CS8803 – Deep RL

FALL 2024

Learning to Communicate in Multi-Agent Environments

Presenters: Mengying Lin, Nhi Nguyen, Wesley Ford, Nathan Chung (Group 12)

Instructor: Animesh Garg

Georgia Tech

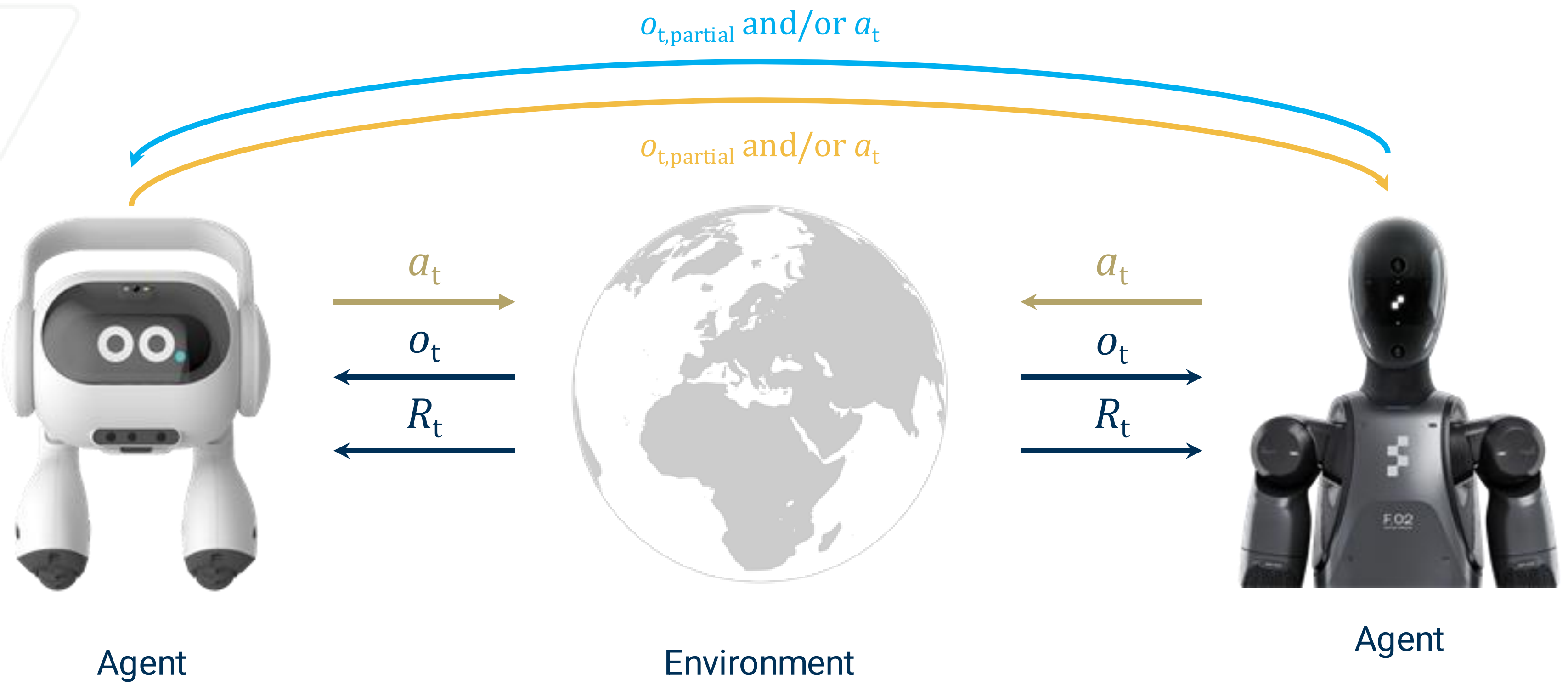
Motivation

Georgia Tech

Motivation and Main Problem

Robots: LG Bipedal Robot (Left), Figure 02 (Right)

Motivation and Main Problem

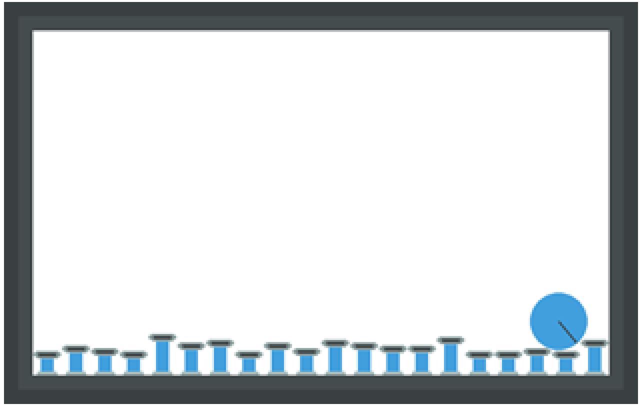


Robots: LG Bipedal Robot (Left), Figure 02 (Right)

Methodology

Methodology

- We aim to investigate the impact of communication strategies and see how it impacts the performance overall. We decided to choose the **Pistonball environment** for its simplicity and abstract settings to focus on the coordinations performance.



- We chose **PPO** as the baseline to train multi-agents policy because it's a simple yet effective method. It also has been successfully used with competitive Multi-Agent Racing.

Problem Settings

Definitions and notations

The Pistonball environment can be modeled as a Markov Decision Process (MDP).

Environment: Markov Decision Process (MDP)

(S, A, P, r, ρ0, γ)

- A finite set of actions
- S finite set of states
- P : S x A x S → $\mathbb{R}$
- r : S → $\mathbb{R}$
- ρ0 : S → $\mathbb{R}$
- γ ϵ (0, 1) (discount factor)

Policy: target to maximize the value function $V^{\pi}(s)$

$$V^{\pi}(s) = E_{\pi}[\sum_{t=0}^{inf} \gamma^{t} r_{t+1} | s_t = s]$$

Network: Actor-Critic

- Actor: output $\pi_{\theta}(a|s)$ to choose next action.
- Critic: predict $V^{\pi}(s)$

Georgia Tech

Methodology

Intuition

- Agents each use the observed state space to generate some emission that is used to augment the input to every agent's policies.

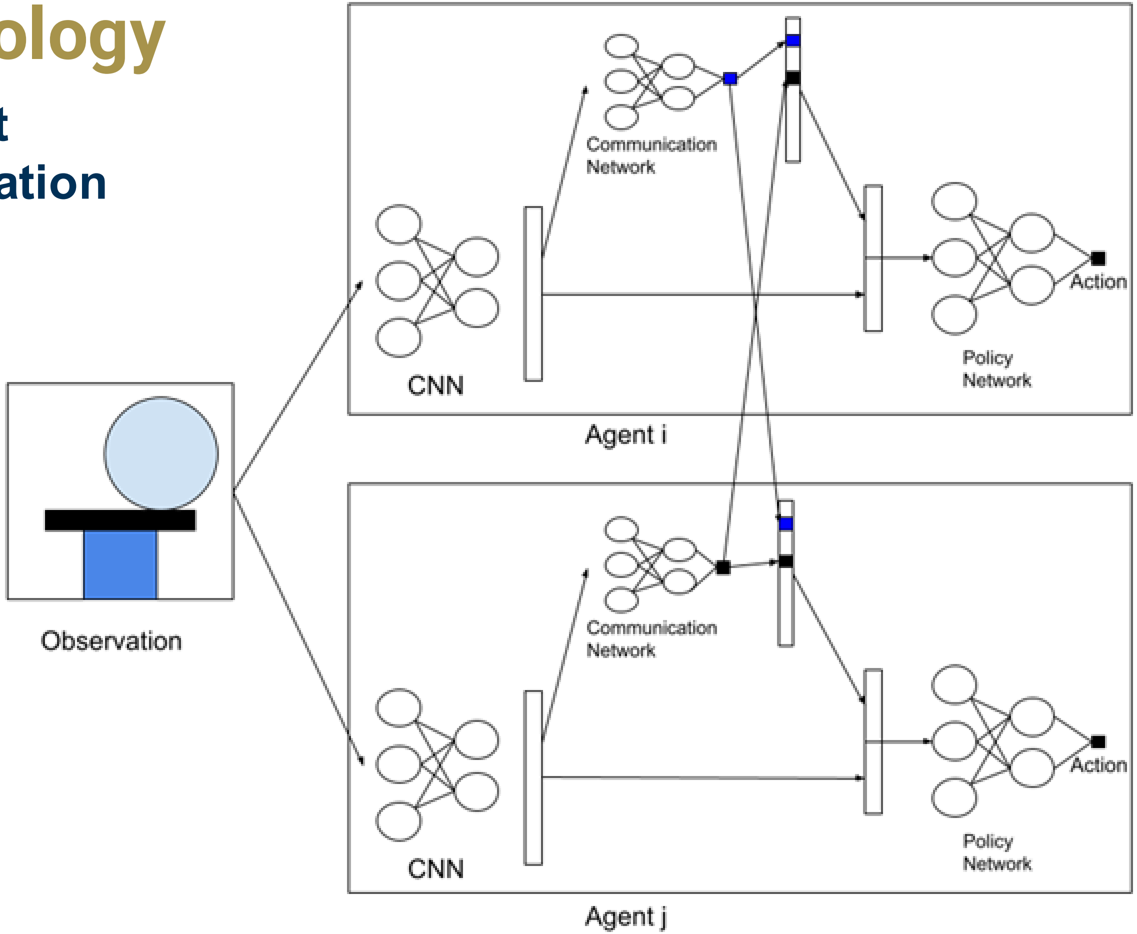
- Using a separate Neural Network, agents condense information from their observation into a single "communication" scalar. Each agent's communicatio scalar is shared with every other agent via a communication vector.

- Simple implementation and does not require much modification to the agents network and policy structure.

→ Multi-Agent Communication Vector

Methodology

Multi-Agent Communication Vector



Georgia Tech

Methodology

Intuition

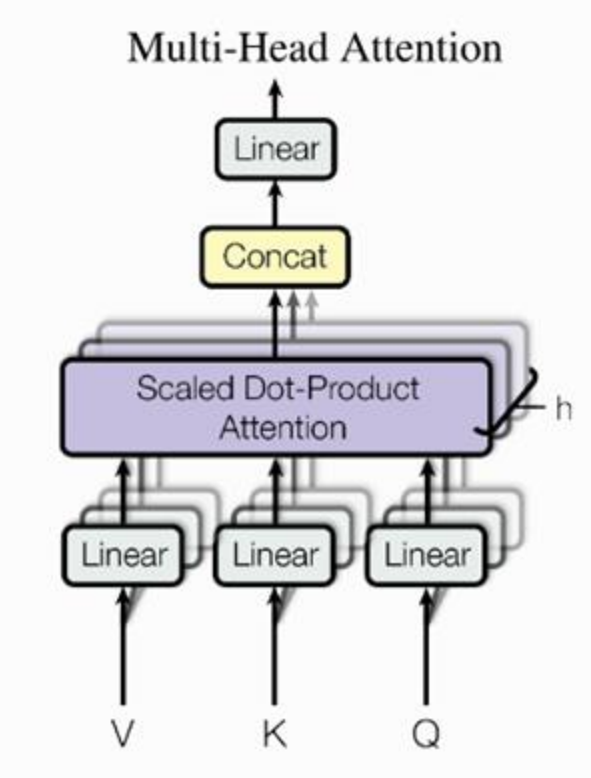
- In multi-agents settings, agents must coordinates their actions to effectively complete the goal.
- Agents are receiving informations from multiple sources, it needs to cherry pick the suitable ones for the moment decision-making.
- As the number of agents increases, redundant information can lead to overload info and suboptimal performance.

→ Multi-head attention for communication processing

Georgia Tech

Methodology

Multi-head attention for communication processing



$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\mathrm{Multihead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)W^O$$
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Methodology

Algorithm 1 PPO with Multi-Agent Communication Vector

procedure PPO-MULTI-AGENT
  Initialize policy, value and communication networks θ, ϕ, ψ for each agent A_i.
  Initialize communication vector c_i for all agents.
  for each training iteration **do**
    for each environment step **do**
      for each agent A_i in $\{A_1, \ldots, A_N\}$ **do**
        Collect local observation s_i.
        Compute communication scalar $c_i = \pi_\psi(s_i)$
        Compute communication vector by aggregating neighbors:

$$C_i \leftarrow \text{Aggregate}(\{c_j\}_{j \in \mathcal{N}(i)})$$

        Form augmented observation: $\tilde{s}_i = [s_i, C_i, I_i]$, conditioned on agent id: I_i
        Select action $a_i \sim \pi_\theta(a_i|\tilde{s}_i)$.
      end for
      Execute joint action $a = \{a_1, \ldots, a_N\}$ and observe rewards.
    end for
    Update θ and ϕ using PPO.
  end for
end procedure

Algorithm 2 PPO with Multi-Head Attention Communication

procedure PPO-MULTI-AGENT
  for each training iteration **do**
    for each environment step **do**
      for each agent $A_i \in \{A_1, \ldots, A_N\}$ **do**
        Collect local observation s_i;
        Extract hidden features: $h_i \leftarrow \text{Network}(s_i)$
        Concatenate hidden features with agent ID:

$$c_i \leftarrow \text{comm\_input}_i \leftarrow \text{Concat}(h_i, \text{id}_i)$$

        Compute communication vector using multi-head attention:

$$c_i \leftarrow \text{Attention}(\{c_j\}_{j \in \mathcal{N}(i)})$$

        Form augmented observation: $\tilde{s}_i = [s_i, c_i]$
        Select action: $a_i \sim \pi_\theta(a_i|\tilde{s}_i)$
      end for
      Execute joint action $a = \{a_1, \ldots, a_N\}$; observe rewards
    end for
    Optimize policy θ and value function ϕ using PPO
  end for
end procedure

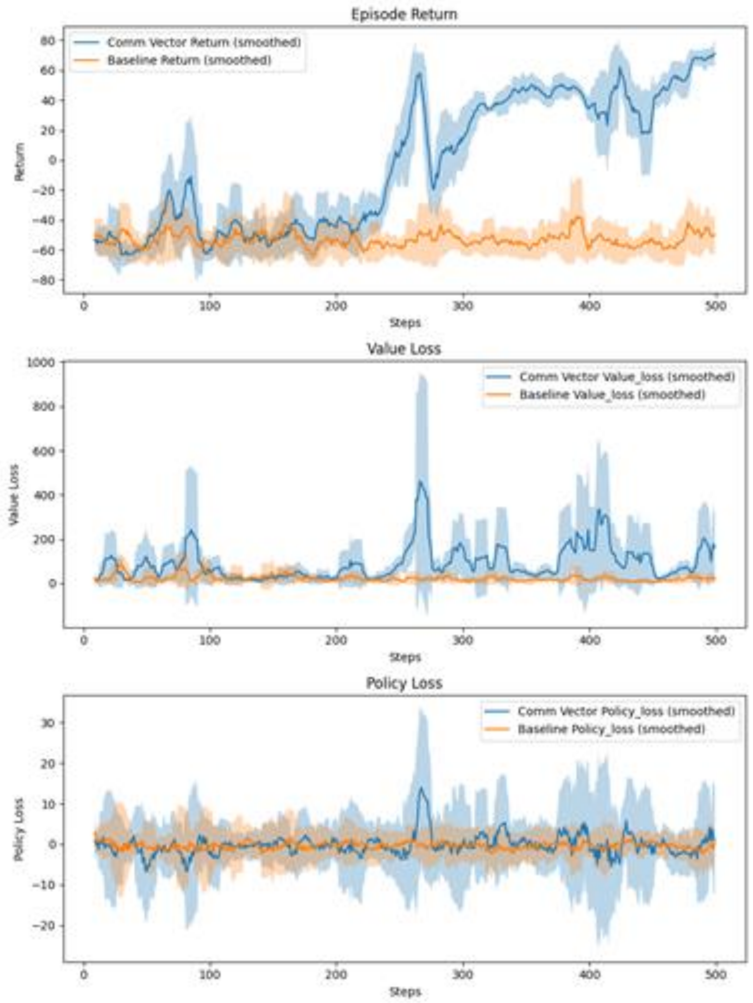
Experimental Results

Experimental Results

Communication Vector

With a communication vector agent was able to converge to a solution within 500 steps.

High Variance Learning with communication. Learning what to communicate can create very large changes in the value function compared to a baseline.

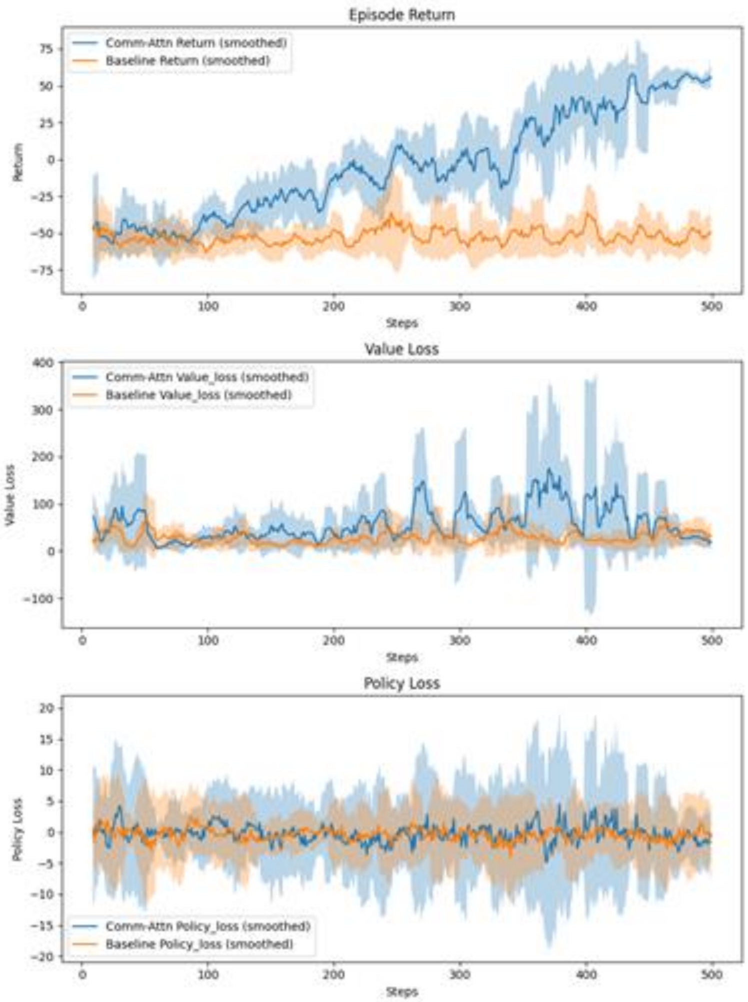


Experimental Results

Multi-head attention

Comm-Attn (blue) learn faster and performs much better than base-line

Since Comm-Attn learn from more complex input (communication vectors and attention), it learns slower from the start. It also has slightly higher variance in policy loss, showing that learning from communication is harder.

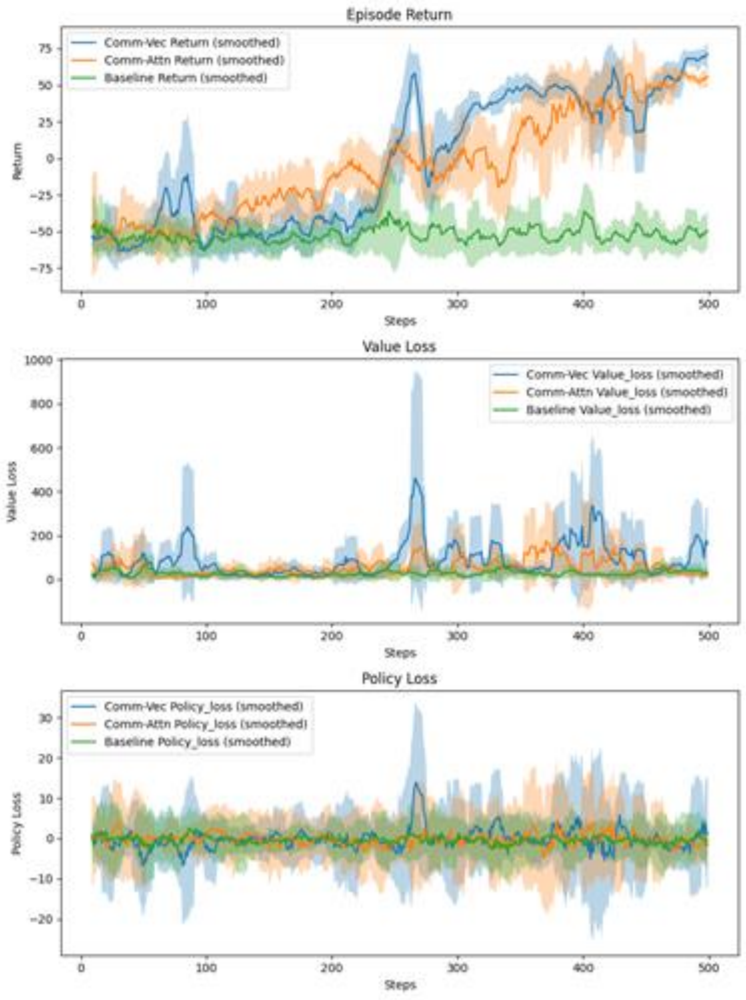


Experimental Results

Multi-head attention vs Communication Vector

Comm-Vec seems to prioritize aggressive learning, reflected in its higher variability in both value and policy losses but achieves better peak performance in returns

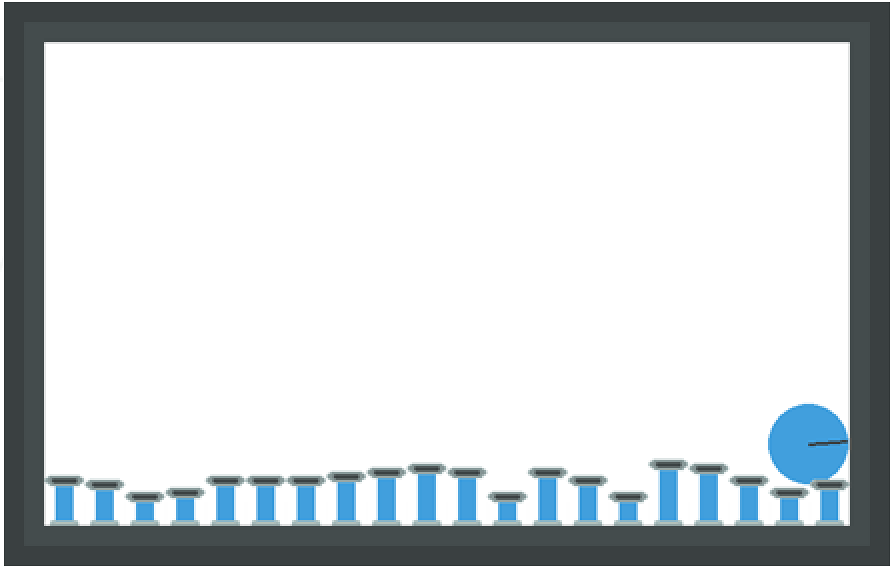
Comm-Attn trades off slightly lower returns for greater stability and consistency in the learning process, making it potentially more reliable.



Experimental Results

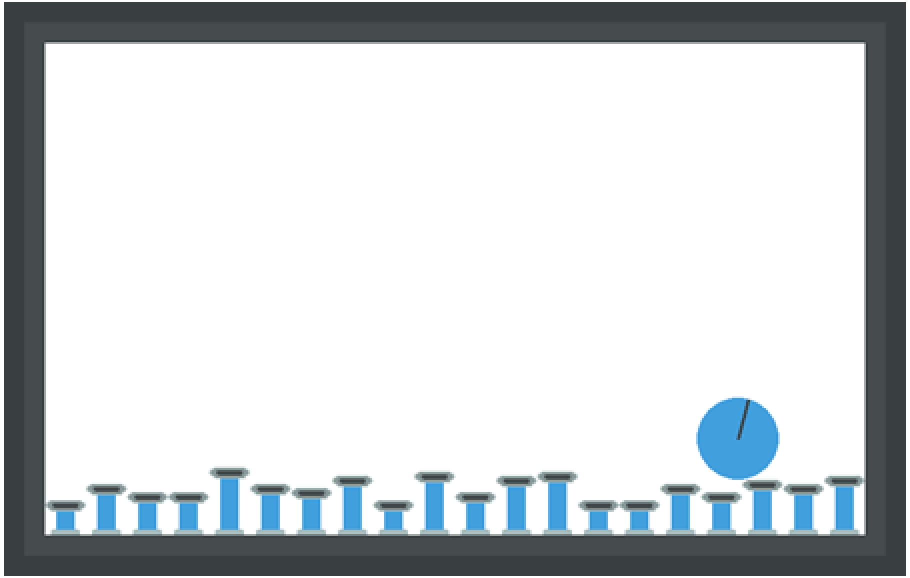
Multi-head attention

Our method



Base-line

(Siqi Liu, et el 2019)



Georgia Tech

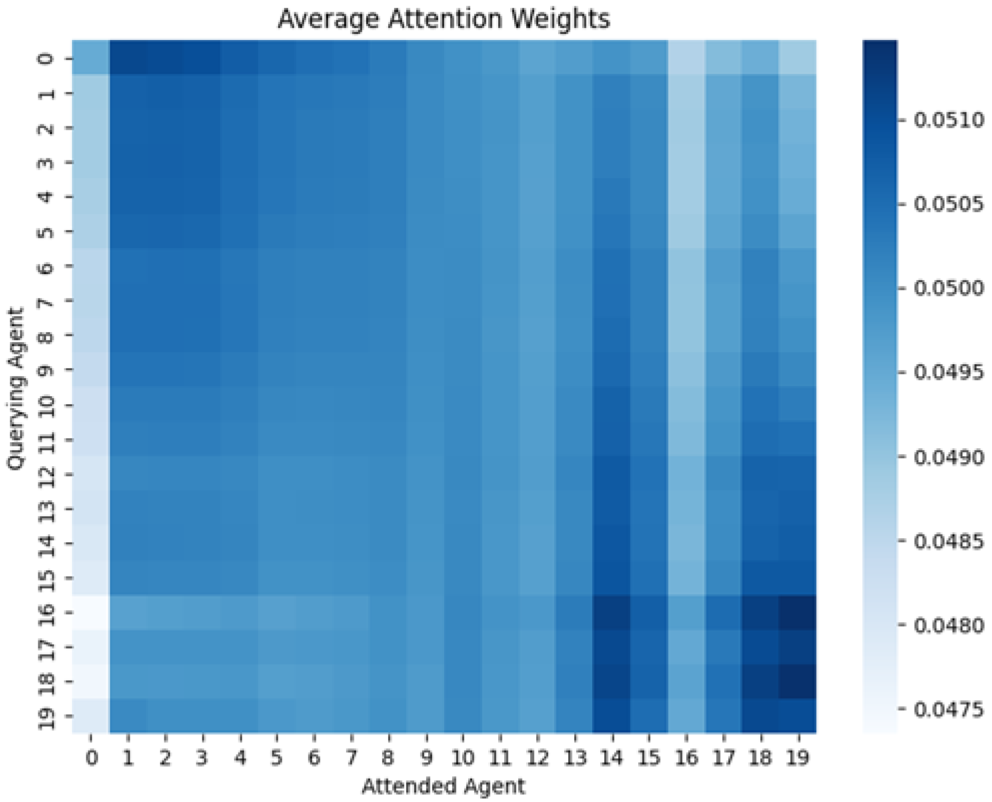
Experimental Results

Multi-head attention - Attention weights

- Diagonal dominance: higher attention to themselves or immediate neighbors

- Agents near the end 17-19 seems to give higher attention to specific attended agents

- Low attention for far agents

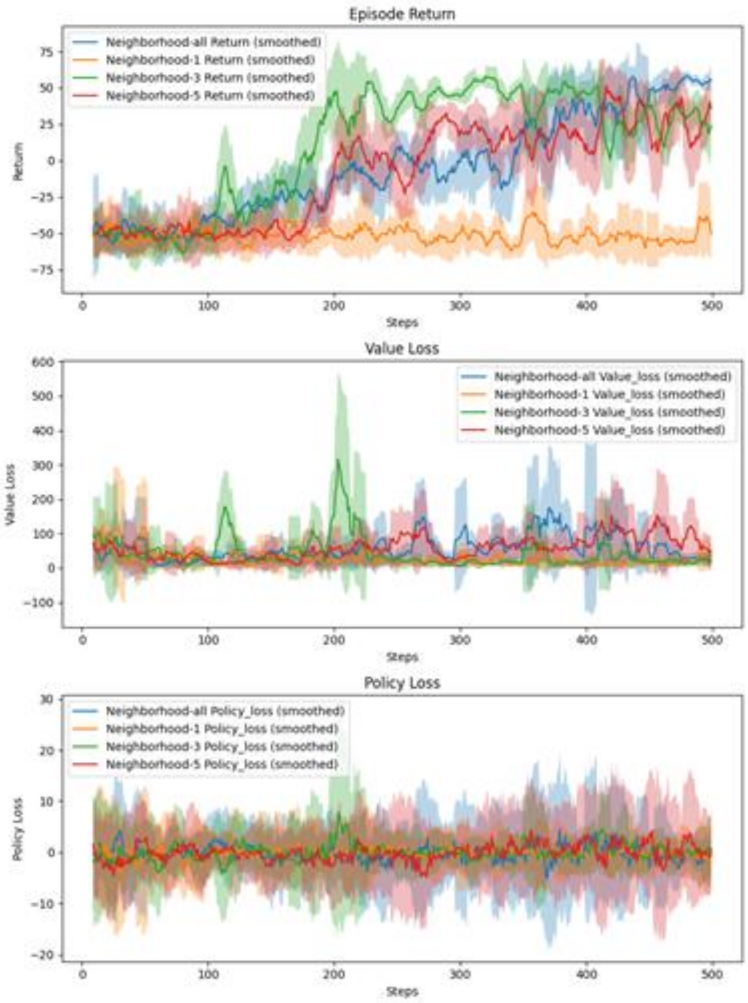


Georgia Tech

Experimental Results

Multi-head attention - Neighbor range

Agents only need to focus on relevant, nearby neighbors (3-5) to perform well. Overloading them with information (all agents) or underloading them (1 agent) hurts performance.



Conclusion

Conclusion

Implementing communication can improve learning in collaborative multi-agent environments (which is observed in the results)

Simple methods such as using a communication vector and more complex methods using Multi Headed Attention were sufficient to solve simple environment

Future work would be to explore more complex multi-agent environments