Improving PPO for Partially Observable Stochastic Multiplayer Tabletop Games through Observation, Reward, and Self-Play Design

Kai-Jun, Zhong[1,a)] Jen-Yeu, Chen[1,b)] Shi-Jim, Yen[2,c)] Wei-Che, Chien[2,d)]

Abstract: Modern tabletop games offer diverse rules and are relatively easy to model and implement, making them suitable testbeds for reinforcement learning. This study investigates how observation representation, reward design, and self-play affect proximal policy optimization (PPO) in Love Letter using the PyTAG framework. We evaluate and revise these components individually to examine changes in agent performance. In addition, we propose a best-response-based self-play procedure that introduces targeted opponents during training to expose weaknesses of the current policy. Our results highlight the importance of observation, reward, and opponent design for PPO in partially observable multiplayer games.

Keywords: reinforcement learning, observation design, reward design, self-play, POSG

1. Introduction

Modern tabletop games often involve hidden information, stochastic events, and interactions among multiple players. These characteristics make them challenging environments for reinforcement learning. In multi-agent reinforcement learning (MARL), these games can often be regarded as partially observable stochastic games, in which an agent must make decisions without complete information. Love Letter provides a compact example of this setting: players act with limited private information while reasoning from public actions, discarded cards, and information revealed during play. Its relatively simple rules, combined with partial observability and multiplayer interaction, make it a useful testbed for studying reinforcement learning in tabletop games.

The Tabletop Games (TAG) framework [2] provides a unified platform for studying modern tabletop games, and PyTAG [3] extends TAG with an interface for reinforcement learning. Balla et al. evaluated proximal policy optimization (PPO) [4] on a subset of these games using self-play and compared the trained policies against Random, OSLA, and MCTS opponents. However, their results on Love Letter show that PPO can perform well against the Random agent but still struggles to reach a strong policy against stronger opponents. This suggests that performance may depend not only on the learning algorithm itself, but also on the design of observations, reward signals, and training opponents.

[1] Department of Electrical Engineering, National Dong Hwa University, Hualien, Taiwan
[2] Department of Computer Science, National Dong Hwa University, Hualien, Taiwan
[a)] 611223031@gms.ndhu.edu.tw
[b)] jenyeu@gms.ndhu.edu.tw
[c)] sjyen@gms.ndhu.edu.tw
[d)] wcc@gms.ndhu.edu.tw

Rather than introducing a novel algorithm or model, this study examines three design factors in PPO training for Love Letter: observation representation, reward design, and self-play. Observation should retain information useful for decision making. Reward should provide a learning signal aligned with game progress. Self-play determines the opponents encountered during training. We evaluate these factors while keeping the PPO architecture as close as possible to the original setting.

Our experiments show that observation design has a substantial effect on PPO performance in Love Letter. The revised score-based reward, which reflects round-level progress rather than accumulated score, outperforms the original score-based setting. The proposed self-play procedure further improves performance in the two-player and three-player settings, although the advantage becomes less consistent with four players. These results suggest that effective learning in partially observable multiplayer tabletop games depends not only on the learning algorithm, but also on how information, rewards, and training opponents are designed.

2. Setting

Love Letter is a two-to-four-player card game played over multiple rounds. On each turn, a player draws one card, chooses one of the two cards in hand to play, and resolves its effect. Card effects can reveal information, force a player to discard, compare card values, provide temporary protection from other players' card effects, or eliminate another player from the current round. A round ends when only one player remains or when the draw pile is empty. The round winner receives a favor token. The game continues until a player reaches the required number of favor tokens [1].

Although Love Letter has relatively simple rules and only

© 2026 Information Processing Society of Japan

eight card types, effective play requires reasoning with both private and public information. Players know their own hand and can observe played cards and public game events, but the remaining cards and opponents' hands are generally hidden. Information revealed by previous card effects may also become relevant to later decisions. A distinctive aspect of Love Letter as a reinforcement learning environment is that its small action space coexists with hidden information, stochastic card draws, elimination, and direct multiplayer interaction.

In PyTAG, the Love Letter observation is represented by a small set of game-specific features, summarized in Table 1 [3]. The representation mainly consists of normalized aggregate values and binary indicators of specific cards held by the current player.

Table 1 Original observation representation for Love Letter.

Observation component	Size
Total value of cards in hand	1
Score relative to the winning threshold	1
Indicators of card types held	8
Visible information about opponents' hands	1
Score advantage over other players	1
Total	12

PyTAG trains PPO policies using self-play, where previously saved policy checkpoints are reused as training opponents [3]. During training, the current policy is periodically evaluated against the baseline agents provided by TAG, including Random, OSLA, and MCTS. TAG determines the legal actions at each decision point, and PyTAG uses an action mask to prevent the PPO policy from selecting invalid actions. We retain the original PyTAG evaluation procedure against the TAG baseline agents throughout our experiments.

3. Design Factors

The baseline observation provides only a small summary of the acting player's current state. We therefore construct the observation from the acting player's viewpoint and retain information available under the game rules that may remain useful for current or later decisions. The revised representation includes remaining-card information, known opponent-hand information, score state, player states, and recent public actions.

Table 2 Revised observation representation for Love Letter.

Observation component	Size
Remaining deck size	1
Current hand	8
Player-view remaining-card pool	8
Score state	3
Known-hand information by player	$9N$
Whether opponents know the player's hand	$N-1$
Alive states	N
Protection states	N
Last played cards	$9N$
Visible reserve cards	8
Total	$27+21N$

Here, N denotes the number of players. For each player, known-hand information is encoded by eight card-type indicators plus one unknown/no-known-card state, while the last played card is encoded by eight card-type indicators plus one none state. The three score-state features are the current player's score relative to the winning score, an indicator of whether the player is one point from winning, and a normalized score-difference feature relative to the highest-scoring opponent.

The normalized score difference is computed as

$$0.5+0.5\frac{s_i-\max_{j\neq i} s_j}{S_{\text{win}}},$$

so that equal scores correspond to 0.5.

Compared with the baseline representation in Table 1, the revised observation preserves more decision-relevant information without revealing hidden game state. In particular, the remaining-card pool is derived from cards that the acting player can account for, including public cards, the player's own hand, and opponent cards that have been explicitly revealed. Known information is updated when card effects such as Priest, Baron, King, and Prince reveal cards, exchange hands, or clear previously known hand information. Thus, the added features represent the player's knowledge rather than using global information unavailable to the acting player.

We next examine the reward definition used for PPO training. Let s_i^t denote the accumulated score of player i after transition t, and let S_{win} denote the score required to win the match. For Love Letter, the terminal reward used in our PyTAG-based training setting [3] is

$$r_{i,t}^{\text{terminal}}=\begin{cases}+1, & \text{if player } i \text{ wins the match,}\\ -1, & \text{if player } i \text{ loses the match,}\\ 0, & \text{otherwise.}\end{cases}$$

The original score-based reward uses the player's current accumulated score at the end of each round [3]:

$$r_{i,t}^{\text{score}}=\begin{cases}s_i^t, & \text{if transition } t \text{ ends a round,}\\ 0, & \text{otherwise.}\end{cases}$$

Because Love Letter is played over multiple rounds, this formulation does not distinguish a newly earned favor token from a score carried over from earlier rounds. Consequently, the same score may be rewarded repeatedly even when no new scoring event has occurred.

To better align the score-based signal with actual progress, we define the revised reward as

$$r_{i,t}^{\text{increment}}=\begin{cases}\dfrac{\Delta s_i^t-\frac{1}{2}\sum_{j\neq i}\Delta s_j^t}{S_{\text{win}}}, & \text{if transition } t \text{ ends a round,}\\ 0, & \text{otherwise.}\end{cases}$$

Thus, gaining a favor token produces positive feedback, whereas an opponent's score increase produces negative

© 2026 Information Processing Society of Japan

feedback at half the magnitude. The normalization by S_{win} expresses the reward relative to the score required to win the match. We revise the formulation to examine whether actual scoring events provide a learning signal that is better aligned with the game objective than accumulated scores.

The original PyTAG self-play setting samples training opponents from a pool of historical policy checkpoints [3]. Although this avoids training against a fixed opponent, the opponent pool is still derived from previous versions of the same learning process. Consequently, the available opponents may remain close to similar strategic behaviors and may not expose weaknesses of the current policy effectively. To examine whether more targeted opponents can improve training, we introduce position-specific best-response (BR) policies.

The main PPO policy uses parameter sharing across all player positions and is updated from transitions collected at those positions. At each BR phase, a snapshot of the current main policy is frozen as the reference policy. For each player position i, a separate BR_i policy is trained against the reference policy controlling the remaining positions, using only transitions collected from position i.

After the BR phase, training of the main policy resumes. At the beginning of each match, one of $N+1$ training configurations is sampled uniformly: either the main policy controls all positions, or BR_i controls position i while the main policy controls the remaining positions. The BR policies are replaced when the next BR phase begins.

Each BR_i is evaluated against the reference policy at the same position, with $G_i = W_{\text{BR},i} - W_{\text{ref},i}$. BR training stops when G_i exceeds the threshold or reaches the step limit, and the resulting BR_i is retained for subsequent main-policy training. The main policy follows a fixed update schedule independent of this criterion.

4. Experimental Setup

The main Love Letter experiments use the same PPO architecture unless otherwise stated. The actor and critic are both multilayer perceptrons with two hidden layers of 256 units and ReLU activations. The actor produces logits over the fixed action space, while the action mask restricts the policy distribution to legal actions. The critic outputs a scalar value estimate. The main PPO settings are summarized in Table 3.

The actor learning rate was tuned separately for the terminal and score-based reward settings, resulting in 1×10^{-3} and 3×10^{-4}, respectively. Each trainable policy is limited to at most 1×10^6 agent–environment interactions. This limit applies independently to the main policy and to each BR policy. The policy is evaluated periodically against the Random, OSLA, and MCTS agents provided by TAG. During evaluation, the trained policy is rotated across all player positions, while the remaining positions are controlled by the same baseline opponent. Evaluation results are used only to measure policy performance and are not included in PPO updates.

Table 3 Main PPO training settings.

Setting	Value
Actor / critic hidden layers	$(256, 256)$, ReLU
Steps per update	4000
Discount factor γ	0.99
GAE parameter λ	0.95
PPO clipping ϵ	0.2
Epochs per update	4
Minibatch size	500
Entropy coefficient	0.005
Value loss coefficient	0.5
Maximum gradient norm	0.5
Actor learning rate (terminal)	1×10^{-3}
Actor learning rate (score)	3×10^{-4}
Critic learning rate	1×10^{-3}

To reduce evaluation variance, we use more evaluation games than the original PyTAG study. For each baseline opponent, each evaluation contains 50, 51, and 52 games in the two-, three-, and four-player settings, respectively, so that each player position is evaluated equally often. The terminal-reward and revised score-reward experiments use five random seeds, while the original score-reward experiments use three. Learning curves show the mean across seeds with standard-error shading. For the Love Letter experiments, tabled win rates are averaged over the final 20% of evaluations and across seeds.

We report win rate as the primary evaluation metric, defined as the fraction of evaluation games won by the trained policy.

The detailed schedule of the proposed self-play procedure is summarized in Table 4. BR policies are evaluated independently for each player position using the gap defined in Section 3, whereas main-policy training follows the fixed step schedule.

Table 4 Self-play training and BR evaluation settings.

Setting	Value
PPO update interval	4×10^3 steps
Main-policy external evaluation interval	2×10^4 steps
BR phase trigger interval	1×10^5 main-policy steps
BR steps before first evaluation	1×10^5 steps
Additional BR training block	4×10^4 steps
BR gap threshold (2 players)	0.10
BR gap threshold (3 players)	0.08
BR gap threshold (4 players)	0.05

As a supplementary observation-design experiment, we use the two-player no-limit Texas Hold'em environment provided by RLCard [5]. We compare a 129-dimensional direct observation with a 161-dimensional decision-oriented observation. The direct observation describes the current hand, visible cards, game stage, chip state, and hand-related features. The decision-oriented observation retains these features and additionally includes the three most recent public actions, chip differences, pot dynamics, and action summaries from the current hand. These additional features contain only publicly available information and do not reveal the opponent's hidden cards.

© 2026 Information Processing Society of Japan

Both Poker representations use the same Actor–Critic architecture, consisting of a CNN encoder for card features, an MLP for non-card features, and a fusion network. The network capacity and PPO settings are kept identical between the two observation representations. Each model is trained for 1000 PPO updates.

For evaluation, we use deterministic action selection. For each evaluation seed, the two policies play two paired games with their player positions exchanged while using the same environment seed. Each position assignment is evaluated for 50,000 games, resulting in 100,000 games in total.

5. Results

We first evaluate the effect of the observation representation while retaining the original PyTAG self-play procedure and terminal reward setting [3]. Over the final 20% of training evaluations, our agent achieves win rates of 0.89 ± 0.01, 0.58 ± 0.02, and 0.58 ± 0.01 against Random, OSLA, and MCTS, respectively. Table 5 summarizes the comparison with the published PyTAG baseline. The revised observation shows substantially larger gains against OSLA and MCTS than against Random.

The upper row of Figure 1 shows the corresponding learning curves; the original PyTAG curves were digitized from the published results in [3].

Table 5 Two-player win rates with terminal reward. PyTAG baseline results are taken from [3].

Method	Random	OSLA	MCTS
PyTAG baseline	0.80	0.22	0.17
Revised observation	0.89	0.58	0.58

With the revised observation fixed, we next compare the terminal, original score-based, and revised score-based rewards in the two-player setting. Table 6 summarizes the win rates, while the lower row of Figure 1 shows the corresponding learning curves. Dotted lines mark the equal-share win rate of 0.5 in all panels.

Table 6 Two-player win rates under different reward settings.

Reward	Random	OSLA	MCTS
Terminal	0.89	0.58	0.58
Original score	0.74	0.30	0.36
Revised score	0.90	0.60	0.63

These results indicate that the poor performance of the original score setting does not arise simply from using score information as reward. Rather, the reward representation matters: repeatedly rewarding accumulated scores can misalign the learning signal with actual scoring events, whereas the revised formulation ties the learning signal to actual round-level scoring outcomes.

We next examine the effect of the number of players under the terminal-reward setting. Since the equal-share win rate decreases as $1/N$, raw win rates across different player counts are not directly comparable. Against Random, the trained policy remains well above the equal-share baseline in all settings. Against OSLA and MCTS, however, its win rate falls from above $1/N$ with two players to below $1/N$ with three and four players.

Increasing the number of players introduces more sources of hidden information, opponents' actions, and interactions that can affect each decision. The available information also becomes more fragmented across multiple opponents, increasing the uncertainty of the game state from the acting player's perspective. These results suggest that the difficulty of Love Letter is determined not only by its observation representation, but also by the increased uncertainty and interaction complexity of multiplayer settings.

Table 7 Win rates with different numbers of players.

Players	Random	OSLA	MCTS
2	0.89	0.58	0.58
3	0.68	0.29	0.31
4	0.47	0.13	0.14

We then compare the original PyTAG self-play procedure with the proposed BR-based self-play while keeping the revised observation and terminal reward fixed. In the two-player setting, BR-based self-play improves the win rates against OSLA and MCTS from 0.58 to 0.63 and 0.62, respectively, while the performance against Random remains similar. In the three-player setting, the win rates increased from 0.68, 0.29, and 0.31 to 0.72, 0.33, and 0.33 against Random, OSLA, and MCTS, respectively.

The improvement becomes less consistent in the four-player setting. The BR-based method obtains win rates of 0.47, 0.15, and 0.13 against Random, OSLA, and MCTS, compared with 0.47, 0.13, and 0.14 using the original self-play procedure. Thus, BR training provides a clearer benefit in the two- and three-player settings; with four players, the greater diversity of hidden information and opponent interactions may reduce the benefit of training against one position-specific BR opponent at a time.

Table 8 Win rates under different self-play methods.

Players	Self-play	Random	OSLA	MCTS
2	Original self-play	0.89	0.58	0.58
	BR-based	0.90	0.63	0.62
3	Original self-play	0.68	0.29	0.31
	Parameter sharing	0.66	0.25	0.29
	BR-based	0.72	0.33	0.33
4	Original self-play	0.47	0.13	0.14
	BR-based	0.47	0.15	0.13

A three-player ablation using parameter sharing without BR policies produces performance close to the original self-play setting rather than the full BR-based method. This suggests that the observed improvement is more closely associated with the targeted BR opponents than with parameter sharing alone.

As a supplementary test of the observation-design principle, we compare the two Poker representations described in Section 4. The decision-oriented observation achieves an average return of 0.502 and a win rate of 71.08%, whereas the

© 2026 Information Processing Society of Japan

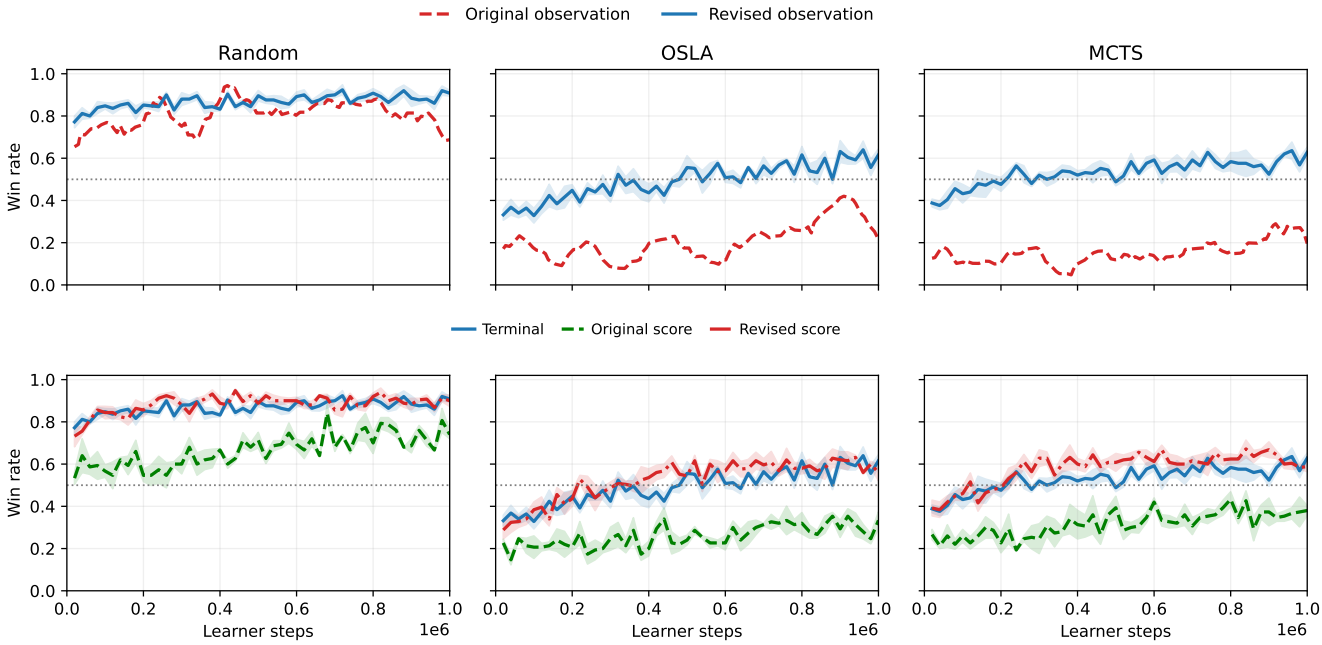


Fig. 1 Two-player Love Letter learning curves.

direct observation obtains -0.502 and 28.23%, respectively. The same performance trend is observed from both player positions, indicating that the difference is not restricted to a particular seating position.

Table 9 Poker results under different observation representations.

Observation	Avg.	First	Second	Win rate
Direct	-0.502	-0.462	-0.543	28.23%
Decision-oriented	0.502	0.543	0.462	71.08%

The main difference between the two representations is that the decision-oriented observation preserves public historical information that may remain relevant to later decisions instead of describing only the current visible situation. This follows the same observation-design principle used in Love Letter. However, this experiment does not show that increasing the observation dimension alone improves performance, nor does it establish that the same effect holds for all partially observable games.

6. Conclusion

This study examined how observation representation, reward design, and self-play affect PPO training in Love Letter. Expanding the observation with decision-relevant information substantially improved performance over the original PyTAG representation, particularly against stronger baseline opponents. For reward design, using score changes associated with actual round outcomes performed better than directly rewarding the accumulated score, showing that the form of the learning signal is important even when both rewards are based on the same game score. The BR-based self-play procedure further improved performance in the two- and three-player settings, while its benefit was less consistent with four players. Overall, the results indicate that effective PPO training in partially observable multiplayer games depends not only on the learning algorithm, but also on how available information, reward signals, and training opponents are constructed.

References

[1] Alderac Entertainment Group, *Love Letter Premium Rulebook*, 2016.

[2] R. D. Gaina, M. Balla, A. Dockhorn, R. Montoliu, and D. Perez-Liebana, "TAG: A Tabletop Games Framework," in *Experimental AI in Games (EXAG), AIIDE 2020 Workshop*, 2020.

[3] M. Balla, G. E. M. Long, J. Goodman, R. D. Gaina, and D. Perez-Liebana, "PyTAG: Tabletop Games for Multiagent Reinforcement Learning," *IEEE Transactions on Games*, vol. 16, no. 4, pp. 993–1002, 2024.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[5] D. Zha, K.-H. Lai, Y. Cao, S. Huang, R. Wei, J. Guo, and X. Hu, "RLCard: A Toolkit for Reinforcement Learning in Card Games," in *AAAI-20 Workshop on Reinforcement Learning in Games*, 2020.

© 2026 Information Processing Society of Japan