties of its legs and joints, as well as LIDAR readings from the space immediately in front and below the robot. The original observation also includes information about neighboring walkers, and the package. To emphasize the necessity of communication, we engineer the observation space to remove LIDAR readings and information about neighboring walkers. *The resulting engineered observation therefore is only composed of the agent's own physical properties and information about the package.* The information about the other agents is now only accessible from communication.

The results are shown in Fig. 7 for 3 agents averaged over 5 random seeds. We ran experiments for various communication protocols. In particular, we ran experiments with no communication **with** neighbor visibility and LIDAR turned on (original observation configuration, labeled with `no_T`), and also no communication **without** neighbor visibility and LIDAR (observation configuration with scarcest info, labeled with `no_F`).

The results show observe that communicative methods *outperform* the original observation configuration `no_T`. This can be explained by the fact that `no_T` only has neighbor information available for each individual agent, while communicative methods can pass the information of non-neighboring agents to each other, providing a wider perception field. Continuous messaging with low bandwidth (`c4`) has poor performance. Continuous messaging with larger bandwidth (≥ 8) has much better performance, within which `c16` performs best. Bit-string messaging performs worse than continuous messaging and one-hot messaging. In contrast with Pulling Levers and Predator-Prey, the performance of bit-string messaging deteriorates when the bandwidth or vocabulary size increases. And the variance in performance also greatly increases accordingly. One-hot messaging has similar performance as continuous messaging in Multi-Walker, among which `o32` performs the best. The performance dramatically drops (below `no_F`) as the bandwidth increases to 256. Similar to Pulling Levers, overly large bandwidths are harmful for learning emergent communication protocols.

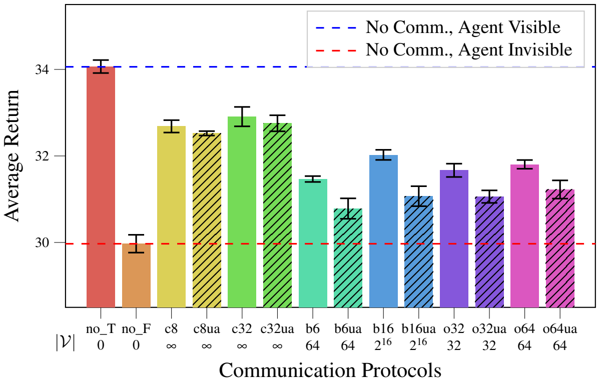

Figure 8: Average return comparison between self-attention and uniform attention from the 4-agent Predator-Prey. Uniform attention bars are shaded and labeled with `ua` suffix.

4.2 ABLATION WITH SELF-ATTENTION

In this analysis, we try to see the contribution of the self-attention to message aggregation. We run self-attention ablation experiments in the 4-agent Predator-Prey environment (the same as Section 4.1.2). In Fig. 8, we show the performance comparison of communication protocols with self-attention (non-shaded) and their counterparts with uniform attention (shaded and with the `ua` suffix). The results are averaged over 5 random seeds. Uniform attention means that we circumvent the multiplicative attention operation and assign equal attention weight $1/n$ to each agent.

We observe that uniform attention poses a much more significant negative performance impact on the discrete messaging than on the continuous messaging. Possible explanations are that continuous messages can make up for the difference in attention weights by messages themselves, but different bit-string messages can have drastically different meanings

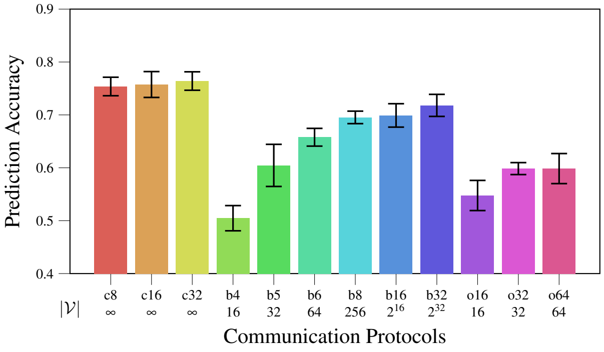

Figure 9: Positive listening analysis on the 4-agent Predator-Prey environment: predicting agent i's actions a_i using the masked aggregated messages $\bar{m}_{aggr,i}$ received by agent i. Data are collected from evaluation trajectories of five random runs.

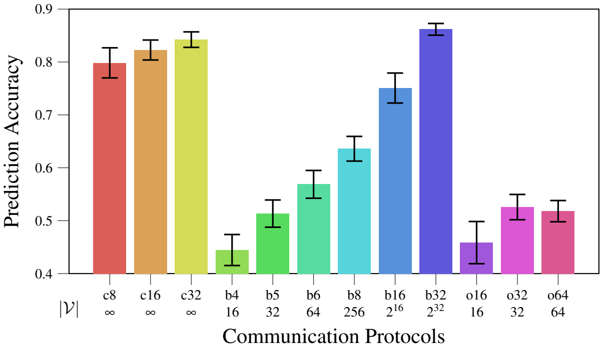

Figure 10: Positive signaling analysis on the 4-agent Predator-Prey environment: predicting agent i's actions a_i using the message m_i sent by agent i. Data are collected from evaluation trajectories of five random runs.

and their limited representational power cannot replace the contributions of learned self-attention weights.

4.3 POSITIVE LISTENING AND SIGNALING OF COMMUNICATION

We design supervised learning tasks [Li et al., 2021] to measure positive listening and signaling of communication.

4.3.1 Positive Listening

Positive listening quantifies the degree to which received messages are influencing an agent's behaviors [Lowe et al., 2019, Jaques et al., 2019]. It can be measured with a supervised learning task by using the masked aggregated message $\bar{m}_{aggr,i}$ *received* by agent i to predict agent i's actions. The masking is done to remove agent i's contribution to the aggregated message.

To compute the masked aggregated message $\bar{m}_{aggr,i}$, we first mask the self-attention weight w_{ii} to zero $w_{ii} = 0$, then re-normalize the attention weights $\bar{w}_{ij} = w_{ij} / \sum_{k=1,\dots,n} w_{ik}$, and finally compute the masked aggregated message using the re-normalized attention weights $\bar{m}_{aggr,i} = \sum_{j=1,\dots,n} \bar{w}_{ij} m_j$.

We formulate the supervised learn task as $\hat{a}_i = f(\bar{m}_{aggr,i}; \phi)$ with loss $L = \text{CrossEntropy}(\hat{a}_i, a_i)$ [Li et al., 2021]. Theoretically, with a finite amount of data, if $m_{aggr,i}$ is more correlated with a_i, i.e. if positive listening is strong, the classifier $f(\cdot; \phi)$ can produce a higher action prediction accuracy. We use a simple multi-layer perceptron (MLP) classifier parameterized by ϕ, with a single 128-unit hidden layer and ReLU as activation.

Results of five random runs from Predator-Prey are shown in Fig. 9. Environment configuration is the same as that in Section 4.1.2. In general, higher bandwidths or larger vocabulary size can bring a stronger positive listening for all communication protocols. In terms of communication bandwidth, continuous messaging shows the best positive listening, closely followed by bit-string messaging. Bit-string messaging exhibits potential to yield positive listening on par with continuous messaging. In terms of the vocabulary size instead of the bandwidth, one-hot messaging has a higher action prediction accuracy than bit-string messaging at low vocabulary sizes ($|\mathcal{V}| \leq 32$).

A high action prediction accuracy is achieved because the broadcast-and-listen architecture allows individual agents to learn other agents' behaviors and intentions through messages they sent.

4.3.2 Positive Signaling

Positive signaling quantifies the degree to which an agent's sent messages are related to its observations or actions [Lowe et al., 2019, Jaques et al., 2019]. It can be measured with a supervised learning task by using the message m_i *sent* by agent i to predict agent i's actions.

Similar to positive listening, we formulate the supervised learn task as $\hat{a}_i = f(m_i; \phi)$ with the same cross entropy loss and network architecture as used for positive listening. With a finite amount of data, if m_i is more correlated with a_i, i.e. if positive signaling is strong, the classifier $f(\cdot; \phi)$ can produce a higher action prediction accuracy.

Results of five random runs from Predator-Prey are shown in Fig. 10. Continuous messaging shows the strongest positive signaling, while one-hot messaging shows the weakest positive signaling. Bit-string messaging shows large gaps between continuous messaging when communication bandwidth is small. The gap decreases as the bandwidth increases. Bit-string messaging outperforms continuous messaging when bandwidth reaches 32. One-hot messaging under performs bit-string messaging by a large margin.

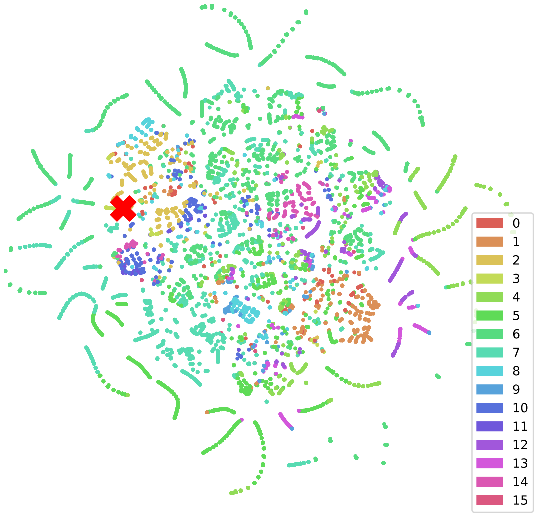

Figure 11: T-SNE clustering of agent observations labeled by their corresponding messages (4-agent Predator-Prey with `b4`). The red cross represents an example of the projection of a new observation.

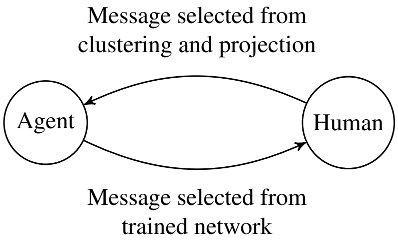
Message selected from clustering and projection
Agent
Human
Message selected from trained network

Figure 12: Human-agent interaction.

4.4 HUMAN INTERPRETATION AND INTERACTION

Discrete messaging has a finite vocabulary size that human can easily interpret. We design a protocol to allow a human to interactively send messages to AI agents.

First, we collect raw observation and message pairs from trained agents. We cluster the raw observations using t-SNE [van der Maaten and Hinton, 2008, Poličar et al., 2019]. Then, we label the raw observations with their corresponding discrete messages. Fig. 11 shows such clustering and labeling from Predator-Prey with `b4` (with a vocabulary size of $2^4 = 16$). Different colors in the clustering plot means different discrete messages. We use this clustering plot as a reference. We observe that the same cluster of raw observations tend to have the same discrete message labels. This provides an entry point for human-agent interaction. We can project new observations into the pre-known reference clusters as shown by the red cross in Fig. 11, and empirically select the most probable messages to send to agents. Fig. 12 shows an outline for the human-agent interaction workflow.

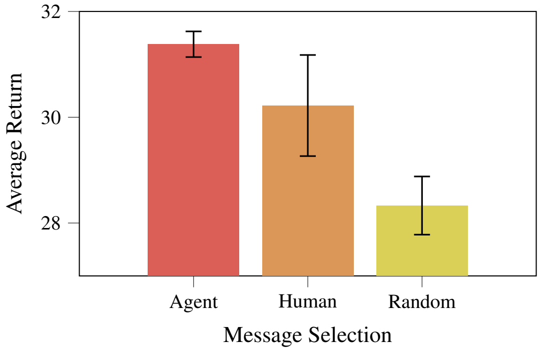

Figure 13: Human-agent interaction results and comparison for Predator-Prey with `b4`.

We design an experiment for letting human to select messages for **one** agent, with the rest of the agents selecting messages using the trained network (actual human-agent interaction experiments have text labels besides colors on the reference clustering points to assist in message selection). We compare the performance with agent selecting messages and random messages for one agent. Fig. 13 shows the results for the 4-agent Predator-Prey with 4-bit-string messaging (vocabulary size $2^4 = 16$). Environment configuration is the same as that in Section 4.1.2. Agent and random selection are averaged over 200 episodes and human selection is averaged over 20 episodes. We can see human message selection has a bit worse performance than agent selection. The performance gap arises from the empirical prediction error when a human selects messages from their estimate using clustering and projection. Human message selection outperforms random selection.

In summary, the human-agent interaction protocol described in this analysis shows the interpretability of discrete messaging and demonstrates a way to integrate both human and AI agents.

5 CONCLUSIONS

In this work, we present a broadcast-and-listen model that enables end-to-end learning of emergent discrete message communication. We demonstrate that the bandwidth and the vocabulary size of discrete messaging affects its performance. In some domains, discrete message communication can yield return performance and positive listening and signaling on par with or exceeding continuous message communication. Since discrete messages are easier to interpret by humans, we propose a human-agent interaction protocol that allows human to send discrete messages to agents. For future work, we would like to try multi-headed attention [Vaswani et al., 2017] for the information aggregation process and new metrics other than average returns to more directly measure communication capabilities [Lowe et al., 2019].

6 ACKNOWLEDGEMENTS

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

References

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning (ICML)*, pages 980–991. PMLR, 2020.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 551–561. Association for Computational Linguistics, 2016.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning (ICML)*, pages 1538–1546. PMLR, 2019.

Katrina Evtimova, Andrew Drozdov, Douwe Kiela, and Kyunghyun Cho. Emergent communication in a multi-modal, multi-step referential game. In *International Conference on Learning Representations (ICLR)*, 2018.

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 66–83. Springer, 2017.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 2961–2970. PMLR, 2019.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 3040–3049. PMLR, 2019.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7254–7264, 2018.

Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Sheng Li, Jayesh K Gupta, Peter Morales, Ross Allen, and Mykel J Kochenderfer. Deep implicit coordination graphs for multi-agent reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2021.

Xihui Liu, Hongsheng Li, Jing Shao, Dapeng Chen, and Xiaogang Wang. Show, tell and discriminate: Image captioning by self-retrieval with partially labeled data. In *European Conference on Computer Vision (ECCV)*, pages 338–354, 2018.

Ryan Lowe, Jakob Foerster, Y-Lan Boureau, Joelle Pineau, and Yann Dauphin. On the pitfalls of measuring emergent communication. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421. Association for Computational Linguistics, 2015.

Frans A Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

Pavlin G. Poličar, Martin Stražar, and Blaž Zupan. openTSNE: a modular python library for t-SNE dimensionality reduction and embedding. *bioRxiv*, 2019. doi: 10.1101/731877.

Jeremy Rosenberg and Richard J. Tunney. Human vocabulary use as display. *Evolutionary Psychology*, 6(3): 147470490800600318, 2008.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *International Conference on Learning Representations (ICLR)*, 2019.

Sainbayar Sukhbaatar and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2244–2252, 2016.

Justin K Terry, Benjamin Black, Mario Jayakumar, Ananth Hari, Luis Santos, Clemens Dieffendahl, Niall L Williams, Yashas Lokesh, Ryan Sullivan, Caroline Horsch, and Praveen Ravi. Pettingzoo: Gym for multi-agent reinforcement learning. *arXiv preprint arXiv:2009.14471*, 2020.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Matthew A Wright and Roberto Horowitz. Attentional policies for cross-context multi-agent reinforcement learning. *arXiv preprint arXiv:1905.13428*, 2019.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations (ICLR)*, 2018.

J. Yu, J. Li, Z. Yu, and Q. Huang. Multimodal transformer with multi-view visual representation for image captioning. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(12):4467–4480, 2020.