OpenHoldem: A Benchmark for Large-Scale Imperfect-Information Game Research

Kai Li, *Member, IEEE*, Hang Xu, Enmin Zhao, Zhe Wu, and Junliang Xing, *Senior Member, IEEE*

***Abstract*— Owing to the unremitting efforts from a few institutes, researchers have recently made significant progress in designing superhuman artificial intelligence (AI) in no-limit Texas hold'em (NLTH), the primary testbed for large-scale imperfect-information game research. However, it remains challenging for new researchers to study this problem since there are no standard benchmarks for comparing with existing methods, which hinders further developments in this research area. This work presents OpenHoldem, an integrated benchmark for large-scale imperfect-information game research using NLTH. OpenHoldem makes three main contributions to this research direction: 1) a standardized evaluation protocol for thoroughly evaluating different NLTH AIs; 2) four publicly available strong baselines for NLTH AI; and 3) an online testing platform with easy-to-use APIs for public NLTH AI evaluation. We will publicly release OpenHoldem and hope it facilitates further studies on the unsolved theoretical and computational issues in this area and cultivates crucial research problems like opponent modeling and human–computer interactive learning.**

***Index Terms*— Artificial intelligence (AI), benchmark, imperfect-information game, Nash equilibrium, no-limit Texas hold'em (NLTH).**

I. Introduction

FROM its inception, artificial intelligence (AI) research has been focusing on building agents that can play games like humans. Both Turing and Shannon developed programs for playing chess to validate initial ideas in AI. For more than half a century, games have continued to be AI testbeds for novel ideas, and the resulting achievements have marked important milestones in the history of AI [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18]. Notable examples include Deep Blue beating Kasparov in chess [1], and AlphaGo defeating Lee Sedol [3] in the complex ancient Chinese game Go. Although substantial progress has been made in solving these large-scale perfect-information games that all players know the exact state of the game at every decision point, it remains challenging to solve large-scale imperfect-information games that require reasoning under the uncertainty about the opponents' hidden information. Our work focuses on learning agents that can make good decisions in imperfect-information games. Why is solving imperfect-information games important for the neural network and learning system community? Because lots of real-world decision-making problems we care about are instances of imperfect-information games. Specifically, real-world interactions, from rock-paper-scissors to business negotiations to diplomacy, involve some amount of hidden information, making research on techniques for imperfect-information games theoretically and practically significant.

Poker has a long history as a challenging problem for developing algorithms that deal with hidden information [19], [20]. The poker game involves all players being dealt with some private cards visible only to themselves, with players taking structured turns making bets, calling opponents' bets, or folding. As one of the most popular global card games, poker has played an essential role in developing general-purpose techniques for imperfect-information games. In particular, no-limit Texas hold'em (NLTH), the world's most popular form of poker, has been the primary testbed for imperfect-information game research for decades because of its large-scale decision space and strategic complexity. For example, heads-up NLTH (HUNL), the smallest variant of NLTH, has 10^{161} decision points [21], making it almost impossible to solve directly.

There have been many efforts to design poker AIs for NLTH over the past few years [22], [23]. Most of these systems exploit some equilibrium-finding algorithms, e.g., counterfactual regret minimization (CFR) [24], with various abstraction strategies to merge similar game states to reduce the size of the game tree. Recently, a series of breakthroughs have been made in the NLTH AI research community. DeepStack [14], which combines the continual resolving and the depth-limited sparse look-ahead algorithms, defeated ten out of 11 professional poker players by a statistically significant margin. Libratus [15] defeated a team of four top HUNL-specialist professionals by using a nested safe subgame solving algorithm with an extensible blueprint strategy. Pluribus [25] defeated elite human professional players in six-player NLTH by extending the techniques behind Libratus.

Manuscript received 13 December 2021; revised 20 June 2022, 29 July 2022, 29 November 2022, and 26 December 2022; accepted 23 May 2023. Date of publication 14 June 2023; date of current version 8 October 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2022ZD0116401; in part by the Natural Science Foundation of China under Grant 62076238, Grant 62222606, and Grant 61902402; in part by the China Computer Federation (CCF)-Tencent Open Fund; and in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA27000000. *(Kai Li, Hang Xu, Enmin Zhao, and Zhe Wu contributed equally to this work.) (Corresponding author: Junliang Xing.)*

Kai Li, Hang Xu, Enmin Zhao, and Zhe Wu are with the Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: kai.li@ia.ac.cn; xuhang2020@ia.ac.cn; zhaoenmin2018@ia.ac.cn; wuzhe2019@ia.ac.cn).

Junliang Xing is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: jlxing@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2023.3280186

Although many important milestones have been achieved in NLTH AI research in recent years, the problem is far from being solved, and many theoretical and computational issues remain to be addressed. For example, the game-theoretic solution for multiplayer NLTH, the best way to game tree abstraction, more efficient equilibrium-finding algorithms that converge faster and consume fewer resources, etc. To solve these challenges, further studies are urgently needed. However, one main obstacle to further research in NLTH AI is the lack of standard benchmarks. First, there are no standard evaluation protocols in this community; different papers use different evaluation metrics, making comparisons of different methods difficult. Second, no publicly available baseline AI can serve as a starting point for future improvements. Third, there are no public easy-to-use platforms for researchers to test the performance of their AIs at any time.

Considering the important role of standard benchmarks in AI development, we present OpenHoldem, the first benchmark for NLTH AI research developed to boost the studies on large-scale imperfect-information games. OpenHoldem provides an integrated benchmark for evaluating NLTH AIs with three main components: the evaluation protocols, the baseline AIs, and a testing platform. For each component, we have made the following contributions to the community.

1) *Comprehensive Evaluation Protocols:* Due to the large variance in imperfect-information games, evaluating the algorithms' performance is challenging. We propose using two variance reduction techniques to alleviate the effects of randomness and ensure statistically significant results. We also propose using two exploitability based evaluation metrics to estimate an algorithm's worst case performance by efficiently calculating its approximate best response. Using these two kinds of evaluation metrics, we can comprehensively test the capabilities and deficiencies of an algorithm.
2) *Strong Baseline AIs:* Although significant progress has recently been made in large-scale imperfect information game research, almost all of these AIs are not publicly available. This situation makes it very challenging for new researchers to study this problem since designing and implementing a decent AI is often very complicated and tedious. To fill this gap, we propose four different types of NLTH AIs, i.e., rule-based AIs, a CFR-based static AI, a DeepStack-like online AI and a novel deep reinforcement learning (RL) based AI, which are rich and strong enough to serve as a good starting point for future research in this area. In particular, the RL-based AI is the first AI that obtains competitive performance solely through RL-based self-play and is more than a thousand times faster than state-of-the-art AIs.
3) *Online Testing Platform:* To compare different AIs more easily, we further propose an online testing platform for standardized model evaluation with all the baseline AIs built in. Researchers can link their AIs to this platform through easy-to-use APIs to play against each other for mutual improvement. This platform can serve as an AI zoo for the research community. Meanwhile, the accumulated data can also facilitate the research of data-driven imperfect-information game solving, imitation learning, and opponent modeling algorithms.

In summary, OpenHoldem makes *systematic contributions* to the imperfect-information game research community in terms of evaluation, algorithm, and platform. The adopted approach, namely to propose an evaluation protocol via several metrics, the provision of baselines tested to have strong performances, and the establishment of an online testing platform, facilitates algorithm improvements and comparisons with the state-of-the-arts, which is impossible to do today without spending much time reimplementing other people's methods. OpenHoldem addresses the challenges of designing learning systems in imperfect-information games. It can potentially significantly impact poker AI research and, more generally, the AI community dealing with decision-making problems under uncertainty. Although we use the imperfect-information poker game as the primary testbed, the techniques we developed are largely domain independent and can be applied to many real-world strategic interactions. We hope OpenHoldem makes NLTH AI research easier and more accessible, and further facilitates the research of other problems in imperfect-information games, such as opponent modeling, which is also an important research topic in the neural network and learning system community [26], [27].

II. Related Work

Standard benchmarks have played an indispensable role in promoting the research in many AI tasks like speech recognition, computer vision, and natural language processing. For example, in the task of speech-to-text, the NIST Switchboard benchmark [28] helps reduce the word error rate from 19.3% in 2000 to 5.5% in 2017; In the task of image classification, the creation of the ImageNet [29] benchmark has helped in the development of highly efficient models which reduce the image classification error rate from 26.2% down to 1.8%; In the task of machine translation, the WMT benchmark helps the machine translation system achieve human-level performance on the Chinese to English translation task [30]. These benchmarks that have greatly influenced the research communities have some common characteristics: clear evaluation metrics, rich baseline models, and convenient online testing platforms. Motivated by this, we propose the OpenHoldem benchmark that meets the above requirements to facilitate the future development of general-purpose techniques for large-scale imperfect-information games.

There are already some benchmarks on game AI. Examples include the Atari environments in OpenAI Gym [31], ViZDoom [32], and MineRL [33], but most of these benchmarks are oriented toward the research of RL algorithms. Recently, some benchmarks for game theory research have been proposed. For example, DeepMind releases the OpenSpiel [34] benchmark, which contains a collection of environments and algorithms for research in n-player zero-sum and general-sum games. Although OpenSpiel implements different kinds of games and state-of-the-art algorithms, it currently does not provide high-performance NLTH AIs. RLCard [35] developed by Texas A&M University includes many large-scale complex card games, such as Dou Dizhu,

Mahjong, and NLTH. However, most of the implemented baseline AIs are relatively weak. In contrast, the proposed OpenHoldem benchmark contains very strong baseline AIs, which can serve as a better starting point for future improvements.

Texas Hold'em, the primary testbed for imperfect information game research, has been studied in the computer poker community for years [20]. The earliest Texas Hold'em AIs are rule-based systems that consist of a collection of if-then rules written by human experts. For example, the early agents (e.g., Loki [36]) produced by the University of Alberta are mostly based on carefully designed rules. While the rule-based approach provides a simple framework for implementing Texas Hold'em AIs, the resulting handcrafted strategies are easily exploitable by observant opponents. Since 2006, the Annual Computer Poker Competition (ACPC) [37] has greatly facilitated poker AI development, and many game-theoretic Texas Hold'em AIs are proposed [22], [23]. These systems first use various abstraction strategies [38], [39] to merge similar game states to reduce the game size, then exploit some equilibrium-finding algorithms (e.g., CFR [24] and its various variants [40], [41], [42], [43]) to find the approximate Nash equilibrium strategies which are robust to different opponents.

Recently, the research on these game-theoretic approaches has made significant breakthroughs. Examples include DeepStack [14] proposed by the University of Alberta that defeats professional poker players by a large margin, Libratus [15] from the Carnegie Mellon University that decisively defeats four top HUNL-specialist professionals, and Pluribus [25] as a direct descendant of Libratus that defeats elite human professional players in six-player NLTH. Nevertheless, almost all of these Texas Hold'em AIs are not publicly available, making it challenging for new researchers to study this problem further. Our OpenHoldem is the first open benchmark with publicly available strong baseline AIs for large-scale imperfect-information game research.

III. Preliminaries

Here, we present some background knowledge needed for the rest of the article. We first provide some notations to formulate imperfect-information games and discuss CFR, the most commonly used equilibrium-finding algorithm for imperfect-information games. Next, we cover the basics of RL. Then, we discuss different categories of algorithms for solving imperfect-information games. Finally, we introduce the game rule of NLTH.

A. Imperfect-Information Games

Imperfect-information games are usually described by a tree-based formalism called **extensive-form games** [44]. In an imperfect-information extensive-form game $\mathcal{G}$ there is a finite set $\mathcal{N} = \{1, \ldots, N\}$ of **players**, and there is also a special player c called **chance**; $\mathcal{H}$ refers to a finite set of histories, each member $h \in \mathcal{H}$ denotes a possible **history** (or **state**), which consists of actions taken by players including chance; $g \sqsubseteq h$ denotes the fact that g is equal to or a **prefix** of h; $\mathcal{Z} \subseteq \mathcal{H}$ denotes the **terminal states** and any member $z \in \mathcal{Z}$ is not a prefix of any other states; $\mathcal{A}(h) = \{a{:}ha \in \mathcal{H}\}$ is the set of available **actions** in the nonterminal state $h \in \mathcal{H} \setminus \mathcal{Z}$; A **player function** $\mathcal{P}{:}\mathcal{H} \setminus \mathcal{Z} \to \mathcal{N} \cup \{c\}$ assigns a member of $\mathcal{N} \cup \{c\}$ to each nonterminal state in $\mathcal{H} \setminus \mathcal{Z}$, i.e., $\mathcal{P}(h)$ is the player who takes an action in state h.

For a state set $\{h \in \mathcal{H}{:}\mathcal{P}(h) = i\}$, $\mathcal{I}_i$ denotes an **information partition** of player i; A set $I_i \in \mathcal{I}_i$ is an **information set** of player i and $I(h)$ represents the information set which contains the state h. If g and h belong to the same information set I_i, then the player i cannot distinguish between them, so we can define $\mathcal{A}(I_i) = \mathcal{A}(h)$ and $\mathcal{P}(I_i) = \mathcal{P}(h)$ for arbitrary $h \in I_i$. We define $|\mathcal{I}| = \max_{i \in \mathcal{N}} |\mathcal{I}_i|$ and $|\mathcal{A}| = \max_{i \in \mathcal{N}} \max_{I_i \in \mathcal{I}_i} |\mathcal{A}(I_i)|$. For each player $i \in \mathcal{N}$, a **utility function** $u_i(z)$ define the payoff received by player i upon reaching a terminal state z. Δ_i is the **range of payoffs** reachable by player i, i.e., $\Delta_i = \max_{z \in \mathcal{Z}} u_i(z) - \min_{z \in \mathcal{Z}} u_i(z)$ and $\Delta = \max_{i \in \mathcal{N}} \Delta_i$.

A **strategy profile** $\sigma = \{\sigma_i | \sigma_i \in \Sigma_i, i \in \mathcal{N}\}$ is a specification of strategies for all players, where Σ_i is the set of all possible strategies for player i, and σ_{-i} refers to the strategies of all players other than player i. For each player $i \in \mathcal{N}$, its strategy σ_i assigns a distribution over $\mathcal{A}(I_i)$ to each information set I_i of player i. The strategy of the chance player σ_c is usually a fixed probability distribution. $\sigma_i(a|h)$ denotes the probability of action a taken by player $i \in \mathcal{N}$ at state h. In imperfect information games, $\forall h_1, h_2 \in I_i$, we have $\sigma_i(I_i) = \sigma_i(h_1) = \sigma_i(h_2)$. The **state reach probability** of h is denoted by $\pi^\sigma(h)$ if all players take actions according to the strategy profile σ. The state reach probability can be composed into each player's contribution, i.e., $\pi^\sigma(h) = \prod_{i \in \mathcal{N} \cup \{c\}} \pi_i^\sigma(h) = \pi_i^\sigma(h)\pi_{-i}^\sigma(h)$, where $\pi_i^\sigma(h) = \prod_{h'a \sqsubseteq h, \mathcal{P}(h')=i} \sigma_i(a|h')$ is player $i's$ contribution and $\pi_{-i}^\sigma(h) = \prod_{h'a \sqsubseteq h, \mathcal{P}(h') \neq i} \sigma_{\mathcal{P}(h')}(a|h')$ is all players' contribution except player i. The **information set reach probability** of I_i is defined as $\pi^\sigma(I_i) = \sum_{h \in I_i} \pi^\sigma(h)$. The **interval state reach probability** from state h' to h is defined as $\pi^\sigma(h', h) = \pi^\sigma(h)/\pi^\sigma(h')$ if $h' \sqsubseteq h$. $\pi_i^\sigma(I_i)$, $\pi_{-i}^\sigma(I_i)$, $\pi_i^\sigma(h', h)$, and $\pi_{-i}^\sigma(h', h)$ are defined similarly.

To facilitate the reader's understanding, we use Kuhn poker as a simple example to illustrate the terminologies and symbols more concretely. Kuhn poker is a simple imperfect-information game with a three-card deck: Jack (J), Queen (Q), and King (K). Each player antes a single chip and has one more chip to bet with, then gets a single private card at random and one is left face down, and players proceed to pass (p), bet (b), call (c), or fold (f). Fig. 1 shows a partial game tree of Kuhn poker, where two players are dealt Q/J in the left sub-tree and Q/K in the right sub-tree. Each black diamond node, i.e., $z_1, \ldots, z_{10}$ represents a terminal node and $h_0, \ldots, h_8$ denote nonterminal nodes. The trajectory from the root to each node is a history of actions. For example, h_7 contains the chance player's action "Deal Q to player 1 and J to player 2," player 1's action "p," and player 2's action "b." We have $h_3 \sqsubseteq h_7$, $\mathcal{A}(h_7) = \{f, c\}$, and $\mathcal{P}(h_7) = 1$. Since Player 2's private card is invisible to player 1, h_1 and h_2 are in the same information set and are indistinguishable from player 1. Similarly, h_7 and h_8 are in the same information set. We use I_0 to denote the information set of h_7 and h_8. Because h_7 and h_8 are undistinguished by player 1, we have $\mathcal{A}(I_0) = \mathcal{A}(h_7) = \mathcal{A}(h_8)$, $\mathcal{P}(I_0) = \mathcal{P}(h_7) = \mathcal{P}(h_8)$, and $\sigma_1(I_0) = \sigma_1(h_7) = \sigma_1(h_8)$. $\pi^\sigma(h_7)$ is the history reach probability of h_7 if all players

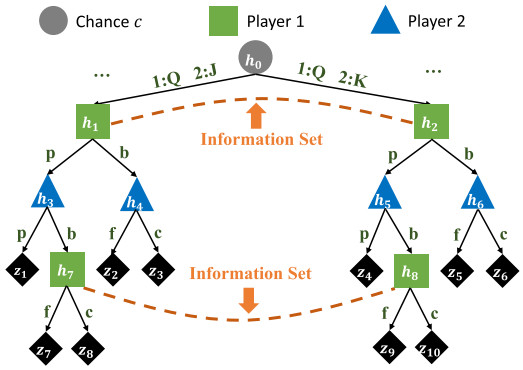

Fig. 1. Partial game tree of Kuhn poker.

play according to σ, we have $\pi^\sigma(h_7) = \pi_1^\sigma(h_7)\pi_2^\sigma(h_7)$ and $\pi^\sigma(h_1, h_7) = \pi^\sigma(h_7)/\pi^\sigma(h_1)$ since $h_1 \sqsubseteq h_7$.

B. Best Response and Nash Equilibrium

For each player $i \in \mathcal{N}$, the **expected utility** $u_i^\sigma = \sum_{z\in\mathcal{Z}} \pi^\sigma(z)u_i(z)$ is the expected payoff of player i obtained at all possible terminal states. The **best response** to the strategy profile σ_{-i} is any strategy σ_i^* of player i that achieves optimal payoff against σ_{-i}, i.e., $\sigma_i^* = \arg\max_{\sigma_i'\in\Sigma_i} u_i^{(\sigma_i', \sigma_{-i})}$. For the two-player zero-sum games, i.e., $\mathcal{N} = \{1, 2\}$ and $\forall z \in \mathcal{Z}, u_1(z) + u_2(z) = 0$, the **Nash equilibrium** is the most commonly used solution concept which is a strategy profile $\sigma^* = (\sigma_1^*, \sigma_2^*)$ such that each player's strategy is the best response to the other. An **ϵ-Nash equilibrium** is an approximate Nash equilibrium, whose strategy profile σ satisfies: $\forall i \in \mathcal{N}, u_i^\sigma + \epsilon \geq \max_{\sigma_i'\in\Sigma_i} u_i^{(\sigma_i', \sigma_{-i})}$. The **exploitability** of a strategy σ_i is defined as $\epsilon_i(\sigma_i) = u_i^{\sigma^*} - u_i^{(\sigma_i, \sigma_{-i}^*)}$. A strategy is unexploitable if $\epsilon_i(\sigma_i) = 0$.

C. Counterfactual Regret Minimization

CFR [24] is an iterative algorithm for computing approximate Nash equilibrium in imperfect-information games and is widely used in NLTH AI. CFR frequently uses **counterfactual value**, which is the expected payoff of an information set given that player i tries to reach it. Formally, for player i at an information set $I \in \mathcal{I}_i$ given a strategy profile σ, the counterfactual value of I is $v_i^\sigma(I) = \sum_{h\in I}(\pi_{-i}^\sigma(h) \sum_{z\in\mathcal{Z}}(\pi^\sigma(h, z)u_i(z))$. The counterfactual value of an action a in I is $v_i^\sigma(a|I) = \sum_{h\in I}(\pi_{-i}^\sigma(h) \sum_{z\in\mathcal{Z}}(\pi^\sigma(ha, z)u_i(z))$.

CFR typically starts with a random strategy σ^1. On each iteration T, CFR first recursively traverses the game tree using the strategy σ^T to calculate the **instantaneous regret** $r_i^T(a|I)$ of not choosing action a in an information set I for player i, i.e., $r_i^T(a|T) = v_i^{\sigma^T}(a|I) - v_i^{\sigma^T}(I)$. Then CFR accumulates the instantaneous regret to obtain the **cumulative regret** $R_i^T(a|I) = \sum_{t=1}^T r_i^t(a|I)$ and uses regret-matching [45] to calculate the new strategy for the next iteration

$$\sigma_i^{T+1}(a|I) = \begin{cases} \dfrac{R_i^{T,+}(a|I)}{\sum_{a'\in\mathcal{A}(I)} R_i^{T,+}(a'|I)}, & \sum\limits_{a'} R_i^{T,+}(a'|I) > 0 \\ \dfrac{1}{|\mathcal{A}(I)|}, & \text{otherwise} \end{cases} \tag{1}$$

where $R_i^{T,+}(a|I) = \max(R_i^T(a|I), 0)$.

In two-player zero-sum imperfect-information games, if both players play according to CFR on each iteration then their **average strategies** $\bar\sigma^T$ converge to an ϵ-Nash equilibrium in $\mathcal{O}(|\mathcal{I}|^2|\mathcal{A}|\Delta^2/\epsilon^2)$ iterations [24]. $\bar\sigma^T$ is calculated as

$$\begin{aligned} S_i^T(a|I) &= \sum_{t=1}^T \left(\pi_i^{\sigma^t}(I)\sigma_i^t(a|I)\right) \\ \bar\sigma_i^T(a|I) &= \frac{S_i^T(a|I)}{\sum_{a'\in\mathcal{A}(I)} S_i^T(a'|T)}. \end{aligned} \tag{2}$$

Therefore, CFR is a ready-to-use equilibrium finding algorithm in two-player zero-sum games.

D. Reinforcement Learning

In RL, an agent attempts to find the optimal policy to maximize its long-term reward through trial and error. This process is formulated by the Markov decision process (MDP). An MDP is a tuple $M = \langle S, A, R, T, \gamma\rangle$, consisting of a set of states S, a set of actions A, a reward function $R{:}S \times A \to \mathbb{R}$, a transition probability model $P(s_{t+1}|s_t, a_t)$, and a discount factor $\gamma \in [0, 1]$. A policy π maps a state to a probability distribution of actions, $\pi{:}S \to \Delta(A)$, where $\Delta(\cdot)$ denotes the probability simplex. At each time step t, the environment has a state s_t. The agent observes this state and chooses an action $a_t \sim \pi(a|s_t)$. The environment returns a reward r_t to the agent, and transits into the next state s_{t+1} sampled from the distribution $P(\cdot|s_t, a_t)$. The reward might be discounted by the discount factor γ at each time step to favor more recent rewards. The goal of the agent is to find the optimal policy that maximizes the discounted cumulative reward $R_t^\gamma = \sum_{k=0}^\infty \gamma^k r_{t+k}$.

E. Algorithms for Solving Imperfect-Information Games

Solving imperfect-information games is an important problem in the neural network and learning system community [46], [47], [48]. There are two main types of algorithms for solving imperfect-information games, i.e., the no-regret based algorithms and the best-response based algorithms. The no-regret based methods are online algorithms that minimize the regrets of both players so that the average strategy gradually approximates the Nash equilibrium. Due to the theoretical guarantee, no-regret based learning is a hot topic in the learning system community, and lots of papers have been published [49], [50], [51], [52]. For example, the Proximal Online Gradient [49] is a recently proposed state-of-the-art no-regret algorithm. Unlike these existing methods, which mainly focus on normal-form games or bandit problems, the CFR-based AI (see Section IV-B2) in OpenHoldem is a no-regret learning algorithm for large-scale extensive-form games, which can serve as a good starting point for large-scale equilibrium-finding and will be interest to the researchers in the online no-regret learning community.

In best-response-based methods, the agents learn to improve their performance by iteratively best-responding to their opponents. The best-response based methods can easily be integrated with deep RL methods and work effectively in

large-scale zero-sum games. For example, the online minimax Q network learning algorithm [46] uses DQN [2] as the best-response learner to find the Nash equilibrium in two-player zero-sum games. The deep RL-based AI (see Section IV-B4) in OpenHoldem is also a best-response-based method which learns from the input states to the output actions directly. The new techniques and underlying principles of this AI are helpful in developing general neural network-based learning algorithms for more imperfect-information games and will be interest to the neural network and learning system community.

F. No-Limit Texas Hold'em

NLTH has been the most widely played type of poker for over a decade. The heads-up (i.e., two-player) variant prevents opponent collusion and allows a clear winner to be determined, so HUNL becomes the primary testbed in the computer poker and game theory communities. HUNL is a repeated game in which the two players play a match of individual games, usually called **hands**. On each hand, one player will win some number of **chips** from the other player, and the goal is to win as many chips as possible throughout the match. In this article, we follow the standard form of HUNL poker agreed upon by the research community [37], where each player starts each hand with a **stack** of U.S. $20 000 chips. Resetting the stacks after each hand allows for each hand to be an independent sample of the same game and is called "Doyle's Game," named for the professional poker player Doyle Brunson who publicized this variant.

HUNL consists of four rounds of betting. On each round of betting, each player can choose to either **fold**, **call**, or **raise**. If a player folds, the game will end with no player revealing their private cards, and the opponent will take the **pot**. If a player calls, he or she places several chips in the pot by matching the number of chips entered by the opponent. If a player raises by x, he or she adds x more chips to the pot than the opponent. A raise of all remaining chips is called an **all in** bet. A betting round ends if each player has taken action and entered the same amount of chips in the pot as every other player still in hand. At the beginning of a round, when there are no opponent chips yet to match, the raise action is called **bet**, and the call action is called **check**. If either player chooses to raise first in a round, they must raise a minimum of U.S. $100 chips. If a player raises after another player has raised, that raise must be greater than or equal to the last raise. The maximum amount for a bet or raise is the remainder of that player's stack, which is U.S. $20 000 at the beginning of a hand.

In HUNL, at the beginning of each hand, the first player, i.e., P1, enters a **big blind** (usually U.S. $100) into the pot; the second player, i.e., P2, enters a **small blind** which is generally half the size of the big blind; and both players are then dealt with two **hole (private) cards** from a standard 52-card deck. There is then the first round of betting (called the **preflop**), where the second player P2 acts first. The players alternate in choosing to fold, call, or raise. After the preflop, three **community (public) cards** are dealt face up for all players to observe, and the first player, P1, now starts a similar round of betting (called the **flop**) to the first round. After the flop round ends, another community card is dealt face up, and the third round of betting (called the **turn**) commences where P1 acts first. Finally, a fifth community card is dealt face up, and a fourth betting round (called the **river**) occurs, again with P1 acting first. If none of the players folds at the end of the fourth round, the game enters a **show-down** process: the private cards are revealed, the player with the best five-card poker hand, constructed from the player's two private cards and the five community cards, wins the pot. In the case of a tie, the pot is split equally among the players. A **match** consists of a large number of poker hands, in which the players alternate their positions as the first and the second player. The rules of Six-player NLTH and HUNL are roughly the same. For the detailed rules of Six-player NLTH, refer to the supplementary materials of [25].

Since NLTH can be played for different stakes, such as a big blind being worth U.S. $0.01 or U.S. $1000, it is inappropriate to measure the performance by chips. Hence, players commonly measure their performance over a match as their average number of big blinds won per hand. The computer poker community has standardized on the unit **milli-big-blinds per hand**, or mbb/h, where one milli-big-blind is one-thousandth of one big blind. For example, a player that always folds will lose 750 mbb/h (by losing 1000 mbb as the big blind and 500 as the small blind).

IV. OpenHoldem

Our proposed OpenHoldem framework consists of three parts, i.e., the evaluation protocols, the baseline AIs, and the online testing platform. The high-level design paradigm of OpenHoldem is shown in Fig. 2, which clearly specifies the design flow of each of its components. Specifically, the design principle of the evaluation protocol is that it should be able to fully test the algorithm's performance from different perspectives. To this end, we design two kinds of evaluation metrics, i.e., the head-to-head-based evaluation metric and the exploitability based metric. The head-to-head-based evaluation metric can accurately measure the algorithm's expected performance through variance reduction techniques. The exploitability based metric can estimate the algorithm's worst case performance by efficiently calculating its approximate best response. Next, the design principle of the baseline AIs is that they should contain a variety of mainstream algorithms. We therefore design four different types of AIs, i.e., rule-based AIs, a CFR-based static AI, a DeepStack-like online AI, and a novel RL-based AI. These baseline algorithms are rich and strong enough to serve as a good starting point for future research. Finally, the design principle of the testing platform is that it should be convenient and easy to use. For this purpose, we design an online testing platform to which researchers can connect through easy-to-use APIs to play against each other for mutual improvement. Next, we will expatiate these three parts, respectively.

A. Evaluation Protocols

Evaluating the performance of different NLTH agents is challenging due to the inherent variance in the game. A better

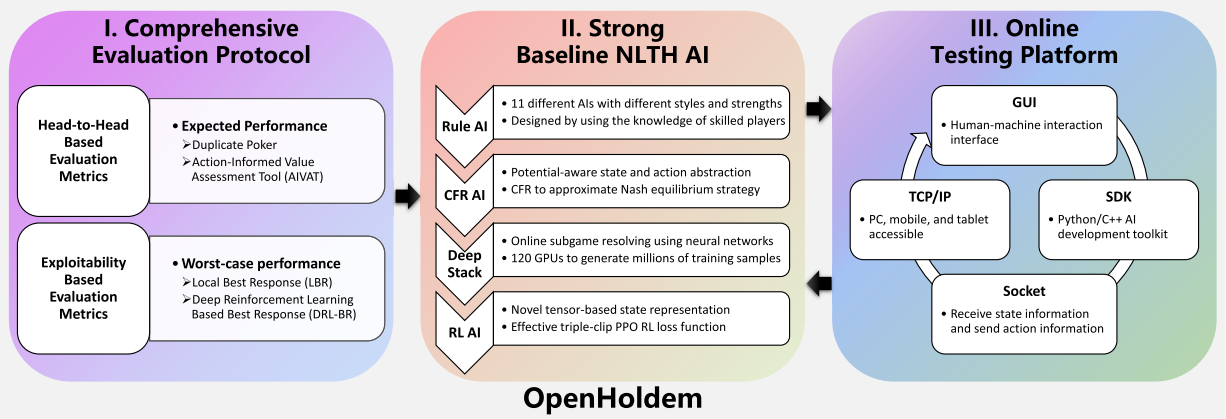

Fig. 2. OpenHoldem provides an integrated toolkit for large-scale imperfect-information game research using NLTH with three main components: the evaluation protocols, the baseline NLTH AIs, and an online testing platform.

agent may lose in a short period simply because it was dealt with weaker cards. Moreover, different papers use different evaluation metrics, making comparisons of different methods difficult. In OpenHoldem, we propose using the following evaluation metrics to thoroughly test different algorithms from different aspects.

1) Head-to-Head Based Evaluation Metrics: One of the main goals of agent evaluation is to estimate the expected utility u_i^σ given a strategy profile σ. If the game is small, one can compute this expectation exactly by enumerating all terminal states, i.e., $u_i^\sigma = \sum_{z\in\mathcal{Z}} \pi^\sigma(z)u_i(z)$. In the large-scale NLTH, however, this approach is unpractical. The most commonly used approach to approximately estimate u_i^σ is sampling. Specifically, the agents repeatedly play against each other, drawing independent samples $z_1, \ldots, z_T$ with the probability $\pi^\sigma(z)$. The estimator $\hat{u}_i^\sigma$ is the average utility

$$\hat{u}_i^\sigma = \frac{1}{T}\sum_{t=1}^{T} u_i(z_t). \tag{3}$$

This estimator is unbiased, i.e., $E[\hat{u}_i^\sigma] = u_i^\sigma$, so the mean-squared-error (mse) of $\hat{u}_i^\sigma$ is its variance

$$\text{mse}\left(\hat{u}_i^\sigma\right) = \text{Var}\left[\hat{u}_i^\sigma\right] = \frac{1}{T}\text{Var}[u_i(z)]. \tag{4}$$

This sampling-based approach is effective when the domain has little stochasticity, i.e., $\text{Var}[u_i(z)]$ is small, but this is not the case in NLTH. We propose to use the following two variance reduction techniques to alleviate the effects of randomness and ensure statistically significant results.

Duplicate Poker is a simple and easy-to-implement variance reduction technique that attempts to mitigate the effects of luck and is widely used in the ACPCs [37]. For example, in HUNL, let us say agent $\mathcal{A}$ plays one seat and agent $\mathcal{B}$ plays the other seat. First, we let $\mathcal{A}$ and $\mathcal{B}$ play M hands of poker, then we switch their seats and play another M hands of poker with the same set of cards for each seat. By doing so, if agent $\mathcal{A}$ is dealt two aces in the first hand, then agent $\mathcal{B}$ will be dealt two aces in the $M+1$th hand, so the effects of luck are significantly alleviated. The process of duplicate poker for multiplayer NLTH is similar.

AIVAT is a more principled variance reduction technique for evaluating agents' performance in imperfect-information games [53]. The core idea of AIVAT is to derive a real-valued function $\tilde{u}_i$ that is used in place of the true utility function u_i. On one hand, the expectation of $\tilde{u}_i(z)$ matches that of $u_i(z)$ for any choice of strategy profile σ, so $\tilde{u}_i^\sigma = (1/T)\sum_{t=1}^{T}\tilde{u}_i(z_t)$ is also an unbiased estimator of the expected utility u_i^σ. On the other hand, the variance of $\tilde{u}_i(z)$ is designed to be smaller than that of $u_i(z)$, so $\text{mse}(\tilde{u}_i^\sigma) < \text{mse}(\hat{u}_i^\sigma)$, i.e., $\tilde{u}_i^\sigma$ is a better estimator than $\hat{u}_i^\sigma$. More specifically, AIVAT adds a carefully designed control variate term for both chance actions and actions of players with known strategies, resulting in a provably unbiased low-variance evaluation tool for imperfect-information games. It is worth noting that duplicate poker and AIVAT can be combined to further reduce the variance.

2) Exploitability Based Evaluation Metrics: Most works on computer poker are to approximate a Nash equilibrium, i.e., produce a low-exploitability strategy. However, head-to-head evaluation is a poor equilibrium approximation quality estimator in imperfect-information games [14]. For example, in the game of Rock-Paper-Scissors, consider the exact Nash equilibrium strategy (i.e., playing each option with equal probability) playing against a dummy strategy that always plays "rock." The head-to-head-based evaluation results are a tie in this example, but the two strategies are vastly different in terms of exploitability. Therefore, the exploitability is also a crucial evaluation metric in imperfect-information games. The exploitability of one strategy can be measured by calculating its best-response strategy, but the large size of NLTH's game tree makes an explicit best-response computation intractable. We propose to use the following two techniques to calculate the exploitability approximately.

Local best response (LBR) is a simple and computationally inexpensive method to find a *lower-bound* on a strategy's exploitability [54]. The most important concept in LBR is the agent's range, i.e., the probability distribution on each of the possible private cards the agent holds. Suppose we want to find the LBR of the agent $\mathcal{A}$ with known strategy σ_a. At the beginning of each hand, it is equally likely that $\mathcal{A}$ holds any pair of private cards. The probabilities of actions performed by $\mathcal{A}$ depend on the private cards it holds. Knowing the strategy

of $\mathcal{A}$, we can use Bayes' theorem to infer the probabilities that $\mathcal{A}$ holds each of the private cards. Based on the range of $\mathcal{A}$, LBR greedily approximates the best response actions, i.e., the actions which maximize the expected utility under the assumption that the game will be checked/called until the end. Thus, LBR best-responds locally to the opponent's actions by looking only at one action ahead, providing a lower bound on the opponent's exploitability. LBR also relies on playing standard poker hands, so the variance reduction techniques (e.g., AIVAT) can be exploited to reduce the number of hands required to produce statistically significant results.

a) Deep RL-based best response (DRL-BR): Because the game tree of NLTH is too large, the LBR algorithm does not explicitly compute a best-response strategy but uses its local approximation to play against the evaluated agent $\mathcal{A}$ directly. In DRL-BR, we try to explicitly approximate the best response strategy by training a DRL agent $\mathcal{B}$ against $\mathcal{A}$. More specifically, by treating $\mathcal{A}$ as part of the environment, then from the perspective of $\mathcal{B}$, the environment can be modeled as an MDP. $\mathcal{B}$ can leverage some suitable DRL algorithms (e.g., DQN [2], PPO [55], etc.) to learn to maximize its payoff from its experience of interacting with the environment, i.e., playing against $\mathcal{A}$. This approach turns the problem of finding the best response strategy into a single-agent RL problem. An approximate solution of the MDP by RL yields an approximate best response to the evaluated agent $\mathcal{A}$. After obtaining the approximate best response $\mathcal{B}$, the head-to-head evaluation result (e.g., AIVAT) can be used to approximate the exploitability of $\mathcal{A}$ by having them repeatedly play against each other.

B. Baseline AIs

Despite significant progress in designing NLTH AIs in recent years, almost all of these AIs are not publicly available. This situation makes it very challenging for new researchers to further study this problem since designing and implementing a decent NLTH AI is often very complicated and tedious. To fill this gap, in OpenHoldem, we design and implement four different types of NLTH AIs, which are strong enough to serve as a good starting point for future research in this area.

1) Rule-Based AI: The rule-based method is probably the most straightforward way to implement NLTH AI. A rule-based NLTH AI consists of a collection of rules designed by domain experts. In OpenHoldem, we develop $\mathcal{A}^{\mathcal{R}}$, a strong rule-based NLTH AI designed by some skilled poker players in our research group. Our rule-based AI $\mathcal{A}^{\mathcal{R}}$ handles about 10^6 different scenarios that are likely to occur in the real play of NLTH and contains tens of thousands of lines of code. As a suggestion, when researchers implement their own NLTH AIs, it is helpful to compare them to $\mathcal{A}^{\mathcal{R}}$ as a sanity check.

Besides the strong rule-based AI $\mathcal{A}^{\mathcal{R}}$, we also designed some other rule-based AIs with different styles and strengths (see Table I). These agents can be used as learning materials for beginners, and more importantly, they can also be used for opponent modeling research. These AIs calculate the expected winning probability at each stage and then make decisions based on these probabilities and different predefined rules.

TABLE I
OPENHOLDEM PROVIDES MANY RULE-BASED AIS WITH DIFFERENT STYLES AND STRENGTHS

NLTH AI Name	Exploitability	Description
CallAgent	Very High	Always Call/Check.
ManiacAgent	Very High	Always raise by half or one pot randomly.
RandomAgent	High	Randomly select legal actions.
TimidAgent	High	Calls when holding the nut; else folds to any bet.
CandidAgent	High	Bets 1/4 to one pot depending on hand strength, checks/calls with marginal hands, folds weak hands.
FickleAgent	High	Randomly change the strategy every N hands.
LooseAggressiveAgent	High	Bets/raises aggressively with a wide range of hands.
LoosePassiveAgent	High	Calls with most hands, folds weak hands, rarely raises.
TightPassiveAgent	High	Calls with good hands, folds most hands, rarely raises.
TightAggressiveAgent	Moderate	Like CandidAgent, with refined ranges and bluffing.
$\mathcal{A}^{\mathcal{R}}$	Low	A relatively strong rule AI designed by using the knowledge of some skilled Texas Hold'em players.

2) CFR-Based Static AI: While the rule-based approach provides a simple framework for implementing NLTH AIs, the resulting strategies are exploitable. Therefore, most recent studies in NLTH AIs focus on approximating the theoretically unexploitable Nash equilibrium strategies. Among them, the most successful approach is the CFR algorithm [24] and its various variants [41], [42], [56]. CFR-type algorithms iteratively minimize the regrets of both players so that the time-averaged strategy gradually approximates the Nash equilibrium. In OpenHoldem, we design and implement $\mathcal{A}^{\mathcal{C}}$, a strong CFR-based NLTH AI, which aims to serve as a starting point for the large-scale equilibrium-finding research. Overall, $\mathcal{A}^{\mathcal{C}}$ first uses the abstraction algorithm to create a smaller abstract game, then approximates the Nash equilibrium strategy in this abstract game, and finally executes the resulting strategy in the original game.

The abstraction algorithm aims to take a large-scale imperfect information game as input and output a smaller but strategically similar game solvable by current equilibrium-finding algorithms. It usually consists of two parts, information abstraction and action abstraction. In $\mathcal{A}^{\mathcal{C}}$, we use the potential-aware information abstraction algorithm [39], which uses the k-means algorithm with the Earth mover's distance metric to cluster cards with similar potential. Action abstraction further reduces the size of the game tree by restricting the available actions, which is especially important in games with large action spaces, such as NLTH. In $\mathcal{A}^{\mathcal{C}}$, we restrict the actions to "fold," "call," "check," "bet 0.5 pot," "bet pot," and "allin."

After obtaining the manageable abstract game $\mathcal{G}$, we use the CFR+ [41] algorithm to approximate the Nash equilibrium in $\mathcal{G}$. As shown in Algorithm 1, given the current strategy profile σ^t, we first calculate the cumulative regret of each action after t iterations in Line 8. Then, the new strategy in the $t+1$th iteration is updated in Line 9 by the regret-matching algorithm. Finally, by normalizing the cumulative strategy S^T in Line 13, the average strategy $\bar{\sigma}^T$ will approach a Nash equilibrium when T is large enough. During the actual play phase, $\mathcal{A}^{\mathcal{C}}$ first finds the abstract state that corresponds to the current real state of the game. Then, the abstract game's approximate Nash equilibrium $\bar{\sigma}^T$ is queried for the probability distribution over different actions. Finally, an action is sampled from this distribution and played in the actual game, if applicable.

3) DeepStack-Like Online AI: In essence, the $\mathcal{A}^{\mathcal{C}}$ agent is a static table calculated offline that contains the probability

Algorithm 1 The CFR+ Algorithm Which Is Used to Train $\mathcal{A}^{\mathcal{C}}$

Input: The abstract game $\mathcal{G}$, the randomly initialized strategy profile σ^1, the zero initialized cumulative regret R^0 and cumulative strategy S^0.
Parameter: The number of iterations T.
Output: The approximate Nash equilibrium $\bar{\sigma}^T = \{\bar{\sigma}_1^T, \bar{\sigma}_2^T\}$.

for $t = 1 \rightarrow T$ **do**
  for $i = 1 \rightarrow 2$ **do**
    $v_i^{\sigma^t}(h) = \sum_{h \sqsubseteq z, z \in \mathcal{Z}} \pi_{-i}^{\sigma^t}(h) \pi^{\sigma^t}(h, z) u_i(z)$
    $v_i^{\sigma^t}(a|h) = v_i^{\sigma^t}(ha)$
    $v_i^{\sigma^t}(I_i) = \sum_{h \in I_i} v_i^{\sigma^t}(h)$
    $v_i^{\sigma^t}(a|I_i) = \sum_{h \in I_i} v_i^{\sigma^t}(ha)$
    $r_i^{\sigma^t}(a|I_i) = v_i^{\sigma^t}(a|I_i) - v_i^{\sigma^t}(I_i)$
    $R_i^t(a|I_i) = \max(0, R_i^{t-1}(a|I_i) + r_i^{\sigma^t}(a|I_i))$
    $\sigma_i^{t+1}(a|I_i) = (R_i^t(a|I_i))/(\sum_{a \in \mathcal{A}(I_i)} R_i^t(a|I_i))$
    $S_i^t(a|I_i) = S_i^{t-1}(a|I_i) + \pi_i^{\sigma^t}(I_i)\sigma_i^t(a|I_i)$
  end for
end for
$\bar{\sigma}_i{}^T(a|I_i) = (S_i^T(a|I_i))/(\sum_{a \in \mathcal{A}(I_i)} S_i^T(a|I_i))$

distributions over possible actions in all situations. During actual play, if the opponent chooses an action that is not in the action abstraction of $\mathcal{A}^{\mathcal{C}}$, i.e., an off-tree action, $\mathcal{A}^{\mathcal{C}}$ round this off-tree action to a nearby in-abstraction action. A more principled approach to calculating the off-tree action's response is solving a subgame that immediately follows that off-tree action. DeepStack [14] is a representative online algorithm based on this idea. In particular, DeepStack allows computation to focus on specific situations raised when making decisions using a sound local strategy computation algorithm called continual resolving. To make continual resolving computationally tractable, DeepStack replaces sub-trees beyond a certain depth with a learned value function based on deep neural network.

The authors of DeepStack [14] do not release the training code or model for NLTH. They only release a pedagogical code for Leduc Hold'em[1] which cannot be transferred directly to NLTH because the game tree of NLTH is much larger than that of Leduc Hold'em, and the pedagogical code does not contain the necessary acceleration techniques for NLTH. Based on this situation, we reimplement DeepStack for NLTH following the original article's key ideas and obtain an online AI called $\mathcal{A}^{\mathcal{D}}$, which aims to serve as a starting point for the research of subgame solving in large-scale imperfect-information games. Specifically, we spent several weeks using 120 high-end GPUs to generate tens of millions of training samples for the river, turn, and flop value networks, which is larger than the amount of data used in the original article. Each training sample is generated by running 1000 CFR+ iterations based on a random reach probability. Since generating these training data requires enormous computing resources, we will provide download links for these training data later. Everyone can freely use these data for research. It is worth noting that

[1]https://github.com/lifrordi/DeepStack-Leduc

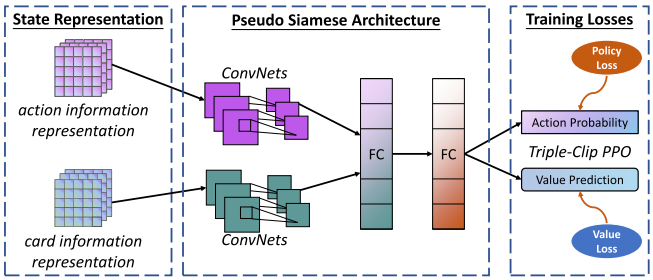

Fig. 3. End-to-end learning architecture of our deep RL based AI $\mathcal{A}^{\mathcal{RL}}$. FC represents fully connected layer. The different colors of the two ConvNets indicate that their parameters are not shared.

Noam Brown, the creator of Libratus, recently coauthored a paper [57], in which they also reimplemented DeepStack. $\mathcal{A}^{\mathcal{D}}$ has achieved similar results to theirs, which validates the correctness of our reimplementation.

4) Deep RL-Based AI: The three agents, i.e., the rule-based AI $\mathcal{A}^{\mathcal{R}}$, the CFR based static AI $\mathcal{A}^{\mathcal{C}}$, and the DeepStack-like online AI $\mathcal{A}^{\mathcal{D}}$, described in Section IV-B1, Section IV-B2, and Section IV-B3 are all based on improvements of existing techniques. These AIs often rely on different kinds of domain knowledge, such as expert rules in $\mathcal{A}^{\mathcal{R}}$ and handcrafted abstraction algorithms in $\mathcal{A}^{\mathcal{C}}$. Besides, there are also computational issues, i.e., in the inference stage of $\mathcal{A}^{\mathcal{D}}$, the CFR iteration process consumes much computation. Specifically, this iteration process often needs to be carried out 1000 times in practice to ensure high-quality prediction.

Based on the above considerations, in OpenHoldem, we further propose a high-performance and lightweight NLTH AI, i.e., $\mathcal{A}^{\mathcal{RL}}$, obtained with an end-to-end deep RL framework. $\mathcal{A}^{\mathcal{RL}}$ adopts a deep neural network to directly learn from the input state information to the output actions. The main technical contributions of $\mathcal{A}^{\mathcal{RL}}$ include a novel state representation of card and betting information, and a novel RL loss function. $\mathcal{A}^{\mathcal{RL}}$ is the first AI that obtains competitive performance in NLTH solely through RL.

a) Overall architecture: $\mathcal{A}^{\mathcal{RL}}$ aims to remove the expensive computation of CFR iteration in both the training and testing stages of an NLTH AI while eliminating the need for domain knowledge. It thus pursues an end-to-end learning framework to perform efficient and effective decision-making in imperfect-information games. Here, *end-to-end* means that the framework directly accepts the game board information and outputs the actions without encoding handcrafted features as inputs or performing iterative reasoning in the decision process. $\mathcal{A}^{\mathcal{RL}}$ adopts the RL framework to achieve this goal, and the only force to drive the model to learn is the reward.

In NLTH, the game board information includes the current and historical card information and the player action information. The agent chooses from a set of betting actions to play the game and try to win more rewards. To capture the complex relationship among the game board information, the desired betting actions, and the game rewards, we design a pseudo-Siamese architecture equipped with the RL schema to learn the underlying relationships from end to end. We illustrate the end-to-end learning architecture of $\mathcal{A}^{\mathcal{RL}}$ in Fig. 3.

As shown in Fig. 3, the input of the architecture is the game state representations of action and card information, which are, respectively, sent to the top and bottom streams of the Siamese architecture. Since the action and card representations provide different kinds of information to the learning architecture, we first isolate the parameter-sharing of the Siamese architecture to enable the two ConvNets to learn adaptive feature representations, which are then fused through fully connected layers to produce the desired actions. This design is the reason why we call it pseudo-Siamese architecture. To train $\mathcal{A}^{\mathcal{RL}}$, we present a novel Triple-Clip loss function to update the model parameters using RL algorithms. We believe these new techniques and underlying principles are helpful in developing general learning algorithms for more imperfect-information games.

b) Effective game state representation: The existence of private information and flexibility of bet size cause the NLTH AI learning extremely challenging. To obtain an effective and suitable feature representation for end-to-end learning from the game state directly to the desired action, we design a new multidimensional feature representation to encode both the current and historical card and bet information.

In NLTH, the card and action information exhibit different characteristics. We thus represent them as two separated three-dimension tensors and let the network learn to fuse them (see Fig. 3). We design the card tensor in six channels to represent the agent's two private cards, three flop cards, one turn card, one river card, all public cards, and all private and public cards. Each channel is a 4×13 sparse binary matrix, with 1 in each position denoting the corresponding card. Since there are usually at most six sequential actions in each of the four rounds, we design it in 24 channels for the action tensor. Each channel is a $4 \times n_b$ sparse binary matrix, where n_b is the number of betting options, and the four dimensions correspond to the small blind's action, the big blind's action, the sum of two player's action, and the legal actions. In the experiments, we adopt nine betting options ($n_b = 9$), i.e., "fold," "check," "call," "bet 0.5 pot," "bet 0.75 pot," "bet pot," "bet 1.5 pot," "bet 2 pot," and "allin." To understand our proposed state representation, Fig. 4(a) illustrates one example that the small blind plays an action "call" after getting a hand "AsKs," followed by a "call" from the big blind, and then three public cards "JcQcKc" are dealt.

Previous CFR-based methods, such as Slumbot and Libratus, often use some abstraction algorithms to represent game state information. They typically exploit some clustering algorithms to abstract the cards and betting information into different groups, which may lose much subtle but essential information. In contrast, our proposed state representation uses multidimensional tensors to encode and reserve all the card and betting information.

Recently, many imperfect-information game algorithms based on deep neural networks have been proposed, such as DeepCFR, single DeepCFR, etc. These methods use vectors to represent the card and betting information. Specifically, they first use an embedding vector to represent each card, then sum up the embeddings of all cards in each round as the round embedding, and finally concatenate the embeddings of all rounds as the card representation. The betting information is represented by a bet occurred vector and a bet size vector. Each position in the bet occurred vector is a binary value specifying whether a bet has occurred, and each position in the bet size vector is a float value specifying the bet size. In Fig. 4(b), we use a schematic way to compare our tensor-based state representation with the vector-based representation more clearly. Compared with the plain vector-based representation, our method uses structured tensors to represent state information, which contains richer spatial and temporal information and is thus more efficient.

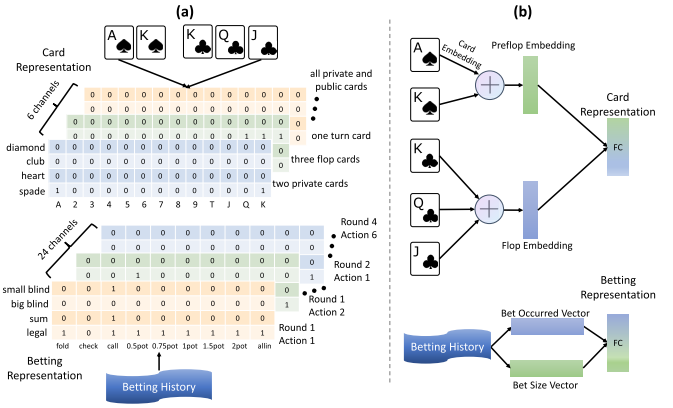

Fig. 4. Comparison of our proposed tensor-based and the existing vector-based state representation methods. In this example, the small blind plays an action "call" after getting a hand "AsKs," followed by a "call" from the big blind, and then three public cards "JcQcKc" are dealt. (a) Tensor-based state representation. (b) Vector-based state representation.

In summary, our state representation has several advantages: 1) there is no abstraction of the card information thus reserves all the game information; 2) the action representation is general and can denote a different number of betting options (though $n_b = 9$ produce satisfactory results in the experiment); 3) all the historical information is encoded to aid reasoning with hidden information; and 4) the multidimensional tensor representation is very suitable for modern deep neural architectures like ResNet [58] to learn effective feature hierarchies, as verified in the AlphaGo AI training.

c) Effective learning with triple-clip PPO: With the multidimensional feature representation, a natural choice is to use state-of-the-art RL algorithms such as PPO [55] to train the deep architecture. PPO is an actor–critic framework which trains a value function $V_\phi(s_t)$ and a policy $\pi_\theta(a_t|s_t)$. PPO defines a ratio function $r_t(\theta) = (\pi_\theta(a_t|s_t))/(\pi_{\theta'}(a_t|s_t))$ as the ratio between the current policy π_θ and the old policy $\pi_{\theta'}$, and a policy loss function $\mathcal{L}^p$ as

$$\mathcal{L}^p(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \quad (5)$$

where $\hat{A}_t$ is the advantage function, $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ ensures r_t lie in the interval $(1-\epsilon, 1+\epsilon)$, and ϵ is a clip ratio hyper-parameter with typical value 0.2. PPO's value loss $\mathcal{L}^v$ is defined as

$$\mathcal{L}^v(\phi) = \mathbb{E}_t\left[\left(R_t^\gamma - V_\phi(s_t)\right)^2\right] \quad (6)$$

in which R_t^γ represents the traditional γ-return [59].

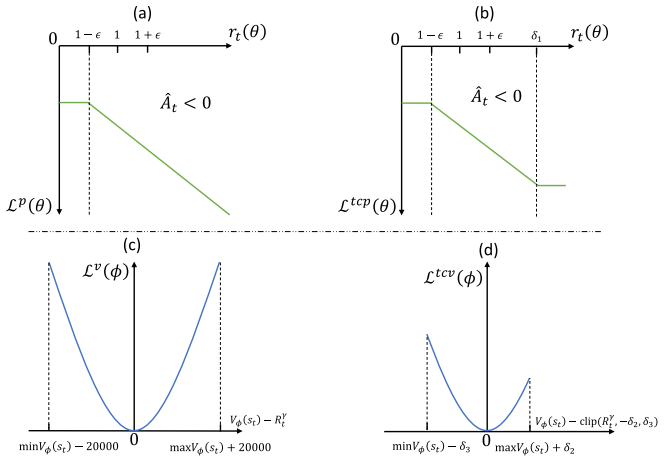

Fig. 5. The graphs of the original PPO loss and our proposed Triple-Clip PPO loss. (a) is PPO's policy loss $\mathcal{L}^p$, (b) is Triple-Clip PPO's policy loss $\mathcal{L}^{tcp}$, (c) is PPO's value loss $\mathcal{L}^v$, and (d) is Triple-Clip PPO's value loss $\mathcal{L}^{tcv}$.

However, the above PPO loss function is difficult to converge for NLTH AI training. We find two main reasons for this problem: 1) when $\pi_\theta(a_t|s_t) \gg \pi_{\theta'}(a_t|s_t)$ and the advantage function $\hat{A}_t < 0$, the policy loss $\mathcal{L}^p(\theta)$ will introduce a large variance and 2) due to the strong randomness of NLTH, the value loss $\mathcal{L}^v(\phi)$ is often too large. To speed up and stabilize the training process, we design a Triple-Clip PPO loss function. It introduces one more clipping hyper-parameter δ_1 for the policy loss when $\hat{A}_t < 0$, and two more clipping hyper-parameters δ_2 and δ_3 for the value loss. The policy loss function $\mathcal{L}^{\text{tcp}}$ for Triple-Clip PPO is defined as

$$\mathcal{L}^{\text{tcp}}(\theta) = \mathbb{E}_t\big[\text{clip}(r_t(\theta), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon), \delta_1)\hat{A}_t\big] \quad (7)$$

where $\delta_1 > 1 + \epsilon$, and ϵ is the original clip in PPO. The clipped value loss function $\mathcal{L}^{\text{tcv}}$ for Triple-Clip PPO is defined as

$$\mathcal{L}^{\text{tcv}}(\phi) = \mathbb{E}_t\Big[\big(\text{clip}\big(R_t^\gamma, -\delta_2, \delta_3\big) - V_\phi(s_t)\big)^2\Big] \quad (8)$$

where δ_2 and δ_3 do not require manual tuning but represent the total number of chips the player and the opponent has placed, respectively. The rationale for $\mathcal{L}^{\text{tcv}}(\phi)$ is that the value should be greater than or equal to the loss (i.e., $-\delta_2$) when the player folds and less than or equal to the gain (i.e., δ_3) when the opponent folds.

We explain the superiority of our proposed Triple-Clip PPO loss in more depth by visualizing and analyzing the graphs of the original PPO loss and the Triple-Clip PPO loss. Fig. 5(a) and (b) shows the graphs of $\mathcal{L}^p(\theta)$ and $\mathcal{L}^{\text{tcp}}(\theta)$, respectively. It is clear that the original PPO's policy loss $\mathcal{L}^p(\theta)$ introduces an unbounded variance when $\hat{A}_t < 0$. By optimizing $\mathcal{L}^p(\theta)$, the new policy will deviate significantly from the old one, making the training quite fragile and unstable. By introducing a clipping parameter δ_1, the Triple-Clip PPO's policy loss $\mathcal{L}^{\text{tcp}}(\theta)$ is lower bounded by a constant value, which significantly reduces the variance and improves the training stability.

Fig. 5(c) and (d) shows the graphs of $\mathcal{L}^v(\phi)$ and $\mathcal{L}^{\text{tcv}}(\phi)$, respectively. Since the γ-return can take a wide range of values from $-20\,000$ (lose all the chips) to $20\,000$ (win all the chips) in NLTH, the original PPO's value loss $\mathcal{L}^v(\phi)$ will also introduce a large variance, which makes the model training unstable. In contrast, the value loss of our Triple-Clip PPO effectively limits the γ-return to a reasonable range by two clipping parameters δ_2 and δ_3, which dramatically improves the training stability.

Some previous works also report that clipping on PPO's policy loss achieves better results in MOBA games and MuJoCo environments. Our proposed Triple-Clip PPO loss further verifies this point in large-scale imperfect-information games. Moreover, this work finds that clipping on the value function further improves the training efficiency and stability significantly, especially for imperfect-information games like NLTH, which contains rewarding signals with high variance. The Triple-Clip PPO loss function improves the learning effectiveness of the actor-critic framework, and we believe it applies to a wide range of RL applications with imperfect information.

d) Training details: We train $\mathcal{A}^{\mathcal{RL}}$ on one computing server with eight NVIDIA TITAN V GPUs and one AMD 2.00 GHz CPU with 64 cores. The mini-batch size per GPU is set to 2048; thus, the total batch size is 16 384. We use the Adam [60] optimizer with an initial learning rate of 0.0003. For the Triple-Clip PPO loss, the hyper-parameter δ_1 is set to 3, δ_2 and δ_3 are dynamically calculated according to the chips played by the players. The discount factor is set to 0.999. For policy updates, we use GAE [61] with $\lambda = 0.95$ as the advantage estimator. The best-performing model is trained for a total of 50 000 iterations. During one iteration, there are eight MPI threads, each of which contains 128 environments and 128 steps. Therefore, $\mathcal{A}^{\mathcal{RL}}$ uses a total of 6.5 billion training samples (about 2.7 billion hands). The complete pseudo-code of $\mathcal{A}^{\mathcal{RL}}$ is outlined in Algorithm 2.

C. Online Testing Platform

In order to make the comparisons between different NLTH AIs easier, we develop an online testing platform with the above four strong baseline AIs, i.e., $\mathcal{A}^{\mathcal{R}}$, $\mathcal{A}^{\mathcal{C}}$, $\mathcal{A}^{\mathcal{D}}$, and $\mathcal{A}^{\mathcal{RL}}$ built-in. Researchers can compare the performances between their own AIs and the built-in baselines through easy-to-use APIs. Fig. 6 shows an example Python code of connecting to the platform for testing NLTH AIs. The NLTH AI designers only need to implement one function, i.e., `act`, without caring about the internal structure of the platform. The input of `act` is the current game state obtained from the platform through TCP sockets. The output of `act` is the action to take in the current game state according to the designer's algorithm. The output action is also sent to the platform through TCP sockets. Fig. 7 shows the system architecture of our testing platform. The server is responsible for playing the poker hands according to the rules of NLTH. It also dynamically schedules requests and allocates resources when necessary. Our platform not only supports testing between different AIs, but also between humans and AIs.

We are more than happy to accept high-performance AIs submitted by everyone to continuously enrich the baseline AIs of OpenHoldem, with the ultimate goal of providing an ***NLTH AI Zoo*** for the research community. Currently, dozens of

Algorithm 2 The Pseudo-Code of $\mathcal{A}^{\mathcal{RL}}$

Input: The randomly initialized policy parameters θ_0 and value function parameters ϕ_0.
Parameter: The number of training iterations K.
Output: The trained policy π_θ.

1: **for** $k = 0 \rightarrow K-1$ **do**
2: Collect a set of trajectories $\mathcal{B}_k = \{\tau_i\}$ by running the policy π_{θ_k} via self-play. Each trajectory τ_i in $\mathcal{B}_k$ corresponds to one hand of poker.
3: For each time step t in each trajectory τ, compute the γ-return $R_t^\gamma = \sum_{i=0}^{|\tau|-t-1} \gamma^i r_{t+i}$.
4: For each time step t in each trajectory τ, compute the advantage $\hat{A}_t$ based on the GAE estimator and the value function V_{ϕ_k}, i.e., $\hat{A}_t = \sum_{i=0}^{|\tau|-t-1} (\gamma\lambda)^i \delta_{t+i}$, where $\delta_t = r_t + \gamma V_{\phi_k}(s_{t+1}) - V_{\phi_k}(s_t)$.
5: Update the policy parameters by maximizing the clipped policy objective via stochastic gradient ascent:

$$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{B}_k|} \sum_{\tau \in \mathcal{B}_k} \frac{1}{|\tau|} \sum_{t=0}^{|\tau|-1} L_t^{\text{tcp}}$$

where $L_t^{\text{tcp}} = \text{clip}(r_t(\theta), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon), \delta_1)\hat{A}_t$ if $\hat{A}_t < 0$, otherwise, $L_t^{\text{tcp}} = \min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)$.
6: Update the value function parameters by minimizing the clipped value loss via stochastic gradient descent:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{B}_k|} \sum_{\tau \in \mathcal{B}_k} \frac{1}{|\tau|} \sum_{t=0}^{|\tau|-1} L_t^{\text{tcv}}$$

where $L_t^{\text{tcv}} = (\text{clip}(R_t^\gamma, -\delta_2, \delta_3) - V_\phi(s_t))^2$, δ_2 and δ_3 are the number of chips the player and the opponent has placed at time step t.
7: **end for**

```
import json
import socket
...
# The IP address and port of the platform
server_ip = "127.0.0.1"
server_port = 1080
# Create socket and connect to the platform
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(server_ip, server_port)
while True:
        # Get state in json format from the platform
        state = recvJson(client)
        ...
        # Use your awesome AI to get the action
        action = act(state)
        ...
        # send your action to the platform
        sendJson(client, action)
# Close the socket
client.close()
```

Fig. 6. Example Python code of connecting to the platform for testing NLTH AIs.

NLTH AI researchers and developers are using this platform. It has accumulated about 20 million high-quality poker data, which increases by about 100 000 per day. We believe these large-scale data will also facilitate the research of data-driven imperfect-information game solving, imitation learning, and opponent modeling algorithms.

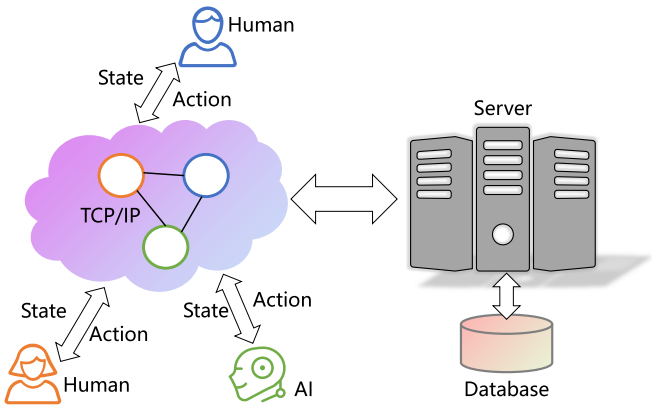

Fig. 7. Schematic of our testing platform's system architecture.

TABLE II
HEAD-TO-HEAD PERFORMANCES (MBB/H) OF THE RULE BASED AI $\mathcal{A}^{\mathcal{R}}$, THE CFR BASED AI $\mathcal{A}^{\mathcal{C}}$, THE DEEPSTACK-LIKE AI $\mathcal{A}^{\mathcal{D}}$, AND THE RL BASED AI $\mathcal{A}^{\mathcal{RL}}$ WHEN PLAYING AGAINST SLUMBOT, RESPECTIVELY

Baseline NLTH AIs	$\mathcal{A}^{\mathcal{R}}$	$\mathcal{A}^{\mathcal{C}}$	$\mathcal{A}^{\mathcal{D}}$	$\mathcal{A}^{\mathcal{RL}}$
Performance (mbb/h)	57	-20	103	110

V. EXPERIMENTS

In this section, we first compare the performance of our baseline NLTH AIs with other publicly available NLTH AIs using the proposed evaluation protocols and online testing platform. Then, we conduct a set of ablation studies to analyze the effects of various design choices in the baseline NLTH AIs.

A. Comparison to the State-of-the-Arts

To the best of our knowledge, Slumbot [22], the champion of the 2018 ACPC, is the only publicly available NLTH AI that provides comparisons through an online website.[2] Slumbot is a strong CFR-based agent whose entire policy is precomputed and used as a lookup table. Similar to our $\mathcal{A}^{\mathcal{C}}$, Slumbot first uses some abstraction algorithm to create a smaller abstract NLTH game. Then it approximates the Nash equilibrium in the abstract game using the CFR-type algorithm and finally executes the resulting strategy in the original game.

The purpose of Slumbot's website is to facilitate human players to compete with it, and there are no open-source tools available to test the performance of AI against Slumbot. Due to the poor stability of Slumbot's website, playing with a simulated browser will lose the connection after a certain number of matches, so we developed a software that uses an alternative method of sending data packets directly. Based on this software, we compare each of our baseline AIs with Slumbot for 100 000 hands, and the head-to-head based evaluation results (AIVAT) are shown in Table II.

We can see that the DeepStack-like AI $\mathcal{A}^{\mathcal{D}}$ outperforms Slumbot by a large margin. This result demonstrates that the real-time subgame solving technique used by DeepStack performs better than Slumbot's offline solving procedure. Although the performance of the CFR-based AI $\mathcal{A}^{\mathcal{C}}$ is not as

[2]https://www.slumbot.com/

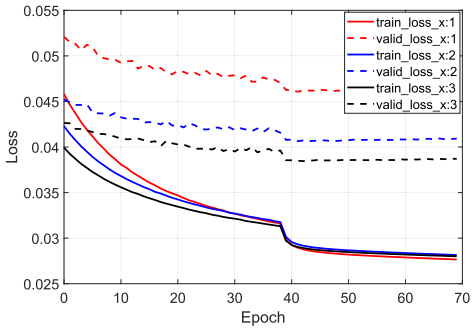

Fig. 8. Training and validation loss curves of the flop network when using $x \in \{1, 2, 3\}$ million training samples, respectively.

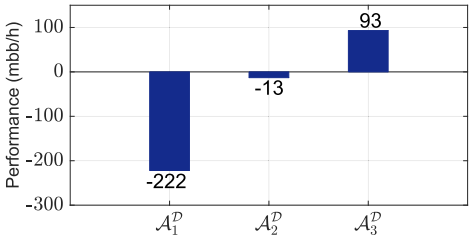

Fig. 9. Head-to-head performances of $\mathcal{A}_1^{\mathcal{D}}$, $\mathcal{A}_2^{\mathcal{D}}$, and $\mathcal{A}_3^{\mathcal{D}}$ when playing against Slumbot, respectively.

good as that of Slumbot, its performance is also commendable because Slumbot exploits a far more fine-grained abstraction algorithm. An interesting result is that the rule-based AI $\mathcal{A}^{\mathcal{R}}$ outperforms Slumbot. This result is not surprising, as it has been reported that the abstraction-based programs from the ACPC are exploitable [54].

Our proposed end-to-end deep RL-based AI $\mathcal{A}^{\mathcal{RL}}$ beats Slumbot by a large margin of 110 mbb/h. Compared with Slumbot, $\mathcal{A}^{\mathcal{RL}}$ does not require domain knowledge for abstraction and achieves better performance while significantly reducing computational and storage resources. $\mathcal{A}^{\mathcal{RL}}$ also outperforms $\mathcal{A}^{\mathcal{D}}$ and beats it by 15 mbb/h. Unlike DeepStack, $\mathcal{A}^{\mathcal{RL}}$ does not need iterative learning in both the training and inference stages. Given input state representation, it performs only one feedforward pass of the neural network to output the action directly.

Similar to DeepStack, Libratus from Carnegie Mellon University also defeated professionals using a nested safe subgame solving algorithm with an extensible blueprint strategy. However, compared with DeepStack, Libratus is a more complex system. It contains multiple sophisticated modules and consumes lots of CPU resources. Due to our limited CPU resources and domain knowledge, we could not implement all the details of Libratus. We will continue reproducing Libratus and deploying it to OpenHoldem when satisfactory performance is obtained. Meanwhile, since OpenHoldem's testing platform is publicly available, we will also try to invite Libratus's author to directly use our platform for performance testing.

The above experimental results illustrate that our baseline NLTH AIs are adequate to serve as a good starting point for NLTH AI research. Since $\mathcal{A}^{\mathcal{D}}$ and $\mathcal{A}^{\mathcal{RL}}$ obtain the best performance among the four baselines and they are also the most complicated baselines in terms of design and implementation, we will conduct some ablation studies to understand the effects of their various design choices next.

B. Ablation Study on $\mathcal{A}^{\mathcal{D}}$

1) Effects of Training Data Size: The training of the river, turn, and flop value networks of $\mathcal{A}^{\mathcal{D}}$ requires a lot of training data. We use $\mathcal{A}_x^{\mathcal{D}}$ to denote the DeepStack-like AIs whose flop networks are obtained by training with x million samples. Fig. 8 shows the loss curves of the flop network during training when $x \in \{1, 2, 3\}$. It is clear that the flop network suffers from severe over-fitting when the training data size is small, and increasing the training data size alleviates this phenomenon. The head-to-head based evaluation results (AIVAT) in Fig. 9 also show that DeepStack-type AI is data-hungry, and more training data results in a stronger AI.

2) Effects of CFR Iterations During Continual Resolving: We use $\mathcal{A}_3^{\mathcal{D}:y}$ to denote the DeepStack-like NLTH AIs, which use y CFR iterations during the continual resolving procedure. We find that $\mathcal{A}_3^{\mathcal{D}:500}$ loses 224 mbb to Slumbot per hand, while $\mathcal{A}_3^{\mathcal{D}:1000}$ wins Slumbot 93 mbb/h. These experimental results demonstrate that the number of CFR iterations during continual resolving is critical to the performance of DeepStack-type AI.

TABLE III
ABLATION ANALYSES OF EACH COMPONENT OF $\mathcal{A}^{\mathcal{RL}}$

Name	Training time (Hours)	ELO
Vector	3.8	85
PokerCNN	5.4	362
Without History Information	6.3	910
Original PPO Loss	8.4	1270
Dual-Clip PPO Loss	8.4	1338
$\mathcal{A}^{\mathcal{RL}}$	8.4	**1611**

C. Ablation Study on $\mathcal{A}^{\mathcal{RL}}$

To analyze the effectiveness of each component of the RL-based AI $\mathcal{A}^{\mathcal{RL}}$, we conduct extensive ablation studies as shown in Table III. The results of each row are obtained by replacing one component of $\mathcal{A}^{\mathcal{RL}}$, and the rest remains unchanged. All models use the same number of training samples, and we use ELO [62] scores to compare their performance.

1) Effects of Different State Representations: We consider three alternative methods for state representation comparison: 1) vectorized state representation like DeepCFR [63]. As shown in Fig. 4(b), it uses vectors to represent card and action information; 2) PokerCNN-based state representation [64] uses tensors to represent card and action information together and uses a single ConvNet to learn features; and 3) state representation without history information is similar to $\mathcal{A}^{\mathcal{RL}}$ except that it does not contain historical action information.

As shown in Table III, state representation significantly impacts the final performance. PokerCNN performs better than the vectorized state representation, demonstrating that it

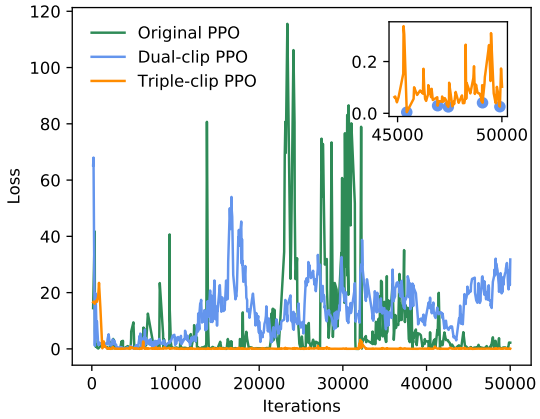

Fig. 10. Loss curves of original PPO, dual-clip PPO and triple-clip PPO during the training process.

TABLE IV
COMPARISONS OF COMPUTATIONAL COST OF DIFFERENT ALGORITHMS

	Training resources (CPU hours)	Training resources (GPU hours)	Time per action	Time per hand
DeepStack	1,533,000	13,140	3.0s	7.2s
Libratus	>3,000,000	No GPUs	-	13.0s
$\mathcal{A}^{\mathcal{RL}}$	2,300	340	2.9ms	7.1ms

is more effective to represent state information using structured tensors. $\mathcal{A}^{\mathcal{RL}}$ outperforms PokerCNN since it uses a pseudo-Siamese architecture to handle card and action information separately. $\mathcal{A}^{\mathcal{RL}}$ is also better than "Without History Information" since historical action information is critical to decision-making in NLTH. $\mathcal{A}^{\mathcal{RL}}$ obtains the best performance thanks to its effective multidimensional state representation, which encodes historical information and is suitable for ConvNets to learn effective feature hierarchies.

2) Effects of Different Loss Functions: For the loss function, we evaluate $\mathcal{A}^{\mathcal{RL}}$'s Triple-Clip PPO loss against two kinds of losses: 1) original PPO loss [55] and 2) dual-clip PPO loss [11]. As shown in Table III, compared with the Original PPO, Dual-Clip PPO has a slight performance boost, and Triple-Clip PPO ($\mathcal{A}^{\mathcal{RL}}$) obtains the best performance. In Fig. 10, we also observe that the Triple-Clip PPO's learning curve is more stable than those of the Original PPO and the Dual-Clip PPO. This performance improvement is mainly because $\mathcal{A}^{\mathcal{RL}}$'s policy-clip and value-clip loss effectively limit its output to a reasonable range, thus ensuring the stability of the policy update. In addition, we find the model with a small overall loss generally performs better after adding the value-clip loss, which is very convenient for model selection during training. This phenomenon also demonstrates that the Triple-Clip loss helps the model converge to a better policy.

3) Computational Cost Analysis: To further illustrate the lightweight properties of our proposed end-to-end deep RL-based AI $\mathcal{A}^{\mathcal{RL}}$, we compare it with the state-of-the-art AIs, i.e., DeepStack and Libratus, from different aspects in Table IV. Both DeepStack and Libratus compute an abstraction of the game and introduce subgame solving with the CFR algorithm. Under the CFR framework, the primary computation cost comes from the CFR iteration process performed in both the model training and testing stages. To ensure high-quality prediction, this iteration process often needs to be

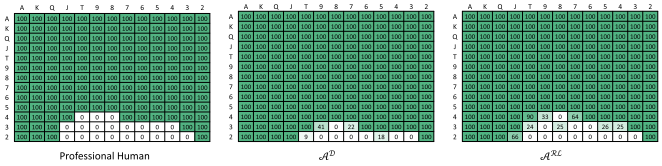

Fig. 11. Probabilities (%) for not folding as the first action for each possible hand. The bottom-left half shows the policy when the suits of two private cards do not match, and the top-right half shows the policy when the suits of two private cards match. (Left to right) Policies of professional human, $\mathcal{A}^{\mathcal{D}}$, and $\mathcal{A}^{\mathcal{RL}}$, respectively.

carried out more than 1000 times in practice, which is very time-consuming. In contrast, $\mathcal{A}^{\mathcal{RL}}$ is very lightweight. It inferences from state information directly to the final action using only a forward pass of the neural network in each decision point, which is more than a thousand times faster than these state-of-the-art AIs.

4) Comparison With Humans: To further demonstrate the performance of $\mathcal{A}^{\mathcal{RL}}$, we invite four professional players to play 10 000 hands against it. Player 1 won the runner-up in WPT DRAGON SERIES 2015, and Player 2 won the fifth place in WPT TEAM DRAGON SERIES 2017. The other two players are both high-end players in online poker. Each player is given two weeks to complete the 2500 game matches. To incentivize players to perform at their best, monetary prizes of U.S. $1000 were awarded to the best performing player. The players were informed of all of these details when they registered to participate, and they were allowed to take as long as they wanted for any decision, but were not allowed to use any software to assist them while playing. On average, for each hand, Player 1 spent 6.6 s, Player 2 spent 6.2 s, Player 3 spent 4.1 s, and Player 4 spent 3.3 s. $\mathcal{A}^{\mathcal{RL}}$ spent 0.01 s per hand on average.

Matches were played between August 14, 2021, and August 29, 2021, and run using our online testing platform, where players can choose to play multiple games simultaneously, as is common in online poker sites. The action history in the hand was displayed on the user interface. This helped the human if he/she forgot what had happened in the hand so far. The humans could claim that they accidentally clicked the wrong action button on the user interface. In each occurrence, we canceled the hand. We allowed these players to choose how many days they would spend playing the 2500 hands. They were also allowed to take breaks at any time.

To reduce the variance caused by all-in, we split the pot by averaging all possible roll-outs of the remaining cards in situations where the players went all-in before the final card was dealt. $\mathcal{A}^{\mathcal{RL}}$ beats these professionals by 10 mbb/h on average, which supports its high performance in beating Slumbot and DeepStack. From the game logs, we observe that $\mathcal{A}^{\mathcal{RL}}$ has learned the abilities to perform bluffing and to recognize the opponent's bluff, which to some extent explains its good performance.

5) Visualization of the Learned Policy: To analyze $\mathcal{A}^{\mathcal{RL}}$'s learned policy, we compare the action frequencies where the agent is the first player to act and has no prior state influencing it [57] with those from human professional[3] and $\mathcal{A}^{\mathcal{D}}$. Fig. 11

[3]Obtained from https://www.wsop.com/how-to-play-poker/

shows the policies on how to play the first two cards from the professional human and the two agents. $\mathcal{A}^{\mathcal{RL}}$'s policy is very similar to that of the human professional, which further explains its superior performance.

VI. Conclusion

This work presents OpenHoldem, a benchmark for large-scale imperfect-information game research using NLTH. OpenHoldem provides an integrated toolkit with three main components: comprehensive evaluation protocols, strong baseline NLTH AIs, and an easy-to-use online testing platform. The algorithms in OpenHoldem cover a lot of important problems in the field of neural networks and learning systems, such as no-regret learning and deep RL under the imperfect-information setting. We plan to add more NLTH AIs to OpenHoldem in the future, with the ultimate goal of providing an NLTH AI Zoo for the research community. We hope OpenHoldem will facilitate further studies on the unsolved theoretical and computational issues in large-scale imperfect-information games.

References

[1] M. Campbell, A. J. Hoane Jr., and F.-H. Hsu, "Deep blue," *Artif. Intell.*, vol. 134, nos. 1–2, pp. 57–83, Jan. 2002.

[2] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[4] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[5] D. Silver et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018.

[6] J. Schrittwieser et al., "Mastering atari, go, chess and Shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, Dec. 2020.

[7] M. Jaderberg et al., "Human-level performance in 3D multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, May 2019.

[8] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, Nov. 2019.

[9] OpenAI et al., "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.

[10] J. Li et al., "Suphx: Mastering mahjong with deep reinforcement learning," 2020, *arXiv:2003.13590*.

[11] D. Ye et al., "Mastering complex control in MOBA games with deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 6672–6679.

[12] D. Ye et al., "Towards playing full MOBA games with deep reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 621–632.

[13] M. Bowling, N. Burch, M. Johanson, and O. Tammelin, "Heads-up limit hold'em poker is solved," *Science*, vol. 347, no. 6218, pp. 145–149, 2015.

[14] M. Moravčík et al., "DeepStack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, vol. 356, no. 6337, pp. 508–513, May 2017.

[15] N. Brown and T. Sandholm, "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals," *Science*, vol. 359, no. 6374, pp. 418–424, Jan. 2018.

[16] D. Zha et al., "Douzero: Mastering DouDizhu with self-play deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12333–12344.

[17] J. Gray, A. Lerer, A. Bakhtin, and N. Brown, "Human-level performance in no-press diplomacy via equilibrium search," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–24.

[18] P. R. Wurman et al., "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, Feb. 2022.

[19] J. Nash, "Non-cooperative games," *Ann. Math.*, vol. 54, no. 2, pp. 286–295, 1951.

[20] J. Rubin and I. Watson, "Computer poker: A review," *Artif. Intell.*, vol. 175, nos. 5–6, pp. 958–987, Apr. 2011.

[21] M. Johanson, "Measuring the size of large no-limit poker games," 2013, *arXiv:1302.7008*.

[22] E. G. Jackson, "Slumbot NL: Solving large games with counterfactual regret minimization using sampling and distributed processing," in *Proc. AAAI Conf. Artif. Intell. Workshops*, 2013, pp. 35–38.

[23] N. Brown, S. Ganzfried, and T. Sandholm, "Hierarchical abstraction, distributed equilibrium computation, and post-processing, with application to a champion no-limit Texas hold'em agent," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, 2015, pp. 7–15.

[24] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1729–1736.

[25] N. Brown and T. Sandholm, "Superhuman AI for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, Aug. 2019.

[26] C. Liu, E. Zhu, Q. Zhang, and X. Wei, "Modeling of agent cognition in extensive games via artificial neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4857–4868, Oct. 2018.

[27] L. Chen, X. Liang, Y. Feng, L. Zhang, J. Yang, and Z. Liu, "Online intention recognition with incomplete information based on a weighted contrastive predictive coding model in wargame," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Feb. 2, 2022, doi: 10.1109/TNNLS.2022.3144171.

[28] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Jun. 1992, pp. 517–520.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[30] H. Hassan et al., "Achieving human parity on automatic Chinese to English news translation," 2018, *arXiv:1803.05567*.

[31] G. Brockman et al., "OpenAI gym," 2016, *arXiv:1606.01540*.

[32] M. Wydmuch, M. Kempka, and W. Jaskowski, "ViZDoom competitions: Playing doom from pixels," *IEEE Trans. Games*, vol. 11, no. 3, pp. 248–259, Sep. 2019.

[33] W. H. Guss et al., "MineRL: A large-scale dataset of minecraft demonstrations," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 2442–2448.

[34] M. Lanctot et al., "OpenSpiel: A framework for reinforcement learning in games," 2019, *arXiv:1908.09453*.

[35] D. Zha et al., "RLCard: A platform for reinforcement learning in card games," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 2442–2448.

[36] D. Billings, D. Papp, J. Schaeffer, and D. Szafron, "Opponent modeling in poker," in *Proc. AAAI Conf. Artif. Intell.*, 2015, pp. 493–499.

[37] N. Bard, J. Hawkin, J. Rubin, and M. Zinkevich, "The annual computer poker competition," *AI Mag.*, vol. 34, no. 2, p. 112, Jun. 2013.

[38] M. Johanson, N. Burch, R. Valenzano, and M. Bowling, "Evaluating state-space abstractions in extensive-form games," in *Proc. Int. Conf. Auto. Agents Multiagent Syst.*, 2013, pp. 271–278.

[39] S. Ganzfried and T. Sandholm, "Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 682–690.

[40] M. Lanctot, K. Waugh, M. Zinkevich, and M. Bowling, "Monte Carlo sampling for regret minimization in extensive games," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1078–1086.

[41] O. Tammelin, "Solving large imperfect information games using CFR+," 2014, *arXiv:1407.5042*.

[42] E. G. Jackson, "Compact CFR," in *Proc. AAAI Conf. Artif. Intell. Workshops*, 2016, pp. 366–370.

[43] M. Schmid, N. Burch, M. Lanctot, M. Moravcik, R. Kadlec, and M. Bowling, "Variance reduction in Monte Carlo counterfactual regret minimization for extensive form games using baselines," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 2157–2164.

[44] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA, USA: MIT Press, 1994.

[45] D. Blackwell, "An analog of the minimax theorem for vector payoffs," *Pacific J. Math.*, vol. 6, no. 1, pp. 1–8, Mar. 1956.

[46] Y. Zhu and D. Zhao, "Online minimax Q network learning for two-player zero-sum Markov games," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 1228–1241, Mar. 2022.
[47] Y. Zhu, D. Zhao, and X. Li, "Iterative adaptive dynamic programming for solving unknown nonlinear zero-sum game based on online data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 714–725, Mar. 2017.
[48] A. Yazidi, D. Silvestre, and B. J. Oommen, "Solving two-person zero-sum stochastic games with incomplete information using learning automata with artificial barriers," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 650–661, Feb. 2023.
[49] Y. Zhao, S. Qiu, K. Li, L. Luo, J. Yin, and J. Liu, "Proximal online gradient is optimum for dynamic regret: A general lower bound," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7755–7764, Dec. 2022.
[50] K. Gokcesu and S. S. Kozat, "An online minimax optimal algorithm for adversarial multiarmed bandit problem," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5565–5580, Nov. 2018.
[51] S. Yang and Y. Gao, "An optimal algorithm for the stochastic bandits while knowing the near-optimal mean reward," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 2285–2291, May 2021.
[52] A. Alipour-Fanid, M. Dabaghchian, and K. Zeng, "Self-unaware adversarial multi-armed bandits with switching costs," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 29, 2021, doi: 10.1109/TNNLS.2021.3110194.
[53] N. Burch, M. Schmid, M. Moravcik, D. Morill, and M. Bowling, "AIVAT: A new variance reduction technique for agent evaluation in imperfect information games," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 949–956.
[54] V. Lisy and M. Bowling, "Eqilibrium approximation quality of current no-limit poker bots," in *Proc. AAAI Conf. Artif. Intell. Workshops*, 2017, pp. 361–366.
[55] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[56] N. Brown and T. Sandholm, "Solving imperfect-information games via discounted regret minimization," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1829–1836.
[57] R. Zarick, B. Pellegrino, N. Brown, and C. Banister, "Unlocking the potential of deep counterfactual value networks," 2020, *arXiv:2007.10442*.
[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[59] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
[61] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.
[62] A. E. Elo, *The Rating of Chessplayers, Past and Present*. New York, NY, USA: Arco, 1978.
[63] N. Brown, A. Lerer, S. Gross, and T. Sandholm, "Deep counterfactual regret minimization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 793–802.
[64] N. Yakovenko, L. Cao, C. Raffel, and J. Fan, "Poker-CNN: A pattern learning strategy for making draws and bets in poker games using convolutional networks," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 360–367.

Kai Li (Member, IEEE) received the Ph.D. degree in pattern recognition and intelligent system from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2018.

He is currently an Associate Professor with the Institute of Automation, Chinese Academy of Sciences. His main research interests are large-scale imperfect-information games and deep multiagent reinforcement learning.

Hang Xu received the bachelor's degree in engineering from Wuhan University, Wuhan, China, in 2020. He is currently pursuing the Ph.D. degree in pattern recognition and intelligent systems with the Institute of Automation, Chinese Academy of Sciences, Beijing, China.

His research interests include computer game and reinforcement learning.

Enmin Zhao received the bachelor's degree in engineering from Tsinghua University, Beijing, China, in 2018. He is currently pursuing the Ph.D. degree in pattern recognition and intelligent systems with the Institute of Automation, Chinese Academy of Sciences, Beijing.

His research interests include computer poker and deep reinforcement learning.

Zhe Wu received the bachelor's degree in engineering from Shandong University, Jinan, China, in 2019. He is currently pursuing the master's degree in pattern recognition and intelligent systems with the Institute of Automation, Chinese Academy of Sciences, Beijing, China.

His research interests include opponent modeling and meta learning.

Junliang Xing (Senior Member, IEEE) received the dual B.S. degree in computer science and mathematics from Xi'an Jiaotong University, Xi'an, Shaanxi, China, in 2007, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2012.

He is currently a Professor with the Department of Computer Science and Technology, Tsinghua University. His research interests include computer vision problems related to human faces and computer gaming problems in imperfect information decision.