Marl Formation using TorchRL

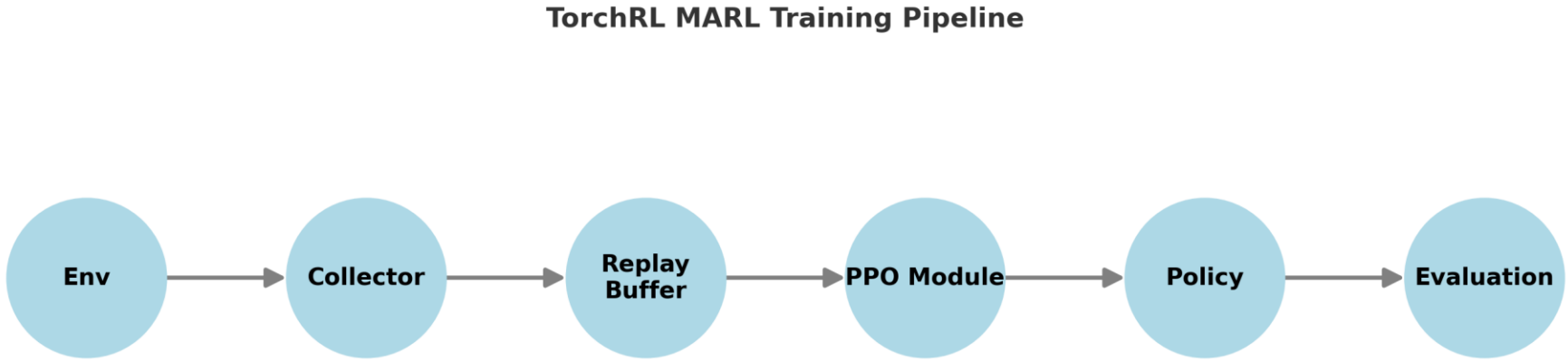
Developers: Dulovits Árpád, Sütő Attila

Objective

- Reimplement a multi-agent reinforcement learning setup where agents learn to form a circle centered at the origin with radius 3.

- **Approach**: Native TorchRL implementation using PPO and custom environment.

System Architecture



- **Framework**: Built entirely using TorchRL, PyTorch's official reinforcement learning library.

- **Algorithm**: Proximal Policy Optimization (PPO) used for stable and efficient training.

- **Components**:
- **Custom Multi-Agent Environment**: Built with Gymnasium and integrated into TorchRL.
- **Replay Buffers**: Managed via TorchRL's data collectors.
- **Training Loop**: Standard TorchRL trainer pipeline with configurable hyperparameters.

- **Parallelism**: Training scaled using vectorized environments and batch rollouts.

Environment & Task Setup

- **Task**: Agents must learn to form a circle of radius 3 centered at the origin.

- **Observation Space**: Each agent perceives:
  - Its own position and velocity
  - Relative positions of other agents

- **Action Space**: Continuous 2D movement vectors (x, y) per agent.

- **Reward Function**:
  - High reward for correct distance from center (radius 3)
  - Penalty for collisions and distance from formation

- **Environment Implementation**:
  - Custom Gym-like environment compatible with TorchRL wrappers

Key Design Choices

- **Custom Environment**: Tailored for full control over agent behavior and reward shaping.

- **TorchRL Usage**:

  Leveraged `MultiaSyncDataCollector` for efficient multi-agent rollout

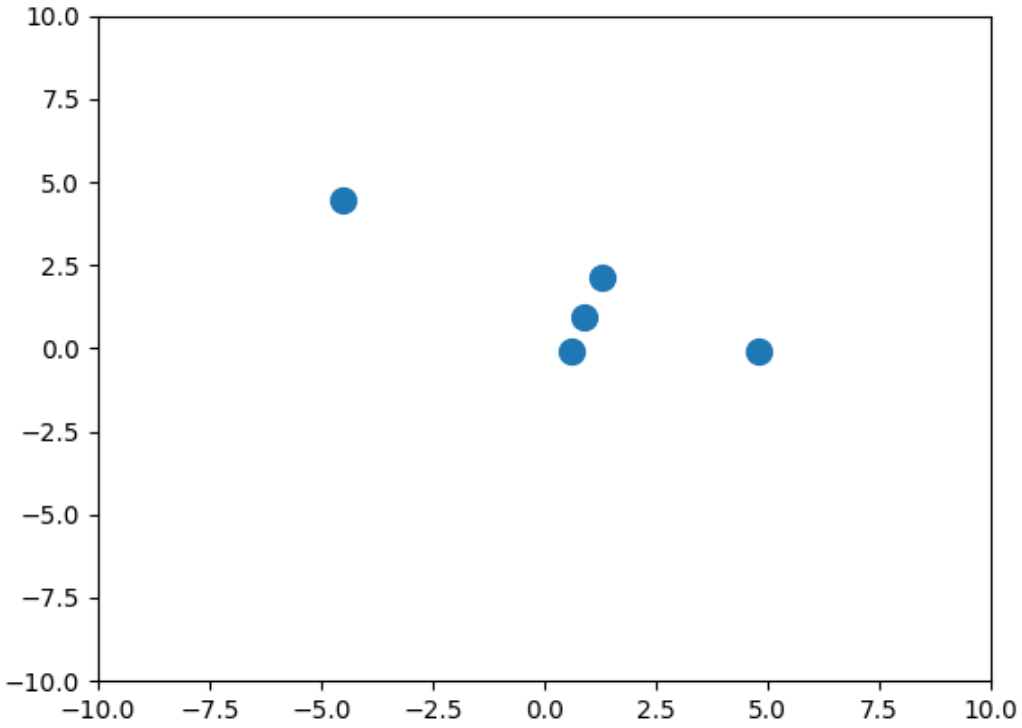
  Integrated `PPOTrainer` with observation transforms and reward normalization

- **Training Optimization**:

  Gradient clipping and entropy regularization

  Careful tuning of rollout length, learning rate, and batch size

Results & Visualizations



Conclusion & Future Work

- **Achievements**

  Successfully trained agents in a decentralized MARL setting using TorchRL

- **Future Directions:**

  Tuning the reward function for better distribution

  Longer training duration or increased rollout length

  Explore scalability to >10 agents with distributed training