# DIAL in Multi-Agent Reinforcement Learning — Project Summary

## 1. Environment / Game

We used the Simple Spread environment from PettingZoo’s MPE (Multi-Agent Particle Environment) suite.  
- Setup: Three cooperative agents must spread out to cover landmarks while avoiding collisions.  
- State: Each agent observes its own position/velocity plus relative positions of landmarks and other agents.  
- Actions: Continuous 2D movement (x/y velocity).  
- Goal: Minimize total distance to landmarks (team reward) while avoiding collisions (penalty).

## 2. Communication Method

We implemented DIAL (Differentiable Inter-Agent Learning) — an explicit communication method.  
- Explicit means: agents send a symbol/message vector each timestep through a dedicated communication channel.  
- In DIAL, message generation is part of the neural policy, and gradients flow through the communication channel during training.  
- Each agent outputs:  
 1. Movement action (continuous)  
 2. Discrete message (one-hot over K\_vocab possible symbols)  
- Messages sent at time t are concatenated to other agents’ observations at time t+1.

## 3. Implementation Steps

1. Environment wrapper:  
 - Extended SimpleSpread to add message passing between agents.  
 - Messages are delivered with a one-step delay.  
2. Policy network:  
 - Based on PPO (Proximal Policy Optimization) per agent.  
 - Added an extra message output head and entropy term for communication.  
3. Training:  
 - 1,000 episodes, horizon 50 steps.  
 - Independent PPO for each agent.  
 - Logged rewards, steps, messages per episode to CSV.  
4. YAML config integration (optional):  
 - Parameters like NUM\_EPISODES, LR, K\_vocab, etc., loaded from a .yaml file.  
 - This allows easy experiment tuning without editing code.  
 - Results from YAML runs saved to a separate CSV file (dial\_simple\_spread\_yaml.csv).

## 4. Results Interpretation

From the First 100 vs Last 100 episodes bar chart and learning curves:  
- Mean reward & team reward:  
 Slight improvement from early episodes to late episodes — but both remain negative.  
- Per-agent rewards:  
 All three agents’ averages improved marginally over training. Performance is symmetric across agents.  
- Messages per episode:  
 Constant at 150 (3 agents × 50 steps), meaning agents always send a message each step.  
- Learning curves:  
 Reward is noisy per episode but moving average shows a slow upward trend.

### Image Descriptions

1. first\_last\_bars.png — Bar chart comparing first 100 vs last 100 episodes for normal config.  
2. first\_last\_bars\_yaml.png — Bar chart comparing first 100 vs last 100 episodes for YAML config.  
3. learning\_curves.png — Learning curves (mean reward, team reward, messages per episode) for normal config.  
4. learning\_curves\_yaml.png — Learning curves for YAML config.

## 5. Comparison: Normal Code vs YAML Config

- Normal code:  
 \* Hyperparameters fixed inside Python script.  
 \* Results saved in dial\_simple\_spread.csv.  
- YAML config version:  
 \* Hyperparameters loaded from external YAML file.  
 \* Easier to experiment with settings.  
 \* Results saved separately in dial\_simple\_spread\_yaml.csv.  
- Observations:  
 \* Performance trends (reward, message usage) are very similar.  
 \* YAML approach makes reproducing experiments and parameter tuning more convenient.

## 6. Key Takeaways

- Game: Simple Spread (cooperative landmark coverage)  
- Communication: Explicit, differentiable (DIAL), fixed rate, discrete symbols  
- Goal: Evaluate whether explicit communication improves coordination  
- Outcome: Small but consistent improvement over training; stable message passing; fair performance across agents  
- Extra: YAML config allows parameter tuning without modifying Python code.