Mastering Complex Control in MOBA Games with Deep Reinforcement Learning

Deheng Ye,[1] Zhao Liu,[1] Mingfei Sun,[1*] Bei Shi,[1] Peilin Zhao,[1] Hao Wu,[1*] Hongsheng Yu,[1] Shaojie Yang,[1] Xipeng Wu,[1] Qingwei Guo,[1] Qiaobo Chen,[1] Yinyuting Yin,[1] Hao Zhang,[1] Tengfei Shi,[1] Liang Wang,[1] Qiang Fu,[1] Wei Yang,[1] Lanxiao Huang[2]
[1]Tencent AI Lab, Shenzhen, China
[2]Tencent Timi Studio, Chengdu, China
{dericye, ricardoliu, mingfeisun, beishi, masonzhao, alberthwu, yannickyu, shaojieyang, haroldwu, leoqwguo, ciaochen, mailyyin, howezhang, francisshi, enginewang, leonfu, willyang, jackiehuang}@tencent.com

Abstract

We study the reinforcement learning problem of complex action control in the Multi-player Online Battle Arena (MOBA) 1v1 games. This problem involves far more complicated state and action spaces than those of traditional 1v1 games, such as Go and Atari series, which makes it very difficult to search any policies with human-level performance. In this paper, we present a deep reinforcement learning framework to tackle this problem from the perspectives of both system and algorithm. Our system is of low coupling and high scalability, which enables efficient explorations at large scale. Our algorithm includes several novel strategies, including control dependency decoupling, action mask, target attention, and dual-clip PPO, with which our proposed actor-critic network can be effectively trained in our system. Tested on the MOBA game *Honor of Kings*, the trained AI agents can defeat top professional human players in full 1v1 games.

Introduction

Deep reinforcement learning (DRL) has been widely used for building agents to learn complex control in competitive environments. In the competitive setting, a considerable amount of existing DRL research adopt two-agent games as the testbed, i.e., one agent versus another (1v1). Among them, Atari series and board games have been widely studied. For example, a human-level agent for playing Atari games is trained with deep Q-networks (Mnih et al. 2015). The incorporation of supervised learning and self-play into the training brings the agent to the level of beating human professionals in the game of Go (Silver et al. 2016). And recently, a more general DRL method is further applied to the Chess and Shogi 1v1 games (Silver et al. 2017).

In this paper, we move on to one type of 1v1 games that has the next level of complexity, i.e., the MOBA 1v1 games. As we know, RTS games are considered as a grand challenge for AI research (Vinyals et al. 2019; Silva and Chaimowicz 2017). MOBA 1v1 is a real-time strategy (RTS) game that requires highly complex action control. Compared with traditional 1v1 games, e.g., board and Atari, MOBA 1v1 games have far more complicated environments and controls. Take the MOBA 1v1 games in *Honor of Kings* as an example, the magnitude of states and actions involved can reach to 10^{600} and 10^{18000}, while these in Go are 10^{170} and 10^{360} (Silver et al. 2016), illustrated in Table 1.

Table 1: Comparing Go and MOBA 1v1

Game	**Go 1v1**	**MOBA 1v1**
Action space	$250^{150} \approx 10^{360}$ (250 pos available, 150 decisions per game on average)	10^{18000} (100+ discretized actions, 9,000 frames per game)
State space	$3^{361} \approx 10^{170}$ (361 pos, 3 states each)	$2^{2000} \approx 10^{600}$ (2 heroes, (1000+ pos)*(2+ states))
Human player data	rich, high-quality	little
Peculiarity	long-term tactics	real-time, complex control

Besides, the complexity of MOBA 1v1 also comes from the playing mechanism. To win a game, in the partially observable environment, agents must learn to plan, attack, defend, control skill combos, induce, and deceive the opponents. Apart from the player's and the opponent's agent, there exists many more game units, e.g., creeps and turrets. This creates challenge to the target selection which requires delicate sequences of decision making and corresponding action controls. Furthermore, different heroes in a MOBA game have very different playing methods. The action control can completely change from hero to hero, which calls for robust and unified modeling. Last but not least, there lacks high-quality human game data for MOBA 1v1 which makes supervised learning unfeasible, because players generally use the 1v1 mode to practice heroes, while the MOBA 5v5 mode is used for formal matches in mainstream MOBA games, like *Dota* and *Honor of Kings*. Note that in this paper we focus on MOBA 1v1 games rather than MOBA 5v5 as the latter emphasizes more on the team collaborative strategy of all agents than the action control of any single agent. In this regard, the MOBA 1v1 setting is more appropriate to study the problem of complex control in games.

To handle these challenges, we design a deep reinforcement learning framework, together with a set of algorithm-level innovations, to enable efficient explorations at massive scale for multi-agent competitive environments like

[*]Work done as research interns at Tencent.

MOBA 1v1 games. We design a neural network architecture including the encoding of multi-modal inputs, the decoupling of inter-correlations in controls, exploration pruning mechanism, and attack attention, to consider the ever-changing game situations in MOBA 1v1 games. To evaluate the upper limit and the robustness of the trained AI agents thoroughly, we invite professional players and a variety of top-amateur human players to compete with our AI agents. We also compare our method with existing state-of-the-art works on building MOBA 1v1 agents (Jiang, Ekwedike, and Liu 2018).

Our contributions are as follows:

- We present a systematic and thorough study on building AI for playing MOBA 1v1 games, which require highly complex action control of agents. On system aspect, we develop a deep reinforcement learning framework which provides scalable and off-policy training. On algorithm aspect, we develop an actor-critic neural network for modeling MOBA action controls. Our network optimizes with a multi-label proximal policy algorithm (PPO) objective, and is featured with the decoupling of control dependency, an attention mechanism for target selection, action mask for efficient exploration, LSTM for learning skill combos, and an improved version of PPO, called dual-clip PPO, for ensured training convergence.
- Extensive experiments show that the trained AI agent can defeat top professional human players on different hero types, tested on the 1v1 mode of *Honor of Kings*, a popular MOBA game.

Preliminaries

Notation

We focus on the two-agent world for multi-agent Markov games (Bansal et al. 2017), which can be extended to multiple agents. We use the tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, P, r, \rho_0, \gamma)$ to denote an infinite-horizon, discounted Markov Decision Process, where $\mathcal{S}$ is the state space, $\mathcal{O}$ is the observable state space of each agent, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denotes the state transition probability, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ represents the reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}$ is the distribution of the initial state s_0, and $\gamma \in (0, 1]$ is the discount factor. A stochastic policy π is a mapping $\mathcal{O} \times \mathcal{A} \to [0, 1]$. In our complex control problem, each agent aims to maximize the cumulative reward returns, i.e., the objective $\mathbb{E}\big[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\big]$, where T is the time horizon.

Related Work

Multi-agent control with reinforcement learning has two settings: cooperative setting and competitive setting. Our work belongs to the latter. A large proportion of existing works use games as the testbed for RL advances.

For the cooperative setting, a survey is done by (Panait and Luke 2005). Recently, Foerster et al. (2016) study multi-agent cooperation to solve riddles with recurrent Q-networks. Collaborative agents for playing 3D FPS games have also been explored (Jaderberg et al. 2019; Lample and Chaplot 2017). Another recent work builds a macro-strategy model for guiding multi-agents in MOBA 5v5 games using supervised learning (Wu et al. 2018).

For the competitive setting, 1v1 games are heavily studied. A typical work is AlphaGo (Silver et al. 2016), which combines supervised learning and RL. RL has also been successfully applied to Atari games (Mnih et al. 2015), which contain both single-agent games, and multi-agent games like the 1v1 Pong game. Further, He et al. (2016) focus on opponent modeling in competitive setting via deep Q-learning, using a simulated soccer game and a trivia game as testbed. Tampuu et al. (2017) use deep Q-learning to train the 1v1 Pong game agents. Bansal et al. (2017) construct four 1v1 games in MuJoCo environment to analyze the emergent complexity in multi-agent competition.

Comparing to these, we study RL systems with a more complex competitive setting, i.e., the MOBA 1v1 games. One published state-of-the-art work on this line proposes a monte-carlo tree search (MCTS) based RL method for playing MOBA 1v1, which uses the game *Honor of Kings* as testbed (Jiang, Ekwedike, and Liu 2018). Recently, OpenAI announced an AI for playing DOTA2 (DOTA2 is a popular MOBA game) that can defeat professionals. The technical details are not open yet, an overview is posted (OpenAI 2018). Apart from this, our work presents the first systematic investigation focusing on the action controls of agents in complex games (micro-management in esports). We play 1v1 full game in *Honor of Kings*, i.e., the game ends until the home base destroyed; And further, the robustness and scalability of our method have been thoroughly tested, as our method has been applied to train various hero types, including Mage, Marksman, Warrior, etc. As mentioned, playing a different MOBA hero is like playing a different game.

This work is also related to research on building AI agents for playing StarCraft 1v1 games which have been significantly explored (Ontanón et al. 2013; Robertson and Watson 2014). By comparison, StarCraft 1v1 games are of a different kind of complexity, i.e., MOBA 1v1 is known for the complex action control of heroes which is in the scope of this paper, while StarCraft 1v1 measures more on the strategy to control many game units simultaneously.

System Design

In this section, we first present an overview of our proposed framework for MOBA 1v1, and then describe each module. To make our presentation easier, we first introduce our system design in this section, and leave the algorithm design for the next section.

Considering the fact that complex agent control problems can introduce high variance of stochastic gradients, e.g. the MOBA 1v1 games, large batch size is necessary to speed up the training (McCandlish et al. 2018). Thus, we design a scalable and loosely-coupled system architecture to construct the utility of data parallelism. Specifically, our architecture consists of four modules, i.e., **Reinforcement Learning (RL) Learner**, **Artificial Intelligence (AI) Server**, **Dispatch Module** and **Memory Pool**, as shown in Fig. 1. AI Server implements how the AI model interacts with the environment. The Dispatch Module is a station for sample collection, compression and transmission. The

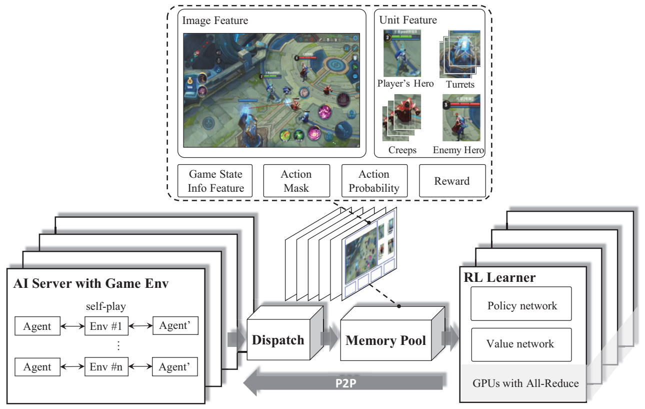


Figure 1: Overview of our System Design

Memory Pool is the data storage module, which provides training instances for the RL Learner. Note that these modules are decoupled and can be flexibly configured, so that our researchers can focus on the algorithm design and the logic of the environment. Such a system design is also applicable to other multi-agent competitive problems. The details of these modules are provided as follows:

AI Server covers the interaction logic between game environment and the AI model. AI server generates episodes via self-play with mirrored policies (Silver et al. 2017). The opponent policy sampling is similar to (Bansal et al. 2017). Based on the features extracted from game state, hero action is predicted using Boltzman exploration (Cesa-Bianchi et al. 2017), i.e., sampling based on softmax distribution. The sampled action is then forwarded to the game core for execution. After execution, the game core returns the corresponding reward value and the next state continuously. In use, one AI Server will bind one CPU core. Because the game logic deduction runs on CPUs, we also run the model inference on CPUs to save the IO cost. In order to generate episodes efficiently, we build a CPU version of the fast inference library FeatherCNN [1]. FeatherCNN can automatically convert AI models trained from mainstream tools like Tensorflow and Caffe, to a customized format for inference.

Each **Dispatch Module** is bounded with several AI Servers on the same machine. It is a server that collects data samples from AI Servers, consisting of reward, feature, action probabilities, etc. These samples are firstly compressed and packed, and then send to Memory Pools. The **Memory Pool** is also a server. Its internals are implemented as a memory efficient circular queue for data storage. It supports samples of varied lengths, and data sampling based on the generated time.

The **RL Learner** is a distributed training environment. To accelerate policy update using large batch sizes, multiple RL Learners are integrated to parallelly fetch data from the same number of Memory Pools. The gradients in the RL learners are averaged through the ring allreduce algorithm (Sergeev and Balso 2018). To reduce IO cost, RL Learners communicate with Memory Pools using shared memory instead of socket, which can deliver 2-3 times of speed boosting. The trained models from the RL Learners are rapidly synchronized to AI Servers in a peer-to-peer manner.

[1]FeatherCNN is a state-of-the-art inference engine for mobile devices: https://github.com/Tencent/FeatherCNN

In our system, the experiences generation is decoupled from the parameters learning. This flexible mechanism makes AI Servers and RL learners scalable with high throughput. To avoid the bottleneck of communication cost between learners and actors, our trained models are synchronized to AI Servers via peer-to-peer from our master RL Learner. To smooth data storage and transmission, we design two mediators, i.e., the Dispatch Server and the Memory Pool Server. In practice, we can scale to millions of CPU cores and thousands of GPUs effortlessly. Note that such design differs from existing system designs like IMPALA (Espeholt et al. 2018). In IMPALA, parameters are distributed across the learners, and actors retrieve the parameters from all the learners in parallel.

Algorithm Design

In the RL Learner, an actor-critic network is implemented to model the action control dependencies in MOBA 1v1 games. Figure 2 illustrates this network, the state and actions. To train this network efficiently and effectively, several novel strategies are proposed. First, the *target attention mechanism* is designed in this network to help with the target selection in MOBA combats. Second, LSTMs are leveraged for the hero to learn the skill combos which are critical to create severe and instant damage. Third, the decoupling of control dependencies is conducted to form a multi-label proximal policy optimization (PPO) objective. Forth, a game-knowledge-based pruning method, called *action mask*, is developed to guide explorations during the reinforcement process. Finally, a dual-clipped version of the PPO algorithm is proposed to guarantee convergence with large and deviated batches (Schulman et al. 2017). The details of our network are provided in the remaining paragraphs.

First, the network encodes image features f_i, vector features f_u, and game state information f_g (the observable game states) as encodings h_i, h_u, and h_g using convolutions, fully-connections, and fully-connections (FC), respectively. Specifically, after a few layers of FC/ReLu, the encoding of f_u is splitted into two parts: the representation of the unit and the attention keys of our target. To handle the varied number of units, the same type units are mapped to a feature vector of fixed length by max-pooling. Then the concatenation of h_i, h_u for all types, and h_g is represented as the encoding vector of an observable game state. The state encoding is then mapped to the final representation h_{LSTM} by a LSTM cell, which further takes the temporal information into consideration. h_{LSTM} is sent to a FC layer to predict the action a. The target unit t of action a is predicted by a **target attention** mechanism over every unit. This mechanism treats a FC output of h_{LSTM} as the query, the stack of all unit encodings as the keys h_{keys}, and calculate the target attention as:

$$p(t|a) = Softmax(FC(h_{LSTM}) \cdot h_{keys}^T) \qquad (1)$$

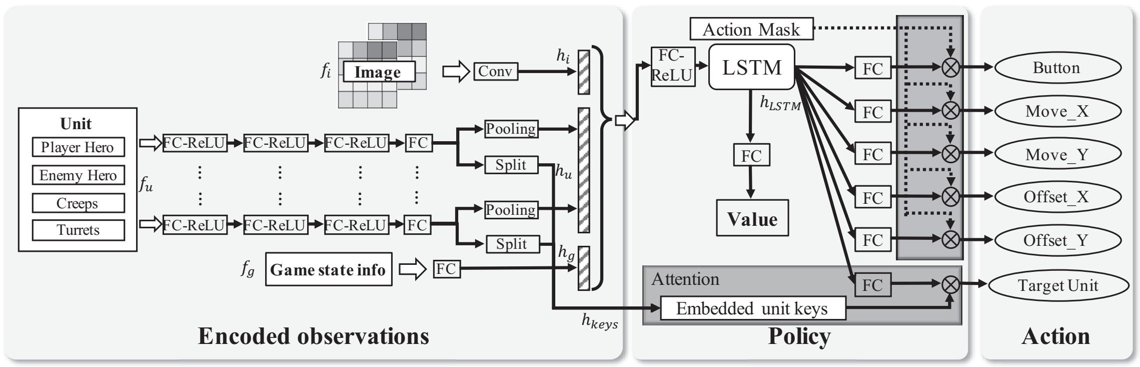


Figure 2: Illustrations of state, action, policy and value. A state $s \in \mathcal{S}$ covers three types of information: local image info (e.g., obstacles in 2D), observable unit attributes (e.g., hero type, health point), and observable game state info (e.g., game time, turrets destroyed, etc.), i.e., $s = [f_i, f_u, f_g]$. An action $a \in \mathcal{A}$ in a MOBA 1v1 game specifies two items: the content (i.e., the action button to press, the Move_X, the Move_Y, Offset_X, and Offset_Y) and the target game unit. The action buttons include move, attack, skill releasing, etc. The policy π_θ is modeled by FCs and an LSTM, which also predicts the values.

where $p(t|a)$ is the attention distribution over units and the dimension of $p(t|a)$ is the number of units in the state.

Second, it is very hard to explicitly model the inter-correlations among different labels in one action of MOBA games in the multi-label policy network, e.g. the correlation between the direction of a skill (Offset_X and Offset_Y), and the skill type (Button). To solve this issue, we treat each label in an action independently to decouple their inter-correlations, i.e., **the decoupling of control dependencies**. Before decoupling the inter-correlations, the PPO objective without clipping is:

$$\max_{\theta} \hat{\mathbb{E}}_{s,a\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right], \tag{2}$$

where the expectation $\hat{\mathbb{E}}_t[...]$ indicates an empirical average over a finite batch of samples, stochastic policy π_θ predicts the probability of taking action a_t at state s_t, and $\hat{A}_t$ is an estimator of the advantage function at timestep t. Suppose each action $a = (a^0, \ldots, a^{N_a-1})$, then the objective after action decoupling becomes:

$$\max_{\theta} \sum_{i=0}^{N_a-1} \hat{\mathbb{E}}_{s,a\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a_t^{(i)}|s_t)}{\pi_{\theta_{\text{old}}}(a_t^{(i)}|s_t)}\hat{A}_t\right]. \tag{3}$$

This decoupled objective brings two advantages. First, it simplifies the policy structure. Specifically, the policy network can be defined without considering the inter-correlations as this dependency can be post-processed. Second, it increases the diversity of actions. As each component has its independent own channel of value output, the actions can be significantly diversified, thus inducing more explorations during training. Furthermore, to force diversity of exploration, we randomize the positions of both agents during training at the beginning of the game.

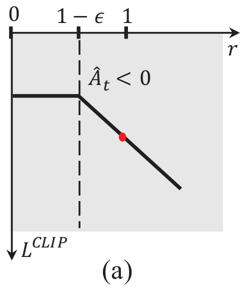
However, the action decoupling further increases the complexity of policy training, while it is originally very high due to the vast action and state spaces in MOBA 1v1 games. To improve the training efficiency, an **action mask** is proposed to incorporate the correlations between action elements at the final output layers of the policy based on prior knowledge of experienced human player, which helps prune the exploration of RL. Specifically, our action mask helps eliminate several unreasonable aspects: 1) physically forbidden areas on map, e.g., suppose the predicted action is to move towards a direction, which cannot be performed as that direction is occupied by obstacles in the map; 2) skill or attack availability, e.g., the predicted action to release a skill within Cool Down time shall be eliminated; 3) being controlled by enemy hero skill or equipment effects; 4) hero-/item-specific restrictions.

Dual-clip PPO Let $r_t(\theta)$ denote the probability ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. Because the ratio $r_t(\theta)$ can be extremely large, maximization of the RL objective may lead to an excessively large policy deviation. To alleviate this issue, the standard PPO algorithm (Schulman et al. 2017) involves a ratio clip as follows:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\Big[\min\Big(r_t(\theta)\hat{A}_t, \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t\Big)\Big], \tag{4}$$

to penalize extreme changes to the policy.

However, in large-scale off-policy training environments like our framework, the trajectories are sampled from various sources of policies, which may differ considerably from the current policy π_θ. In such situations, the standard PPO will fail to work with such deviations since it was originally proposed for on-policy (Schulman et al. 2017). For example, when $\pi_\theta(a_t^{(i)}|s_t) \gg \pi_{\theta_{\text{old}}}(a_t^{(i)}|s_t)$, the ratio $r_t(\theta)$ is a huge number. When $\hat{A}_t < 0$, such a large ratio $r_t(\theta)$ will introduce a big and unbounded variance since $r_t(\theta)\hat{A}_t \ll 0$. As a result, even using the objective of PPO, the new policy deviates significantly from the old policy, which makes it very difficult to insure the policy convergence. We thus propose a dual-clipped PPO algorithm to support large-scale distributed training, which further clip the ratio $r_t(\theta)$ with a



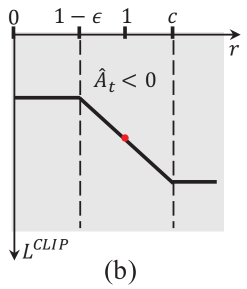


Figure 3: (a) Standard PPO (clip with ϵ); (b) Our Proposed Dual-clip PPO (clip with ϵ and c when $\hat{A}_t < 0$)

lower bound of the value $r_t(\theta)\hat{A}_t$, illustrated in Fig. 3. When $\hat{A}_t < 0$, the new objective of our dual-clipped PPO is:

$$\hat{\mathbb{E}}_t\Big[\max\Big(\min\Big(r_t(\theta)\hat{A}_t, \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t\Big), c\hat{A}_t\Big)\Big] \tag{5}$$

where $c > 1$ is a constant indicating the lower bound.

Experiments

System Setup

We test our method by using the 1v1 mode in *Honor of Kings*, which is the most popular MOBA game nowadays, and has been actively used as the testbed for recent RL advances (Eisenach et al. 2019; Wang et al. 2018; Jiang, Ekwedike, and Liu 2018).

Our framework runs over a total number of 600,000 CPU cores encapsulated in Dockers and 1,064 Nvidia GPUs (a mixture of Tesla P40 and V100). The volume of our framework allows parallel experiments. We have 1600 vector features containing observable unit attributions and game information, and 2 channels of image features read from game core (the obstacle channel and the hero position channel). We mainly use vectors to represent observations as they they are lightweight. We adopt FP16 for data transmission to save bandwidth, and revert to FP32 for training. To train one hero, we use 48 P40 GPU cards and 18,000 CPU cores. The mini-batch size per GPU card is 4096. The time step and unit size of the LSTM are 16 and 1024, respectively. We train using full rollouts, i.e., one episode ends until the termination of the game, and we use zero-start, i.e., the agent starts the game from Frame 0. The training speed is about 80000 samples per second per GPU card. We use the game core of *Honor of Kings* directly to execute the game. With the high throughput of our framework, we have experiences collected per day per hero is about 500 years human data in the 1v1 mode of *Honor of Kings*.

We use Adam optimizer with initial learning rate 0.0001. In the dual-clipped PPO, the two clipping hyperparameters ϵ and c are set as 0.2 and 3, respectively. The discount factor is set as 0.997. For the case of *Honor of Kings*, this discount is valuing future rewards with a half-life of about 46 seconds. We use generalized advantage estimation (GAE) (Schulman et al. 2015) for reward calculation, and we set $\lambda = 0.95$ in GAE to reduce the variance caused by delayed effects.

To evaluate the trained AI's ability in real world, we deploy the AI model into *Honor of Kings* to play against professional and top-amateur human players. We predict actions via the AI model every 133 ms which is about the response time of top-amateur players. We also compare our AI with baseline methods used in existing works (Jiang, Ekwedike, and Liu 2018) for playing 1v1 games in *Honor of Kings*, including behavior-tree (implemented as the internal AI in the game), MTCS and its variants. We also use Elo rating (Coulom 2008) for comparing different versions of models, similar to that of AlphaGo (Silver et al. 2016).

Table 2: Information of human professional testers. Here "the league" refers to the *Honor of Kings* Professional League, known as KPL.

Player	Info
eStarPro.Cat	professional, a.k.a, best Mage in the league
QGhappy.Hurt	professional, a.k.a, best Marksman in the league
TS.NuanYang	professional, famous top Assassin player
QGhappy.Fly	professional, a.k.a best Warrior in the league
WE.762	professional, famous top Warrior player

Experimental Results

Exploring the Upper Limit of Control Ability We invite 5 top professional *Honor of Kings* human players to play BO5 (best of five) matches against our AI. They are `QGhappy.Hurt`, `WE.762`, `TS.NuanYang`, `QGhappy.Fly`, `eStarPro.Cat` (name in `CLUB.PLAYER` format). The information of these players are briefed in Table 2.

Table 3 shows the match results. Note that these professional players play the heroes they are specialized in. We see that our AI can defeat professional players on heroes of various types. Take the Mage hero `DiaoChan` for instance, our AI defeats eStarPro.Cat with the score 3:0. `eStarPro.Cat` is the current best Mage player in the professional league, and is particularly good at controlling DiaoChan. `DiaoChan` controlled by AI dominates the game. It achieves 5 kills per game, but gets killed only 1.33 times on average. The game lasts for 6 minutes and 56 seconds on average. AI also prevails significantly in terms of gold and experience gained in game.

Evaluating the Robustness of Control Ability We further evaluate whether the policies learned by our AI could counter to a diversity of top human players. In ChinaJoy 2019, we held large public matches where the public was allowed to face off against our AI, from Aug. 2 to Aug. 5, 2019, held in Shanghai, China [2]. We showcased eight AI heroes of different types in total. Human players who defeat AI in 1v1 games will be rewarded with a 600 USD smart phone. Participants must have demonstrated rankings in *Honor of Kings* as the entry condition (top 1% minimum). The statistics results of our public experiment are provided

[2]This event was held jointly by Vivo, Qualcomm and Tencent.

Table 3: Match Statistics of our AI vs. Professional Players on Different Types of Heroes

Hero	DiaoChan	DiRenjie	LuNa	HanXin	HuaMulan
Hero Type	Mage	Marksman	Warrior+Mage	Assassin	Warrior
Score	3:0 (AI:eStarPro.Cat)	3:0 (AI:QGhappy.Hurt)	3:0 (AI:QGhappy.Fly)	3:1 (AI:TS.NuanYang)	3:0 (AI:WE.762)
Kill	5.0:1.3	2.3:0.7	2.7:1.0	2.5:1.5	4.0:1.3
Game Length	6'56"	6'23"	7'53"	6'41"	6'48"
Gold/min	852.7:430.6	869.3:606.6	969.7:724.0	954.1:754.2	945.2:654.2
Exp/min	900.0:573.0	895.3:661.7	979.0:817.2	965.4:802.5	921.4:723.1

Table 4: Results of AI vs. Various Top Human Players

Hero Name	Hero Type	#Matches	#Win	Rate
DiaoChan	Mage	445	445	100%
DiRenJie	Marksman	264	264	100%
HuaMuLan	Warrior	256	256	100%
HanXin	Assassin	221	220	99.55%
LuNa	Warrior+Mage	260	260	100%
HouYi	Marksman	79	78	98.70%
LuBan	Marksman	354	354	100%
SunWukong	Assassin	221	219	99.09%
		2100	2096	99.81%

in Table 4. Our AI achieves a 99.81% win rate among 2100 matches, with only 4 games lost. Five of the eight heroes achieve a 100% win rate through hundreds of matches.

Comparison with Baselines We also compare our method with existing methods. As mentioned, a recent work applies monte-carlo tree search (MCTS) based RL to build AI for playing 1v1 games in *Honor of Kings* (Jiang, Ekwedike, and Liu 2018). Particularly, they built four AI agents with MCTS and its variants, which are named as "FBTS", "NR", "DPI", and "AVI", respectively. Their evaluation method is to compare which of these agents can defeat the same baseline opponent faster, while the detailed match statistics have not been reported. Specifically, their baseline opponent consists of six Marksman heroes that can be defeated by the internal "DiRenJie" AI, implemented using behavior-trees. Following the same experiment settings, we compare the averaged length of time for our AI to defeat the same baseline opponent. The result is shown in Fig. 4. We see that the AI trained from our method significantly outperforms the FBTS, NR, DPI, AVI and the internal AI.

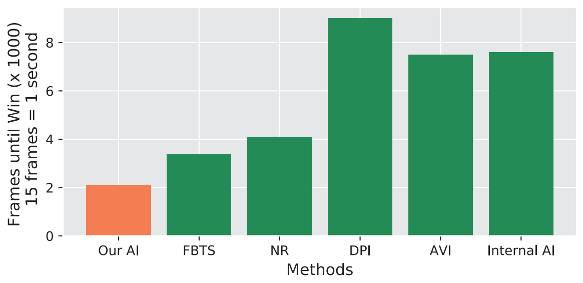


Figure 4: Comparing Averaged Time Length to Defeat the Same Baseline Opponent

Table 5: Results of Ablation Experiments

Item	Win rate vs Base	Time to converge
Base	-	80 h
Base + AM	50.5%	**65 h**
Base + TA	**75%**	90 h
Base + LSTM	73%	100 h
Full version	90%	80 h

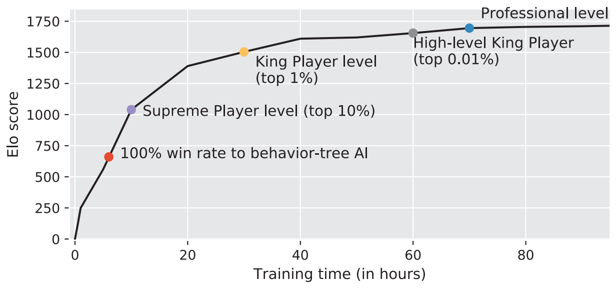


Figure 5: Elo Change during Training

Progression During Training In Fig. 5, we show the change of Elo rating during training, using the Marksman hero "DiRenJie" as a case. We observe that the Elo score grows with the training length and becomes relatively steady after about 80 hours. Further, the growth rate of Elo is inversely proportional to training time. With about 6 hours training, the AI begins to defeat the internal behavior-tree based AI with 100% win rate. The AI ability rapidly increases to King Player's level (top 1% human player for *Honor of Kings*) after about 30 hours, and becomes comparable to professionals after 70 hours.

Ablation Study We conduct ablation experiments to understand the effect of different components and settings in our method.

We first analyze three components from the model network, including the *action mask* (AM), *target attention* (TA) and *LSTM* we developed. In Table 5, we show the results of a tournament between different "DiRenJie" AI versions using the same amount of training resources. The Full version refers to the strongest AI we trained, while the Base version means the Full version without the aforementioned three components. We see that using *action mask* can largely reduce the training time, while achieving the same AI ability as the Base (win rate 50.5%). We see that the TA and the

LSTM are both useful to improve the AI ability.

We also analyze hyperparameter settings. Particularly, we compare *full rollouts* (FR) with *partial rollouts* (PR) with fixed N frames, and we compare *random initial frame* (RIF) with zero-start (ZS), i.e., selecting random frame of the whole game as the start of the Markov decision process (RIF) versus starting from the beginning of the game (ZS). We find that: 1) FR improves the AI's ability to a large margin, with which the win rate increases to 70%∼80% when compared to PR with 1000, 2000 and 3000 frames; 2) RIF can speed up the convergence by 15% but at a cost of slightly lower AI ability (win rate 40% when compared to ZS).

Conclusion and Future Work

In this paper, we present a deep reinforcement learning (DRL) approach to handle the complex action control of agents in MOBA 1v1 games, from the perspectives of both system and algorithm. We propose a scalable and off-policy DRL system architecture for massive episode exploration. We propose an actor-critic multi-label neural network, containing several strategies for modeling MOBA combats, and a dual-clipped version of the PPO algorithm for ensuring convergence. The resulting AI from our framework can defeat top professional esports players in MOBA 1v1 games, tested on the popular MOBA game *Honor of Kings*.

As a next step, we will make our framework and algorithm open source, and the game core of *Honor of Kings* accessible to the community to facilitate further research on complex games; and we will also provide part of our computing resources via virtual cloud for public use [3].

Appendix

MOBA 1v1 Games

Real-time Strategy (RTS) games are considered as a grand challenge for AI research (Vinyals et al. 2019). MOBA is one type of RTS games, and is the most played game type (Mora-Cantallops and Sicilia 2018). Popular MOBA games include *Dota*, *Honor of Kings*, *League of Legends*, etc.

MOBA 1v1 mode is a pure arena for competing one's level of action control. The formal 1v1 matches, for fairness, are mirror games, i.e., two players pick the same hero and control their own hero individually to fight against each other. When the game begins, each player sets out from the base, gains gold and experience by killing or destroying other game units (e.g., enemy heroes, creeps, turrets). The goal is to destroy the opponent's turrets and base while protecting own turrets and base. Briefly, creeps are a small group of computer-controlled creatures periodically travel along the predefined lane to attack the opponent. Turrets, i.e., defensive buildings, are designed to attack any creeps or heroes moving into their sight area. Heroes are player-controlled units which can move around and have abilities to release various attacks and healing skills.

Each MOBA hero requires very complicated control mechanism, known as micro-management in esports. All

[3]By Nov. 21, 2019, the Beta version is open to 4 universities in China for user feedback.

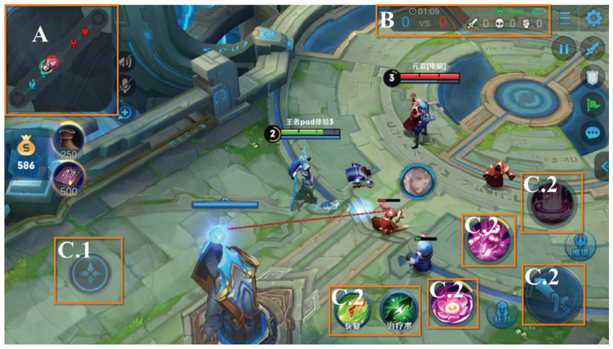


Figure 6: Game UI of *Honor of Kings* 1v1. In the main screen, there are four sub-parts: mini-map (A) on the top-left, dashboard (B) on the top-right, movement controller (C.1) on the bottom-left, and ability controller (C.2) on the bottom-right, as highlighted in each box.

Table 6: Reward Design

Reward	Weight	Type	Description
hp_point	2.0	dense	the health point of hero
tower_hp_point	10.0	sparse	the health point of turrets and base
money (gold)	0.008	dense	the gold gained
ep_rate	0.8	dense	the rate of mana
death	-1.0	sparse	being killed
kill	-0.5	sparse	kill an enemy hero
exp	0.008	dense	the experience gained
last_hit	0.5	sparse	last hitting to enemy units

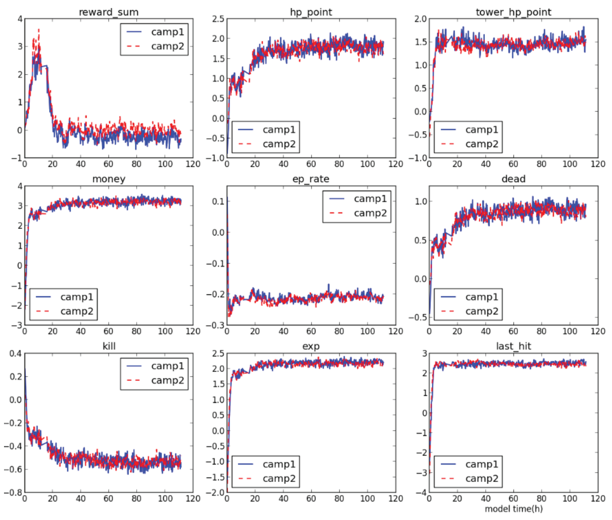


Figure 7: A case of reward change during training; the x-axis is the training time in hours, the y-axis is reward; camp1 and camp2 refer to the two camps (teams) in MOBA games.

MOBA games have various types of heroes. For example, in *Honor of Kings*, there are six types of heroes, including Mage, Assassin, Warrior, Marksman, Tank, Support. Different hero types have different playing method. Generally, Tank and Support are durable heroes and are mainly for defense, thus not suitable for MOBA 1v1 solo games. The rest four types are generally selected for MOBA 1v1 esports. In Fig. 6, we show the UI of the MOBA game *Honor of Kings* as an example.

Reward

All the trained 1v1 heroes use the same reward, shown in Table 6. The reward design is inspired by OpenAI Five's Dota reward (OpenAI 2018). The reward is zero-sum, i.e., one's mean reward is subtracted from that of the opponent. Our framework allows on-the-fly reward analysis during training. A case of the `DiaoChan` hero is shown in Fig. 7.

References

Bansal, T.; Pachocki, J.; Sidor, S.; Sutskever, I.; and Mordatch, I. 2017. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*.

Cesa-Bianchi, N.; Gentile, C.; Lugosi, G.; and Neu, G. 2017. Boltzmann exploration done right. In *Advances in Neural Information Processing Systems*, 6284–6293.

Coulom, R. 2008. Whole-history rating: A bayesian rating system for players of time-varying strength. In *International Conference on Computers and Games*, 113–124. Springer.

Eisenach, C.; Yang, H.; Liu, J.; and Liu, H. 2019. Marginal policy gradients: A unified family of estimators for bounded action spaces with applications. In *The Seventh International Conference on Learning Representations (ICLR 2019)*.

Espeholt, L.; Soyer, H.; Munos, R.; Simonyan, K.; Mnih, V.; Ward, T.; Doron, Y.; Firoiu, V.; Harley, T.; Dunning, I.; et al. 2018. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*.

Foerster, J. N.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016. Learning to communicate to solve riddles with deep distributed recurrent q-networks. *arXiv preprint arXiv:1602.02672*.

He, H.; Boyd-Graber, J.; Kwok, K.; and Daumé III, H. 2016. Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning*, 1804–1813.

Jaderberg, M.; Czarnecki, W. M.; Dunning, I.; Marris, L.; Lever, G.; Castaneda, A. G.; Beattie, C.; Rabinowitz, N. C.; Morcos, A. S.; Ruderman, A.; et al. 2019. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science* 364(6443):859–865.

Jiang, D.; Ekwedike, E.; and Liu, H. 2018. Feedback-based tree search for reinforcement learning. In *International Conference on Machine Learning*, 2289–2298.

Lample, G., and Chaplot, D. S. 2017. Playing fps games with deep reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

McCandlish, S.; Kaplan, J.; Amodei, D.; and Team, O. D. 2018. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Mora-Cantallops, M., and Sicilia, M.-A. 2018. Moba games: A literature review. *Entertainment Computing* 26:128–138.

Ontanón, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; and Preuss, M. 2013. A survey of real-time strategy game ai research and competition in starcraft. *IEEE Transactions on Computational Intelligence and AI in games* 5(4):293–311.

OpenAI. 2018. Openai five. https://blog.openai.com/openai-five/. [Online; accessed on 05-Sept-2019].

Panait, L., and Luke, S. 2005. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems* 11(3):387–434.

Robertson, G., and Watson, I. 2014. A review of real-time strategy game ai. *AI Magazine* 35(4):75–104.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sergeev, A., and Balso, M. D. 2018. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*.

Silva, V. d. N., and Chaimowicz, L. 2017. Moba: a new arena for game ai. *arXiv preprint arXiv:1705.10443*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature* 529(7587):484–489.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; and Vicente, R. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12(4):e0172395.

Vinyals, O.; Babuschkin, I.; Chung, J.; Mathieu, M.; Jaderberg, M.; Czarnecki, W.; Dudzik, A.; Huang, A.; Georgiev, P.; Powell, R.; Ewalds, T.; Horgan, D.; Kroiss, M.; Danihelka, I.; Agapiou, J.; Oh, J.; Dalibard, V.; Choi, D.; Sifre, L.; Sulsky, Y.; Vezhnevets, S.; Molloy, J.; Cai, T.; Budden, D.; Paine, T.; Gulcehre, C.; Wang, Z.; Pfaff, T.; Pohlen, T.; Yogatama, D.; Cohen, J.; McKinney, K.; Smith, O.; Schaul, T.; Lillicrap, T.; Apps, C.; Kavukcuoglu, K.; Hassabis, D.; and Silver, D. 2019. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.

Wang, Q.; Xiong, J.; Han, L.; Liu, H.; and Zhang, T. 2018. Exponentially weighted imitation learning for batched historical data. In *Advances in Neural Information Processing Systems (NIPS 2018)*, 6288–6297.

Wu, B.; Fu, Q.; Liang, J.; Qu, P.; Li, X.; Wang, L.; Liu, W.; Yang, W.; and Liu, Y. 2018. Hierarchical macro strategy model for moba game ai. *arXiv preprint arXiv:1812.07887*.