MARLo Polo: Emergent Communication in Multi-Agent Reinforcement Learning for Complex Tasks

Oliver Greenwood

September 11, 2025

Proposal

1 Introduction

Since Google Deepmind released AlphaGo in 2016, there has been a steady increase of interest into the field of reinforcement learning. This has led to breakthroughs in areas of finance, trading, robotics and even video games. A clear next step for reinforcement learning is multi-agent reinforcement learning where multiple agents are placed into the same environment and have to learn to interact with one another efficiently. This area of research has potential to improve surgery robots, traffic optimisations and search and rescue operations. However, there exist a range of challenges which make multi-agent reinforcement learning more difficult in practice such as: non-stationary problem, credit assignment issues, and the one I will focus on in this project, communication problems. Unlike single-agent reinforcement learning, multi-agent reinformcent learning has the ability to communicate to improve the agent's knowledge on the environment and/or teach them the optimal next step. A very common issue with communication is the "Joint Exploration Problem", this is a struggle for agents to communicate effectivly in novel environments. A recent paper[1] described a technique which is able to counter this issue was suggested.

In this project, I propose to extend AI Mother Tongue to a realistic situation for a more rigourous evaluation of the paper's proposed method. I will simulate an area for mulptiple agents with the goal of exploration. I will train the agents using a VDN (Value Decomposition Network) to work together to explore the area. I will have a continuous communication protocol which broadcasts raw information from agent to agent and use this as a baseline to compare to my emergent communication protocol. This other protocol will use a vector quanitsed variational autoencoder (VQ-VAE) to generate discrete symbols to broadcast from agent to agent.

This new method using a VQ-VAE allows communication protocols to emerge naturally rather than relying on manually designed rules or incentives. The limited alphabet forces agents to communicate with more compact, informative, and interpretable message that prioritize essential information sharing to achieve coordination. Hence we expect a improvement over the baseline communication protocol.

2 Structure of the Project

There are five main components to this project:

- Simulation: Building
- Value Decomposition Network
- Communication Protocols
- Training
- Evaluation: refer to Evaluation section

Simulation

The simulation is required as it will be how the reinforcement learning is both trained and evaluated. It will be responsilbe for the state management and local observations of each agent. Interactions will be handled through discrete timesteps, where agents produce actions informed by their policy and their communication inputs. As training multi agent systems will require a lot of runs, the simulation must be optimised highly. It will be written in Rust for this reason and mainly function as a headless program (only providing a simple graphical UI when demoing single runs).

Value Decomposition Network

The reinforcement learning will optimise the agents policy (the agents mapping from state space to action space). I have chosen to use Value Decomposition Network to facilitate decentralised policy learning as well as simultaneously optimising for a joint reward. Each agent will maintain its own Q function. TODO

Communication Protocols

This is the core of the project. I will primarily produce two distinct communication protocols. A simple protocol which has the goal of providing a baseline to the next protocol. It will emit raw continuous vectors. Our other protocol will emit symbols. These will be produced from a VQ-VAE (Vector Quantised - Variational Autoencoders), providing a discrete alphabet for agents to communicate via. Both of the protocols will initially broadcast every message to every other agent, unless the GNN extension is reached. This will allow a GNN to learn an optimal message passing structure.

Training

The training will manage the model's optimisation process. It will use the simulation to collect data and update both the agents' Q functions and, when required, the VQ-VAE. I will add a simple logging and checkpointing framework to prevent loss of results.

3 Evaluation

Quantitive

I will gather a small subset of environments not yet seen by the training, and run each model with the different communication protocols on each. I will then compare the average output of each model.

Qualitive

I will observe the evaluation runs and note down strategies I see the agents come up with. I will then compare the two model's strategies.

4 Starting Point

I have limited experience in using reinforcement learning. However I have previously built an autoencoder when analysing the stock market which will help my developing of the variational autoencoder. Furthermore I have a small amount of experience using Rust from a three month internship I completed, but I have a strong fluency in Python. This will help my development of a high performence backend (Rust) and the ML (Python).

5 Success Criteria

For this project to be deemed a success, the following must be successfully completed:

1. Design and implement a simulation to train and demo multi agent interactions. The simulation should be able to have single runs visualised. The simulation should control the state of every agent and collect data.
2. The training and evaluation pipeline should be automated end-to-end and only require a single config file as input. The pipeline should record checkpoints in training and log relevant output.
3. The training and evaluation pipeline should be able to interchangeably use both communication protocols as described earlier.

6 Possible Extensions

- Implement a third communication protocol using HQ-VAE.
- Add a GNN to the simulation to improve message passing between agents.
- Analyse quantised vectors for meaning

7 Timetable and Milestones

TODO

Weeks 1 to 2

Proposal submitted

Weeks 3 to 4

Weeks 5 to 6

Weeks 7 to 8

Weeks 9 to 11

Weeks 12 to 13

Weeks 13 to 15

Weeks 16 to 17

Weeks 18 to 19

Weeks 20 to 21

Weeks 22 to 23

Weeks 24 to 25

Weeks 26 to 27

Weeks 28 to 29

8 Resource Declaration

I will be using my personal desktop computer (AMD Ryzen 7 3700X, 32GB RAM, NVIDIA RTX 5060 Ti) as my primary machine for software development. As a backup, I will use my personal laptop and resources provided by SRCF (student run computing facility). I will continuously backup my code and dissertation with Git version control onto GitHub.