Advancing DRL Agents in Commercial Fighting Games: Training, Integration, and Agent-Human Alignment

Chen Zhang [1 2]   Qiang He [3]   Yuan Zhou [2]   Elvis S. Liu [2]   Hong Wang [2]   Jian Zhao [1]   Yang Wang [1 4 5]

Abstract

Deep Reinforcement Learning (DRL) agents have demonstrated impressive success in a wide range of game genres. However, previous research primarily focuses on optimizing DRL competence rather than addressing the challenge of prolonged player interaction. In this paper, we propose a practical DRL agent system for fighting games, named *Shūkai*, which has been successfully deployed to Naruto Mobile, a popular fighting game with over 100 million registered users. *Shūkai* quantifies the state to enhance generalizability, introducing Heterogeneous League Training (HELT) to achieve balanced competence, generalizability, and training efficiency. Furthermore, *Shūkai* implements specific rewards to align the agent's behavior with human expectations. *Shūkai*'s ability to generalize is demonstrated by its consistent competence across all characters, even though it was trained on only 13% of them. Additionally, HELT exhibits a remarkable 22% improvement in sample efficiency. *Shūkai* serves as a valuable training partner for players in Naruto Mobile, enabling them to enhance their abilities and skills.

1. Introduction

Deep Reinforcement Learning (DRL) has demonstrated its capability in sequential decision-making, ranging from robotics (Schulman et al., 2015; Haarnoja et al., 2018), dialogue systems (Ouyang et al., 2022; Touvron et al., 2023), and games (Mnih et al., 2015; Silver et al., 2017; Vinyals et al., 2019). Games (Mnih et al., 2015; Moravčík et al., 2017; Silver et al., 2017; Brown & Sandholm, 2018; Vinyals et al., 2019), in particular, provide a cost-effective and natural environment for training DRL agents due to the ease of data generation. However, previous work (Berner et al., 2019; Jaderberg et al., 2019; Ye et al., 2020b) primarily focuses on enhancing agent competence to professional levels and often overlooks the necessity of improving the human player experience in commercial applications (Csikszentmihalyi, 2000; Bakker et al., 2011; Cheng et al., 2016).

In gaming, DRL's potential extends beyond achieving high competence. The integration of DRL agents as adversaries in Player versus Environment (PvE) settings promises a richer gaming experience by offering dynamic and unpredictable interactions (Shafer, 2012). However, this integration is remarkably challenging. The computational demands of training DRL agents in modern games with extensive character pools are formidable. For instance, training scenarios can escalate in complexity with the increase in character matchups, as seen in cases requiring substantial resources (Silver et al., 2017; Vinyals et al., 2019; Ye et al., 2020a). Furthermore, the sole pursuit of surpassing human performance in DRL agents may not necessarily enhance the player experience and restricts the practical deployment of DRL agents in games (Cheng et al., 2016; Bakker et al., 2011). For example, players with lower competitive skill levels may find it frustrating to face overly strong agents. The inability to overcome such opponents can rapidly diminish a player's interest in the game. Therefore, it is imperative to ensure that DRL agents exhibit varying levels of competitiveness, aligning with the diverse skill levels of players and providing an enjoyable gaming experience.

To address these challenges, this paper proposes a DRL agent system called *Shūkai*, which is specifically designed for fighting games. *Shūkai* utilizes a unified DRL model capable of managing a diverse roster of characters, thereby significantly reducing the complexity inherent in large-scale character sets. *Shūkai* significantly reduces the complexity associated with extensive character rosters and keeps a promising generalization ability. To counter potential overfitting due to training on a subset of characters (Bejani & Ghatee, 2021; Cobbe et al., 2019), our approach

[1]University of Science and Technology of China (USTC), Hefei, China. [2]Tencent Games [3]Institute of Automation, Chinese Academy of Sciences, China [4]Key Laboratory of Precision and Intelligent Chemistry, USTC, China [5]Suzhou Institute for Advanced Research, USTC, China. Correspondence to: Yang Wang <angyan@ustc.edu.cn>, Elvis S. Liu <elvissyliu@tencent.com>.

includes measures to enhance the generalizability of the unified model. Notably, *Shūkai* features Heterogeneous League Training (HELT), a novel method that amalgamates agents of diverse structures, broadening the policy space and achieving a balance between competitive performance (competence) and policy generalization. Aligning agent behavior with human expectations is achieved through meticulously designed rewards, promoting a human-like gaming experience. A behavior evaluation system, informed by expert insights, is utilized to identify and align the behavioral nuances of DRL agents with human players.

Shūkai has been extensively evaluated and deployed in Naruto Mobile, a renowned fighting game featuring over 400 characters and attracting more than 100 million registered players. The generalization of *Shūkai* is evidenced by its consistent competence across all characters, despite being trained on only 13% of them. HELT also presents 22% improvement in terms of sample efficiency. Moreover, *Shūkai* serves as a training partner for players in Naruto Mobile, beneficial for players to improve their abilities. The human-agent alignment of *Shūkai* has positively impacted player engagement, as evidenced by significant improvements in player retention rates through the A/B test. Specifically, the game has experienced a significant next-day retention rate of over 50%, a 7-day retention rate of over 20%, and a 30-day retention rate of over 10%. It has also experienced a maximum growth rate of 5% and an average growth rate of 4%. These enhancements in player retention hold significant importance for a commercial game with a user base in the millions. These improvements demonstrate the real-world impact of DRL and give an example of the ground of DRL.

In summary, our contributions are threefold. **i)** We present a practical DRL agent system, *Shūkai*, introducing HELT to improve agents' ability to generalization, and design rewards that align agents with human-player. **ii)** *Shūkai* demonstrates robust generalizability and improved training efficiency in Naruto Mobile. The interaction with human players and the alignment of agent behavior with player expectations are comprehensively validated. **iii)** This work marks a further step in the deployment of DRL systems in a large-scale commercial fighting game. To the best of our knowledge, this is the first example of a DRL system being deployed in a large-scale commercial fighting game.

2. Preliminaries

In this section, we first introduce the commercial fighting game Naruto Mobile and then discuss the problem formulation.

2.1. Naruto Mobile

Naruto Mobile is an online fighting game developed by Tencent Games with over 100 million registered users. Naruto

Mobile has a large-scale character pool with more than 400 characters (ninjas). Each ninja has its special unique characteristics. Figure 1 shows the interface of Naruto Mobile. Each episode of Naruto Mobile consists of two adversarial ninjas. Players of Naruto Mobile can choose a ninja from the character pool and use the ninja to fight against opponents. The winning condition for all players is to defeat their opponents, and the episode terminates when the condition is satisfied or the timeout. For more details of Naruto Mobile, readers can find in Appendix A.

Fighting games such as Naruto Mobile can be considered as a Markov decision process (Sutton & Barto, 2018). At each time step, the agent records various pieces of information in their observation, including basic information, skill information, current skill information (referring to the activated skills), hit information (including the hitbox and hurt box), element information (describing entities with motion logic independent of characters, such as the behavior of skills), and buff information (referring to temporary enhancement effects).
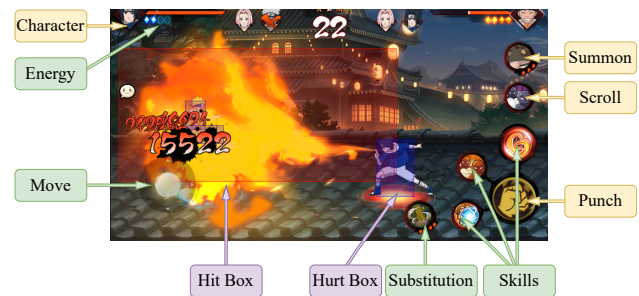

Figure 1. The interface of Naruto Mobile. The player selects a ninja from the character pool to fight against the opponent. The winning condition is to defeat the opponent. Each ninja has three skill buttons and a punch button, controlled by a virtual joystick. The hitbox and hurt box serve as fundamental game mechanics in Naruto Mobile, although they are not visible to the players. The substitution consumes energy and can be used to counter enemy attacks, creating opportunities for counterattacks. Scrolls and summons are additional skills with special effects, such as providing buffs.

In the state design, all this information is categorized into distinguishable categories to effectively model both the self and opponents (Figure 3). This differentiation involves the utilization of ID information, which provides directed references to characters, skills, actions, etc., and attribution information, which represents the numerical values of entities (e.g., health points, cooldown values of skills, remaining time and location of the hurt box, etc.). The action space is categorized into movement, skills, and skill direction (specifically included to cater to skills that require specific

orientations), ensuring the versatility of the model. For more details about the state, please refer to Appendix C.1.

2.2. Problem Formulation

Considering the opponent as a component of the environment, the MDP for Naruto Mobile's training agent is formalized as a six-element tuple $\langle S, A, P, R, T, \gamma \rangle$. Here, S represents the state space and A denotes the action space. R is defined as the reward function, P encapsulates the state transition probabilities, T signifies the time horizon, and γ is the discount factor. At each discrete time step t, the agent observes the current state of the environment, denoted as $s_t \in S$. The agent's policy π, a mapping from states to actions, selects an action a_t based on the current state, formalized as $a_t \sim \pi(\cdot|s_t)$. The state then transitions to the next state $s_{t+1} \sim P(\cdot|s_t, a_t)$, and the agent receives a reward r_{t+1}.

The state value function V and the action function Q are always leveraged to optimize policy. They are defined as:

$$V_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{T} \gamma^k R_{t+k+1}|S_t = s],$$

$$Q_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{T} \gamma^k R_{t+k+1}|S_t = s, A_t = a].$$

The advantage function $A(s, a) = Q(s, a) - V(s)$ is used to measure the quality of action a compared to the average quality.

3. Method

The application of the DRL agent system to Naruto Mobile presents several key challenges. Naruto Mobile features a large-scale character pool with over 400 characters, which is uncommon in other fighting games. While it may seem natural to assign a separate model to each character, deploying over 400 models in the game is unfeasible.

An intuitive approach is to use a unified model to control multiple characters, but this introduces another challenge. As the number of characters increases, the number of potential matchups between different characters also grows. In the case of 400 characters, the possible matchups reach up to 16×10^4, which is not favorable to training efficiency. To address this issue, one possible solution is to train on a subset of characters. However, this approach may lead to overfitting of the model to the subset (Cobbe et al., 2019), making the generalizability of the model decrease, resulting in the failure to adapt to unfamiliar characters who are out of the training subset.

Therefore, the tricky challenge in designing the DRL agent system for Naruto Mobile lies in striking a balance between

training efficiency, model competence, and generalization ability.

To address these challenges, *Shūkai* applies different learning techniques such as heterogeneous neural network architecture, a unified model for controlling multiple characters, quantifying the network input to enhance the generalizability, and Heterogeneous League Training (HELT) to improve the training efficiency and competence.

3.1. Heterogeneous Agents

To use a unified model to control multiple characters while generalizing to the characters out of the training subset, it is necessary to enable the network to learn more generalized knowledge rather than focusing on finding optimal responses within a subset of characters. A feasible approach is to imperfectize the network's input to enhance the model's generalization ability (Schaul et al., 2015; Dosovitskiy & Koltun, 2016; Finn et al., 2017).

As mentioned in Section 2.2, in Narotu Mobile, all the information from the client is divided into two parts, the ID information directly refers to characters, skills, actions, etc., and the attribution information represents the numerical values of game entities. Based on ID information and attribution information, three heterogeneous networks for agents are employed:

- **Full-ID-State (FIS):** Modeling self and opponent utilizing ID information and attribution information.

- **Quantitative-State (QS):** Modeling self with the ID information and attribution information while modeling the opponent only with the attribution information.

- **Full-Quantitative-state (FQS):** Modeling self and opponent utilizing only attribution information, without any ID information.

Quantitative states essentially refer to the utilization of incomplete information to construct representation. By employing quantitative states, the network shifts its focus from finding optimal decisions for specific actions of a designated character to learning knowledge about the game itself, thus enhancing its generalization ability. These agents with different structures will be applied in Heterogeneous League Training.

3.2. Heterogeneous League Training

To balance the competence and generalization ability of the DRL model, while considering training efficiency, *Shūkai* introduce Heterogeneous League Training (HELT). HELT adopts a similar organizational structure to AlphaStar (Vinyals et al., 2019), utilizing the main agent, main
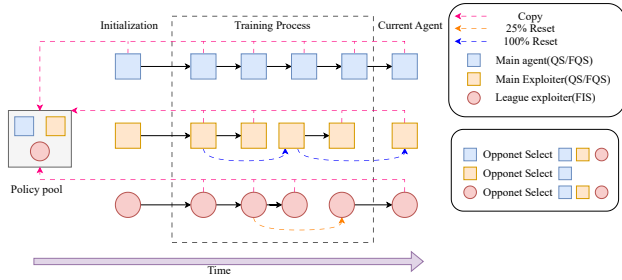
Figure 2. Illustration of HELT. As time progresses, the main agent undergoes continuous training. Once it meets the win rate condition or reaches the timeout, a copy of the main agent is added to the policy pool. The main agent's objective is to defeat all opponents. Simultaneously, the main exploiter engages in battles with the main agent to discover its weaknesses. After meeting the win rate condition or reaching the timeout, the main exploiter is reset, and its copy is added to the policy pool. The league exploiter fights with all agents, and once the win rate condition or timeout is met, its copy is also added to the policy pool, with a 25% probability of being reset. More details can find in Appendix C.1

exploiter, and league exploiter to construct the league. The main agent serves as the primary training agent, tasked with defeating all opponents within the league. The main exploiter engages exclusively in battles against the contemporary main agent, with the primary objective of identifying weaknesses in the contemporary main agent. The league exploiter is employed to enhance the diversity within the league, ensuring that the main agent encounters a wide range of opponents.

However, all the agents are homogeneous in AlphaStar, which may lead to overlapping exploration among agents within the league, resulting in reduced diversity. This reduction in diversity can have implications for the competence and generalization ability of the main agent and ultimately hinder the efficiency of training (Lanctot et al., 2017). In *Shūkai*'s setting, the league of HELT consists of three distinct types of agents, differing primarily both in their **network structure** and in their **mechanism for selecting opponent mixture**. Different structures mentioned in Section 3.1 are used in HELT.

The illustration of HELT is shown in Figure 2. Firstly, the QS agent or the FQS agent serves as the main agent to enhance generalizability. The main agents leverage the PFSP (Vinyals et al., 2019) mechanism that adapts the mixture probabilities proportionally to the win rate of each opponent against the agent. Secondly, the main exploiter shares the same structure (not the parameters) as the main agent and exclusively plays against the contemporary main agent. By engaging in battles against the contemporary main agent, the main exploiter assists in exploring overlooked aspects and helps the main agent uncover potential areas of improvement. Thirdly, the FIS agent serves as the league exploiter. The FIS agent utilizes ID information, enabling it to

learn optimal responses for all characters within the training subset. As a result, it can quickly improve the competence of the league exploiter, thereby enhancing the overall training efficiency. When any agent achieves a winning rate of 80% or encounters a timeout in the settings, it will load its copy into the league. After loading the current copy into the league, the main exploiter will reset, whereas the league exploiter has a 25% probability of resetting. The main agent never reset.

Due to the structural differences between the league exploiter and the main agent, their exploration directions differ, thereby expanding the diversity within the policy space (Lanctot et al., 2017). This diversity better assists the main agent in learning generalized knowledge. Also, the FIS agent can quickly improve its competence cause it can find the best response to the characters in the training subset. Therefore, the league exploiter with the FIS structure will provide more valuable data to the main agent resulting in accelerating the training efficiency. The opponent policy selecting mechanism is PFSP (Vinyals et al., 2019), readers can find the details in Appendix C.2.

3.3. Agent-Human Alignment

Deploying *Shūkai* into Naruto Mobile presents another key challenge of aligning with human players' expectations. The goal of the DRL agent system is to assist players in improving their abilities, so it is essential to provide an agent that offers a certain level of challenge while still being beatable. To achieve this alignment, we introduce additional action masks to constrain the behavior of the DRL agent. In *Shūkai*, we reduce the frequency of agent predictions to align with the reaction speed of human players. We also designed a mechanism that controls the execution probability of the action to generate different levels of *Shūkai*.

To enhance the human-like behavior of the DRL agent, we collected a significant amount of player gameplay data and discovered that players of Naruto Mobile primarily fall into three archetypes $\langle balanced, cautious, aggressive \rangle$. Aggressive players tend to favor proactive attacks, cautious players lean towards defensive counterattacks, and balanced players adapt their actions based on the current situation. Based on these observations, we designed three different types of reward functions to align with the behavioral styles of human players.

- **Balanced.** The agent will trade off the opponent's remaining health point, its health point, and the available resources. It aims to strategically defeat the opponent at the right moment and in a suitable manner.

- **Cautious.** The agent will prioritize its resources and remaining health points, aiming to minimize damage taken while winning the game.
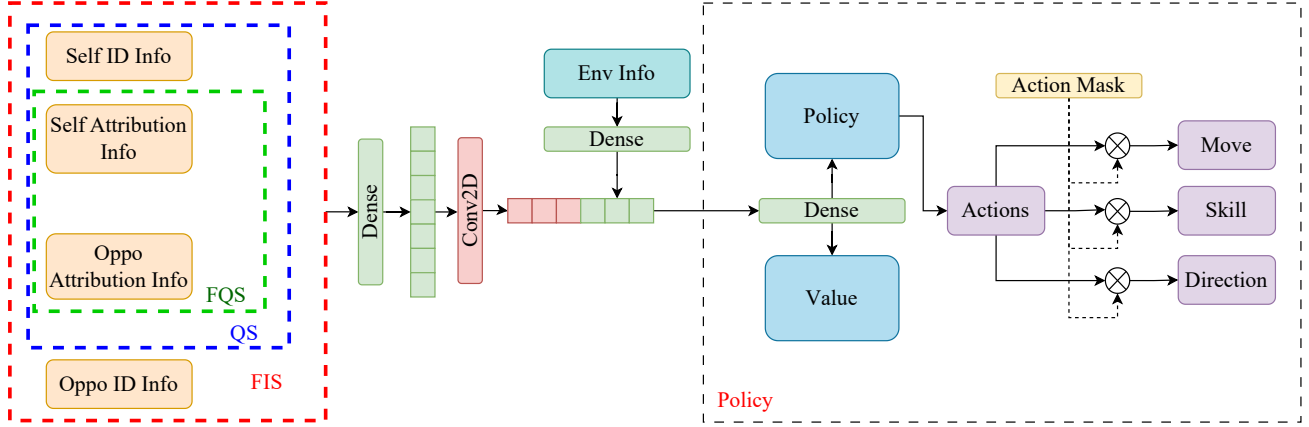
Figure 3. The learning structure of FIS, QS, and FQS. FIS uses both ID and attribution information (numerical information) to model self and opponent. QS uses ID information to model self and only attribution information to model opponent. FQS models self and opponent only with attribution information. In this figure, the red dashed box contains FIS, the blue dashed box contains QS, and the green dashed box contains FQS. These features are processed by the network and then concatenated with the environment feature, the concatenated features are used to predict actions by the policy.

- **Aggressive.** The agent will focus on the opponent's remaining health points. The objective of the agent is to defeat the opponent as quickly as possible.

These reward functions can be used as modular components and integrated into HELT. By utilizing HELT with these reward functions, we can facilitate the development of multi-style agents. Also, we can prevent DRL agents from making confusing or seemingly illogical actions that players may find off-putting. This helps to maintain the illusion of human-like behavior and avoid detection by the players. Please check Section 5 for the agent-human alignment.

3.4. Policy Improvement

In Section 2.2, by modeling the fighting game as a Markov process, we can leverage reinforcement learning algorithms to optimize the agent's strategies. We use the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017) in our system. PPO trains a value function $V_\phi(s_t)$ with a policy $\pi_\theta(a_t|s_t)$, the utilization of importance sampling makes it highly feasible to apply reinforcement learning algorithms in distributed scenarios.

In the training process, the trajectories are sampled from multiple policies, which can exhibit significant deviations from the current policy π_θ. Our updated objectives have remained consistent with PPO (Schulman et al., 2017):

$$\mathcal{L}^\pi(\theta) = \mathbb{E}_t[min(\rho_t(\theta)\hat{A}_t, clip(\rho_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)],$$

where $\rho_t(\theta) = \pi_\theta(s_t|a_t)/\pi_{old}(s_t|a_t)$ is the importance sampling weight and $\hat{A}_t$ is the advantage estimation (Sutton & Barto, 2018) that is calculated by Generalized Advantage

Estimation (GAE) (Schulman et al., 2015). More details can be found in Appendix C.5.

4. Experiments

4.1. Experimental Setup

In this section, we present the experimental results of HELT, using Naruto Mobile as the evaluation platform. For clarity, we define the HELT agent as the final version of the main agent, while the league agents consist of the main agent, main exploiter, and league exploiter. The terms 'FIS agent', 'QS agent', and 'FQS agent' refer to the FIS, QS, and FQS network structures, respectively, when they serve as the main agent. The familiar characters refer to the characters who are in the training subset, and the unfamiliar characters refer to the characters who are out of the training subset.

Naruto Mobile has over 400 characters, making it impractical to train them all simultaneously. To manage this, we categorized the characters into four distinct levels (S, A, B, C) based on their in-game strength and expert prior. We constructed a training subset by selecting 15 characters from the S-level, 15 from the A-level, 10 from the B-level, and 10 from the C-level. In total, 50 characters have been chosen across the four levels (13% of all characters).

In our experimental setup, all agents were trained using 4 NVIDIA T4 GPUs and 3000 CPU cores. The league training consisted of a main agent, a main exploiter, and a league exploiter. A total of 12 GPUs and 9000 CPU cores were utilized for each league training session. The training process comprised 10 rounds, the first 7 rounds lasting 6 hours each, and the final 3 rounds lasting 12 hours each.
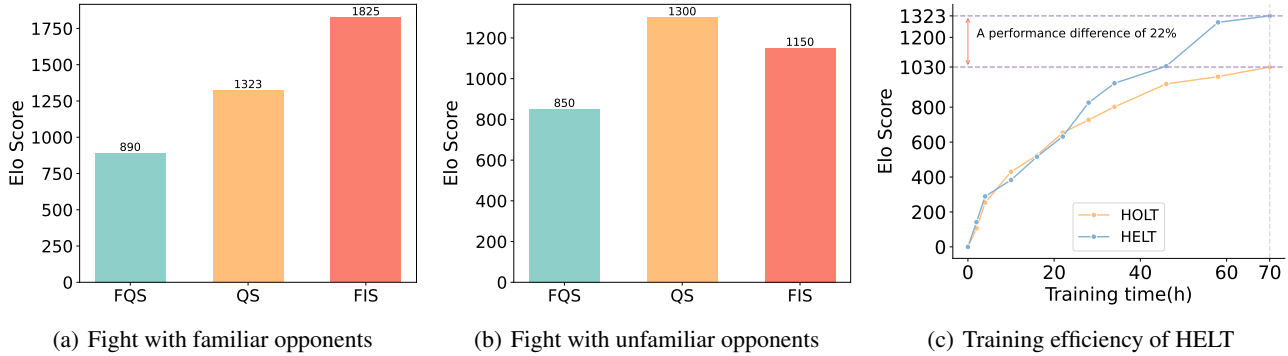
(a) Fight with familiar opponents   (b) Fight with unfamiliar opponents   (c) Training efficiency of HELT

Figure 4. (a) Competence of different structures serve as the main agent in HELT, FIS agent outperforms other agents. (b) Generalizability of different structures serves as the main agent in HELT, the competence of the FIS agent dropped by 36%, while the competence of the QS and FQS agents remained stable. (c) Training efficiency of HELT. After 70 hours of training, HELT achieved a 22% improvement in competence compared to the QS network with homogeneous league training (HOLT).

4.2. Results

To demonstrate the specific competence of HELT, we constructed an "oracle agent" that had access to all information, including cheat information from the client. Additionally, the oracle agent features an expanded character pool, allowing it to choose any character regardless of whether it was in HELT's character pool and training a longer time than HELT agents. HELT agents with different structures are evaluated by having them compete against the oracle agent, the competence results are presented in Figure 4(a), and the generalizability results are presented in Figure 4(b). Each evaluation consisted of 2500 matches. The characters in the character pool were uniformly sampled (all the matchups have been sampled). The competence of the model can be represented by its Elo score (Elo, 1978), with a higher Elo score indicating a higher level of competence.

Competence. The competence result is shown in Figure 4(a). The FIS agents outperformed agents with other structures. This finding suggests that utilizing ID information to model the opponent can significantly enhance competence when dealing with familiar characters. By incorporating this approach, the agents can effectively learn and adapt to the unique characteristics and optimal strategies of different characters in the training subset, resulting in improved overall competence.

In contrast, the FQS and QS agents, which quantify the states, did not perform satisfactorily when facing familiar characters. Particularly, the FQS network exhibited the worst competence. This suggests that quantifying the states can lead to a decrease in competence, and the higher the degree of quantization, the more pronounced the competence decline.

Generalizability. Figure 4(b) displays the result of generalizability, the results indicate an interesting phenomenon: when facing the unfamiliar characters, the competence of the FIS agent dropped by 36%, while the competence of the QS and FQS agents remained stable, with negligible competence decline. This result can be attributed to the FIS agent's over-reliance on ID information for modeling different characters, which reduced its generalization capability. Conversely, appropriately quantifying the states allowed the QS and FQS agents to effectively learn the patterns of gameplay. The excellent generalization performance of QS and FQS agents is crucial in real-game scenarios where targeted training for all characters is not feasible. Therefore, when dealing with a large-scale character pool, a purely ID-based FIS structure may not be the optimal choice.

Training Efficiency. To compare the improvement in training efficiency brought by HELT, we trained a HELT agent with a QS network as the main agent and compared it with a QS network that constructs homogeneous league training (Vinyals et al., 2019). During the training process, the training resources and settings for both models are the same. Specifically, each agent had access to the same number of GPUs and CPUs and set the same number of training epochs and time limits to ensure a fair comparison. The result is shown in Figure 4(c). By comparing the training results, we observed a significant competence difference. After 70 hours of training, the HELT system achieved a 22% improvement in competence compared to the QS network with homogeneous league training. This indicates that the HELT system can converge faster and reach a higher level of competence within the same training time.

The above experimental results indicate that by quantifying states, DRL agents can enhance generalization while

maintaining competence. The introduction of HELT can improve training efficiency and enhance the competence of DRL agents, thereby mitigating key challenges in implementing DRL systems in real-world applications to games. Due to the considerations of both competence and generalization, the QS network has been chosen as the main agent for training in the practical deployment.

5. Real-world application of Shūkai

5.1. Human Evaluation

The main application of *Shūkai* in the Naruto Mobile is to serve as a player's training partner. The primary purpose is to enhance the gaming abilities and skills of players through fighting with *Shūkai*. Therefore, ensuring a satisfying player gaming experience is of paramount importance, and the competence of DRL agents needs to be challenging while still beatable (Bakker et al., 2011). We classify *Shūkai* into three levels: beginner, intermediate, and advanced, based on the number of frames for delay prediction and the probability of action execution. The higher the level of the agent, the lower the delay in frame prediction and the higher the probability of executing actions.
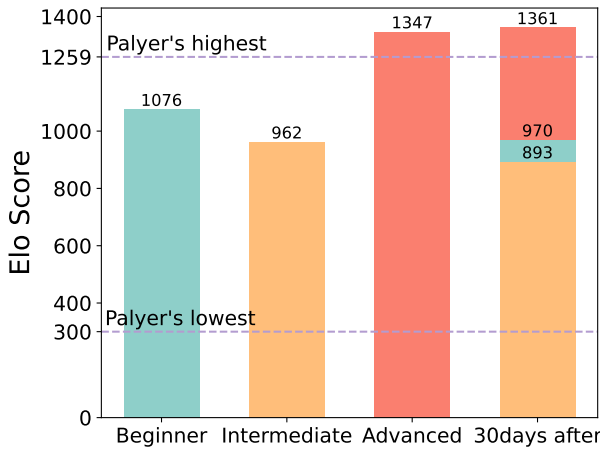

Figure 5. The competence of three different levels of *Shūkai* compete against human players in-game matches. The Elo scores of different level agents after 30 days are represented in the rightmost column. After 30 days, *Shūkai* beginner and *Shūkai* intermediate experience a decrease in Elo score, indicating *Shūkai* can help players to enhance their abilities. *Shūkai* advanced maintain its Elo score, suggesting advanced *Shūkai* is challenging even for the skillful player.

The ranking system of Naruto Mobile uses Elo scores, ranging from 0 to 2599. *Shūkai* are primarily deployed in the score range of 0-1299. Within this range, we deploy *Shūkai* beginner for scores 0-600, *Shūkai* intermediate for scores 600-999, and *Shūkai* advanced for scores 999-1299. *Shūkai* has been online in Naruto for over a year and has partici-

pated in more than 500 million player-agent matches. We conducted a competence analysis using a sample of 40 million game matches from player gameplay data, collected over time. The results are displayed in the Figure 5.

In each match, players have a certain probability of being matched with either *Shūkai* or other human players as opponents. With players having different skill levels, talents, and experiences, using human players' performance against each other as a baseline could introduce certain inconsistencies. We ultimately chose to focus on *Shūkai*'s Elo score change trend. This aim to showcase *Shūkai*'s impact within specific score ranges during 30 days, rather than individual player performance changes.

According to the Figure 5, *Shūkai* beginner achieved Elo scores surpassing their respective rating brackets. This can be attributed to the novice players' initial lack of proficiency which needs competence improvement. After 30 days, *Shūkai* beginner experienced a 9% decrease in Elo score, providing evidence of player skill enhancement through the utilization of *Shūkai*. Similarly, *Shūkai* intermediate followed a similar pattern to *Shūkai* beginner. With the training provided by *Shūkai* intermediate, players demonstrated an improvement in skill level for 30 days. Conversely, the Elo scores of the *Shūkai* advanced increased after 30 days. However, the difference in Elo scores from the highest player rating was less than 10%, suggesting that though *Shūkai* advanced provided a challenge within that rating bracket, they were not unbeatable. The results also demonstrate the generalizability of *Shūkai* due to the inclusion of characters selected by players that were not exclusively from the training subset which is the same as the experiment.

5.2. Agent-Human Alignment

To discuss the consistency between DRL agents and human players, we have designed a scoring system for detecting in-game behaviors based on human expert priors. This scoring system can score based on the semantic-level behavior of human players or DRL agents in the game, such as the rationality of using skills in a certain situation, etc. Figure 6 displays the differences in cumulative distribution function (CDF) between human players and different DRL agents across Five metrics. The x-axis represents the reward scores, while the y-axis represents the probabilities. Higher reward scores indicate better behaviors.

Metrics. Substitution indicates the release of ultimate defensive skills, **Special** represents the release of the highest damage-dealing skill, **Blitz** indicates a preference for taking the initiative at the beginning of the game, **Counter** indicates a tendency to counter-attack at the beginning of the game, and **Attack** represents regular attacks without using skills. Agents that were induced with behavior through three types of rewards mentioned in Section 3.3, over 120,000
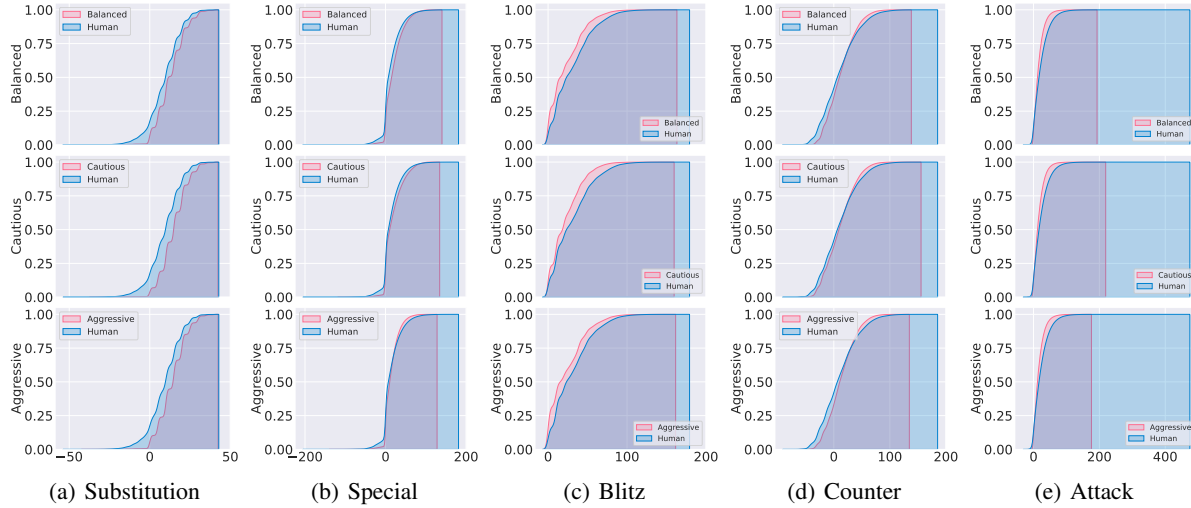
(a) Substitution   (b) Special   (c) Blitz   (d) Counter   (e) Attack

Figure 6. A comparison of the cumulative distribution function (CDF) between human players and different agents across different metrics. The x-axis represents the reward scores, while the y-axis represents the probabilities. Higher reward scores indicate better behaviors. Agents that were induced with behavior through three types of rewards mentioned in Section 3.3, over 120,000 game instances have been collected for comparison with human players.

game instances have been collected for comparison with human players. These players, with Elo scores of 1500 or above, exhibit their experience and expertise in the Naruto Mobile game.

The result in Figure 6(d) indicates that DRL agents are generally more cautious than human players. However, different reward functions still have an impact on the behavior of the DRL agents. The aggressive agent receives lower rewards in the **Counter** metric compared to the cautious agent, while the balanced agent falls in between the two. The results in Figure 6(a) describes *Shūkai* are more precise in releasing defensive skills compared to human players, which is why they consistently outperform in Figure 6(d). In Figure 6(e), *Shūkai* do not prioritize regular attacks. This can be attributed to the *Shūkai*'s precise skill releases (Figure 6(b)).

In conclusion, *Shūkai* exhibit more precise skill releases but tend to be more cautious than humans. However, this also demonstrates that we can induce different behavioral styles in the agents through special reward functions, which can guide us in designing agents with specific styles. The evaluation system works to effectively observe the gap between human players and DRL agents, helping design more human-like agents to enhance the gaming experience. Overall, *Shūkai* still has some gap compared to human players in terms of high reward scores, indicating the need for further improvement in our algorithms to encourage more advanced exploration by the agents. Additional detailed data can be found in the Appendix E.

6. Related Work

6.1. DRL for Games

With the development of DRL, there are several successful applications in different domains such as robotic (Andrychowicz et al., 2017), autonomous vehicles (Aradi, 2020), and video games. DQN (Mnih et al., 2015) proposed trying its hand at Atari games using DRL from pixel input. AlphaGo (Silver et al., 2016; 2017; 2018) solved the game of Go with self-play. Researchers have discovered the ability of self-play to solve game problems, which has sparked interest among researchers to apply it in the field of game theory. applied similar methods to card games. (Moravčík et al., 2017; Zha et al., 2021; Zhao et al., 2022) applied similar methods to card games such as Texas Hold'em and DouDizhu (a popular Chinese card game). AlphaStar (Vinyals et al., 2019) and OpenAI Five (Berner et al., 2019) used reinforcement learning to train game AI in two famous games, StarCraft II and Dota 2. Jaderberg et al. (2019) achieved human performance in Capture the Flag. Subsequently, the AI they designed in the MOBA game King of Glory demonstrated outstanding performance. Subsequently, the AI designed by (Ye et al., 2020b) and (Ye et al., 2020a) in the MOBA game King of Glory demonstrated outstanding performance. Additional discussion can be found in the Appendix B.

6.2. DRL for Fighting Games

In the field of game AI for fighting games, there are numerous notable precedents, primarily focused on the applica-

tion scenario of FightICE(Khan et al., 2022). Kim & Ahn (2018) combined genetic algorithms and Monte Carlo Tree Search (MCTS) to find general solutions that address the constraints of game AI while maintaining competitiveness. Ishii et al. (2018) introduced the concept of aligning with user personas in game AI, which has gained significant attention in recent research. Kim et al. (2020) introduced a reinforcement learning approach combined with MCTS to enhance the performance of the game AI and proposed novel evaluation metrics to ensure the stability of game AI. Oh et al. (2021) introduced self-play curriculum learning and three types of rewards to further enhance AI personas and performance. Halina & Guzdial (2022) introduced a diversity-based deep reinforcement learning approach for generating a set of agents with similar difficulty but utilizing diverse strategies. Ishii et al. (2018) and Halina & Guzdial (2022) share similarities with our research, but the core attention of HELT lies in addressing the training efficiency and generalization issues caused by a large pool of characters. Additionally, we introduce multi-style rewards to align with players and ensure a satisfactory gaming experience.

7. Conclusions

In this paper, we propose a practical DRL agent system called *Shūkai* to address key challenges in its deployment to the fighting game Naruto Mobile, focusing on achieving a balance between training efficiency, model competence, and generalizability. This is achieved by utilizing a unified model to control multiple characters simultaneously, quantifying input states to enhance generalization, and introducing HELT to accelerate training efficiency. Experimental results demonstrate the powerful capabilities of *Shūkai*, which has been adopted by Naruto Mobile since 2022 for player training, aiding in skill improvement. Furthermore, we present the concept of aligning DRL agent behavior with human players, analyzing the differences between DRL agents and human players through reward and practical evaluations. This is crucial for the practical application of DRL agents in commercial games. With the integration of *Shūkai*, the game has witnessed a notable 5% maximum increase in the retention rate and an average improvement of 4%.

8. Future Work

Shūkai has been employed by Naruto Mobile to train its agents and have achieved great results. Our future work lies in extending the principles of *Shūkai* and applying them to more games and domains. We have successfully deployed *Shūkai* to two new fighting games that are currently under development at Tencent. Additionally, *Shūkai* is also planned to be used in Arena Breakout, a first-person shooter game published by Tencent.

We will continue to promote the practical implementation of reinforcement learning and further its theoretical development, enabling reinforcement learning to better empower the real world.

Acknowledgement

Impact Statement

This paper presents work that aims to advance the field of Machine Learning. This work marks a further step in the deployment of DRL systems in a large-scale commercial fighting game. To the best of our knowledge, this is the first example of a DRL system being employed by a commercial fighting game.

Reinforcement learning has seen applications and advancements in various domains as a means to solve sequential decision-making problems. We aim to promote the coexistence of better reinforcement learning methods with human applications. In the Introduction section, we mentioned that most previous work focused on training stronger and more competitive agents. However, as game players, we don't necessarily want to compete against unbeatable agents. It is based on this understanding that we introduced *Shūkai*. To address the training and generalization challenges posed by a large character pool, we proposed HELT and conducted initial explorations of human-agent alignment. It's important to emphasize that this is a practical implementation of reinforcement learning. While our focus was on Naruto Mobile in this paper, the HELT approach and the concept of human-agent alignment can be applied to reinforcement learning in general, regardless of whether it's within a game or in other contexts.

Returning to games, having controllable anthropomorphic agents in the game ecosystem benefits both players and game developers. It was observed that human players are enthusiastic about immersing themselves in the story and becoming part of it. This means that highly anthropomorphic agents can help players derive more enjoyment from their gaming experience, thereby promoting player engagement and social exploration within the game. This represents a positive influence. For game developers, having controllable anthropomorphic agents can contribute to the creation of better activities and gameplay designs, ultimately producing superior games.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Aradi, S. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):740–759, 2020.

Bakker, A. B., Oerlemans, W., Demerouti, E., Slot, B. B., and Ali, D. K. Flow and performance: A study among talented dutch soccer players. *Psychology of Sport and Exercise*, 12(4):442–450, 2011.

Bejani, M. M. and Ghatee, M. A systematic review on overfitting control in shallow and deep neural networks. *Artificial Intelligence Review*, pp. 1–48, 2021.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Brown, G. W. Iterative solution of games by fictitious play. *Act. Anal. Prod Allocation*, 13(1):374, 1951.

Brown, N. and Sandholm, T. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Cheng, T.-M., Hung, S.-H., and Chen, M.-T. The influence of leisure involvement on flow experience during hiking activity: Using psychological commitment as a mediate variable. *Asia Pacific Journal of Tourism Research*, 21 (1):1–19, 2016.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 1282–1289. PMLR, 2019.

Csikszentmihalyi, M. *Beyond boredom and anxiety.* Jossey-bass, 2000.

Dosovitskiy, A. and Koltun, V. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

Elo, A. The rating of chess players, past and present (arco, new york). 1978.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference On Machine Learning (ICML)*, pp. 1126–1135. PMLR, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pp. 1861–1870, 2018.

Hafner, D., Lillicrap, T. P., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2467–2475, 2019.

Halina, E. and Guzdial, M. Diversity-based deep reinforcement learning towards multidimensional difficulty for fighting game ai. *arXiv preprint arXiv:2211.02759*, 2022.

He, Q., Su, H., Zhang, J., and Hou, X. Frustratingly easy regularization on representation can boost deep reinforcement learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pp. 20215–20225. IEEE, 2023. doi: 10.1109/CVPR52729.2023.01936. URL https://doi.org/10.1109/CVPR52729.2023.01936.

He, Q., Zhou, T., Fang, M., and Maghsudi, S. Adaptive regularization of representation rank as an implicit constraint of bellman equation. *CoRR*, abs/2404.12754, 2024. doi: 10.48550/ARXIV.2404.12754. URL https://doi.org/10.48550/arXiv.2404.12754.

Heinrich, J. and Silver, D. Deep reinforcement learning from self-play in imperfect-information games, 2016.

Heinrich, J., Lanctot, M., and Silver, D. Fictitious self-play in extensive-form games. In *International Conference on Machine Learning (ICML)*, 2015.

Houthooft, R., Chen, X., Isola, P., Cruz, E., and Abbeel, P. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.

Ishii, R., Ito, S., Ishihara, M., Harada, T., and Thawonmas, R. Monte-carlo tree search implementation of fighting game ais having personas. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8. IEEE, 2018.

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. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

Khan, I., Van Nguyen, T., Dai, X., and Thawonmas, R. Dare-fightingice competition: A fighting game sound design and ai competition. In *2022 IEEE Conference on Games (CoG)*, pp. 478–485. IEEE, 2022.

Kim, D.-W., Park, S., and Yang, S.-i. Mastering fighting game using deep reinforcement learning with self-play. In *2020 IEEE Conference on Games (CoG)*, pp. 576–583. IEEE, 2020.

Kim, M.-J. and Ahn, C. W. Hybrid fighting game ai using a genetic algorithm and monte carlo tree search. In *Proceedings of the Genetic And Evolutionary Computation Conference Companion*, pp. 129–130, 2018.

Lan, S., Zhang, R., Yi, Q., Guo, J., Peng, S., Gao, Y., Wu, F., Chen, R., Du, Z., Hu, X., et al. Contrastive modules with temporal attention for multi-task reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Leslie, D. S. and Collins, E. J. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006.

Li, L., Lyu, J., Ma, G., Wang, Z., Yang, Z., Li, X., and Li, Z. Normalization enhances generalization in visual reinforcement learning. *arXiv preprint arXiv:2306.00656*, 2023.

Lyu, J., Bai, C., Yang, J., Lu, Z., and Li, X. Cross-domain policy adaptation by capturing representation mismatch. *arXiv preprint arXiv:2405.15369*, 2024a.

Lyu, J., Wan, L., Li, X., and Lu, Z. Towards understanding how to reduce generalization gap in visual reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 2369–2371, 2024b.

Lyu, J., Wan, L., Li, X., and Lu, Z. Understanding what affects generalization gap in visual reinforcement learning: Theory and empirical evidence. *arXiv preprint arXiv:2402.02701*, 2024c.

McMahan, H. B., Gordon, G. J., and Blum, A. Planning in the presence of cost functions controlled by an adversary. In *International Conference on Machine Learning (ICML)*, pp. 536–543, 2003.

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. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Moravčík, M., Schmid, M., Burch, N., Lisỳ, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

Oh, I., Rho, S., Moon, S., Son, S., Lee, H., and Chung, J. Creating pro-level ai for a real-time fighting game using deep reinforcement learning. *IEEE Transactions on Games*, 14(2):212–220, 2021.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:27730–27744, 2022.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 16–17, 2017.

Peng, S., Hu, X., Yi, Q., Zhang, R., Guo, J., Huang, D., Tian, Z., Chen, R., Du, Z., Guo, Q., Chen, Y., and Li, L. Self-driven grounding: Large language model agents with automatical language-aligned skill learning. *ArXiv*, abs/2309.01352, 2023a. URL https://api.semanticscholar.org/CorpusID:261530737.

Peng, S., Hu, X., Zhang, R., Guo, J., Yi, Q., Chen, R., Du, Z., Li, L., Guo, Q., and Chen, Y. Conceptual reinforcement learning for language-conditioned tasks. In *AAAI Conference on Artificial Intelligence*, 2023b. URL https://api.semanticscholar.org/CorpusID:257427058.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International Conference On Machine Learning (ICML)*, pp. 1312–1320. PMLR, 2015.

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

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017.

Shafer, D. M. Causes of state hostility and enjoyment in player versus player and player versus environment video games. *Journal of Communication*, 62(4):719–737, 2012.

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. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Sun, X., Hoffman, J., Saenko, K., and Darrell, T. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 675–684, 2017.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Wang, K., Liang, Y., Li, X., Li, G., Ghanem, B., Zimmermann, R., Yi, H., Zhang, Y., Wang, Y., et al. Brave the wind and the waves: Discovering robust and generalizable graph lottery tickets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Wang, X. Balancing the ai strength of roles in self-play training with regret matching+, 2024.

Ye, D., Chen, G., Zhang, W., Chen, S., Yuan, B., Liu, B., Chen, J., Liu, Z., Qiu, F., Yu, H., et al. Towards playing full moba games with deep reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:621–632, 2020a.

Ye, D., Liu, Z., Sun, M., Shi, B., Zhao, P., Wu, H., Yu, H., Yang, S., Wu, X., Guo, Q., et al. Mastering complex control in moba games with deep reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pp. 6672–6679, 2020b.

Yi, Q., Zhang, R., Peng, S., Guo, J., Hu, X., Du, Z., Zhang, X., Guo, Q., and Chen, Y. Object-category aware reinforcement learning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Yi, Q., Zhang, R., Peng, S., Guo, J., Gao, Y., Yuan, K., Chen, R., Lan, S., Hu, X., Du, Z., Zhang, X., Guo, Q., and Chen, Y. Online prototype alignment for few-shot policy transfer. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 39968–39983. PMLR, 2023. URL https://proceedings.mlr.press/v202/yi23b.html.

Zha, D., Xie, J., Ma, W., Zhang, S., Lian, X., Hu, X., and Liu, J. Douzero: Mastering doudizhu with self-play deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 12333–12344, 2021.

Zhao, Y., Zhao, J., Hu, X., Zhou, W., and Li, H. Douzero+: Improving doudizhu ai by opponent modeling and coach-guided learning. In *IEEE Conference on Games (CoG)*, pp. 127–134, 2022.

A. Naruto Mobile

A.1. Introduction of Naruto Mobile

Naruto Mobile is an online fighting game published by Tencent Games with over 100 million registered users. Naruto Mobile has a large-scale character pool with more than 400 characters(ninjas). Each ninja has its special unique characteristics. Players of Naruto Mobile can choose a ninja from the character pool and use the ninja to fight against other characters. The winning condition for all players is to defeat their opponents. In every episode, two players choose a character of their own, and they control their character against the other player(i.e., the opponent). The episode will terminate when a player's health drops to 0 or the game time ends. Our objective is to create an AI system for Naruto Mobile that is seamlessly integrated into the game and actively participates in battles against players. This AI system will be consistently deployed, ensuring its continuous presence and engagement within the game environment. Besides, the system is required to have sufficient generalization capabilities to be able to control all characters through only one model.

Each ninja is controlled using a virtual joystick to manage movement and direction. They possess a basic attack, two primary skills, one ultimate skill, one summoning skill, and one defensive skill. Furthermore, certain skills have a mechanism that allows for multiple subsequent activations after their initial release. Additionally, ninjas have the option to choose different scroll items to enhance the diversity of their combat strategies.

A.2. Hitbox and Hurt Box

The hurt box and hitbox play crucial roles in determining the outcome of attacks. The hurt box represents the vulnerable area around a character's body where they can take damage. It acts as an invisible or transparent box that detects incoming attacks. When an opponent's attack connects with the hurt box, the character will suffer damage. The size and shape of the hurt box can vary based on the character's stance, animation, and the specific move being executed.

On the other hand, the hitbox represents the area around a character's attacks where they can deal damage to the opponent. Similar to the hurt box, it is an invisible or transparent box that detects potential hits. When a hitbox overlaps with the hurt box of the opponent's character, it registers as a successful hit, causing damage. The size and shape of the hitbox can vary depending on the specific move being executed.

Both the hurt box and hitbox are integral components of the hit detection system in fighting games. They determine whether an attack successfully lands on an opponent, resulting in damage, or if it misses. Precise placement and sizing of the hitbox and hurt box are essential for balanced and fair gameplay, as they directly impact the effectiveness and reliability of attacks and defensive maneuvers.

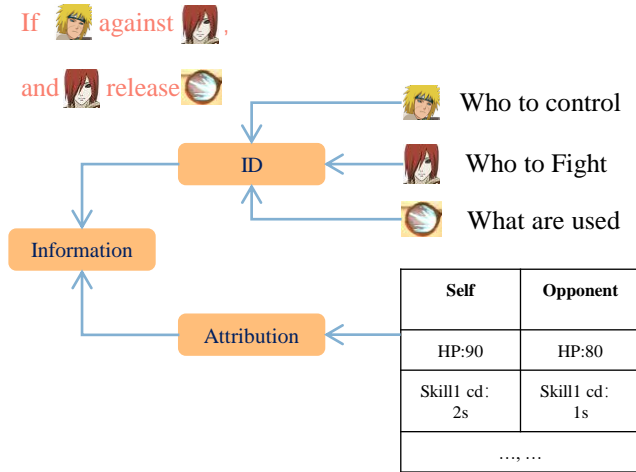## A.3. Different Information in Naruto Mobile

Figure 7. ID information and attribution information

All the information from the client is divided into distinguishable categories to model both self and opponents effectively(Figure 7). This differentiation involves the utilization of ID information, which provides directed references to characters, skills, actions, etc., and general numerical information, which represents less explicit inputs to the network.

From Figure 7, it was observed that the ID formation represents the figuration that which character is being controlled, which character is fighting with, and what skills are used. The attribution information is the specific numerical value of the entity.

A.4. MDP of Naruto Mobile

Considering the opponent as a component of the environment, the Markov Decision Process (MDP) framework for Naruto Mobile's training agent is formalized as a six-element tuple $\langle \mathcal{S}, \mathcal{A}, P, R, T, \gamma \rangle$. Here, $\mathcal{S}$ represents the state space and $\mathcal{A}$ denotes the action space. R is defined as the reward function, P encapsulates the state transition probabilities, T signifies the time horizon, and γ is the discount factor. At each discrete time step t, the agent observes the current state of the environment, denoted as $s_t \in \mathcal{S}$. The agent's policy π, a mapping from states to actions, selects an action a_t based on the current state, formalized as $a_t \sim \pi(\cdot|s_t)$. The state then transitions to the next state $s_{t+1} \sim P(\cdot|s_t, a_t)$, and the agent receives a reward r_{t+1}. The state value function is $V_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{T} \gamma^k R_{t+k+1}|S_t = s]$, and the action value function is $Q_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{T} \gamma^k R_{t+k+1}|S_t = s, A_t = a]$ are always leveraged to optimize policy. The advantage function $A(s, a) = Q(s, a) - V(s)$ is used to measure the quality of action a compared to the average quality.

The reward function r_t is defined based on the variations in health points r_{HP}^t and the game's outcome. r_t^{HP} is defined as:

$$r_t^{HP} = HP_{t+1}^{self} - HP_t^{self} + HP_t^{oppo} - HP_t^{oppo}, \tag{1}$$

Where the HP^{self} is the health points of the agent, and HP^{oppo} is the health points of the opponent. Then the reward function r_t is defined as:

$$r_t = \begin{cases} r_t^{HP}, & t < T \\ 7 \cdot r_t^{HP}, & t = T \end{cases} \tag{2}$$

The agent is constructed by a neural network with parameter θ, i.e., the policy $\pi_\theta(a_t|s_t)$. The optimization goal of the RL algorithm is to find an optimal policy π^* to maximize the cumulative reward:

$$\pi^* = \arg\max_{\pi_\theta} \mathbb{E}_{(s_t, a_t) \sim \pi_\theta}[\sum_{t=0}^{n} \gamma^t r(s_t, a_t)] \tag{3}$$

A.5. Action Representation of Naruto Mobile

The action space of Naruto Mobile is composed of three action heads, each represented by a one-hot vector. When predicting an action, the network simultaneously predicts all the action heads. These action heads are then passed through an action mask processing and delivered to the client for execution. Table 1 shows the action head.

B. Related Work

B.1. Policy Improvement in Multi-agent Competition

Multi-agent environment means there exists more than one agent in the environment. These agent needs to cooperate or compete to achieve their goals. In the competition setting, agents defeat other agents to win. Every agent has to improve their policy to make sure that is un-exploitable. There exists plenty of research to achieve un-exploitable. Naive self-play is a common approach to policy improvement, it essentially does best response in an iterated way. Self-play only focuses on finding the best response to the opponent's latest strategy, which would lead to circular behaviors in in-transitive games. To address the limit of self-play, Fictitious Play (FP) (Brown, 1951) was proposed. FP maintains a belief over the historical action that the opponent has played, the learning agent then takes the best response to this empirical distribution. Generalized Weakened Fictitious Play (GWFP) (Leslie & Collins, 2006) was proposed to weaken the constraint of FP,

Table 1. Action representation of Naruto Mobile

Action Space	Action Head	Action Name		Key mapping
Move	Action_ud	None		JoyStick (left hand)
		Up		
		Down		
	Action_lr	None		
		Left		
		Right		
Skill	Skill	None	Substitute	Fight buttons (right hand)
		Punch	Summon	
		Skill1	Scroll	
		Skill2	Subskill1	
		Skill3	Subskill2	
Direction	Direction	Direction1	Direction5	Joystick skill
		Direction2	Direction6	
		Direction3	Direction7	
		Direction4	Direction8	

which was accomplished by allowing approximate best response and perturbed average strategy updates while preserving the convergence guarantee under certain conditions.

The methods mentioned above are in the formulation of Norm-formal Game (NFG). To expand NFG to Extensive-form Game (EFG) while utilizing modern machine learning, two variants of FP are implemented by Fictitious Self-play (FSP) (Heinrich et al., 2015). The first variant introduces full-width extensive-form fictitious (XFP). It is realization equivalent to a normal-form fictitious play and therefore inherits its convergence guarantees. The second variant is FSP based on the first variant but leveraging reinforcement learning (RL) to do the best response and supervised learning (SL) to average strategy update. With the development of deep learning, Heinrich & Silver (2016) leverages the neural networks to implement the RL and SL of FSP. Prioritized Fictitious self-play (PFSP) (Vinyals et al., 2019) calculates weight based on winning rate and selects opponents based on weight to avoid the waste of computing resources.

Double Oracle (DO) (McMahan et al., 2003) is also an iterated best response method, different from self-play, it best responds to the opponent's Nash equilibrium at each iteration. Although DO can guarantee the convergence, it will traverse the entire policy space in the worst case. Policy Space Response Oracle (PSRO) (Lanctot et al., 2017) is a generalization of DO on meta-games. DO focuses on the action at each iteration, but PSRO focuses on policy space for every agent. Thus, PSRO can generalize all previous methods by setting different forms of the distribution of policy space.

B.2. DRL Application in Games

With the development of deep reinforcement learning (DRL), there are several successful applications in different domains such as robotic (Andrychowicz et al., 2017), autonomous vehicles (Aradi, 2020), and video games. Video games are the most suitable scene serving as DRL environment owing to their data accessibility and timely feedback. Leveraging DRL to develop game AI in video games has been stimulating researchers' interest. DQN (Mnih et al., 2015) proposed trying its hand at atari games using DRL from pixel input. Subsequently, AlphaGo (Silver et al., 2016), efficiently combined the policy and value networks with MCTS and defeated the world champion of Go. Without human knowledge, AlphaGo Zero (Silver et al., 2017) defeated AlphaGo solely through self-play to improve policy. Silver et al. (2018) generalizes AlphaGo Zero to AlphaZero which can achieve superhuman competence in chess, shogi, and Go through self-play. The result of AlphaZero shows the power of DRL and self-play.

After tackling the simple perfect-information game, researchers' interest began to focus on more complex game scenarios such as poker, Dota2, and StarCraft II. These environments have more complex state-action space. DeepStack (Moravčík et al., 2017) defeated with statistical significance professional poker players in heads-up no-limit Texas hold 'em which combines recursive reasoning, decomposition, and learned intuition through self-play. DouZero (Zha et al., 2021) and DouZero+ (Zhao et al., 2022) are proposed to solve DouDizhu through enhancing traditional Monte-Carlo methods with deep neural networks, action encoding, parallel actors, and opponent modeling. For real-time strategy (RTS) games, AlphaStar

(Vinyals et al., 2019) achieved the grandmaster level in StarCraft II through league training with PFSP. And for the sub-genre of RTS games, Multi-player Online Battle Arena (MOBA) games are also challenging. OpenAI Five (Berner et al., 2019) is the first AI system to defeat the world champions in a famous MOBA game as known as Dota2. OpenAI Five leveraged existing reinforcement learning techniques, scaled to learn from batches of approximately 2 million frames every 2 seconds. Ye et al. (2020b) tackled MOBA game from the perspectives of both system and algorithm. They further proposed a MOBA AI learning paradigm that methodologically enables playing full MOBA games with deep reinforcement learning (Ye et al., 2020a).

In this paper, we focus on the Fighting game which is an imperfect-information two-player zero-sum game called Naruto Mobile. Naruto Mobile has a large pool of characters and complex state-action space, with a high demand for real-time feedback. We design an AI system with heterogeneous league training to tackle these challenges. Our system trained on only 15% of the characters but achieves high-level player competence across all characters. In particular, our system is consistently deployed in the game and engages in battles with players.

B.3. Generalization in Reinforcement Learning

Previous research has extensively investigated the challenges of generalization in reinforcement learning. One line of work focuses on addressing the problem of overfitting, where an RL agent fails to generalize its learned policy to unseen states or tasks. Approaches such as experience replay (Mnih et al., 2015) and regularization techniques (Houthooft et al., 2016) have been proposed to mitigate this issue. Another direction of research explores the concept of transfer learning in RL, aiming to enable knowledge transfer from one task to another. Methods like domain adaptation (Sun et al., 2017) and meta-learning (Finn et al., 2017; Lan et al., 2024) have been employed to facilitate better generalization across tasks. He et al. (2023; 2024) discusses the capacity loss problem in reinforcement learning to ensure better stability and generalization of the network. Additionally, recent studies have investigated the role of model-based RL algorithms (Hafner et al., 2019) and intrinsic motivation (Pathak et al., 2017) in enhancing generalization capabilities. Li et al. (2023) and Lyu et al. (2024a;b;c) discuss the generalization caused by vector input and vision input.There are some methods that employ causal inference and language models to enhance the multi-task generalization of DRL (Peng et al., 2023b;a), and some object-oriented generalization works (Yi et al., 2022; 2023). In addition, there is also some work on network pruning that can be considered for enhancing the generalization of models in reinforcement learning (Wang et al., 2023). Despite these efforts, the problem of achieving robust and reliable generalization in RL remains a challenging and active research area.

C. Details about HELT

C.1. Learning Architecture

Figure 8 shows the learning architecture of Naruto Mobile. The input of the QS network comprises three components: self-modeling, opponent-modeling, and env information. Self-modeling still utilizes the ID information. Opponent modeling of the QS network only with attribution information, thereby liberating the QS network from the constraints of the opponent's character pool and enhancing the generalizability of the QS network.

Env information incorporates spatial and relative characteristics of the two agents, in addition to their health and time attributes. All of these features are encoded through fully connected layers and concatenated. The agent's actions, encompassing movement direction and optional skills, are generated through fully connected layers. The purpose of the QS network is to improve generalization while ensuring strength. The QS agent is expected to be able to serve as a universal model to control a part of characters against all characters.

The FQS network is similar to the QS network and does not use any ID information, only the attributes information. FQS agent is expected to obtain models that can be used across all characters.

Different from the QS and FQS networks, FIS uses full ID information to model all the entities whether self-modeling or opponent-modeling. The advantage is that FIS can find the optimal model to defeat the opponent. The disadvantage is that when facing an opponent FIS has never seen before, the competence will decrease.
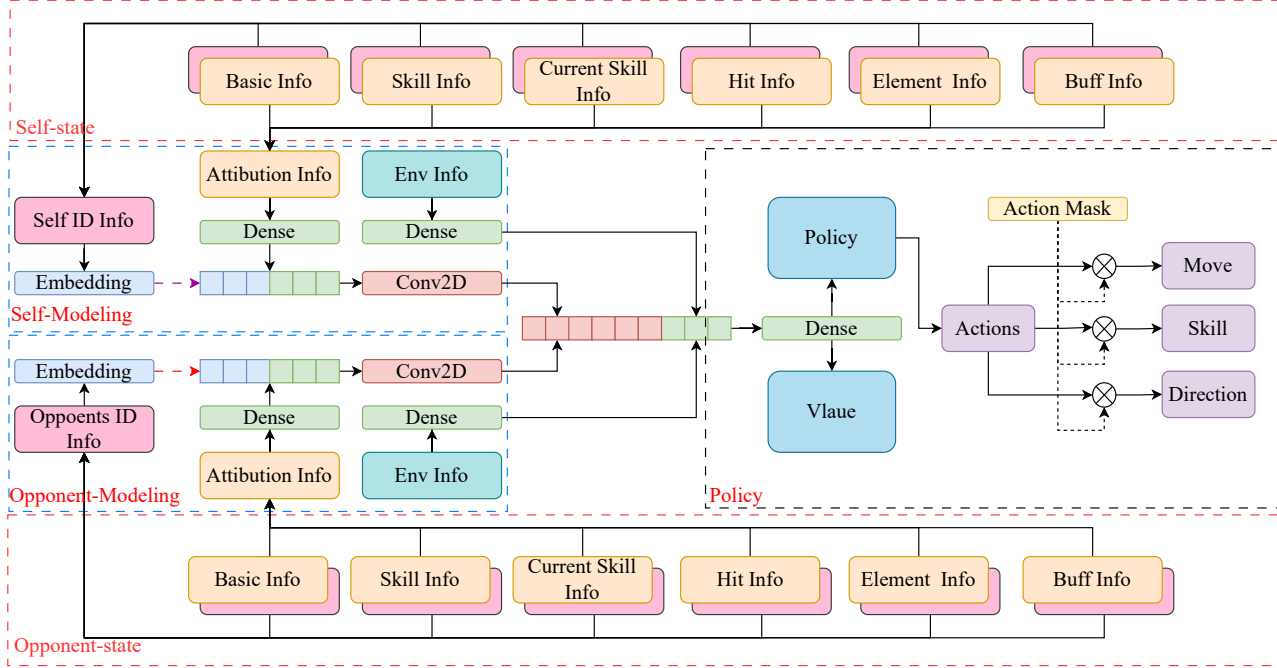
Figure 8. The structure of FIS, QS, and FQS. while the purple dashed line and the red dashed line are activated, the structure is FIS. When the purple dashed line is activated and the red dashed line deactivated, the structure is QS. While the purple dashed line and the red dashed line are both deactivated, the structure is FQS.

C.2. Opponent Policy Selecting

Prioritized fictitious self-play(PFSP) In this paper, our PFSP algorithm is similar to AlphaStar. Giving a learning agent A, we sample the frozen opponent O from a candidate set $\mathbf{S}$ with probability

$$\frac{f(\mathbb{P}[A \ beats \ O])}{\sum_{S \in \mathbf{S}} f(\mathbb{P}[A \ beats \ S])} \tag{4}$$

Where $f : [0, 1] \to [0, \infty]$ is the weighting function.

In this paper, we apply $f_{hard}(x) = (1 - x)^p$ as the hard weighting to encourage focusing on the hardest opponents; and $f_{var}(x) = x(1 - x)$ as the var weighting to encourage focusing on the opponents around the same level. Note that not only do we dynamically utilize PFSP based on win rates during the training process, but we also employ PFSP weights to streamline the model pool when determining opponents for each iteration.

C.3. Populating the League

Same as Alphastar, throughout the training process, we employed three distinct types of agents that primarily differed in their opponent distribution during training, the snapshotting procedure used to create a new player, and the probability of resetting to the previous parameters. Due to the lack of high-quality player data in Nurotu Mobile, we have designed different resetting mechanisms to optimize our exploitation.

Main agents are trained with a proportion of 100% PFSP against all past players in the league. Although Alphastar leverages different proportions of PSFP and SP, applying heterogeneous agents ensures diversity among agents within the entire league. When timeout hours or the main agent defeats all agents in the league in more than 80% of games, a copy of the main agent is added as a new agent to the league. The main agent never reset.

League exploiters are trained with PFSP and their frozen copies are added to the league when they beat all agents in the league in more than 80% of games or after a timeout. At the beginning of the next iteration, the league exploiters have a
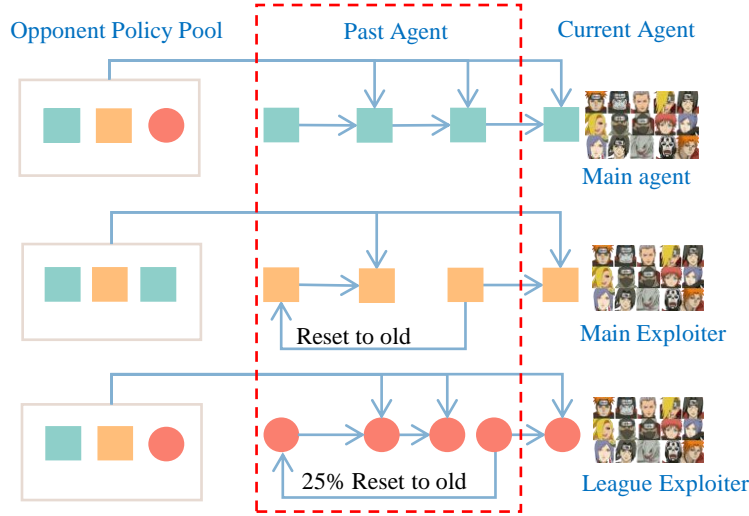
Figure 9. Population the league.

25% probability that the agent is reset. Owing to the lack of high-quality player data, the league exploiter randomly selects a model from the models of the previous three iterations to serve as the current model. This mechanism allows the league exploiter to balance convergence speed and the ability to explore diversity effectively. The league exploiter has a reset probability of 0 for the first three generations of training.

Main exploiters always play against main agents. To address the issue of low win rates during the initial stages of training for the main exploiters, they are trained by PFSP with f_{var} in the first three iterations. Main agents are added to the league when the main agents are defeated in more than 80% of games, or after a timeout. The reset mechanism is similar to the league exploiter, but the main exploiter will reset at every start of the iteration. The primary objective of main exploiters is to identify and expose weaknesses within the main agents, thereby facilitating their enhancement and fortification.

C.4. Opponent Character Selecting

The opponent character selecting strategy is accomplished by Wang (2024). The idea is based on regret matching (RM). For details, readers can refer (Wang, 2024).

Given a match between i and j, the return to j is defined as:

$$r_{s_t}(i,j) = \begin{cases} 1, j \text{ wins} \\ 0, \text{otherwise} \end{cases}$$

Once the result of the match comes out, exponential smoothing is applied to the return value and this averaged result serves as an updated win rate:

$$p_{s_t}(win_{i>j}) = \bar{r}_{s_t}(i,j) = \bar{r}_{s_t-1}(i,j) * \gamma + r_{s_t}(i,j) * (1-\gamma)$$

where $\gamma \in [0,1]$ is the smoothing factor.

Then the expected utility is defined as the overall win rate, which can be calculated as:

$$\mathbb{E}(p_{s_t}) = p_{s_t}(win_{all}) = \sum_{i,j \in I} p_{s_t}(win_{i>j}) w_{s_t-1}(i,j)$$

The regret is defined as the difference between the smoothed win rate and the expected win rate:

$$\Delta p_{s_t}(i,j) = \bar{r}_{s_t}(i,j) - \mathbb{E}(p_{s_t})$$

and the regret matrix is updated as:

$$R_{s_t}(i,j) = \max(R_{s_t-1}(i,j) + \Delta p_{s_t}(i,j), 0)$$

Finally, new weights for selecting combinations in next match is updated as:

$$w_{s_t}(i,j) = \begin{cases} \frac{1}{N^2}, \text{if } \sum_{i,j \in I} R_{s_t}(i,j) = 0, \\ \frac{R_{s_t}(i,j)}{\sum_{i,j \in I} R_{s_t}(i,j)} * (1 - \eta) + \frac{1}{N^2} * \eta \text{ otherwise} \end{cases}$$

where η is a weight factor.

C.5. Policy Improvement

In Section 2.2, by modeling the fighting game as a Markov process, we can leverage reinforcement learning algorithms to optimize the agent's strategies. We use the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017) in our system. PPO trains a value function $V_\phi(s_t)$ with a policy $\pi_\theta(a_t|s_t)$, the utilization of importance sampling makes it highly feasible to apply reinforcement learning algorithms in distributed scenarios.

Policy updates. In our large-scale distributed system, the trajectories are sampled from multiple sources of policies, which can exhibit significant deviations from the current policy π_θ. While the method proposed by (Ye et al., 2020b) demonstrates superior competence in MOBA games, we have discovered that the original Proximal Policy Optimization (PPO) algorithm is more effective for fighting games. Hence, our update objectives have remained consistent with PPO:

$$\mathcal{L}^\pi(\theta) = \mathbb{E}_t[min(\rho_t(\theta)\hat{A}_t, clip(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] + \beta\mathcal{L}^E(\theta) \tag{5}$$

Where $\rho_t(\theta) = \frac{\pi_\theta(s_t|a_t)}{\pi_{old}(s_t|a_t)}$ is the important weighting and $\hat{A}_t$ is the advantage estimation that is calculated by Generalized Advantage Estimation (GAE) (Schulman et al., 2015). $\mathcal{L}^E$ is the entropy of policy which enhances the exploration.

Value updates. Same as AlphaStar, we use full information to decrease the variance of value estimation:

$$V^{\pi_\theta}(s_t) = \mathbb{E}_{a \sim \pi_\theta}[r(s_t, a) + \gamma V^{\pi_\theta}(s_{t+1})] \tag{6}$$

We use GAE to estimate the advantage function $\hat{A}$:

$$A^{\pi_\theta}(s_t, a_t) = r(s_t, a_t) + \gamma V^{\pi_\theta}(s_t + 1) - V^{\pi_\theta}(S_t),$$
$$\hat{A} = \sum_{t=0}^{T} \gamma\lambda^t A_{t+1}, \tag{7}$$

and the value loss can be expressed as follows:

$$\mathcal{L}^V(\phi) = \mathbb{E}_{a \sim \pi_\theta}[\sum_{k=t}^{T} \gamma^{k-t} r(s_k, a_k) - V(s_t)]^2. \tag{8}$$

D. Experimental Details

D.1. Experimental Setting

In Table 2, we list the parameters used in the experiments. All the parameters are the same for three heterogeneous agents. During the training process, the main agent, league exploiter, and main exploiter are initialized with different structures. These three types of league agents maintain the same policy pool. After each agent completes an iteration, its copy is added to the overall policy pool. We evaluate the policies in the policy pool pairwise, and based on the evaluation results, we use the PFSP to select the next round of strategies from the policy pool. Main exploiters will reset at every start of the iteration and league exploiters will reset at the same moment with a probability of 25%. The main agents never reset. The three league agents are trained asynchronously, and each agent will undergo 10 training iterations. The maximum duration for each iteration is limited to 12 hours.

Table 2. Eperimental Parameters

Parameter	Value	Parameter	Value
n-steps	100	Batch size	5120
γ	0.995	Actor number	1000
λ	0.95	Env number per actor	10
Learning rate	2e-4	Learner number	2
CPU core num	9000	GPU per Learner	0.5

D.2. The Characters Selecting of Training Subset

In Naruto Mobile, characters are categorized into ranks (S, A, B, C) based on their relative strength and skill mechanics by professional human experts. Taking this ranking system into consideration, higher-ranked characters are generally more powerful than lower-ranked ones. Building upon this foundation, we further selected characters suitable for our training based on their popularity within each rank, using expert priors. We will include these details in the appendix section of our paper for easy reference and further discussion.

D.3. Ablation Study

We need to do ablation studies to figure out where the HELT's competence and generalization ability come from. For the generalization ability, we can take another look at Figure 4(a) and Figure 4(b), we can observe that while the ID-based FIS structure can lead to competence improvements, quantifying the states can enhance generalization, especially when it comes to opponent modeling. By quantifying opponent modeling while ensuring competence, we can improve the overall generalization capability.
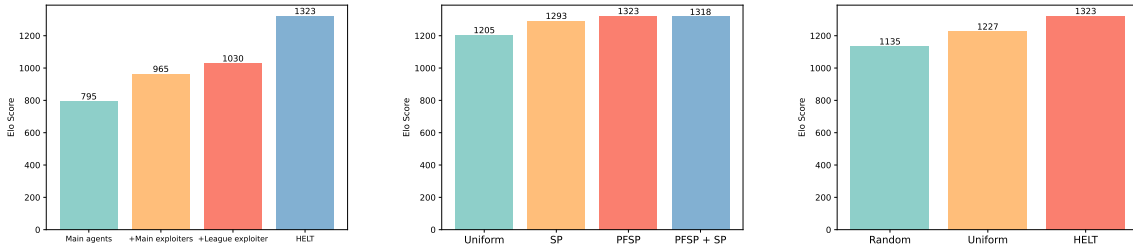
(a) Impact of different league agents. (b) Impact of different opponent policy select strategies. (c) Impact of different character select strategies

Figure 10. Ablation studying of different components for QS agent. (a) Impact of different league agents. (b) Impact of different opponent policy select strategies. (c) Impact of different character select strategies

For competence, we conducted ablation experiments on different components using the QS agent as the focus of our study. This analysis aimed to observe the competence improvements brought by each component. Figure 10(a) displays the impact of different league agents on the model's competence. We can observe that although the same-structure league exploiter contributes to around a 20% competence improvement, the primary source of competence in HELT comes from the league exploiter with the FIS structure. The inclusion of the FIS agent increases the overall diversity of strategies, significantly enhancing the competence of the QS agent. Figure 10(b) showcases the impact of different opponent policy selection strategies on the competence of the QS agent. From the Figure 10(b), we can observe that the competence differences caused by different opponent policy selection methods are not significant. The strategy used in AlphaStar, PFSP+SP, shows similar competence to pure PFSP in our environment. Therefore, we primarily adopt the PFSP opponent policy selection in our study.

Figure 10(c) displays the impact of different character selection strategies on the competence of the QS agent. We compared

the strategies of random selection, uniform selection, and HELT's methodAppendix C.2. HELT's method encourages stronger characters to be trained more frequently, which results in the highest competence improvement.

Based on the above analysis, HELT's method can improve generalization while maintaining competence, which are both crucial factors for practical implementation and online gaming.

D.4. Evaluating in Publication Platform

Shūkai primarily focuses on catering to the needs of commercial fighting games by aligning with player expectations, enhancing game content, and driving continuous game development. HELT is primarily developed to tackle the challenges of generalization and training efficiency that arise from the large character pool in commercial games. With its expansive roster of over 400 characters, Naruto Mobile is more challenging than FightingICE (Khan et al., 2022), which features only 4 characters. As a result, our paper significantly focuses on deploying reinforcement learning systems in commercial fighting games. In contrast, FightingICE (Khan et al., 2022) places more emphasis on competition, aiming to ensure the generation of stronger agents. Nevertheless, we still plan to evaluate HELT on the FightingICE platform to demonstrate its computational efficiency and generalization. However, since the framework we used is primarily tailored for game customization, we need to invest more time in integrating with FightingICE. We will release the performance of HELT on FightingICE in the future.

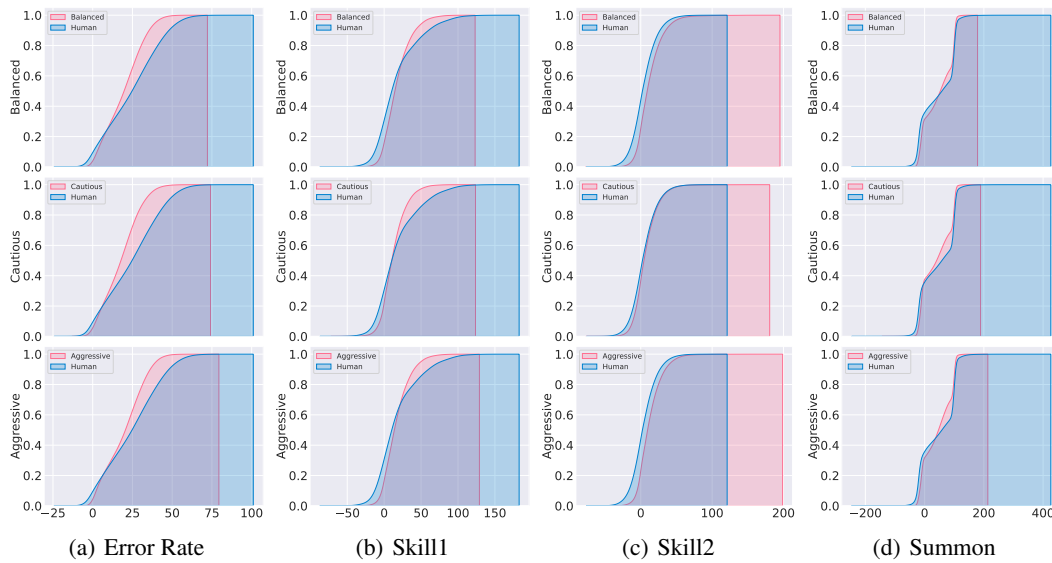## E. Agent-Human Alignment with Extra Metrics

Figure 11. A comparison of the cumulative distribution function (CDF) between human players and different agents across different metrics.

The error rate refers to the probability of making an error or failing to successfully perform a particular action or skill in a game. In the context of using substitution in Naruto Mobile, the error rate would indicate the likelihood of failing to execute the substitution properly.

Based on the previous discussion, it has been observed that our intelligent agents exhibit fewer errors compared to human players. To further analyze this, we compared the release of Skill 1 and Skill 2 between human players and the intelligent agents, considering the same character.