Asset Allocation in Reinforcement Learning Report

Zhinuo Zhou
`zzhoudj@connect.ust.hk`

Linbing Xiang
`lxiangac@connect.ust.hk`

Contents

1 Introduction

This project implements an AI agent for Tic-Tac-Toe based on reinforcement learning techniques, specifically utilizing the Proximal Policy Optimization (PPO) algorithm - a state-of-the-art policy gradient method in modern reinforcement learning. The primary objective of this research is to develop and train a high-performing AI agent capable of exhibiting superior gameplay in the CrossTicTacToe variant.

1.1 Game Environment

The game environment is designed as a 12Œ12 board where only specific regions are designated as valid placement areas: top, bottom, left, right, and center regions. The game follows these rules:

- Players take turns placing their pieces on the board
- Victory is achieved by connecting 4 pieces horizontally or vertically, or 5 pieces diagonally

1.2 Training Process

The environment is formulated as a Markov Decision Process (MDP):

- States represent the current board configuration

- Actions correspond to piece placement in valid positions
- Reward structure: +1 for winning, -1 for losing
- A small penalty of -0.01 is applied per action to incentivize faster victories

2 Related Work

2.1 Reinforcement Learning in Board Games

Reinforcement learning has a long history of applications in board games. Early pioneering work can be traced back to Samuel's (Samuel, 1959) checkers program, which first demonstrated that computers could learn to improve strategies through self-play. TD-Gammon (Tesauro et al., 1995) was a milestone achievement that applied temporal difference (TD) learning to backgammon, with the system reaching near human expert level through self-play. These early works established the fundamental application patterns of reinforcement learning in board games.

With the rise of deep learning, AlphaGo by Silver et al. (Silver et al., 2016) combined deep reinforcement learning with Monte Carlo Tree Search (MCTS) to defeat a human world champion in Go for the first time. Its improved version, AlphaGo Zero (Silver et al., 2017), no longer relied on human expert data and learned completely from scratch through self-play, further demonstrating the powerful potential of deep reinforcement learning. These works have inspired extensive research applying deep reinforcement learning to various board games.

2.2 Policy Optimization Algorithms

Policy gradient methods have undergone significant evolution in the field of reinforcement learning. The REINFORCE algorithm proposed by Williams (Williams, 1992) was an early policy gradient method, but it faced training instability issues due to high variance. The Actor-Critic method introduced by Konda and Tsitsiklis (Konda and Tsitsiklis, 1999) improved training stability by combining policy gradients with value function estimation to reduce variance.

In recent years, policy optimization algorithms have achieved major breakthroughs. Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) addressed the performance collapse problem caused by large step updates by introducing trust region constraints for policy updates. Building on TRPO's ideas, Schulman et al. (Schulman et al., 2017) proposed Proximal Policy Optimization (PPO), which achieved performance similar to TRPO through a simplified objective function while significantly reducing computational complexity. Due to its high sample efficiency, simple implementation, and stable performance, PPO has become one of the mainstream algorithms in reinforcement learning research and applications.

3 Algorithm Design

This research employs the Proximal Policy Optimization (PPO) algorithm as the core training method, which achieves a balance between stability and sample efficiency by limiting policy update step sizes. We have constructed a network architecture that integrates modern deep learning techniques, specifically optimized for discrete state space environments like Tic-tac-toe.

3.1 Network Architecture Design

We implemented a dual-headed network structure based on the Actor-Critic paradigm:

- **Feature Extraction Module**: Consists of three convolutional neural network layers, using progressive channel expansion (1326464), coupled with padding strategies to maintain spatial dimensions. This module efficiently extracts features from the 12Œ12 game board state, capturing spatial patterns, piece distribution, and potential connection opportunities.
- **Policy Network (Actor)**: Receives the extracted feature representations and maps them to a 512-dimensional latent space through non-linear transformations (ReLU activation), ultimately outputting the logarithmic probabilities (logits) of action distribution. This design enables the network to accurately express complex policy functions.
- **Value Network (Critic)**: Shares feature extraction layers with the policy network, but independently maps to state value assessments, forming the foundation for complete advantage function estimation.

The shared parameter design reduces model complexity while accelerating the representation learning process.

3.2 Architecture Optimization and Stability Design

To improve training stability and model performance, we introduced several technical improvements:

- **Dueling Architecture**: Decomposes Q-values into state value function V(s) and advantage function A(s,a), reducing overestimation problems

- **Activation Function Optimization**: Uses parameterized LeakyReLU (α=0.01) instead of standard ReLU, effectively mitigating gradient vanishing and neuron death issues

- **Advanced Initialization Strategy**: Applies He initialization (Kaiming normal distribution), providing appropriate initial gradient flow for deep networks

- **Normalization Techniques**: Integrates LayerNorm and scale factors in the output layer, enhancing numerical stability and training consistency

3.3 Experience Replay Mechanism

We implemented an efficient Experience Replay Buffer to optimize sample utilization:

- Fixed-capacity storage structure based on double-ended queue (deque), saving state transition tuples (s, a, r, s', d)

- Random sampling strategy that breaks temporal correlations, reducing overfitting risk and increasing sample diversity

- Batch data preprocessing pipeline, improving computational efficiency and training stability

4 Experimental Design

4.1 Hyperparameter Configuration and Optimization

After systematic tuning, we determined the following key hyperparameters:

- Discount factor (γ=0.99): Balances near-term and long-term rewards, suitable for medium temporal length tasks like Tic-tac-toe

- GAE smoothing coefficient (λ=0.95): Achieves balance between bias and variance, providing stable advantage estimates

- PPO clipping parameter (ϵ=0.1): Limits policy update magnitude, preventing performance collapse due to excessive deviation

- Multi-epoch optimization (epochs=3): Fully utilizes each batch of data while avoiding overfitting

- Batch size ($batch\_size$=128): Provides sufficient statistical information while maintaining update flexibility

- Value loss coefficient ($value\_coef$=0.25): Balances policy improvement with state evaluation accuracy

- Entropy regularization coefficient ($entropy\_coef$=0.02): Encourages policy exploration, preventing premature convergence to suboptimal policies

- Gradient norm ceiling (max_grad_norm=0.5): Prevents abnormal gradients from disrupting the training process

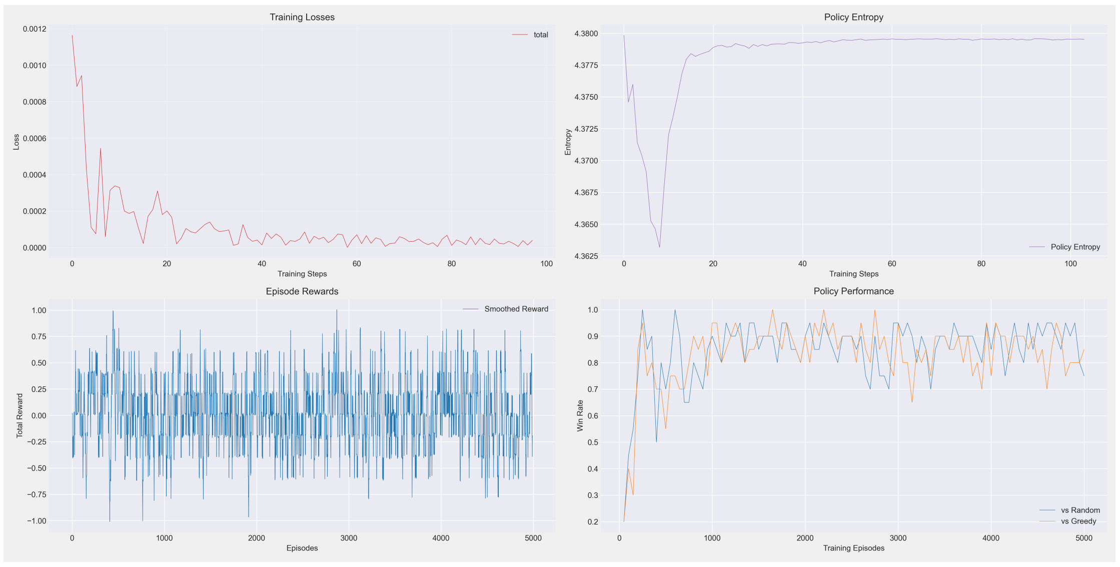

Figure 1: Training result

4.2 Training Optimization Strategies

We implemented multiple training stability strategies:

1. Action Space Masking Mechanism: Enforces policy distribution within valid action space through log-masking techniques, effectively handling Tic-tac-toe's dynamic action sets. This method elegantly achieves zero probability for invalid actions numerically while maintaining gradient flow continuity.

2. Generalized Advantage Estimation (GAE): Combines the advantages of n-step returns and temporal difference estimates, adaptively balancing bias and variance. The implementation includes outlier detection and handling mechanisms to ensure training stability.

3. PPO Core Algorithm Optimization:
   - Implements policy ratio clipping objective, constructing a conservative update interval
   - Adopts Huber loss instead of traditional MSE, enhancing robustness against reward scale outliers
   - Introduces KL divergence monitoring and early stopping mechanism to prevent excessive policy deviation
   - Applies multi-epoch data iteration and mini-batch training to improve sample efficiency
   - Implements adaptive learning rate decay strategy to optimize the convergence process

Through these designs, our PPO implementation can train stably in complex Tic-tac-toe environments and learn efficient policy representations, demonstrating strong adversarial performance and generalization capabilities.

5 Training Result

The figure 1 illustrates the training metrics of our PPO agent. The training curves demonstrate steady performance improvement as the number of training iterations increases. Key observations include:

- **Training Loss**: The training loss curve demonstrates a typical PPO training pattern of "initial volatility followed by gradual stabilization." Specifically, in the initial phase, loss values are high with significant fluctuations and pronounced oscillations. During the middle phase, the trend shifts downward with reduced yet rhythmic fluctuations. In the later phase, loss values maintain a relatively low and stable level with minimal variations. This pattern is expected as PPO's total loss comprises policy loss, value loss, and entropy regularization components, with learning rate decay implemented every 200 rounds in the code. This behavior aligns with theoretical expectations for PPO algorithms, indicating that the training process successfully balanced exploration and exploitation before converging to an effective policy.

- **Policy Entropy**: The entropy trend follows a pattern of initially high values, followed by a decrease, and then a gradual increase while remaining below initial values. This reflects three distinct phases of policy learning, consistent with typical PPO training trajectories:
  - Initial high-entropy phase: When training begins, the policy is random with action probabilities approaching a uniform distribution, resulting in high entropy values.
  - Rapid decline phase: As learning progresses, the model develops preferences for certain actions, concentrating the distribution and causing entropy to decrease quickly.
  - Gradual rise phase: When the model identifies states requiring additional exploration, it appropriately increases entropy, though typically not returning to initial random levels.
  - The entropy settling below its initial value while maintaining a certain level is ideal, indicating that the model has achieved a balance between deterministic decision-making and appropriate exploratory capacity.
- **Episode Reward**: The relatively stable reward curve can be attributed to:
  - Reward Shaping Mechanism: The project employs a sophisticated RewardShaper class to enhance the original reward signals, taking into account multiple factors including positional value, connection potential, blocking potential, and regional control.
  - Stable Policy Performance: The policy learned by the agent tends to stabilize in the later stages of training, resulting in a balanced distribution of shaped rewards.
- **Win Rate**: The win rate curve exhibits an upward trajectory, confirming that the agent has learned effective gameplay strategies.

Through approximately 5,000 training iterations, the agent successfully acquired effective strategies for the CrossTicTacToe game and demonstrates the ability to defeat baseline opponent strategies.

6 Policy Testing

To evaluate the performance of the trained PPO agent, we designed a comprehensive testing methodology, primarily measuring its performance by having the agent compete against opponents with two different strategies (random strategy and rule-based greedy strategy). The testing process uses a deterministic policy (non-exploration mode), records complete match data, and calculates win rate metrics, while also generating intuitive GIF animations of the match process, including board state evolution, move position highlights, and game result statistics. By analyzing the agent's win rates, game lengths, and decision patterns against different opponents, we can comprehensively evaluate the agent's strategy quality and adaptability. Test results show that the agent has successfully learned effective game strategies, is able to anticipate opponent intentions, and execute reasonable offensive and defensive decisions.

6.1 Random Policy

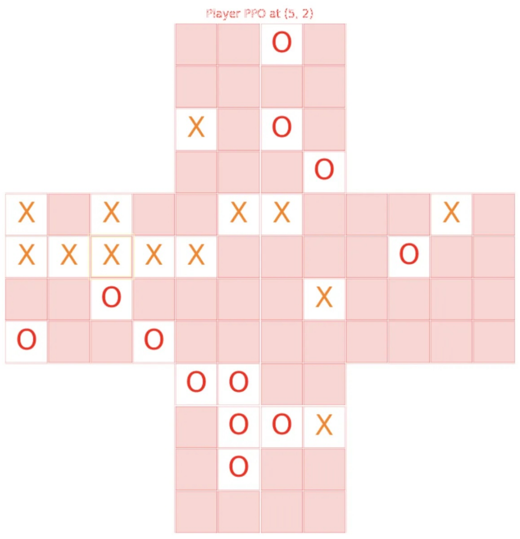

Figure 2: vs Random Policy

The figure 2 above demonstrates our PPO agent competing against an opponent using a random strategy. It can be observed that the agent consistently defends against random placements while effectively building its own connecting lines, ultimately achieving victory. The agent exhibits strong capabilities in identifying critical positions and demonstrates strategic planning abilities throughout the gameplay.

6.2 Greedy Policy

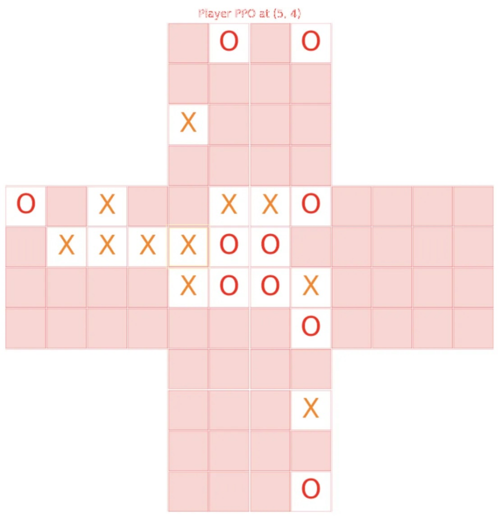

Figure 3: vs Greedy Policy

This demonstration showcases the PPO agent competing against an opponent using a greedy strategy. The greedy strategy prioritizes positions that lead to immediate victory or blocks the opponent from winning, thus presenting a significantly more challenging adversary than a random policy. Even when facing such a sophisticated opponent, our agent demonstrates excellent strategic thinking, anticipating the opponent's moves to set up traps and secure victories.

7 Conclusion

In conclusion, this project not only demonstrates the successful application of PPO to the Cross Tic-Tac-Toe game but also provides insights into the practical considerations for implementing reinforcement learning in discrete action space environments. The methodologies developed herein contribute to the broader understanding of how strategic decision-making can emerge from reinforcement learning processes, with potential implications for more complex domains beyond board games.

Code is available in https://github.com/ZhinuoNunu/RL$_t tt$.

References

Vijay Konda and John Tsitsiklis. 1999. Actor-critic algorithms. *Advances in neural information processing systems*, 12.

A. L. Samuel. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

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

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Gerald Tesauro et al. 1995. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256.