An Efficient Training Strategy for Multi-Agent Reinforcement Learning in Card Games

Jiyuan Shen
School of Computer Engineering and Science
Shanghai University
Shanghai, China
shenjiyuan@shu.edu.cn

***Abstract*—Most of the previous researches on reinforcement learning focused on modifying the learning mechanism of the Markov Decision Process. Researches on training strategy to improve the performance of the MARL algorithm has not received enough attention. Therefore, this paper focuses on the improvement during training process and proposes a gradual promotion training strategy. It is mainly divided into two stages, single combat stage and multi combat stage. Scenario-transfer training, rule-based training, self-playing training and mixed training are used at single combat stage to obtain a strong single agent, namely the pretrain process. At multi combat stage, multi-agent training is introduced, which increases the complexity of the game, so that the strong single agent gradually adapts to the multi-agent task, and the strong multi agent is obtained. This paper combines these two stages of learning tasks with two popular multi-agent reinforcement learning methods, namely Deep Q-learning and Neural Fictitious Self-Play. The experiment found that the gradual promotion training strategy can effectively improve the winning rate and average reward of the agent. Compared with the un-pretrained agent, the average reward is improved by 25% and winning rate is improved by 44%; at the same time, it is an extremely convenient and easy training strategy to implement.**

Keywords-MDRL; Training Strategy; DQN; NFSP; Texas Hold'em

I. INTRODUCTION

Reinforcement learning (RL) has achieved outstanding results in recent years, especially in competitive games such as the Go game, Atari games and Card games [1, 9]. These achievements will definitely contribute to wider uses, including industrial production and daily life, not limited to games. However, it is precisely because of the complexity of RL in practical application, so Stone and Veloso [2] proposed multiagent systems (MAS), focusing on much opener problems to simulate the realistic situations. Common background settings for MAS include fully cooperative, fully competitive, mixed cooperative and competitive, and self-interested [9]. However, if applies traditional reinforcement learning methods directly to MAS, such as model-based and model-free methods [17-20], a series of problems may appear. Firstly, the state and action space expand exponentially with the increase of agents, making it hard to express in a fixed algorithm. Secondly, the computation complexity grows greatly as each agent has their own policy to make action, which makes the environment state more intricate and affects the convergence of the algorithm. As a result, multiagent deep reinforcement learning (MDRL) [3,4] has been proposed to approximate the optimal policy and/or the value function, depending on the powerful express capacity of neural network.

For MDRL, there are three most used methods to control agents, centralized method, decentralized method and mixed method [1]. The biggest distinction is whether agents can communicate with each other and know their next action. Centralized learners use a central controller to train all agents and the controller makes decision based on all the agents' observation [10]. Decentralized learners are similar to the single-agent algorithms with each agent independently learns its own policy [11]. Mixed learners mean centralized training and decentralized executing [12]. Although the control methods and algorithms may different, yet it can all apply to neural network to solve and so various approximation algorithms have appeared [5-8].

This paper studies a traditional card game named Texas Hold'em Poker and despite of its extensive research it still has enough room for improvement in performance and complexity degree. In a game of Texas hold'em, each player is dealt two cards face down (the 'hole cards'). Throughout several betting rounds, five more cards are eventually dealt face up in the middle of the table. These face-up cards are called the community cards. Each player is free to use the community cards in combination with their hole cards to build a five-card poker hand. During the game players have three choices to make call, raise or fold. The aim is to compose the biggest card and win the game. For the Texas Hold'em, two popular MDRL methods are adopted as baselines, i.e., Deep Q-learning (DQN) and Neural Feural Fictitious Self-Play (NFSP). These two methods and their variants have been used in many other tasks. DQN uses neural network to approximate Q function and observations are shared by all agents [13, 14]. NFSP is a learning method for learning approximate Nash equilibria of imperfect-information games. NFSP combines fictitious self-play (FSP) with neural network function approximation [7].

During the training process, this paper proposes a gradual promotion training strategy including two stages: at the first stage, it is a single combat stage (like single agent reinforcement learning) and four techniques are applied that is scenario-transfer training, rule-based training, self-playing training and mixed training. Through these pretrained process, the agent can gradually learn the basic way to learn the game,

which focuses much more on self-improvement. At the second stage, it is a multi-combat stage and the environment is enhanced in its difficulty and complexity through adding more powerful agents. Through this stage, the agent can gradually learn advanced playing way, such as making full use of others' call and bet size and viewing the Texas Hold'em more exhaustively. Actually, these two stages work in totally different aspect during the training process and the agents can perform much better and get a higher reward through the gradual promotion training strategy.

In general, the main novelties of this paper are summarized below:

- Consider Texas Hold'em as a multi agent task and do thorough experiments using DQN and NFSP as baselines.
- Propose gradual promotion training strategy including two training stages and improve the performance of DQN and NFSP on Texas Hold'em game.
- The proposed training strategy is easy to transplant and operate and has a well effect on other similar scenarios.

In this paper, Section 2 introduces some common improvement methods and tricks using in the RL; Section 3 elaborates the detail of gradual promotion training strategy; Section 4 presents the result of gradual promotion training strategy and a series of ablation experiments; Section 5 makes a conclusion about this paper.

II. Related work

Generally speaking, the problem of reinforcement learning can be converted to the problem of solving Markov Decision Process (MDP), that is to say, it can be solved using traditional algorithms. However, the card games have obvious characteristic of imperfect information [6], which makes it require more complex reasoning than similar sized perfect information games. Therefore, if card games directly use the common algorithm like Temporal Difference Learning (TDL) [17, 18], Policy Gradient (PG) [19, 20], Full-Width Extensive-Form Fictitious Play (XFP) [25] and Counterfactual Regret Minimization (CFR) [21] etc., the results are often not ideal. Consequently, You *et al.* [15] considered introducing expert knowledge and combining it with RL algorithms such as Combination Q-network (CQN) in order to reduce the uncertainty of action information and try to make the agent do the right decision. Busoniu *et al.* [22] applied prior knowledge to accelerate the RL algorithm called online least-squares policy iteration (LSPI). The LSPI with prior knowledge learns much faster than the original algorithm. However, these methods still cannot defeat the agent that has been trained by heuristic learning [16].

With the improvement of computation capacity, the current mainstream method is to use the powerful expression ability of neural networks to simulate Q and/or policy functions with neural networks, such as Deep Monte Carlo (DMC) [5], Deep Q-learning (DQN) [13, 14], Neural Fictitious Self-Play (NFSP) [7, 26], and Deep Counterfactual Regret Minimization (DCFR) [23, 24]. However, these types of algorithms are mostly used in single agent tasks. For multi-agent tasks like card games, it is necessary to consider whether it is separate training or centralized training and the correlation between each agent. As a result, it will lead to a problem: due to the policy changes, an unbalanced environment is prone to appear and it's hard to achieve Nash Equilibrium. Therefore, some researchers proposed algorithms such as multi-agent deep Q-learning (MDQN) [27-29] and multi-agent deep deterministic policy gradient (MADDPG) [30]. Their main idea is to aggregate the observations of all individual agents for training. This is an improvement to the multi agent reinforcement learning (MARL) problem, but they rarely pay attention to different training strategy problems, especially in card games, which is also a gap that this paper will fill.

In addition, in view of the amnesia nature of neural networks [1], many researchers have proposed corresponding methods to compensate for it to a certain extent. Hasselt *et al.* [31] proposed double estimators that used double Q-learning to reduce the overestimation of action values. This idea has been proven to converge to the optimal policy, but the amount of calculation and complexity have also increased double. Also, experience replay was proposed [32-34] and was widely used in many RL scenarios, but Lin and Foerster *et al.* [34, 35] showed that only when the environment remains constantly unchanged, experience replay will play an effective role. For card games, in fact, past experience may not be fully relevant to the current state, and may even be harmful. Besides, there are some other methods such as: multiagent credit assignment [36] and multitask learning [37, 38], but they are all not particularly effective in the imperfect information scenarios.

Besides, Zhang *et al.* [39] proposed an efficient training technique for MARL in combat tasks, which mainly focused on the improvement of training techniques and used some methods kind of similar to this paper. However, Zhang's method only contains one stage and ignores the continuity of gradual training. As a result, this paper goes a step further and firstly combines single and multi-combat stage together, which achieve a better performance without adding extra computation complexity.

III. methods

A. Preliminary

In the game of Texas hold'em, the agent is simulated using DQN and NFSP as baselines. According to the structure of the neural network, this paper do not specifically design an overly complex neural network structure. One is because that a relatively simple network can also achieve a good effect. The second reason is that this paper focuses on explaining a training strategy: by improving the training strategy, using an efficient gradual promotion training strategy, the model can perform better without changing the algorithm structure. Therefore, the network mainly adopts a simplest structure.

B. Single Combat Stage

The whole pretrained process of single combat stage is presented in the Fig. 1 and it consists of three main parts: scenario-transfer training, rule-based training and self-playing training, and the mixed part. The mixed agent4 comes from the original agent that combat against the agent1, agent2 and

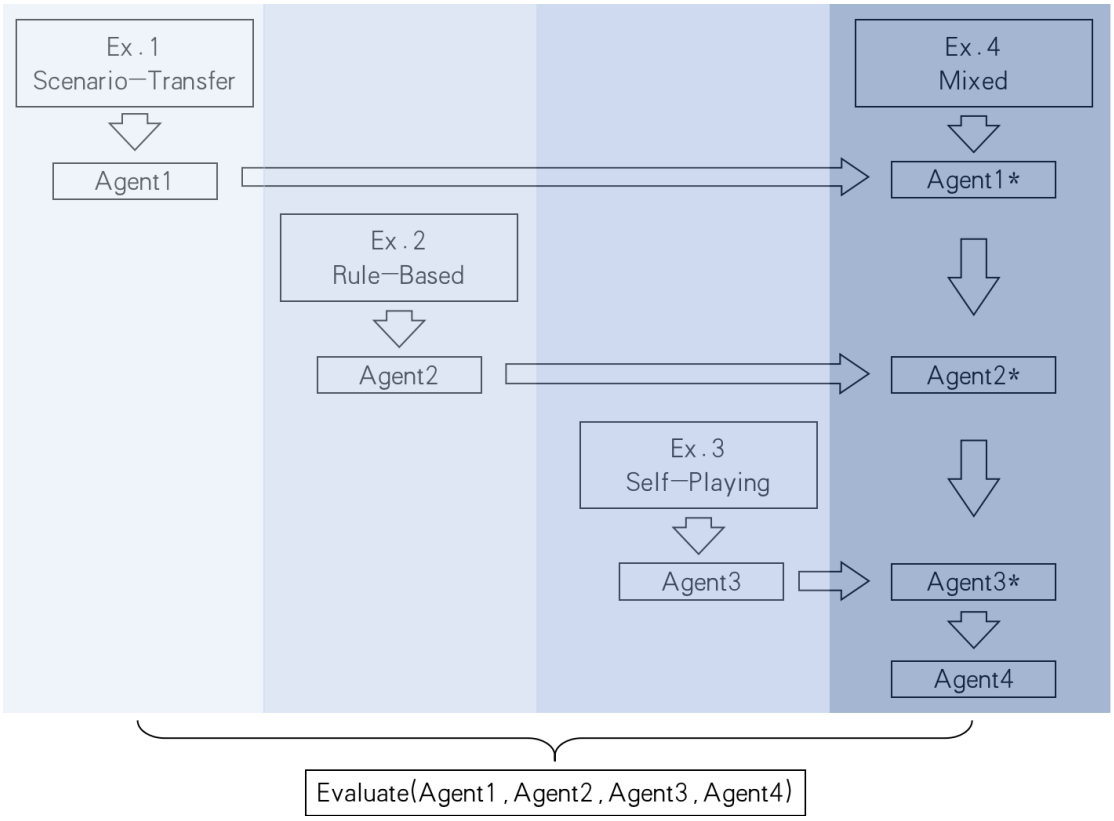

Figure 1. Single Combat Stage. Ex.1,2,3,4 refers to different training technique part (like scenario-transfer training, rule-based training, self-playing training and the mixed part). In the Ex.4, the * refers to the agent that combat against the former generated agent (like agent1,2,3,4). Let it play against the previously generated agents one by one, so finally get the Agent 4.

agent3 one by one and update its parameters through each process.

After getting these four agents, evaluate these through combat and return their average reward. Through the evaluation, which agent performs best can be seen clearly by the converge speed and average reward.

1) Scenario-Transfer Training

For the much more complex environment like card games, if you start training from scratch, it will not only slow down the convergence speed, but also lead to poor learning effects [40]. In fact, it is difficult to directly train a good model for imperfect information scenarios. Therefore, scenario-transfer training [39] is used. The detail algorithm is in Table I.

A few simple and easy-to-beat scenarios are designed as transition training, including random agents and some low-level agents. The reinforcement learning model is trained in these simple scenes one by one, and the training experience in each scene is stored, forming a mechanism that gradually adjusts the environment as the scene changes. To be specific, the model is actually a neural network model used to determine the next action the agent should take. The training of the model is to update its parameters. The parameters of model are randomly initialized at first. Then train a set of models of pre-designed scenes one by one. The parameters of the model are updated through these trainings to achieve the purpose of scenario-transfer training.

TABLE I. THE ALGORITHM OF SCENARIO-TRANSFER TRAINING

Algorithm 1: Scenario-Transfer training

Input: A set of pre-designed models
Output: Agent 1

```
Randomly initialize the parameters of model
For each pre-designed model do
    For each episode do
        Train the Agent 1 for this scenario
        Update parameters of Agent 1
    End for
End for
```

2) Rule-Based Training

This paper also establishes an agent that is based on the basic rules of Texas hold'em and set an agent to combat against with the rule-based agent in order to make sure that agent can learn the basic rules through the error and trial. See Table 2 for detail. The rules knowledge can actually be viewed as a set of prior knowledge. At any time, it can guarantee the agent not do some odd actions and choose the right action that at least satisfy the basic rules even though this action may not be the optimal choice.

TABLE II. THE ALGORITHM OF RULE-BASED TRAINING

Algorithm 2: Rule-Based training

Input: A set of rules R (including N rules)
Output: Agent 2

```
Randomly initialize the parameters of Agent 2
For each episode do
    Get observation S for the opponent
    i = 1, (i = 1, 2, 3 … N)
    For i < N do
        If a rule r ∈ R is matched then
            Get the action according to r
        Else
            Compute action based on Agent 2
        End if
        i ++
    End for
    Train the Agent 2 for this scenario
    Update parameters of Agent 2
End for
```

The designed rules contain the basic knowledge of Texas hold'em, including when to call or fold and when to raise the bet size. By applying the basic rules, the agent cannot become exceptionally excellent but can become one that knows to maintain a certain profit, being relatively conservative.

3) Self-Playing Training

Self-playing training technique has already been successfully applied in many games such as Go game, chess and shogi game. The main usage is easy. The detail algorithm is demonstrated in Table 3. Copy the trained model for one agent and view it as an opponent and then train the model by combat against each other. Also, update the parameters of models simultaneously in order to improve the agent through the training.

TABLE III. THE ALGORITHM OF SELF-PLAYING TRAINING

Algorithm 3: Self-Playing training

Input: An initial model
Output: Agent 3

```
Randomly initialize the parameters of model
Copy the model as the opponent
For each episode do
    Get the observation S
    Compute the action through Agent 3
    Compute the opponent action
    Train the Agent 3 for this scenario
    Update parameters of Agent 3 and its opponent
End for
```

4) Mixed Training

Finally, the above three algorithms are combined. To be specific, the agent generated at each step is used as the intensive opponent agent at this part, and a randomly initialized agent is used to conduct adversarial training with it, and the parameters are updated synchronously. Through this process, the agent gradually learns more from the difficult environments through the single combat. These three mixed learning processes are actually complementary. The first thing that agent learns is the simplest scenario, which enables the agent to make actions without randomness and learn some basic skills; then, through the learning of the prior knowledge, the agent can learn the most basic rules and beat the agent consisted by the rules. However, rule learning is relatively conservative to a certain extent. Therefore, a third process, self-playing training, is set up. Through self-playing, the right to make actions is handed over to the model itself, so that it can learn the winning method suitable in the single agent combat.

C. Multi Combat Stage

At the multi combat stage, the environment's difficulty and complexity is enhanced through adding more powerful agents. Through this stage, the agent can gradually learn advanced playing way, such as making full use of others' call and bet size and viewing the Texas Hold'em more exhaustively. As the Fig. 2 has shown, there are five different agents in the environment and the most powerful agent is obtained through this process. The Agents 1-4 are just the agents getting from the former stage. The strongest agent (Agent *) from Multi combat stage is obtained, then replace the agent that generating the strongest agent with a totally new and untrained agent, and evaluate these two agents in order to see whether the gradual promotion strategy makes the agent work better.

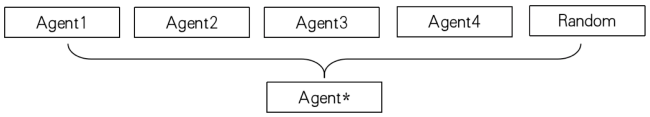

Figure 2. Multi Combat Stage. Agent 1-4, generated by various techniques presented in the Section 3.2, are put together with a random agent at the multi combat stage in order to further improve the agents' capacity and hope to get a strong agent, named Agent *.

For the five different agents chosen to add in the environment, it is not totally random and arbitrary. After considering various aspects of the Texas Hold'em game, five different agents are added into the environment. Agent1 is like a "rookie" in the game. Although it is not strong, it is also indispensable in training. Through the observation, some of its card playing methods will surprise others. If a trained strong model has not seen it or is not familiar with it, it may be misjudged by it, leading to bad and unpredicted consequences. Agent2 is mainly based on rules. It resembles the well-behaved players in the game. However, through observation, Agent2 has always been conservative and will adopt the fold strategy when the cards are not particularly good. In the long term, if play 10,000 rounds and calculate the average reward, the reward for Agent2 is relatively good. So, this agent is added to suppress the over-aggressive performance of the trained model. For Agent3, it is obtained through two neural networks fighting against each other. They are good at finding opportunities, discovering the opponents' loopholes and operational errors. As a result, Agent3 is put into the multi environment. Agent 4 is trained through the mixed process. From the intuition, it should be the most powerful agent generated at the single combat stage. It is widely acknowledged that better opponents make better

agents. Therefore, it is added into the environment, hoping to train the strongest agent, being expert at multi-players Texas Hold'em game. Besides, it is worth noting that a random agent is added, which is very necessary and important. Because the model must be robust and cannot be disturbed by obvious deviations. Just like some existing reinforcement learning models, the trained model will conduct targeted learning for obvious outliers, and need to guarantee that the model is not disturbed by these points.

IV. Results & Discussion

A series of experiments are conducted based on the card games environment developed by RLCard [8]. The experiment will combine the training techniques with DQN and NFSP respectively. A fully connected neural network is adopted as neural network structure. To be specific, a total of three full linear connection layers are set up, each layer contains 128 neurons, ReLU is used as the activation function, and Dropout is added to the last layer in order to increase the robustness and generalization ability of the model. Comprehensive experiments are made so that can illustrate the performance of gradual promotion training strategy clearly.

A. Performance of Single Combat Stage

Firstly, construct the combat scenario including the random agent and some easy-to-beat scenarios. Calculate the average reward and winning rates during the evaluation process, as shown in the Fig. 3.

According to the experimental results, it can be clearly found that both DQN and NFSP have learned the scene information well, and the average reward and winning rates increase gradually, eventually to 2.5 average reward and 96% winning rates. Moreover, the convergence effect was good, and the convergence was basically achieved after 250 episodes of training. In addition, it is worth noting that the training effect of DQN is better than NFSP in scenario-transfer training: NFSP's winning rate is about 5% higher than DQN's.

Next, rule-based training technique is used. A series of rules are set according to the Texas Hold'em game. For example, if the card in hands is a pair, raise will be chosen; if my cards are two small points and don't have much relation with the public cards, it triggers a fold, etc. The designed rules are relatively conservative, not particularly radical. Through observation, it can be found that the rules train the agent to make fold judgment in most cases in order to avoid itself losing too much money when the cards in hand aren't so good. This is actually the thought of most normal player, therefore, the agent can learn normal card playing strategies through this process.

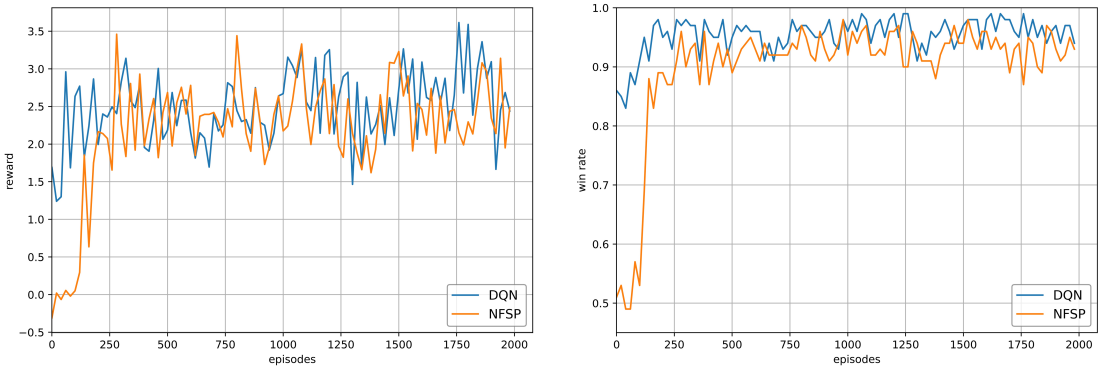

Figure 3. The Result of Scenario-Transfer training.

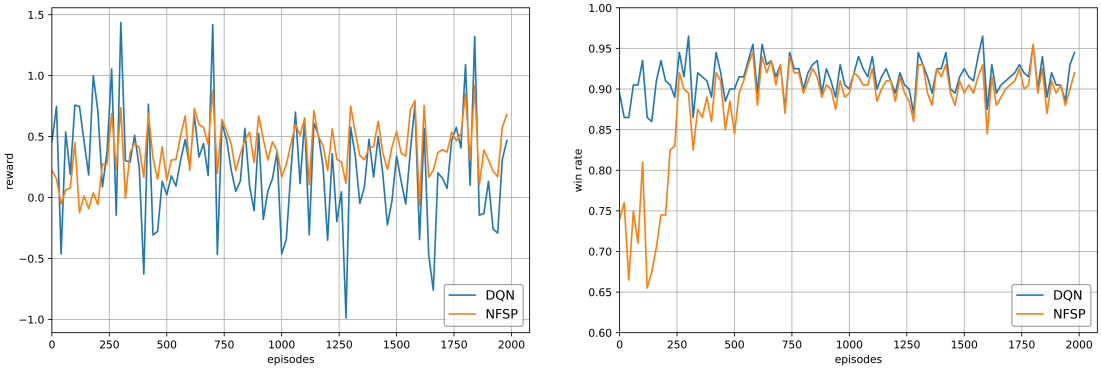

Figure 4. The Result of Rule-Based training.

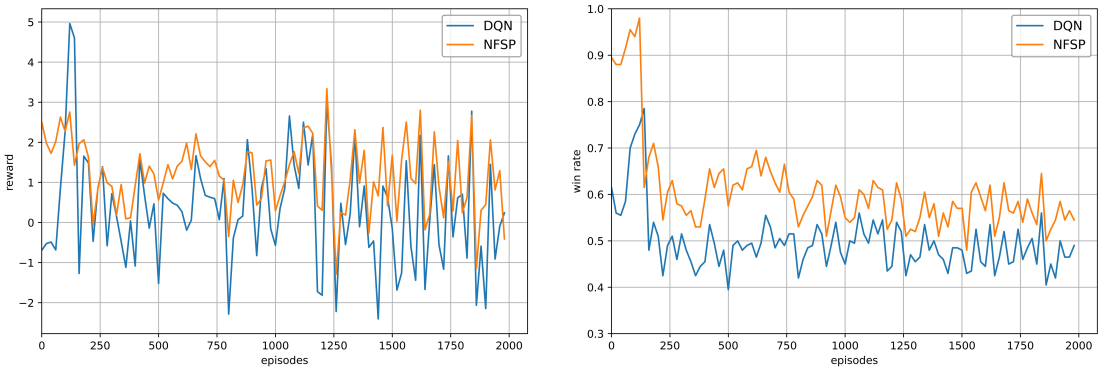

Figure 5. The Result of Self-Playing training.

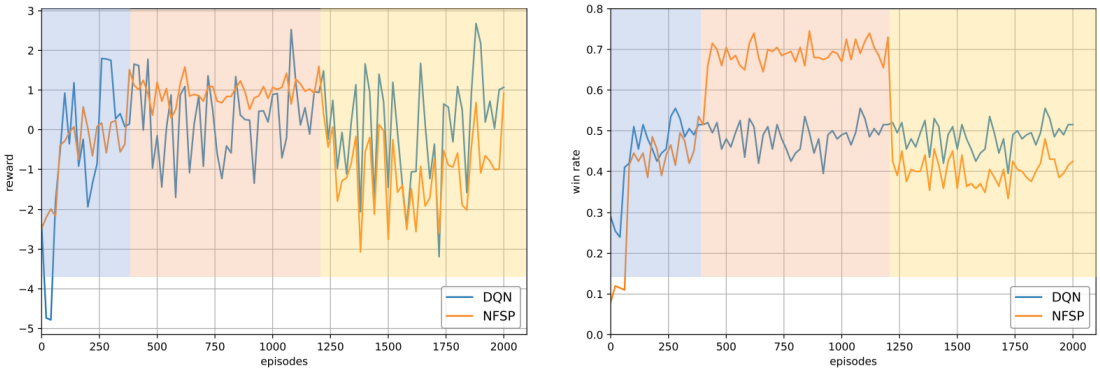

Figure 6. The Result of Mixed training. The blue part represents Scenario-Transfer training, the orange part represents Rule-Based training and the yellow part represents Self-Playing training. Scenario-Transfer training contains 400 episodes and the rest contains each 800 episodes.

The experiments showed in the Fig. 4 indicate that both DQN and NFSP can learn quickly from the rule and master the basic playing strategies. DQN and NFSP's winning rate through the rule-based agent are all very high, almost up to 93%. Also, the average reward is above the zero, which means that there may be a few games where the game is lost due to the lack of luck, but overall, if play 2000 games (as have been done in the experiment), the average return is always positive and around 0.5. It is quite good given that only the conservative rules are applied as the agent's opponent.

Thirdly, a totally same agent is used as the opponent added into the environment and conduct the single combat stage. Initialize both the agents and train these with 2000 episodes. Conduct the evaluation every 20 episodes and evaluate 100 episodes. The result is showed in Fig. 5.

As can be clearly seen from the result, with the opponent becoming stronger, the performance of agent decreased first and then increased subsequently. In fact, it is reasonable: the structure of agents on both sides of the combat is actually the same, and both sides are constantly learning from rewards to modify their network parameters. However, it can be found that the final average reward of agent through self-playing training can be around 1 and the winning rate is above 50% in the middle and late stages of training. In addition, NFSP obviously performed better in self-playing training. The reason may be that NFSP has introduced the technique of self-play itself and is therefore more adaptable to this training process.

After presenting the basic performance of three different training techniques, it comes to the mixed training , illustrated in the Fig. 6. The training process is split and a new agent is trained by playing against the models obtained from scenario-transfer training, rule-based training, self-playing training respectively. By comparing Fig. 3, 4 and 5, several interesting phenomena are found.

First of all, the jitter range of NFSP is significantly larger than that of DQN, especially in the adversarial training with Agent2, which shows a sharp increase. It indicates that NFSP after the adversarial training of Agent1 was obviously stronger than that of Agent2, and the improvement that Agent2 can bring to it was not very obvious. And in the Fig. 6, it can be seen that DQN performs significantly better than NFSP at this stage, with a winning rate almost 10% higher than DQN.

Additionally, when using the enhanced Agent generated in the above three training processes, the evaluation during the training process shows that it seems to have a certain decrease in average reward and winning rate, which in fact is actually related to the strength of the opponent. Therefore, verification experiments are conducted in Table 4. Agent1, 2, 3 and mixed agent are put into an environment for experiment, and the results show that the agent obtained through mixed training is obviously much better than other agents in DQN, which is reflected in that the average reward is 0.99 and the winning rate is 27%. However, although the agents using NFSP have a great improvement compared with the agent got by scenario-transfer training and rule-based training, it is still worse than the agent obtained by self-playing training.

TABLE IV. THE EVALUATION EXPERIMENTS

Algorithm \ Evaluation	DQN		NFSP	
	Average reward	Winning rate	Average reward	Winning rate
Scenario-Transfer	-0.2866	25.74%	-7.2926	13.2%
Rule-Based	-0.797	25.34%	-8.6784	11.32%
Self-Playing	0.0914	26.16%	**15.9261**	**57.46%**
Mixed	**0.9922**	**27.04%**	0.0449	20.86%

B. *Performance of Multi Combat Stage*

After the single combat stage, it comes to the multi combat stage. A total of five agents are set up on the Multi combat stage. For details, please refer to Section 3.3. The intuitive idea is to use the strongest agent generated in the previous stage and put it into the second stage for training. Also, Agent1, 2, 3 generated by single combat stage and a random agent are added at the same time. The purpose is to make the agent grasp the overall situation and not be disturbed by outliers. Then, conduct the ablation experiments by replacing the strongest agent with a completely new and untrained agent. See Fig. 7 - 9 for details.

Experiments related to DQN are showed in Fig. 7 - 8. In Fig. 7, the strongest agent generated by mixed process is used. However, it shows that the strongest agent seems not to perform better than the untrained agent, and the average reward curve and winning rate curve are almost the same. It may be a little puzzled. So another experiment is conducted. The Agent3 generated by self-playing training is replaced with a completely new and untrained agent and the ablation experiments are made identically, shown in Fig. 8. It can be seen clearly that Agent3 outperforms a lot, approximate to 10% higher than the new DQN in both average reward and winning rate. It shows that the gradual promotion training strategy actually works. The agent can perform better in the multi combat stage through the pretraining, namely the single combat stage.

In Fig. 9, because the most powerful agent is Agent3 generated by self-playing training, the ablation experiments are conducted to compare the new NFSP with the Agent3. From Fig. 9, the results are even better than the same DQN's experiments. The agent trained through the single combat stage thoroughly defeat the un-pretrained agent. Compared with the un-pretrained agent, the average reward is improved by 25% and winning rate is improved by 44%. It also proves that the single combat stage is useful and necessary, and makes the agent perform better in the multi agents environment.

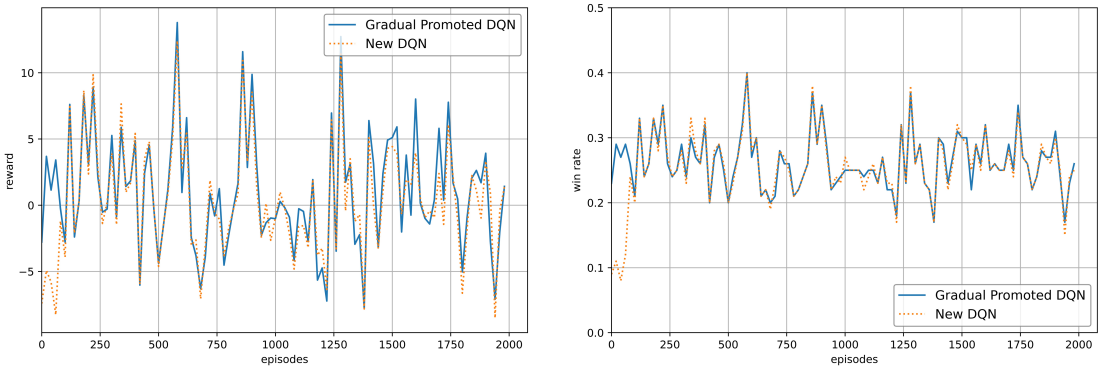

Figure 7. Comparison of DQN Agent obtained by mixed training and New DQN agent.

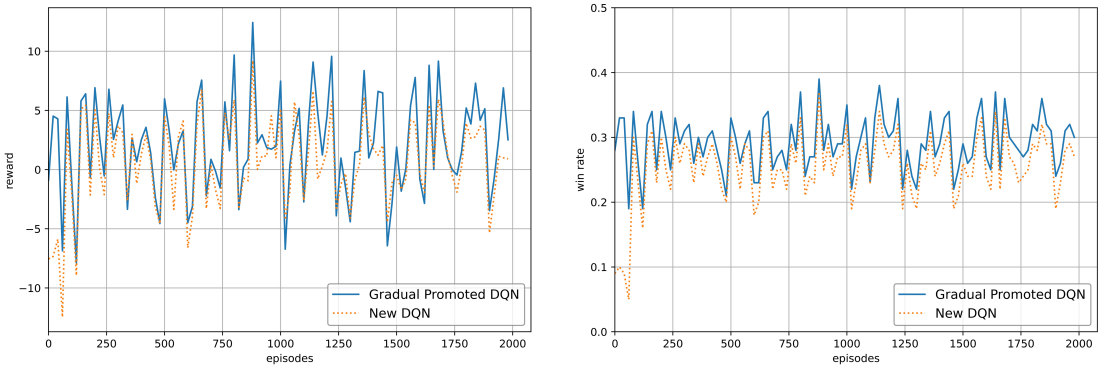

Figure 8. Comparison of DQN Agent obtained by self-playing training New DQN agent.

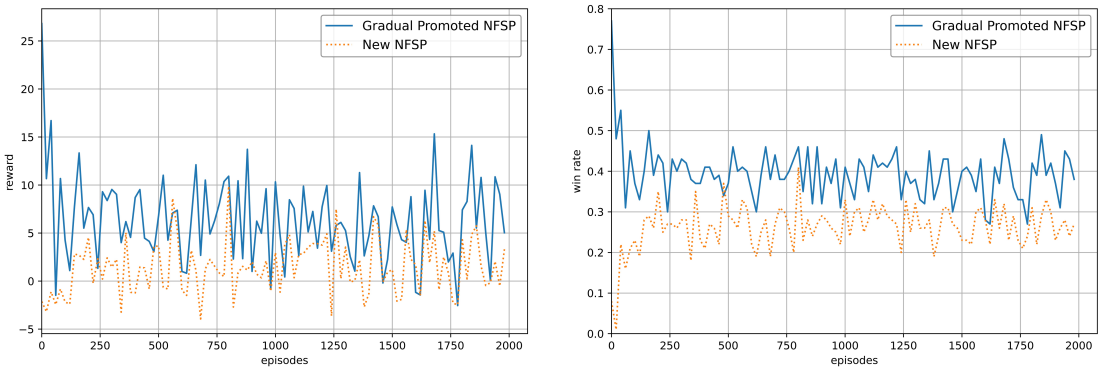

Figure 9. Comparison of NFSP Agent obtained by self-playing training New NFSP agent.

V. CONCLUSION

In conclusion, this paper study the card games problem with multi-agent reinforcement learning methods. Different from some existing studies, which focuses on using different types of learning methods, this paper focuses on exploring a training strategy that is universal, easy to operate and easy to transplant. So, a gradual promotion training strategy is proposed, which includes two stages: single combat stage and multi combat stage. Scenario-transfer training, rule-based training, self-playing training and mixed training are used at single combat stage to obtain a strong single agent, namely the pretrain process. Then, at multi combat stage, multi-agent training is introduced, which increases the complexity of the game, so that the strong single agent gradually adapts to the multi-agent task, and the strong multi agent is obtained. Compared with the un-pretrained agent, the average reward is improved by 25% and winning rate is improved by 44%. In fact, the gradual promotion training strategy are a general method that can have a wide use not only in the card games, but also in other kinds of tasks. It is an interesting process to explore the effectiveness of this strategy in various fields.

REFERENCES

[1] Hernandez-Leal, P. , B. Kartal , and M. E. Taylor . "A survey and critique of multiagent deep reinforcement learning." Autonomous Agents and Multi-Agent Systems 33.6(2019):750-797.

[2] Shoham, Yoav , R. Powers , and T. Grenager . "If multi-agent learning is the answer, what is the question?." Artificial Intelligence 171.7(2006):365-377.

[3] Papoudakis, G. , et al. "Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning." (2019), unpublished.

[4] Nguyen, T. T. , N. D. Nguyen , and S. Nahavandi . "Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications." IEEE Transactions on Cybernetics PP.99(2020):1-14.

[5] Zha, Daochen , et al. "DouZero: Mastering DouDizhu with Self-Play Deep Reinforcement Learning." (2021), unpublished.

[6] J Niklaus, et al. "Survey of Artificial Intelligence for Card Games and Its Application to the Swiss Game Jass." Swiss Conference on Data Science Document Image and Voice Analysis Group (DIVA), University of Fribourg, Switzerland; Docu, 2019.

[7] Heinrich, J. , and D. Silver . "Deep Reinforcement Learning from Self-Play in Imperfect-Information Games." (2016).

[8] Zha, D. , et al. "RLCard: A Toolkit for Reinforcement Learning in Card Games." (2019).

[9] Busoniu, L. , R. Babuska , and B. D. Schutter . "A Comprehensive Survey of Multiagent Reinforcement Learning." IEEE transactions on systems, man and cybernetics. Part C, Applications and reviews 38.2(2008):p.156-172.

[10] Moradi, M. . "A centralized reinforcement learning method for multi-agent job scheduling in Grid." ICCKE 2016 IEEE, 2016.

[11] Zhao, Y. , et al. "Decentralized Online Learning: Take Benefits from Others' Data without Sharing Your Own to Track Global Trend." (2019).

[12] Chen, G. . "A New Framework for Multi-Agent Reinforcement Learning -- Centralized Training and Exploration with Decentralized Execution via Policy Distillation." (2019).

[13] Mnih, V. , et al. "Playing Atari with Deep Reinforcement Learning." Computer Science (2013).

[14] Kurek, M. , and W. Jaskowski . "Heterogeneous team deep q-learning in low-dimensional multi-agent environments." 2016 IEEE Conference on Computational Intelligence and Games (CIG) IEEE, 2016.

[15] You, Y., Li, L., Guo, B., Wang, W., and Lu, C. Combinational q-learning for dou di zhu. arXiv preprint arXiv:1901.08925, 2019.

[16] Jiang, Q., Li, K., Du, B., Chen, H., and Fang, H. Deltadou: Expert-level doudizhu ai through self-play. In Interna- tional Joint Conferences on Artificial Intelligence, 2019.

[17] R. Sutton and A. G. Barto, “Reinforcement learning: An introduction,” IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council, vol. 9, p. 1054, 02 1998.

[18] R. Sutton and A. G. Barto, “Reinforcement learning: An introduction,” IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council, vol. 9, p. 1054, 02 1998.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” CoRR, vol. abs/1707.06347, 2017.

[20] H. Charlesworth, “Application of self-play reinforcement learning to a four-player game of imperfect information,” CoRR, vol. abs/1808.10442, 2018.

[21] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, “Regret minimization in games with incomplete information,” in Advances in Neural Information Processing Systems 20. Curran Associates, Inc., 2008, pp. 1729–1736.

[22] L. Busoniu, B. De Schutter, R. Babuska, and D. Ernst, ‘‘Using prior knowledge to accelerate online least-squares policy iteration,’’ inProc.IEEE Int. Conf. Automat., Qual. Testing, Robotics, vol. 1, May 2010,pp. 1–6.

[23] N. Brown, A. Lerer, S. Gross, and T. Sandholm, “Deep counterfactual regret minimization,” CoRR, vol. abs/1811.00164, 2018.

[24] N. Brown and T. Sandholm, “Superhuman ai for heads-up no-limit poker: Libratus beats top professionals,” Science, 2017.

[25] Hendon, E. , H. J. Jacobsen , and B. Sloth . "Fictitious Play in Extensive Form Games." Games & Economic Behavior 15.2(2013):177-202.

[26] K. Kawamura, N. Mizukami, and Y. Tsuruoka, “Neural fictitious self-play in imperfect information games with many players,” in Computer Games. Cham: Springer International Publishing, 2018, pp. 61–74.

[27] K. Mateusz and J. Wojciech, ‘‘Heterogeneous team deep q-learning inlow-dimensional multi-agent environments,’’ inProc. IEEE Conf. Comput.Intell. Games, Sep. 2016, pp. 1–8.

[28] Lin, Kaixiang , et al. "Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning." the 24th ACM SIGKDD International Conference ACM, 2018.

[29] M. Aleksandra, T. Tegg, S. Chae-Bong, K. Daniel, and S. Aleksei,‘‘Deep multi-agent reinforcement learning with relevance graphs,’’ 2018,arXiv:1811.12557. [Online]. Available: https://arxiv.org/abs/1811.12557

[30] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, ‘‘Multi-agent actor-critic for mixed cooperative-competitive environments,’’ inProc. Adv. Neural Info. Process. Syst., Long Beach, CA, USA, Dec. 2017,pp. 6379–6390.

[31] H. V. Hasselt, Double Q-learning, in: Advances in Neural Information Processing Systems, 2010, pp. 2613– 2621.

[32] Mnih, V. , et al. "Playing Atari with Deep Reinforcement Learning." Computer Science (2013).

[33] L. J. Lin, Programming robots using reinforcement learning and teaching., in: AAAI, 1991, pp. 781–786.

[34] Lin, L. J. . "Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching." Machine Learning 8.3-4(1992):293-321.

[35] J. N. Foerster, Y. M. Assael, N. De Freitas, S. Whiteson, Learning to communicate with deep multi-agent reinforcement learning, in: Advances in Neural Information Processing Systems, 2016, pp. 2145–2153.

[36] A. K. Agogino, K. Tumer, Analyzing and visualizing multiagent rewards in dynamic and stochastic domains, Autonomous Agents and Multi-Agent Systems 17 (2) (2008) 320–338.

[37] M. E. Taylor, P. Stone, Transfer learning for reinforcement learning domains: A survey, The Journal of Machine Learning Research 10 (2009) 1633–1685.

[38] Rich, and Caruana. "Multitask Learning." Machine Learning (1997).

[39] Zhang, G. , et al. "Efficient Training Techniques for Multi-Agent Reinforcement Learning in Combat Tasks." IEEE Access 7(2019):109301-109310.

[40] Pan and Q. Yang, ‘‘A survey on transfer learning,’’IEEE Trans. Knowl.Data Eng., vol. 22, pp. 1345–1359, Nov. 2010