Proceedings of the 4th International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/2026.38760

From Board Games to Esports: Multi-agent Deep Reinforcement Learning Across Game Types

Jiayi Qi

Silesian College of Intelligent Science and Engineering, Yanshan University, Qinhuangdao, China
qijiayi191@outlook.com

Abstract. Multi-agent deep reinforcement learning has made significant progress in game applications. However, it is still unknown that why some algorithms work well in a specific game but not in others. This paper proposes a classification-based survey covering five game categories: perfect-information board games, imperfect-information card games, real-time strategy games, sports games, and multiplayer online competitive games. Moreover, it analyzes its key features, technical challenges, and representative algorithms for these kinds of game categories .Board games use Monte Carlo Tree Search which is accompanied by self-play; card games use counterfactual reasoning for information that is imperfect; real-time strategy games use centralized critics, policy optimization, and value decomposition; sports games have competitive self-play function in continuous action environments; multiplayer online games use methods that use large-scale self-play and population learning to make systems more scalable. Comparison between different types shows shared challenges including non-stationarity, credit assignment, and scalability, and introduce three future directions to lead algorithm design between different game types.

Keywords: Multi-agent deep reinforcement learning, Game AI, Game types

1. Introduction

Deep reinforcement learning (DRL) combines deep neural networks with reinforcement learning, and it has helped AI make big progress in recent years. As researchers start to study systems with multiple agents instead of just one, multi-agent deep reinforcement learning (MADRL) has become a key way to understand how groups of decision-makers interact in shared spaces [1]. The gaming industry is a good place to test these ideas. Newzoo's Global Games Market Report 2026 says the global games market will reach $213.9 billion in 2026, up 6.1% from last year. About 3.70 billion people play games around the world, which is 62.0% of everyone online [2]. These players use different devices: 977 million on PC, 651 million on consoles, and 3.10 billion on mobile phones [2]. This large and varied group of players has created a lot of demand for better game AI [3].

MADRL has been used in many games with good results, but it is still not clear why some algorithms work well in certain games but not in others.Board games with full information have huge search spaces, so tree search methods work best. Silver et al.'s AlphaZero used self-play and Monte Carlo Tree Search (MCTS) to master chess, shogi, and Go [4], and Anthony et al.'s Expert Iteration used tree search with deep networks for the game of Hex [5]. Card games like poker work

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.38760

differently because players cannot see all the information. Moravčík et al.'s DeepStack used depth-limited search and counterfactual value networks to play heads-up no-limit poker in an expert level [6], and Brown et al.'s Pluribus beat professional players in six-player poker which uses similar search and self-play [7]. These methods would not work well for board games. MOBA games like Honor of Kings have very large state and action spaces.Therefore, they need methods that can work with many agents. Ye et al. used curriculum self-play and policy distillation to handle it [8]. Sports games like Kurach et al.'s Google Research Football have continuous actions and 3D physics, so they also need different algorithm abilities [9]. Most surveys which exist now organize papers by algorithm type rather than by game type, so they do not compare between different game types [3, 10, 11]. In addition, common problems like non-stationarity, credit assignment, and scalability show up differently depending on the game [1]. These differences suggest that a survey organized by game type is needed.

This paper looks at how MADRL works in different game types to help fill this gap. Five game categories are covered: perfect-information board games, imperfect-information card games, real-time strategy games, sports games, and multiplayer online competitive games. This paper analyzes the main technical challenges and reviews representative algorithms . Comparisons across game types are also made to find shared and challenges of specific type, calculate how well algorithms transfer between types, and point out future research directions based on open problems from existing studies [1, 3].

2. Methodology

2.1. Perfect-information board games

AlphaZero [4]. AlphaZero combines Monte Carlo Tree Search (MCTS) with self-play and deep neural networks to reach a level beyond what humans can achieve. The core idea is that MCTS helps to improve the neural network's decisions. The network guides where to look in the search tree, and the results from each search are then used to train the network further which means creating a cycle of steady improvement. This method works for chess, shogi, and Go all in the same way. It shows that one general reinforcement learning algorithm can beat top human players between different board games, without needing pre-built opening strategies or endgame databases.

Expert Iteration (ExIt) [5]. ExIt works through repeated cycles of learning. In each cycle, a tree search produces high quality moves and a neural network which is called the "apprentice" learns from those moves by imitation. Once the apprentice gets better, it then leads the next round of tree search and the process repeats and improves steadily over time. The main idea that sets ExIt apart is keeping the search step and the learning step separate, so each one can make the other better. Tested on the game of Hex, ExIt showed that this step-by-step method can reach a high level of play when using much less computing power than methods that rely purely on self-play.

2.2. Imperfect-information card games

DeepStack [6]. DeepStack is mainly used for heads-up no-limit Texas Hold'em. It uses depth-limited search and counterfactual value networks to play imperfect information games. The main idea is easy: instead of computing the entire game tree which would be so large, the algorithm only looks a few ahead. Deep neural networks are used to estimate the value of game states at those deeper points, resulting in the algorithm can make decisions quickly during play. This method removes the need to precompute a full strategy before the game starts. The main contribution of DeepStack is

DOI: 10.54254/2755-2721/2026.38760

showing that deep networks can be regarded as a fast way to estimate counterfactual values, making the algorithm work well even for large games where older methods which based on tables would require too much memory.

Pluribus [7]. Pluribus beats professional players in six-player no-limit Texas Hold'em using self-play and depth-limited search. It was the first AI to reach level which human can not achive in a multiplayer game with hidden information. At its core, Pluribus uses a "blueprint strategy" which means a simplified version of the full game that serves as a starting point for decisions. During a game, Pluribus bases on this blueprint to search only a few moves ahead at each decision point and it adjusts its strategy as the game goes on.The main contribution of Pluribus is showing that a fairly straightforward self-play approach, accompanied by smart ways to make the game easier, can handle the complexity of multiplayer poker without massive computing ability. This was a big step improvement compared to earlier methods of only working for two-player games.

2.3. Real-time strategy games

MADDPG [12]. MADDPG follows a simple rule: train together, run separately. Each agent has its own decision-making network and its own evaluation network. During training, the evaluation networks can see everything which means the full state of the game and what every agent is doing. But when actually playing, each agent only uses its own local view to make decisions. This setup solves a common problem in multi-agent learning: when all agents are learning at the same time, the environment keeps changing from each agent's perspective, which make it hard to learn steadily. By having a central critic that sees everything during training, every agent gets a clear, stable target to learn from.MADDPG also trains multiple different strategies for each agent at the same time, which leads to more flexible and reliable behavior overall. The main contribution of MADDPG is showing that this "train together, play alone" approach works well for learning complex team strategies, whether the agents are working together or competing against each other.

MAPPO [13]. The famous Proximal Policy Optimization (PPO) algorithm is adapted by MAPPO for cooperative multi-agent settings. One shared evaluation network is used for all agents during training, while each agent's own decision network is kept for actual play. Drawing on PPO's core ideas — including generalized advantage estimation and a clipping mechanism — training stability is maintained by ensuring that each policy update does not change too much at once. The main takeaway from MAPPO is surprising: very strong performance across four widely used multi-agent benchmarks can be achieved by a straightforward PPO-based method, with no special add-ons or custom-built structures. The StarCraft Multi-Agent Challenge, Google Research Football, simple particle environments, and the Hanabi card game challenge are included among them.The common belief that off-policy methods are necessary for multi-agent scenarios is challenged by this result. Instead, it shows that standard on-policy methods can work just as well as long as researchers adjust their settings carefully.

QMIX [14]. QMIX is a method for cooperative multi-agent reinforcement learning. It works by breaking down the overall team value into individual values for each agent with one most significant rule: the overall value must always move in the same direction as each agent's individual value.To make this work, QMIX uses a "mixing network" where all weights are positive or zero. This structure guarantees that the best team action which is found by looking at the overall value matches what each agent would choose based on its own individual value. In other words, the team level decisions and individual level decisions stay consistent. The main contribution of QMIX is that it provides a mathematically solid way to decompose values, without losing too much flexibility. Compared to easier methods like VDN, QMIX's non-linear mixing network captures more complex

DOI: 10.54254/2755-2721/2026.38760

relationships between agents. QMIX achieved top results on the StarCraft Multi-Agent Challenge benchmark and has become a standard baseline that other methods are compared against.

2.4. Sports games

TLeague [15]. TLeague is a framework for training multiple reinforcement learning agents through competitive self-play. It was built because multi-agent training often needs huge amounts of data, and TLeague is designed to handle that efficiently. The framework includes several common algorithms for competitive self-play, and it can run on a single machine or on large clusters with both CPUs and GPUs, managed through Kubernetes. The main contribution of TLeague is that it achieves fast training speeds and works well when scaled up, while keeping its code structure organized enough that researchers can easily adapt it to other multi-agent problems. TLeague has performed well on several benchmarks, such as Google Research Football. These results show that competitive self-play works well as a general approach for multi-agent learning in team sports games with continuous actions.

2.5. Multiplayer online competitive games (MOBA/FPS)

2.5.1. Self-play based methods

OpenAI Five [16]. OpenAI Five is a reinforcement learning system that plays Dota 2 at a level beyond what humans can reach. It learns by playing against itself. This method is called self-play. The training happens on a very huge scale. The system processes roughly two million frames every two seconds and the training runs nonstop for ten months. It uses the PPO algorithm as its base, but the real achievement is showing that scaling up self-play can work for complex games that involve long sequences of decisions and hidden information. OpenAI Five showed that self-play reinforcement learning can beat world champions in esports games.

Honor of Kings AI [8]. This system plays Honor of Kings, a popular MOBA game with many different heroes. It learns through two main techniques: curriculum self-play and policy distillation.

Curriculum learning means the AI starts by playing against easy opponents and gradually faces stronger ones. This step-by-step method helps learning stay stable, even though the game has a huge number of possible situations. Policy distillation takes what many different strong strategies have learned and combines it into a single model. The main point is that by putting curriculum learning together with distillation, then it becomes possible to train AI agents for large, commercially released MOBA games.

2.5.2. Population learning methods

For the Win (FTW) [17]. FTW is a multi-agent AI that plays Quake III Arena Capture the Flag at about the same level as a human player. Instead of training just one agent, it trains a whole group of agents that play against each other. Each agent develops its own way of judging which moves are good. This helps them gradually learn cooperative behaviors like working with teammates and capturing the flag without humans having to tell them exactly what to do. The agents also use LSTM memory so they can remember what happened earlier in the game. FTW shows that training a diverse group of agents together can produce solid team strategies, even when no one teaches them how to cooperate.

DOI: 10.54254/2755-2721/2026.38760

Anytime PSRO [18]. PSRO is a multi-agent reinforcement learning method based on game theory, mainly for two-player zero-sum games. It grew out of a method called double oracle, which can find Nash equilibria but sometimes performs worse in the middle of training before it gets better. Anytime PSRO fixes this problem by making sure the strategy gets better at every single step, instead of only being good at the very end. It uses a regret-matching approach and adds new policies one at a time. When tested on Leduc poker, PSRO showed that population-based methods can work both quickly and correctly in theory at the same time.

3. Discussion

3.1. Challenges

3.1.1. Challenge 1

Non-stationarity. When multiple agents learn at the same time, the environment keeps changing from each agent's point of view. This breaks a basic assumption of traditional reinforcement learning which means that the environment follows stable and predictable rules.This problem shows up in different ways depending on the game type. In competitive card games, it happens because opponents keep changing their strategies [6, 7]. In cooperative real-time strategy games, it comes from all agents updating their policies at the same time [12]. MADDPG helps reduce this problem by using centralized critics during training [12], but the issue still exists when there are too many agents. In large-scale settings, training with a central critic becomes too slow and expensive to be practical [13].

3.1.2. Challenge 2

Credit assignment. In cooperative games, a key problem is figuring out which agent deserves credit when the team does well — or blame when it does poorly. The team only gets one overall reward, but it needs to be split up so each agent knows how it contributed.Some methods like QMIX [14] split the team value by following a strict rule about how individual values relate to the overall value. This works but limits what the method can express. Other methods like COMA use gradients and offer more flexibility, but their results can be uneven depending on the game [3, 10]. In MOBA games with huge state spaces, credit assignment gets especially hard because there are so many agents taking so many different actions at once [8].

3.1.3. Challenge 3

Scalability and cross-type transfer. Algorithms that work well in small, simple game simulations often do not work as well when scaled up to larger, more realistic settings. OpenAI Five succeeded by using a huge amount of computing power [16], but that approach costs too much for most people to use. Also, algorithms do not move from one type of game to another easily. For example, MCTS works well in board games [4] but does not apply to games with hidden information [6, 7]. Similarly, value decomposition methods built for discrete-action strategy games [14] have not yet been tested in sports games where actions are continuous.

DOI: 10.54254/2755-2721/2026.38760

3.2. Future directions

3.2.1. Direction 1: dynamic adaptation mechanisms

There are several promising ways to fix the non-stationarity problem. Curriculum learning starts with simple tasks and slowly increases difficulty, giving agents time to learn each level before moving on [8]. Opponent modeling means trying to predict what other agents will do, so you can plan better even when their strategies keep changing [6, 7]. Expert Iteration uses a step-by-step approach. A tree search expert teaches a neural network apprentice, and they improve together over time [5]. FTW uses a whole population of different agents that play against each other, so no single strategy can dominate and learning stays stable [17]. Anytime PSRO also uses population-based methods but adds theoretical guarantees, making sure strategies keep getting better instead of sometimes getting worse [18]. TLeague's competitive self-play framework is another example. It matches agents of similar skill levels so they can learn from fair fights [15]. In the future, researchers could try combining these ideas. They could adjust difficulty based on how well agents are doing automatically, build pools of diverse opponents to prevent overfitting, and use meta-learning so agents can quickly adapt when the environment changes. These techniques could help agents keep performing well even in games and real-world systems where conditions never stop changing.

3.2.2. Direction 2: unified credit assignment frameworks

Attention mechanisms [3] and learnable value decomposition [3, 14] are promising directions. They work by dynamically figuring out which agents contribute the most to the team's outcome. Adding interpretability techniques [10, 11] would also help. If it is possible to see exactly how credit is being assigned to each agent, it becomes easier to build frameworks that are clearer and work across more types of games. Bio-inspired approaches might also bring new ideas. They take inspiration from how natural selection shapes group behavior in nature [11].

3.2.3. Direction 3: sim-to-real transfer and cross-domain applications

MADRL first got started in games, but it can be used in many more real-world situations. Zhang et al. [19] showed this by applying CTDE-MASAC to multi-microgrid energy systems. They got nearly optimal results, and the system could make decisions in just milliseconds without needing a central controller. This proves that MADRL techniques first developed for games can work in real industrial settings. In the future, researchers should look into domain adaptation and sim-to-real transfer. These techniques would let algorithms trained in game simulations be used in areas like energy management, robotics, and transportation. Researchers should also add safety rules and interpretability features [10, 11]. These are necessary before people can trust MADRL in real-world systems, where mistakes can have serious consequences.

4. Conclusion

This survey looks at MADRL by game type. It reveals one key insight. Choosing a MADRL algorithm is not just a technical decision. It is a response to the basic rules and limits of the game itself. This finding matters for both researchers and people building real systems. Algorithm design should start with a clear understanding of the game. People need to know how information works in the game, what actions players can take, and how much coordination the game needs. A "one size

DOI: 10.54254/2755-2721/2026.38760

fits all" approach does not work well. The line between game AI and real-world uses keeps getting less clear. The lessons learned from different game types will become more and more valuable. They can help solve complex multi-agent problems in energy systems, robotics, and other areas. In the end, building MADRL algorithms that are more general, easier to understand, and easier to transfer will require connecting ideas across game types. The field needs to move from solutions that only work for one specific game. It should move toward shared principles that is suitable for anywhere multiple agents learn and interact together.

References

[1] Gronauer, S., & Diepold, K. (2022). Multi-agent deep reinforcement learning: A survey. *Artificial Intelligence Review, 55*, 895–943.

[2] Newzoo. (2026). Newzoo's global games market report 2026 (Free edition). Newzoo, Amsterdam.

[3] Li, Z., et al. (2025). A comprehensive review of multi-agent reinforcement learning in video games. *IEEE Transactions on Games*. arXiv:2509.03682.

[4] Silver, D., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science, 362*(6419), 1140–1144.

[5] Anthony, T., Tian, Z., & Barber, D. (2017). Thinking fast and slow with deep learning and tree search. *Advances in Neural Information Processing Systems (NeurIPS)*.

[6] Moravčík, M., et al. (2017). DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science, 356*(6337), 508–513.

[7] Brown, N., & Sandholm, T. (2019). Superhuman AI for multiplayer poker. *Science, 365*(6456), 885–890.

[8] Ye, D., et al. (2020). Towards playing full MOBA games with deep reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*.

[9] Kurach, K., et al. (2020). Google Research Football: A novel reinforcement learning environment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 4501–4511.

[10] Wang, Y., et al. (2025). Intelligent games meeting with multi-agent deep reinforcement learning: A comprehensive review. *Artificial Intelligence Review, 58*, 165.

[11] Li, H., et al. (2025). Multi-agent reinforcement learning in games: Research and applications. *Biomimetics, 10*, 375.

[12] Lowe, R., et al. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems (NeurIPS)*.

[13] Yu, C., et al. (2022). The surprising effectiveness of PPO in cooperative multi-agent games. *arXiv Preprint*, arXiv:2103.01955.

[14] Rashid, T., et al. (2020). Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research, 21*, 1–51.

[15] Sun, P., et al. (2020). TLeague: A framework for competitive self-play based distributed multi-agent reinforcement learning. *arXiv Preprint*, arXiv:2011.12895.

[16] Berner, C., et al. (2019). Dota 2 with large scale deep reinforcement learning. *arXiv Preprint*, arXiv:1912.06680.

[17] Jaderberg, M., et al. (2019). Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science, 364*(6443), 859–865.

[18] McAleer, S., et al. (2021). Anytime PSRO for two-player zero-sum games. *Advances in Neural Information Processing Systems (NeurIPS)*.

[19] Zhang, P., et al. (2026). A cooperative game-based low-carbon optimal operation strategy for multi-microgrids based on multi-agent deep reinforcement learning. *Energies, 19*, 3683.