

LOOK-AHEAD REASONING WITH A LEARNED MODEL IN IMPERFECT INFORMATION GAMES

Ondřej Kubíček

Artificial Intelligence Center, FEE
Czech Technical University in Prague
Czech Republic
kubicon3@fel.cvut.cz

Viliam Lisý

Artificial Intelligence Center, FEE
Czech Technical University in Prague
Czech Republic
viliam.lisy@agents.fel.cvut.cz

ABSTRACT

Test-time reasoning significantly enhances pre-trained AI agents’ performance. However, it requires an explicit environment model, often unavailable or overly complex in real-world scenarios. While MuZero enables effective model learning for search in perfect information games, extending this paradigm to imperfect information games presents substantial challenges due to more nuanced look-ahead reasoning techniques and large number of states relevant for individual decisions. This paper introduces an algorithm LAMIR that learns an abstracted model of an imperfect information game directly from the agent-environment interaction. During test time, this trained model is used to perform look-ahead reasoning. The learned abstraction limits the size of each subgame to a manageable size, making theoretically principled look-ahead reasoning tractable even in games where previous methods could not scale. We empirically demonstrate that with sufficient capacity, LAMIR learns the exact underlying game structure, and with limited capacity, it still learns a valuable abstraction, which improves game playing performance of the pre-trained agents even in large games.

1 INTRODUCTION

Strategic reasoning and planning are key components of human intelligence, encompassing our ability to reason about possible outcomes of actions in complex situations, often with incomplete information and uncertain consequences. Although humans navigate such decision-making naturally, replicating this process in artificial intelligence remains a fundamental challenge. Games, with their well-defined rules and yet complex strategic landscapes, serve as ideal testbeds for developing and evaluating AI planning and reasoning methods (Mnih et al., 2015; Silver et al., 2018; Perolat et al., 2022; Schultz et al., 2025).

In perfect information games like Chess, Go or Shogi, look-ahead search algorithms as Minimax and Monte Carlo Tree Search (MCTS) have achieved superhuman performance by systematically exploring future states (Russell & Norvig, 2003; Silver et al., 2018). These methods traditionally rely on access to game rules to implement state transitions in the search. MuZero demonstrated that an agent can learn a model of the environment dynamics implicitly through interaction and use this learned model to perform MCTS planning, removing the dependency on explicit implementation of the rules (Schrittwieser et al., 2020).

Not requiring explicit pre-programmed representation of the environment greatly expands applicability of AI methods. A method that does not require explicit rules representation can be applied, for example, to create an AI opponent in a proprietary video game without access to its source code; to create AI opponents for a large database of games for an online game playing platform, where programming a suitable representation for each of them would be prohibitively expensive; or in a game design setting, where the game is repetitively modified without the need for a programmer to reflect the changes in the implementation.

However, extending the model learning paradigm to imperfect information games such as Poker or Stratego presents fundamental difficulties. Since players lack complete knowledge of the state of the game, theoretically sound look-ahead reasoning methods need to reason about the distribution

of all possible hidden states consistent with shared knowledge (Kovářík et al., 2023), which differs from the MCTS used by MuZero.

Our work aims to **enable look-ahead reasoning in two-player imperfect information games using a learned abstract model**, thereby **eliminating the need for explicit game rules** and also **enabling look-ahead reasoning in parts of the game intractable without abstraction**. Following the approach of Schrittwieser et al. (2020), we focus only on games without chance events, which is a large class that includes commonly used benchmarks like Dark Chess, Stratego, Battleship or Imperfect Information Goofspiel. This allows us to tackle the unique difficulties of learning an effective abstraction for imperfect information without conflating it with the separate challenges introduced by chance events. Our contributions are: We identify the necessary components that a learned model must capture to facilitate look-ahead reasoning under imperfect information. We introduce a training procedure to learn these components from sampled game trajectories. We demonstrate how tractably small, domain-independent abstractions can be learned concurrently with the model. Finally, we introduce a way to conduct look-ahead reasoning with the learned model.

Our empirical evaluation shows that in small games, given sufficient capacity, the strategies produced by look-ahead reasoning are less exploitable than those of concurrently trained Regularized Nash Dynamics (RNAD). Furthermore, in large games with shared knowledge consistent with over 10^{18} states for some decisions, the proposed look-ahead reasoning is still applicable in all decision points and significantly improves over RNAD, reaching up to 80% win rate in head-to-head play.

1.1 RELATED WORK

Direct Policy Optimization One approach for computing strategies in large imperfect information games stores the strategy implicitly in neural network weights and directly optimizes this policy based on self-play traces. These methods include: policy-gradient algorithms with reward regularization, like Regularized Nash Dynamics (Perolat et al., 2021; 2022; Sokota et al., 2023; Masaka et al., 2025); training networks to approximate CFR results, like Deep CFR or DREAM (Brown et al., 2019; 2020; Steinberger et al., 2020); or iteratively training best responses to growing strategy pools, like PSRO (Lanctot et al., 2017; Vinyals et al., 2019). Critically, these approaches rely solely on the trained actor during gameplay and cannot refine decisions with additional test-time computation. Pre-trained agents without test-time reasoning are usually very exploitable (Wang et al., 2023; Lisý & Bowling, 2017), and adding test-time reasoning greatly improves their capabilities in games (Silver et al., 2016; Kubíček et al., 2024), and beyond (Snell et al., 2024). This paper enables adding test-time reasoning to policies created by direct policy optimization algorithms.

Look-ahead reasoning Reasoning algorithms in imperfect information games, such as Counterfactual Regret Minimization (Zinkevich et al., 2007), iteratively improve player’s policies by systematically iterating over all possible future action sequences in all possible (unobserved) states of the game. In large games, this requires either domain-specific abstractions, like Libratus (Brown & Sandholm, 2018) or depth-limited reasoning, like DeepStack, ReBeL, Student of Games, SePoT (Moravčík et al., 2017; Brown et al., 2020; Schmid et al., 2023; Kubíček et al., 2024). In either case, all these algorithms require explicit implementation of game rules to construct game trees and manage belief states. It limits their applicability when exact rules are unavailable, computationally prohibitive or if the amount of possible hidden states is too huge. Knowledge-limited subgame solving (Zhang & Sandholm, 2021; 2026; Liu et al., 2023) can reduce the complexity, but even this reduced state space remains intractable in the games we study here. In contrast, our approach does not require explicit game rules and automatically learns a tractably small abstraction of the game just from full traces of game play.

Model learning In single-player settings, Dreamer, TD-MPC and subsequent works showed that learning models and generating artificial traces can match purely model-free approaches (Hafner et al., 2020; 2021; 2025; Hansen et al., 2022; 2024). MuZero demonstrated similar results in perfect information games, using learned models for reasoning during gameplay (Schrittwieser et al., 2020; Antonoglou et al., 2022). Our work extends these approaches to imperfect information games without chance. The model we learn is similar to TD-MPC (Hansen et al., 2022), but it trains additional components to account for the abstractions and more nuanced imperfect information search.

2 BACKGROUND

We define two-player zero-sum simultaneous move game as $\mathcal{G} = (\mathcal{N}, \mathcal{W}, w^{\text{INIT}}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O})$ (Kovářík et al., 2022), where $\mathcal{N} = \{1, 2\}$ are the players, \mathcal{W} is the set of world states in the game and $w^{\text{INIT}} \in \mathcal{W}$ is the initial world state. $\mathcal{A} = \prod_{i \in \mathcal{N}} \mathcal{A}_i$ is the set of joint actions,. We use $\mathcal{A}(w) \subseteq \mathcal{A}$ to denote the set of joint legal actions in world state w . $\mathcal{T} : \mathcal{W} \times \mathcal{A} \rightarrow \mathcal{W}$ is the transition function and $\mathcal{R} : \mathcal{W} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, which corresponds to the reward of player 1. We use $\mathcal{R}_1(w, a) = \mathcal{R}(w, a)$ and $\mathcal{R}_2(w, a) = -\mathcal{R}(w, a)$ as rewards for player 1 and 2 respectively. $\mathcal{O} : \mathcal{W} \times \mathcal{A} \times \mathcal{W} \rightarrow \mathbb{O}$ is the observation function. $\mathbb{O} = \mathbb{O}_0 \times \mathbb{O}_1 \times \mathbb{O}_2$ is the set of joint public and private observations. \mathcal{O} can be factored as $\mathcal{O} = (\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2)$, where \mathcal{O}_0 is a public part of observation and $\mathcal{O}_1, \mathcal{O}_2$ are private parts of observations for each player. Even though we focus on simultaneous-move games, this does not limit the generality, since any sequential-move game can be converted into a simultaneous-move game by adding fictitious moves for the non-acting player in each decision node.

History $h = w^0, a^0 \dots a^{l-1} w^l \in (\mathcal{W}\mathcal{A})^* \mathcal{W}$ is a finite sequence of world states and actions, which starts in the initial world state $w^0 = w^{\text{INIT}}$ and for each timestep $t \in \{0, \dots, l-1\}$ holds $a^t \in \mathcal{A}(w^t)$ and $w^{t+1} = \mathcal{T}(w^t, a^t)$. \mathcal{H} is the set of all possible histories within the game. We use $h \sqsubseteq h'$ to denote that h' contains h as a prefix. We will use $h^{\text{INIT}} = w^{\text{INIT}}$ to denote the initial history. Each history h ends with some world state w^l . We will sometimes use history h in the game functions instead of the world state w^l . For example $\mathcal{A}(h) := \mathcal{A}(w^l)$ corresponds to the set of joint legal actions in the final world state of the history. Each player i does not observe the whole world state at each timestep, but only observes public observations \mathcal{O}_0 and its private observations \mathcal{O}_i . This means that the player may not be able to distinguish between several different histories. We will use $s_i \in \mathcal{S}_i$ to denote the set of all histories consistent with the observations of the player i , which we will call the information set. \mathcal{S}_i is the set of all the information sets of the player i . $s_i(h)$ is the information set that corresponds to history h . Similarly, the public state $s_0 \in \mathcal{S}_0$ is an information of an external player, which does not have private observations, so it contains all the histories consistent with public observations. Each public state contains one or more information sets for each player.

Strategy of player i is a function $\pi_i : \mathcal{S}_i \rightarrow \Delta \mathcal{A}_i$ that maps each information set to a probability distribution over the actions. We will sometimes use $\pi_i(s_i, a_i)$ as a probability that player will play a_i in information set s_i if it follows the strategy π_i . $\pi = (\pi_1, \pi_2)$ is a joint strategy profile of both players. If $h \sqsubseteq h'$, then the reach probability of reaching history h' from history h under strategy profile π is $P^\pi(h'|h) = \prod_{h'' \sqsupseteq h} \prod_{i \in \mathcal{N}} \pi_i(s_i(h''), a_i)$. We also use $P^\pi(h) := P^\pi(h|h^{\text{INIT}})$. Any reach probability can be factored into the contribution by each player $P^\pi(h|h') = \prod_{i \in \mathcal{N}} P_i^\pi(h|h')$. Expected utility of a history h if all players follow strategy profile π is $u_i^\pi(h) = \sum_{h' \sqsupseteq h} P^\pi(h'|h) \mathcal{R}_i(h', a) \prod_{i \in \mathcal{N}} \pi_i(s_i(h'), a_i)$.

Best response against a strategy π_i is a strategy $\pi_{-i}^{BR} \in BR_{-i}$, which maximizes the opponent's utility $u_{-i}^{(\pi_i, \pi_{-i}^{BR})}(h^{\text{INIT}}) \geq u_{-i}^{(\pi_i, \pi'_{-i})}(h^{\text{INIT}})$ for any π'_{-i} . We use a $-i$ here to symbolize the other player than i , which is a standard notation in games. If all players play a best response to each other, the resulting strategy profile is known as Nash Equilibrium π^* (Nash, 1950; Kuhn & Tucker, 1951). In two-player zero-sum games, this is usually the sought after solution concept. As a metric to evaluate quality of a strategy, we use exploitability $\mathcal{E}(\pi_i) = u_{-i}^{(\pi_i, \pi_{-i}^{BR})}(h^{\text{INIT}}) - u_{-i}^{\pi^*}(h^{\text{INIT}})$, which is how much can opponent gain, when it plays best response as compared to the Nash equilibrium. In two-player zero-sum games, the exploitability is always nonnegative and is zero if and only if the π_i is a part of Nash equilibrium.

3 LEARNING THE GAME MODEL

In perfect information games, players possess complete knowledge of the current game state represented by a history h . Consequently, search algorithms initiate from a single, known root state, simplifying the search. In contrast, imperfect information games (IIGs) grant players only partial observability through an information set s_i , which typically corresponds to multiple possible underlying world states. As established in prior works, approximating optimal strategies via look-ahead reasoning in IIGs requires a more sophisticated approach than in perfect information settings (Kovářík et al., 2023; Moravčík et al., 2017).

Specifically, it is insufficient to restrict the reasoning to only those histories consistent with the player’s i information set. Instead, the reasoning must encompass all histories that share the public state s_0 . This necessity arises because sound reasoning algorithms compute strategies for all players simultaneously, aiming for mutual best responses characteristic of an equilibrium. Consider this situation in two-player Poker: if player i holds two Kings, their information set s_i includes all histories consistent with this hand but with varying opponent hands. A reasoning restricted only to those histories would implicitly grant the opponent knowledge of i ’s hand when computing opponent’s strategy, leading to suboptimal strategies. Therefore, the look-ahead reasoning must operate over the broader set of histories consistent with public state to compute valid equilibrium strategies (Moravčík et al., 2017; Schmid et al., 2023; Kovařík et al., 2023; Milec et al., 2024).

We adopt a reinforcement learning paradigm where an agent learns from interaction with the environment. During training, we assume access to a game simulator capable of generating the whole game trajectories. During testing (gameplay), the agent receives only its own information set s_i at each step and must rely entirely on its learned model to plan, without access to the simulator or explicit rules. This means that the agent does not use any domain-specific knowledge. at any point.

We propose a model inspired by MuZero (Schrittwieser et al., 2020) but adapted for the IIG setting, which requires additional structures necessary to model the imperfect information. Our model comprises three core learnable functions, parameterized by θ :

- Representation function $\Lambda_\theta^I : \mathcal{S}_i \rightarrow \overline{\mathcal{S}}_i$. Maps a player i ’s potentially high-dimensional information set s_i to a fixed-size latent representation $\overline{s}_i \in \overline{\mathcal{S}}_i$.
- Dynamics function $\Upsilon_\theta : \overline{\mathcal{S}}_1 \times \overline{\mathcal{S}}_2 \times \mathcal{A}_1 \times \mathcal{A}_2 \rightarrow \overline{\mathcal{S}}_1 \times \overline{\mathcal{S}}_2 \times \mathbb{R} \times \mathbb{B}$. Given the latent representations for all players $(\overline{s}_1, \overline{s}_2)$ and the joint action taken (a_1, a_2) , this function predicts the resulting next latent representations for both players, the immediate reward r (e.g., for player 1), and a binary termination flag l . This models the joint evolution of the game across possible hidden states. We use $\mathbb{B} = \{0, 1\}$.
- Legal actions function $\Gamma_\theta : \overline{\mathcal{S}}_i \rightarrow \mathbb{B}^{|\mathcal{A}_i|}$. Predicts the mask of legal actions \mathcal{A}_i available to player i from their latent representation \overline{s}_i . This is crucial for constraining the look-ahead reasoning only to the feasible parts of the game.

Each training episode a game trajectory is sampled. This trajectory is $h = w^{\text{INIT}} a^0 \dots a^{l-1} w^l$, where $a^t = (a_1^t, a_2^t)$ is the joint action at step t . For any sub-history $h^t \sqsubseteq h$, the simulator provides the true information sets $s_i^t(h^t)$, legal actions $\mathcal{A}_i^t(s_i^t)$, and the reward r^t . The model is trained to predict these quantities through recurrent application of its components.

Starting from an initial latent state $\overline{s}_i^{t,0} = \Lambda_\theta^I(s_i^t)$, the dynamics function is unrolled for k steps using the actual actions from the trajectory:

$$\overline{s}_1^{t,k+1}, \overline{s}_2^{t,k+1}, r^{t,k+1}, l^{t,k+1} = \Upsilon_\theta(\overline{s}_1^{t,k}, \overline{s}_2^{t,k}, a_1^{t+k}, a_2^{t+k})$$

Here, $\overline{s}_i^{t,k}$ is the predicted latent state after k unrolls from step t , and $\overline{r}^{t,k}$ and $\overline{l}^{t,k}$ are the predicted reward and termination logit. The legal actions function predicts logits $\overline{A}_i^t = \Gamma_\theta(\overline{s}_i^{t,0})$ from the initial latent state.

The model parameters θ are optimized by minimizing a combined loss function over the trajectory:

$$\begin{aligned} \mathcal{L}_\theta^M = \sum_{t=0}^{l-1} & \left[\sum_{i \in \mathcal{N}} \underbrace{\mathcal{L}_\theta^L(\overline{A}_i^t, \mathcal{A}_i^t)}_{\text{Legal Action Prediction}} \right. \\ & \left. + \sum_{k=1}^{l-t} \left(\underbrace{\mathcal{L}_\theta^T(\overline{l}^{t,k}, \mathbb{I}[t+k=l])}_{\text{Termination Prediction}} + \underbrace{\mathcal{L}_\theta^R(\overline{r}^{t,k}, r^{t+k})}_{\text{Reward Prediction}} + \sum_{i \in \mathcal{N}} \underbrace{\mathcal{L}_\theta^D(\overline{s}_i^{t,k}, \Lambda_\theta^I(s_i^{t+k}))}_{\text{Latent State Prediction}} \right) \right] \end{aligned} \quad (1)$$

where \mathcal{L}_θ^L and \mathcal{L}_θ^T are binary cross-entropy losses, while \mathcal{L}_θ^R and \mathcal{L}_θ^D are mean squared errors. The target for the dynamics loss is the latent representation of the actual subsequent information set.

Minimizing \mathcal{L}_θ^M trains the functions $\Lambda_\theta^I, \Upsilon_\theta, \Gamma_\theta$ to collectively act as a learned simulator. This learned model captures the necessary dynamics within the game, enabling test-time look-ahead reasoning algorithms (discussed in Section 5) to effectively plan over the required set of public states without recourse to the original game rules or simulator.

4 LEARNING THE ABSTRACT MODEL

A primary limitation of sound look-ahead reasoning in imperfect information games is the potential size of the public states, as the number of information sets consistent with public information may be exponential in the history length, making the look-ahead reasoning intractable (Moravčík et al., 2017; Schmid et al., 2023).

Although traditional abstraction techniques often rely on domain expertise or require offline enumeration and analysis of all information sets (Čermák et al., 2020; Kroer & Sandholm, 2018; Ganzfried & Sandholm, 2014; Brown & Sandholm, 2015; Bard et al., 2014; Johanson et al., 2013), we aim to learn domain-independent abstraction directly from the game experience during training. Our goal is to partition the vast space of information sets sharing a public state into a manageable number L of abstract information sets, enabling tractable reasoning. Such an abstraction may contain imperfect recall as discussed in Section 7.

Consider Texas Hold’em Poker as an example. A player might hold any of the 1326 private hands, each corresponding to a different information set. Our abstraction aims to represent this multitude using only L representatives, learned based on similarity within the training process rather than predefined rules.

We adapt the model from Section 3, but we will use mechanisms inspired by online clustering to limit the amount of information sets in each public state. We hypothesize that information sets behaving similarly, e.g. having similar optimal strategies or leading to similar future states, should be grouped. We formalize this using a function $\kappa : \mathcal{S}_i \rightarrow \mathbb{R}^K$, which maps any information set to a K -dimensional space, in which the clustering will be performed.

To satisfy the condition that each public state consists of at most L information sets of each player, we split the representation function into two parts as shown in Figure 1. The public state representation $\Lambda_{i,\theta} : \mathcal{S}_0 \rightarrow \overline{\mathcal{S}}_i^L$ maps a public state s_0 to L latent abstract information sets for a player i . The information set representation $\Lambda_{i,\theta}^I : \mathcal{S}_i \rightarrow \Delta \overline{\mathcal{S}}_i$ maps a real information set to the probability distributions on the abstract information sets provided by $\Lambda_{i,\theta}$. Crucially, despite $\Lambda_{i,\theta}^I$ providing a probability distribution, we enforce the many-to-one mapping, so we represent any real information set by the single abstract information set, corresponding to the highest probability. This representative will then be used for training dynamics Υ_θ and legal actions Γ_θ . We opted to this, so that the dynamics function constructs the search tree, which is compatible with look-ahead reasoning algorithms like Counterfactual Regret Minimization.

In order to perform the clustering, we require the same function κ as for the real information sets. We introduce $\kappa_\theta : \overline{\mathcal{S}}_i \rightarrow \mathbb{R}^K$, which will be trained to predict this clustering property for each abstract information set. The dynamics function Υ_θ and legal actions function $\Gamma_{i,\theta}$ are defined as in Section 3.

To learn the abstraction that approximates the proposed clustering based on κ , we train $\Lambda_{i,\theta}, \Lambda_{i,\theta}^I$ and κ_θ jointly. Each training step, the simulator provides trajectory $h = w^{\text{INIT}} a^0 \dots a^{l-1} w^l$. s_0^t, s_i^t represents the public state and information set of player i at time t in the same trajectory.

The training involves two additional losses, the first one \mathcal{L}_θ^A updates the $\Lambda_{i,\theta}$ and κ_θ using a soft clustering objective similar to fuzzy c-means (Bezdek et al., 1984). It minimizes a mean squared error between real and abstract properties weighted by the softmax. The second loss \mathcal{L}_θ^S updates the $\Lambda_{i,\theta}^I$ by minimizing the cross entropy loss between the predicted probability distribution over the abstract information sets and the one-hot encoded vector of the abstract information set that is the nearest neighbor of the real information set based on the κ .

$$\mathcal{L}_\theta^A = \sum_t^{l-1} \sum_{i \in \mathcal{N}} \sum_{\overline{s}_i^t \in \Lambda_{i,\theta}(s_0^t)} \|\kappa_\theta(\overline{s}_i^t) - \kappa(s_i^t)\|^2 \frac{e^{-\gamma \|\kappa_\theta(\overline{s}_i^t) - \kappa(s_i^t)\|^2}}{\sum_{\overline{s}_i^{t'} \in \Lambda_{i,\theta}(s_0^t)} e^{-\gamma \|\kappa_\theta(\overline{s}_i^{t'}) - \kappa(s_i^t)\|^2}} \quad (2)$$

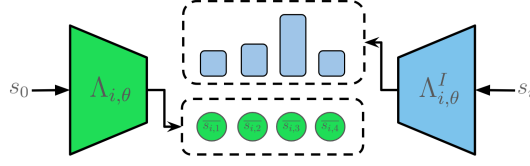


Figure 1: The public state and information set representations functions of player i . First the $\Lambda_{i,\theta}$ predicts 4 abstract information sets and then $\Lambda_{i,\theta}^I$ predicts the probability distribution over those abstractions.

$$\overline{s_i^{t,*}} = \arg \min_{\overline{s_i^t} \in \Lambda_{i,\theta}(s_0^t)} \|\kappa_\theta(\overline{s_i^t}) - \kappa(s_i^t)\|^2 \quad \mathcal{L}_\theta^S = \sum_t \sum_{i \in \mathcal{N}} \mathcal{L}_\theta^{CE}(\text{sg}(\overline{s_i^{t,*}}), \Lambda_{i,\theta}^I(s_i^t)) \quad (3)$$

Here, $x = \text{sg}(x)$ is an identity operator that stops the gradient flow through x . The γ controls the softness of the clustering and as $\gamma \rightarrow \infty$ the clustering becomes hard. The gradients from \mathcal{L}_θ^A propagate through both κ_θ and $\Lambda_{i,\theta}$, but the gradients from \mathcal{L}_θ^S are only propagated through $\Lambda_{i,\theta}^I$.

$$\mathcal{L}_\theta^{MA} = \mathcal{L}_\theta^M + \mathcal{L}_\theta^A + \mathcal{L}_\theta^S \quad (4)$$

The overall loss also includes the model learning loss from Section 3. Importantly, \mathcal{L}_θ^M is computed using the dynamics based on the selected abstract information set. Furthermore, the gradients from \mathcal{L}_θ^M are not backpropagated through $\Lambda_{i,\theta}$, $\Lambda_{i,\theta}^I$ or κ_θ . This decouples the learning of the abstraction structure from the learning of the model dynamics.

5 DEPTH-LIMITED SOLVING

While the learned model and abstraction allow reconstructing the whole game tree of the abstracted game, practical applications in large imperfect information games rely on depth-limited reasoning combined with a learned value function to estimate payoffs beyond the reasoning horizon. Defining and training optimal value function for imperfect information games is challenging, as they theoretically depend on belief states (Brown et al., 2020; Kovařík et al., 2023). The belief states are public states with corresponding probability distribution of reaching each history. Such value function is often trained by repeatedly sampling varied belief states and performing depth-limited reasoning in each of them Moravčík et al. (2017); Schmid et al. (2023), which is generally intractable.

In this section we present an algorithm *Learned Abstract Model for Imperfect-information Reasoning* (LAMIR), which learns the abstracted model from Section 4 along with a value function that enables depth-limit reasoning. We first show what additional components are necessary to learn the value function, then we summarize the whole training procedure and lastly we show how to use these trained components for depth-limited reasoning.

LAMIR uses a value function based on the multi-valued states (Brown et al., 2018; Kubíček et al., 2024; Milec et al., 2025). Training of this value function just from samples requires these additional components:

- Strategy function $\pi_\theta : \mathcal{S}_i \rightarrow \Delta \mathcal{A}$, which for given information set from the original game returns the strategy trained with some policy-gradient algorithm, like RNAD (Perolat et al., 2022).
- Transformations function $\tau_\theta : \mathcal{S}_i \rightarrow \mathbb{R}^{|\mathcal{A}_i| \times |T|}$, representing T characteristic directions in strategy space explored by the policy-gradient algorithm during training. For a single transformation $\chi \in \tau_\theta$ the transformed strategy is computed locally as $\pi_i^\chi(s_i, a_i) = \pi_i(s_i, a_i) + \chi(s_i, a_i)$. The resulting strategy $\pi_i^\chi(s_i)$ is clipped to non-negative numbers and then normalized (Kubíček et al., 2024).
- Value function $v_\theta : \overline{\mathcal{S}}_1 \times \overline{\mathcal{S}}_2 \rightarrow \mathbb{R}^{|T| \times |T|}$, which approximates the expected value of each combination of transformed strategies between players (Kubíček et al., 2024).

We train π_θ and τ_θ using real information sets from sampled trajectories. However, v_θ is trained using the corresponding joint abstract information sets instead of histories. It is possible that several

real histories map to the same abstract state with varying reach probabilities. This may introduce bias to the trained values. While importance sampling could correct this, we found in our experiments it did not affect the results in any significant way. This is most likely due to transformations being just a heuristic approach to approximate different parts of the strategy space.

We have used Regularized Nash Dynamics (RNaD) as the policy-gradient algorithm, which includes Neural Replicator Dynamics (NeuRD) loss for strategy training and mean squared error for the associated baseline value function \mathcal{L}_θ^{PG} (Hennes et al., 2020; Perolat et al., 2021; 2022). Note, that the value function from the RNaD cannot be used as a value function for look-ahead reasoning, because it represents value for a specific belief based on the networks strategy and not for an arbitrary belief. We use mean squared error for the training of the value function \mathcal{L}_θ^V and the targets are computed by the V-trace (Espeholt et al., 2018; Kubíček et al., 2024), which estimates the value of different policies in an off-policy setting.

Following Kubíček et al. (2024), transformations represent the strategy changes during training. For each player i we compute the difference vector $\delta_i^t = \pi_i^{t, \text{NEW}} - \pi_i^{t, \text{OLD}}$, where $\pi_i^{t, \text{OLD}}$ and $\pi_i^{t, \text{NEW}}$ are concatenated strategies along the whole trajectory before and after the training step. Instead of the hard clustering proposed originally, we use the soft clustering from Section 4.

$$\mathcal{L}_\theta^T = \sum_{i \in \mathcal{N}} \sum_{\chi_i \in T} \|\chi_i - \delta_i^t\|^2 \frac{e^{-\gamma \|\chi_i - \delta_i^t\|^2}}{\sum_{\chi'_i \in T} e^{-\gamma \|\chi'_i - \delta_i^t\|^2}} \quad (5)$$

5.1 TRAINING

A single training iteration of LAMIR begins by sampling a trajectory $h = w^{\text{INIT}} a^0 \dots a^{l-1} w^l$ according to a sampling strategy μ . In the on-policy setting, this strategy is the current policy, $\mu = \pi_\theta$. While our approach is not limited to on-policy learning, off-policy settings would require scaling the policy and value function updates with importance sampling.

For any sub-history $h^t \sqsubseteq h$, the simulator provides the true public state representation $s_0^t(h^t)$, true information set representation $s_i^t(h^t)$, legal actions $\mathcal{A}_i^t(s_i^t)$, and the reward r^t . Our method uses a two-step update per episode. This is necessary because the transformation loss \mathcal{L}_θ^T depends on strategy changes induced by the policy-gradient step. First step trains the strategy function π_θ using a policy-gradient loss \mathcal{L}_θ^{PG} of the underlying policy-gradient algorithm. In the specific case of RNaD, this loss is NeuRD (Hennes et al., 2020). Second step computes the transformation loss \mathcal{L}_θ^T and value function loss \mathcal{L}_θ^V (Kubíček et al., 2024). The second step also includes all losses related to learning the game abstraction.

At each timestep t , the abstraction network $\Lambda_{i,\theta}$ predicts a set of L possible abstract information sets for each player: $\overline{s_{i,j}^t} = \Lambda_{i,\theta}(s_0^t)$. The goal is to cluster information sets in a K -dimensional space. A trainable function $\kappa_\theta(\overline{s_{i,j}^t})$ maps the abstract information sets into this space. Similarly $\kappa(s_i^t)$ maps the true information set. κ could be a separate provided function either by environment or user, but similarly it could be the trained strategy $\kappa = \pi_\theta$. \mathcal{L}_θ^A is computed using the K -dimensional representations. It updates both $\Lambda_{i,\theta}$ and κ_θ .

Concurrently, a separate network $\Lambda_{i,\theta}^I$ predicts a probability distribution over the L possible abstract sets $\overline{s_{i,j}^t}$ given the current real information set s_i^t . The \mathcal{L}_θ^S is a cross-entropy loss that trains $\Lambda_{i,\theta}^I$ to increase the likelihood of selecting the closest abstraction, $\overline{s_{i,j}^{t,*}}$ in the K -dimensional space. Finally, the \mathcal{L}_θ^M is computed as described in Section 3. The dynamics model Υ_θ is trained to predict the most likely future abstraction, $\overline{s_{i,j}^{t+k,*}}$, after k steps. The final losses for the two-step update are

$$\mathcal{L}_\theta^1 = \mathcal{L}_\theta^{PG} \quad \mathcal{L}_\theta^2 = \mathcal{L}_\theta^{MA} + \mathcal{L}_\theta^V + \mathcal{L}_\theta^T \quad (6)$$

5.2 LOOK-AHEAD REASONING

At test time, LAMIR employs continual resolving (Moravčík et al., 2017; Schmid et al., 2023) within the learned abstract game. The process begins at the initial history h^{INIT} , which defines

the starting public state $s_0(h^{\text{INIT}})$ and information sets $s_1(h^{\text{INIT}}), s_2(h^{\text{INIT}})$. The root of the initial abstract subgame is generated using $\Lambda_{i,\theta}, \Lambda_{i,\theta}^I$.

The following process then repeats until the terminal state is reached. The algorithm constructs the depth-limited game tree using the learned dynamics Υ_θ and legal actions Γ_θ . At the depth limit, a final "decision layer" is added for non-terminal states. This layer consists of T artificial actions where both players choose their strategy for the remainder of the game. Each combination leads to a terminal state with a reward defined by the learned value function v_θ (Brown et al., 2018; Kubíček et al., 2024).

In this newly created abstracted game LAMIR runs Counterfactual Regret Minimization+ (CFR+) or some of its variant (Zinkevich et al., 2007; Tammelin, 2014; Farina et al., 2019) to get a strategy at each decision node. The actual current information set of the acting player s_i is mapped to its corresponding abstract information set \bar{s}_i using $\Lambda_{i,\theta}^I$. The strategy corresponding to \bar{s}_i is used to sample an action that advances the real game to the new state.

Out of the previous game tree, all the histories consistent with s_0 are used to create a new gadget game. Each of those histories is associated with joint abstract information set. Thanks to $\Lambda_{i,\theta}$, there are at most L^2 unique combinations. Any two histories that share the same joint abstract information set are merged so that the new subgame root contains at most L^2 nodes. Counterfactual values and the acting player's reaches are reused from the previous subtree to provide necessary statistics to the gadget game. The algorithm repeats this process until it reaches the terminal state (Burch et al., 2014; Moravčík et al., 2017).

6 EXPERIMENTS

6.1 EXPLOITABILITY IN SMALLER GAMES

To ensure that the strategies found by LAMIR approximate Nash equilibria, we applied it in games small enough to compute exact exploitability, which serves as a distance from the Nash equilibrium in two-player zero-sum games. For various abstraction sizes L and different properties for clustering κ we trained LAMIR with 10 different random seeds for 100,000 episodes. In all of our experiments we used Regularized Nash Dynamics (RNaD) to train the strategy for multi-valued states value function. The rules of the games used in experiments are in Appendix C.

Starting at episode 80,000 we have computed exploitability every 1000 episodes. In each public state, the algorithm uses the trained functions to construct the depth-limited subgame with depth limit 1. Then this subgame is solved using CFR+ and the strategy from abstract infosets is mapped to real ones. Then we compute the exploitability of the final composed strategy in the original game. The results for different κ and L are displayed in Figure 2.

Compared to RNaD, the LAMIR has a more profound training. As a result single training step of LAMIR takes more time. In our implementation, the training is roughly 2-2.5 times slower per-iteration depending on the size of the abstraction L . We further discuss this in Appendix B.2. Still, the exploitability of RNaD reported in Figure 2 does not improve even with increased training time. The main likely cause are network approximation errors. Similarly in the original results, RNaD could not decrease the exploitability in Leduc Poker beyond a certain point (Perolat et al., 2021).

We used three different κ , which served as a basis for clustering, first the legal actions, second the legal actions with the current RNaD strategy, and third the legal actions, RNaD strategy, and the player's action history. Each choice of κ is capable of outperforming the concurrently trained RNaD with the same amount of training episodes given sufficient L . When using the action history as κ , each information set is uniquely defined in a given public state, which means that, with sufficient capacity, the dynamics network should model the underlying game. This was evaluated in II Goofspiel 5 with $L = 30$, which indeed has mapping one-to-one for each abstract and information set. The main reason why the exploitability is greater than 0 arises from the discrepancies in the rewards produced by the dynamics function.

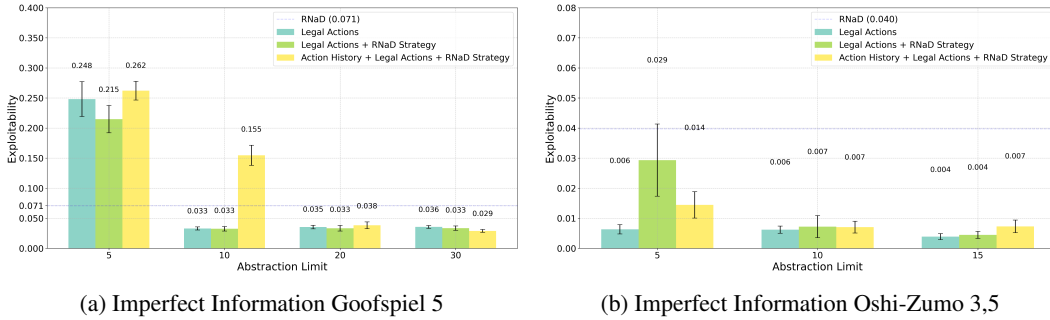


Figure 2: Exploitability of LAMIR in a different games by using continual resolving with depth-limit 1 in each subgame with different choice of abstraction limit L and κ . The largest public state in II Goofspiel 5 contains 30 infosets and in Oshi-Zumo 3,5 it contains 625 information sets.

Algorithm	II Goofspiel 10	II Goofspiel 13	II Goofspiel 15
LAMIR $\kappa = \text{Legal actions}$	$54.47 \pm 0.25 \%$	$60.68 \pm 0.34 \%$	$80.49 \pm 0.26 \%$
LAMIR $\kappa = \text{RNAD strategy}$	$61.60 \pm 0.29 \%$	$58.33 \pm 0.27 \%$	$61.80 \pm 0.36 \%$

Table 1: Average win rate with 2-sigma error bars of LAMIR against RNAD in different games.

6.2 HEAD-TO-HEAD IN LARGE GAMES

The main usage of LAMIR is in very large games, where the exact exploitability cannot be computed. This experiment evaluates LAMIR in this exact setting, where we compare it with Regularized Nash Dynamics (RNAD) in head-to-head play. For each game and κ , we have trained LAMIR with 3 different random seeds for 3 million episodes. Similarly, we have trained RNAD with 6 different random seeds for the same number of episodes. The hyperparameter settings remained the same for each game. Then we matched each trained seed of LAMIR with each trained seed of RNAD and played more than 100,000 matches. Note that when we use $\kappa = \text{RNAD strategy}$, it is the strategy that is learned concurrently for the value function and not the RNAD strategy against which the algorithm is compared later. The resulting win rates with 2-sigma error bars are in Table 1.

LAMIR consistently outperforms RNAD in each of the tested games. Imperfect Information Goofspiel is known for its large public states, so continual resolving without abstractions is not applicable. SePoT (Kubíček et al., 2024) was also evaluated in such large domains, but we did not compare against it, as it only uses CFR if the subgame is small enough. The authors showed that in II Goofspiel 13, SePoT has a win rate of only 52%, which is likely caused by not resolving almost any subgame due to the limit on the size of the subgame. Furthermore, we show that even in larger games than those tested with SePoT, LAMIR improves over RNAD even more.

7 LIMITATIONS AND FUTURE WORK

LAMIR advances the scalability of the continual resolving paradigm to larger games by integrating learned models and abstractions. However, it presents several interesting avenues for future research.

The computational complexity of the look-ahead reasoning, when using CFR is linear in the amount of information sets in the game. This complexity still remains, as LAMIR only reduces the size of the game in each public state but uses CFR in the abstract game. Each subgame LAMIR construct with depth D contains at most $\sum_{d \in \{0, \dots, D\}} L^2 |\mathcal{A}|^{2d}$ unique nodes and at most $L^2 |\mathcal{A}|^{2D} T^2$ terminal histories, where T is the amount of transformations. Most of the subgames would be smaller, but this is the main limitation of LAMIR as it limits how large L can be.

Currently, LAMIR’s dynamics network Υ_θ does not explicitly model chance nodes within the game. LAMIR could still be applied in games with chance events, but this is not an intended setting as the absence of chance nodes will result in poor abstraction, regardless of the abstraction capacity L . Although algorithms like Stochastic MuZero (Antonoglou et al., 2022) demonstrate that modeling chance is feasible in learned models for perfect information games, integrating chance nodes into

our framework, particularly in conjunction with learned abstractions, requires careful consideration and is a key area for future work.

The effectiveness of the learned abstraction depends on the chosen property function κ for clustering. In games without chance, if this property function could perfectly distinguish two different information sets, L is greater than the size of the largest public state and neural networks have sufficient capacity, then LAMIR learns a near-perfect model of the game. When L is reduced to achieve scalability, the learned abstraction may not capture all crucial strategic nuances, which can affect the strength of the derived strategy. In experiments, we have used simple proxies present in any game and it still yielded strong performance. Ideally, κ would also be learned during the training process.

Reducing L necessarily introduces imperfect recall, meaning that the player in the abstract game may "forget" information it previously knew. Algorithms like Counterfactual Regret Minimization (CFR) guarantee convergence to a Nash equilibrium strategy only in perfect recall games. While CFR's convergence is not generally guaranteed in imperfect recall settings, it has been shown for subclasses such as A-loss recall games (Čermák et al., 2020). Our abstractions are A-loss recall games if the public observations in the original game depend only on prior public information and the actions taken in the current round. Games like Imperfect Information Goofspiel or Oshi-Zumo satisfy these conditions, but many others, including Battleships, Dark Chess, or Stratego, do not. Thus, for games that do not fall into the A-loss recall category after abstraction, theoretical convergence guarantees for CFR-based methods within LAMIR are not assured.

LAMIR focuses on abstracting information sets but does not inherently abstract action spaces. In games with very large or continuous action spaces (e.g., bet sizing in Poker), the sheer number of actions can remain a bottleneck for the look-ahead reasoning, regardless of the information set abstraction. While action abstractions have been extensively studied (Brown & Sandholm, 2014; 2018; Li et al., 2024), integrating them with LAMIR is out of the scope of single paper, but it presents another direction for future improvement.

8 CONCLUSION

In this paper, we have introduced LAMIR, an algorithm designed to enable look-ahead reasoning by learning the model of the game with a suitable abstraction directly from experience, without the need for any domain-specific knowledge.

Our core contributions are fourfold. First, we have proposed a method for learning the fundamental components of a model of dynamics of imperfect information games without chance. Second, we developed a technique for automatically learning an abstraction by clustering information sets, effectively reducing the size of the game. Third, we integrated these components with a learning of value function in the abstracted game that enables depth-limited look-ahead reasoning. Lastly, we demonstrate how LAMIR facilitates the continual resolving paradigm by performing a depth-limited look-ahead reasoning in each decision node encountered.

We empirically verify that, when given sufficient capacity, LAMIR learns a nearly perfect model. Still, the game-playing capabilities degrade gracefully when the abstraction capacity is reduced. We have also shown that LAMIR manages to perform look-ahead reasoning even in intractably large public states. Thanks to that, it achieved up to 80% win rate in large games compared to RNaD, a strong baseline that was successfully used to create a human-expert level player in Stratego.

The primary impact of LAMIR lies in scaling look-ahead reasoning techniques to larger games by learning abstraction and its model directly from experience. This overcomes the most notable limitations of the continual resolving paradigm, which requires each subgame to be tractable and to have explicit access to the game model. LAMIR has several limitations, which can be addressed in future work. Still, LAMIR is, to the best of our knowledge, the first algorithm that enables look-ahead reasoning in large-scale games like Imperfect Information Goofspiel 15 without any domain-specific knowledge, substantially outperforming the model-free policy-gradient algorithm.

ACKNOWLEDGMENTS

This research is supported by Czech Science Foundation (GA25-18353S) and Grant Agency of the CTU in Prague (SGS23/184/OHK3/3T/13). Computational resources were supplied by (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic and also (CZ.02.1.01/0.0/0.0/16 019/0000765)

REFERENCES

- Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K Hubert, and David Silver. Planning in stochastic environments with a learned model. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=X6D9bAHhBQ1>.
- Nolan Bard, Michael Johanson, and Michael Bowling. Asymmetric abstractions for adversarial settings. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 501–508, 2014.
- James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & geosciences*, 10(2-3):191–203, 1984.
- James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL <http://github.com/google/jax>.
- Noam Brown and Tuomas Sandholm. Regret transfer and parameter optimization. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI’14, pp. 594–601. AAAI Press, 2014.
- Noam Brown and Tuomas Sandholm. Simultaneous abstraction and equilibrium finding in games. In *International Joint Conference on Artificial Intelligence*, 2015. URL <https://api.semanticscholar.org/CorpusID:12321994>.
- Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018. doi: 10.1126/science.aao1733. URL <https://www.science.org/doi/abs/10.1126/science.aao1733>.
- Noam Brown, Tuomas Sandholm, and Brandon Amos. Depth-limited solving for imperfect-information games. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/34306d99c63613fad5b2a140398c0420-Paper.pdf.
- Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. In *International conference on machine learning*, pp. 793–802. PMLR, 2019.
- Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. *Advances in neural information processing systems*, 33:17057–17069, 2020.
- Neil Burch, Michael Johanson, and Michael Bowling. Solving imperfect information games using decomposition. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI’14, pp. 602–608. AAAI Press, 2014.
- DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL <http://github.com/google-deepmind>.

- Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.
- Gabriele Farina, Christian Kroer, Noam Brown, and Tuomas Sandholm. Stable-predictive optimistic counterfactual regret minimization. In *International conference on machine learning*, pp. 1853–1862. PMLR, 2019.
- Sam Ganzfried and Tuomas Sandholm. Potential-aware imperfect-recall abstraction with earth mover’s distance in imperfect-information games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), Jun. 2014. doi: 10.1609/aaai.v28i1.8816. URL <https://ojs.aaai.org/index.php/AAAI/article/view/8816>.
- Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=S1lOTC4tDS>.
- Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=0oabwyZbOu>.
- Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pp. 1–7, 2025.
- Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=Oxh5CstDJU>.
- Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, pp. 8387–8406. PMLR, 2022.
- Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2024. URL <http://github.com/google/flax>.
- Daniel Hennes, Dustin Morrill, Shayegan Omidshafiei, Rémi Munos, Julien Perolat, Marc Lanctot, Audrunas Gruslys, Jean-Baptiste Lespiau, Paavo Parmas, Edgar Duñez Guzmán, and Karl Tuyls. Neural replicator dynamics: Multiagent learning via hedging policy gradients. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS ’20*, pp. 492–501, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450375184.
- Michael Johanson, Neil Burch, Richard Valenzano, and Michael Bowling. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS ’13*, pp. 271–278, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450319935.
- Vojtěch Kovařík, Martin Schmid, Neil Burch, Michael Bowling, and Viliam Lisý. Rethinking formal models of partially observable multiagent decision making. *Artificial Intelligence*, 303:103645, 2022.
- Vojtěch Kovařík, Dominik Seitz, Viliam Lisý, Jan Rudolf, Shuo Sun, and Karel Ha. Value functions for depth-limited solving in zero-sum imperfect-information games. *Artificial Intelligence*, 314: 103805, 2023.
- Christian Kroer and Tuomas Sandholm. A unified framework for extensive-form game abstraction with bounds. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/aa942ab2bfa6ebda4840e7360ce6e7ef-Paper.pdf.

- Ondřej Kubíček, Neil Burch, and Viliam Lisý. Look-ahead search on top of policy networks in imperfect information games. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 4344–4352. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/480. URL <https://doi.org/10.24963/ijcai.2024/480>. Main Track.
- Harold W. Kuhn and Albert William Tucker (eds.). *Contributions to the Theory of Games, Volume I*. Princeton University Press, Princeton, 1951. ISBN 9781400881727. doi: 10.1515/9781400881727. URL <https://doi.org/10.1515/9781400881727>.
- Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3323felle9595c09af38fe67567a9394-Paper.pdf.
- Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Paul Muller, Timo Ewalds, Ryan Faulkner, János Kramár, Bart De Vylder, Brennan Saeta, James Bradbury, David Ding, Sebastian Borgeaud, Matthew Lai, Julian Schrittwieser, Thomas Anthony, Edward Hughes, Ivo Danihelka, and Jonah Ryan-Davis. OpenSpiel: A framework for reinforcement learning in games. *CoRR*, abs/1908.09453, 2019. URL <http://arxiv.org/abs/1908.09453>.
- Boning Li, Zhixuan Fang, and Longbo Huang. RI-cfr: improving action abstraction for imperfect information extensive-form games with reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org, 2024.
- V. Lisý and Michael Bowling. Equilibrium approximation quality of current no-limit poker bots. *The AAAI-17 Workshop on Computer Poker and Imperfect Information Games*, WS-17-06, 2017. URL <https://api.semanticscholar.org/CorpusID:6206144>.
- Weiming Liu, Haobo Fu, Qiang Fu, and Yang Wei. Opponent-limited online search for imperfect information games. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 21567–21585. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/liu23k.html>.
- Wataru Masaka, Mitsuki Sakamoto, Kenshi Abe, Kaito Ariu, Tuomas Sandholm, and Atsushi Iwasaki. The power of perturbation under sampling in solving extensive-form games. *arXiv preprint arXiv:2501.16600*, 2025.
- David Milec, Ondřej Kubíček, and Viliam Lisý. Continual depth-limited responses for computing counter-strategies in sequential games. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS ’24*, pp. 2393–2395, Richland, SC, 2024. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400704864.
- David Milec, Vojtěch Kovařík, and Viliam Lisý. Adapting beyond the depth limit: Counter strategies in large imperfect information games, 2025. URL <https://arxiv.org/abs/2501.10464>.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017. doi: 10.1126/science.aam6960. URL <https://www.science.org/doi/abs/10.1126/science.aam6960>.

- John F. Nash. Equilibrium points in n -person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950. doi: 10.1073/pnas.36.1.48. URL <https://www.pnas.org/doi/abs/10.1073/pnas.36.1.48>.
- Julien Perolat, Remi Munos, Jean-Baptiste Lespiau, Shayegan Omidshafiei, Mark Rowland, Pedro Ortega, Neil Burch, Thomas Anthony, David Balduzzi, Bart De Vylder, et al. From poincaré recurrence to convergence in imperfect information games: Finding equilibrium via regularization. In *International Conference on Machine Learning*, pp. 8525–8535. PMLR, 2021.
- Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.
- Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003. ISBN 0137903952.
- Martin Schmid, Neil Burch, Marc Lanctot, Matej Moravcik, Rudolf Kadlec, and Michael Bowling. Variance reduction in monte carlo counterfactual regret minimization (vr-mccfr) for extensive form games using baselines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 2157–2164, 2019.
- Martin Schmid, Matej Moravčík, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, G. Zacharias Holland, Elnaz Davoodi, Alden Christianson, and Michael Bowling. Student of games: A unified learning algorithm for both perfect and imperfect information games. *Science Advances*, 9(46):eadg3256, 2023. doi: 10.1126/sciadv.adg3256. URL <https://www.science.org/doi/abs/10.1126/sciadv.adg3256>.
- Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. ISSN 0028-0836. doi: 10.1038/s41586-020-03051-4.
- John Schultz, Jakub Adamek, Matej Jusup, Marc Lanctot, Michael Kaisers, Sarah Perrin, Daniel Hennes, Jeremy Shar, Cannada Lewis, Anian Ruoss, Tom Zahavy, Petar Veličkovic, Laurel Prince, Satinder Singh, Eric Malmi, and Nenad Tomašev. Mastering board games by external and internal planning with language models, 2025. URL <https://arxiv.org/abs/2412.12119>.
- David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
- David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404. URL <https://www.science.org/doi/abs/10.1126/science.aar6404>.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.
- Samuel Sokota, Ryan D’Orazio, Chun Kai Ling, David J. Wu, J. Zico Kolter, and Noam Brown. Abstracting imperfect information away from two-player zero-sum games. In *International Conference on Machine Learning (ICML)*, 2023.
- Eric Steinberger, Adam Lerer, and Noam Brown. Dream: Deep regret minimization with advantage baselines and model-free learning. *arXiv preprint arXiv:2006.10410*, 2020.
- Oskari Tammelin. Solving large imperfect information games using cfr+. *arXiv preprint arXiv:1407.5042*, 2014.
- Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

- Jiří Čermák, Viliam Lisý, and Branislav Bošanský. Automated construction of bounded-loss imperfect-recall abstractions in extensive-form games. *Artificial Intelligence*, 282:103248, 2020. ISSN 0004-3702. doi: <https://doi.org/10.1016/j.artint.2020.103248>. URL <https://www.sciencedirect.com/science/article/pii/S0004370220300126>.
- Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.
- Tony Tong Wang, Adam Gleave, Tom Tseng, Kellin Pelrine, Nora Belrose, Joseph Miller, Michael D Dennis, Yawen Duan, Viktor Pogrebniak, Sergey Levine, et al. Adversarial policies beat superhuman go ais. In *International Conference on Machine Learning*, pp. 35655–35739. PMLR, 2023.
- Brian Hu Zhang and Tuomas Sandholm. Subgame solving without common knowledge. In *Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS ’21*, Red Hook, NY, USA, 2021. ISBN 9781713845393.
- Brian Hu Zhang and Tuomas Sandholm. General search techniques without common knowledge for imperfect-information games, and application to superhuman fog of war chess. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026. URL <https://arxiv.org/abs/2506.01242>. Early version on arXiv, 2025.
- Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In J. Platt, D. Koller, Y. Singer, and S. Roweis (eds.), *Advances in Neural Information Processing Systems*, volume 20, 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/08d98638c6fcd194a4b1e6992063e944-Paper.pdf.

A EXPERIMENTAL DETAILS

Our implementation of LAMIR was done in Python 3.12.3 (Van Rossum & Drake, 2009) with libraries from JAX ecosystem (Bradbury et al., 2018; DeepMind et al., 2020; Heek et al., 2024) for automatic differentiation and GPU acceleration. Our implementation of RNAD is derived out of the implementation in OpenSpiel (Lancot et al., 2019). All experiments were run in cluster with several GPUs Nvidia Tesla A100 and CPUs AMD EPYC 7543 with 1TB memory. Each experiment always used only one CPU core and one GPU at most. The training used at most 8GBs of memory per training. Gameplay evaluation and exploitability computation took at most 64GBs of memory. The hyperparameters used are in Tables 2 and 3

A.1 SOFT CLUSTERING CHANGES

When we have used the soft clustering as described in Section 4. However, we have encountered that in some instances when the abstraction limit L was greater than the amount of information sets in a public state, each cluster center. This caused a problem with the dynamics network, which sometimes mapped to two different continuations and therefore revealed some information to the player. Since some information sets are sampled more often than others, it may happen that the soft clustering pulls several clusters more to this sample than to others. To avoid these problems, we have introduced three changes to the clustering loss. First, we added an additional loss $\mathcal{L}_\theta^{\text{Rep}}$ that repels the clusters that are closer than d_r . Second, we changed \mathcal{L}_θ^A so that if some cluster is closer than d_h , we changed the soft clustering to hard, e.g. we move only the closest cluster, for that particular sample. Third, we added Gaussian noise to each sampled point with mean 0 and scale $\sigma = 0.02$.

$$\mathcal{L}_\theta^{\text{Rep}} = \sum_t \sum_{i \in \mathcal{N}} \sum_{\bar{s}_i^t \in \Lambda_{i,\theta}(s_0^t)} \sum_{\bar{s}_i^{t'} \in \Lambda_{i,\theta}(s_0^t)} \max\{0, d_r - \|\kappa_\theta(\bar{s}_i^t) - \kappa_\theta(\bar{s}_i^{t'})\|^2\} \quad (7)$$

A.2 PUBLIC STATE DECODER

With the abstraction loss \mathcal{L}_θ^A as we defined it in Section 4, it may happen that two different public states will produce the same abstract information set. This may not cause any issues, but we decided to avoid it by training a decoder $\Lambda_i^{-1} : \bar{\mathcal{S}}_i \rightarrow \mathcal{S}_0$ with loss of L2, which is also propagated through Λ_i . As a result in each \bar{s}_i is embedded the information to which public state it belongs. In other words, it also means that information about the public state is also used within κ , but since it is the same for each information set with the same public state, it does not affect the clustering.

A.3 OFF-POLICY REGULARIZED NASH DYNAMICS

The Regularized Nash Dynamics (RNAD) algorithm, which builds upon the Follow the Regularized Leader paradigm, typically updates a player’s strategy based on an advantage function (Perolat et al., 2021). In the successful Stratego implementation (DeepNash), the authors adapted this by training an information set value function, which returns a scalar value for an information set. This function approximates the expected game outcome if all players follow the current network strategy in the whole game (both before and after this decision point). This value was then used to derive an advantage function for policy updates, maintaining convergence guarantees within their on-policy training framework.

A strictly on-policy approach may suffer from insufficient exploration, potentially leaving some parts of the game state unvisited. As LAMIR aims to find an abstraction in each part of the game, we employed an off-policy sampling strategy. Specifically, we used ϵ -on-policy sampling: at each decision point, an action was chosen uniformly at random with probability ϵ , or sampled from the current network strategy with probability $1 - \epsilon$. This ensures that all parts of the game can be visited with non-zero probability.

However, this off-policy sampling introduces a challenge: a value function trained naively would estimate the value of the ϵ -on-policy sampling strategy, rather than the target network strategy. To obtain an unbiased estimator of the network strategy we trained a history value function. This

value function was trained using targets derived from the V-trace off-policy estimator (Espeholt et al., 2018), which corrects for the discrepancy between the sampling policy and the target policy. The advantage for taking an action at information set was then computed similarly to how RNaD derives advantages from an information set value function (Schmid et al., 2019; Perolat et al., 2022). Specifically, the advantage relies on counterfactual values - the expected outcome if a specific action is taken and then the network strategy is followed, weighted by the probability opponent have played to this decision point. To ensure unbiased estimates of these counterfactual values under our off-policy sampling scheme, the estimation of this value uses a importance sampling correction for each opponent decision preceding this value (Schmid et al., 2019; Masaka et al., 2025).

A.4 DYNAMICS NETWORK

In our experiments, we worked with games exhibiting a specific property: the public observation (and thus the public state) at the next step depends only on the previous public state and the joint action taken by all players. In other words, the transition of the public state is independent of the private information distinguishing different information sets within the same current public state.

We leveraged this property to refine our dynamics network (Υ_θ). Instead of directly predicting the subsequent abstract information set representation for each player, the modified dynamics function performs a two-stage prediction:

1. Predicts the next public state identifier (s'_0) within the original game.
2. For each player i , it predicts a probability distribution over the L abstract information set within that predicted next public state s'_0 .

To determine the actual subsequent abstract information set representation for player i , we first use public state representation function $\Lambda_{i,\theta}(s'_0)$ to get the L abstract information sets. Then we select the \bar{s}_i , which corresponds to the highest probability from the dynamics network (argmax selection). This modification effectively separates the prediction of the public state transition from the prediction of the players' abstracted private states within that future public context. It can simplify the learning task for the dynamics network when the underlying game structure supports this decomposition.

B ADDITIONAL EXPERIMENTAL DETAILS

B.1 TABULAR K-MEANS

In this experiment, we evaluate abstraction performance separately to show that tabular clustering of the real information sets in each public state to a limited number of abstract ones leads to graceful degradation of the quality of the strategies. As a clustering property κ , we have used either legal action or strategy computed by 4,000 iterations of Counterfactual Regret Minimization. We performed the tabular K-means as a clustering algorithm and then we constructed the original game tree, while changing the information structure to use the abstraction. This new abstract game was solved using CFR and then the exploitability of this final strategy was computed in the original game. The results are displayed in Figure 3.

The maximal amount of information sets per player in public state is 7, 30 and 168 for Goofspiel 4, 5 and 6 respectively. Increasing the abstraction limit beyond 10 in Goofspiel 5 and 20 in Goofspiel 6 increases the performance only slightly, so it suggests that this size is sufficient in those games. We have used this knowledge in our large experiments and we set the abstraction limit $L = 20$. Also, it is important to note that legal actions do not provide sufficient information to create an optimal abstraction in larger Goofspiels. However, in large games, as tested in Section 6.2, it still outperformed the RNaDs strategy as κ .

Computational resources: For each value of K in the experiment, we have used a single core of AMD EPYC 7543 for all 10 seeds. For each K in Goofspiel 4, the computation took less than a minute, in Goofspiel 5 it took less than hour and in Goofspiel 6 it was less than 8 hours. However, all of those were ran in parallel. Approximately the resulting computational time was at most 1375 CPU hours.

Group	Parameter	Value
RNaD	Regularization η	0.2
	NeuRD threshold β	2
	NeuRD clipping c	10000
	V-trace clipping c	1.0
	V-trace clipping ρ	∞
	Target network update γ_a	10^{-3}
	Off-policy sampling ϵ	0.5
	Amount of transformations T	10
Training	Learning rate	$3 \cdot 10^{-4}$
	Architecture	MLP
	Activation functions	ReLU
	Optimizer	ADAM (ADAMW for κ_θ)
	Adam decay rate β_1	0.9 (0 for π_θ)
	Adam decay rate β_2	0.999
	Weight decay of ADAMW	10^{-5}
	Gradient clipping	100 (1 for $\kappa_\theta, \tau_\theta$)
Abstractions	Sample noise scale σ	0.02
Soft clustering	Softmax temperature γ	1
	Hard clustering threshold d_h	0.3
	Closest cluster distance d_r	0.5
Look-ahead reasoning	CFR+ iterations	1000
	Depth limit	1

Table 2: Common hyperparameters

Parameter	Exploitability	Leduc	Head-to-head
<i>Architecture Parameters</i>			
π_θ MLP layers	256, 256	256, 256	512, 512
v_θ MLP layers	512, 512	512, 512	4096, 4096
Λ_θ^I MLP layers	256, 256	256, 256	512, 512
Λ_θ MLP layers	1024, 1024	256, 256	4096, 4096
Γ_θ MLP layers	128, 128	128, 128	512, 512
Υ_θ MLP layers	256, 256	256, 256	2048, 2048
κ_θ MLP layers	256, 256	256, 256	1024, 1024
τ_θ MLP layers	512, 512	512, 512	4096, 4096
Λ_θ^{-1} MLP layers	128, 128	128, 128	512, 512
<i>Training Parameters</i>			
Batch size	64	128	128
Episodes	10^5	10^5	$3 \cdot 10^6$
<i>Other Parameters</i>			
Abstract information set dimension	64	64	256
Regularization policy change each	1000	1000	20000

Table 3: Specific hyperparameters for each experiment.

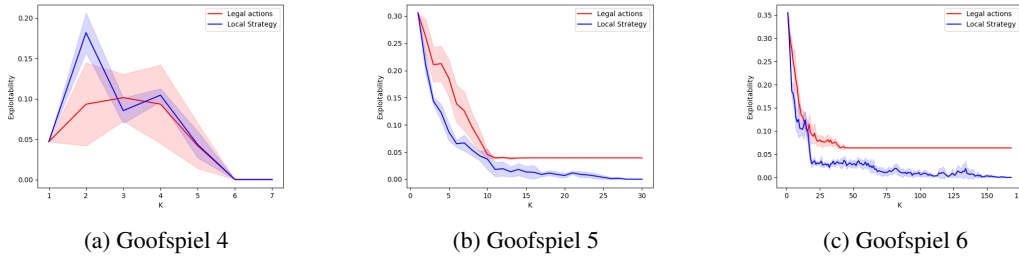


Figure 3: Exploitability in different games when performing tabular K-means to get abstraction using different property functions κ .

B.2 EXPLOITABILITY IN SMALLER GAMES

We provide additional experimental results for Section 6.1, which were omitted from the main body of the paper due to limited space. We show a performance of different κ from those in the main body. Besides that we evaluate different depth-limits in the look-ahead reasoning. Lastly we show ablation studies to computing exploitability from the same training runs.

Figure 4 contains the exploitability of the abstraction in three different settings. First, we construct the original game tree, and then we map each info set with public state representation function Λ_θ and information set representation Λ_θ^I to the abstract one. Then, we solved this changed game with CFR+ and computed the exploitability of this new strategy in the original one. The Second was to use a dynamics network to construct the whole game, which is equivalent to using ∞ depth-limit and then using CFR+ in this game. This setting does not use a value function, so it evaluates the quality of the abstraction Λ_θ and the dynamics Υ_θ . Third, we used the LAMIR, which is the same setting as in Section 4.

In Goofspiel 5, the experiments have shown that using just the RNAD strategy π_θ , which was trained for the v_θ for clustering κ performs worse than the RNAD itself. This is partially due to changes to the clustering to avoid collapsing clusters together, which we discussed in Appendix A.1 and also because the strategy itself was not stationary during the training, as it changed from the RNAD dynamics.

Using just the abstraction and then mapping it to the original game shows that with sufficient capacity, the learned abstraction mirrors the game’s underlying structure, as evidenced by $L = 30$ in Goofspiel. In Oshi-Zumo, even with $L \geq 5$ it is enough to model the game perfectly when using the legal actions for clustering κ . This suggests that even if Oshi-Zumo is quite a large game, as the largest public state contains 625 information sets, it is not important to distinguish between them.

Constructing the whole game tree sometimes produces worse results than using the continual resolving. This mainly happens if the abstraction is worse than the value function, which may fix some mistakes that the poor abstraction caused only further in the game. This mainly happened in Oshi-Zumo. In goofspiel it occurred only when using only the action history as κ .

Figure 5 suggests that increasing the depth-limit can increase exploitability, if the abstraction is small. It has been shown in prior works on depth-limited solving, that increasing the depth-limit improves the performance (Burch et al., 2014; Kubíček et al., 2024). However, since the game tree constructed by LAMIR is not the real game tree, but only the abstracted version, then the solution depends on the abstraction itself. Again, well trained value function with poor abstraction may prevent some of the mistakes that would happen further in the game. Still, using larger depth-limit with larger abstraction size L manages to either be on-par with the depth-limit 1 or improves upon it.

In our experiments, we have trained LAMIR on top of RNAD, so the training time of LAMIR was higher than that of RNAD. We conducted an experiment to demonstrate the time difference between the two training times, where we ran 1000 iterations of RNAD and LAMIR with different L , but with the same hyperparameters in Goofspiel 5. The results are shown in Table 4. The LAMIR trained 10 separate networks. Yet the training time did not increase tenfold, as the RNAD requires computing

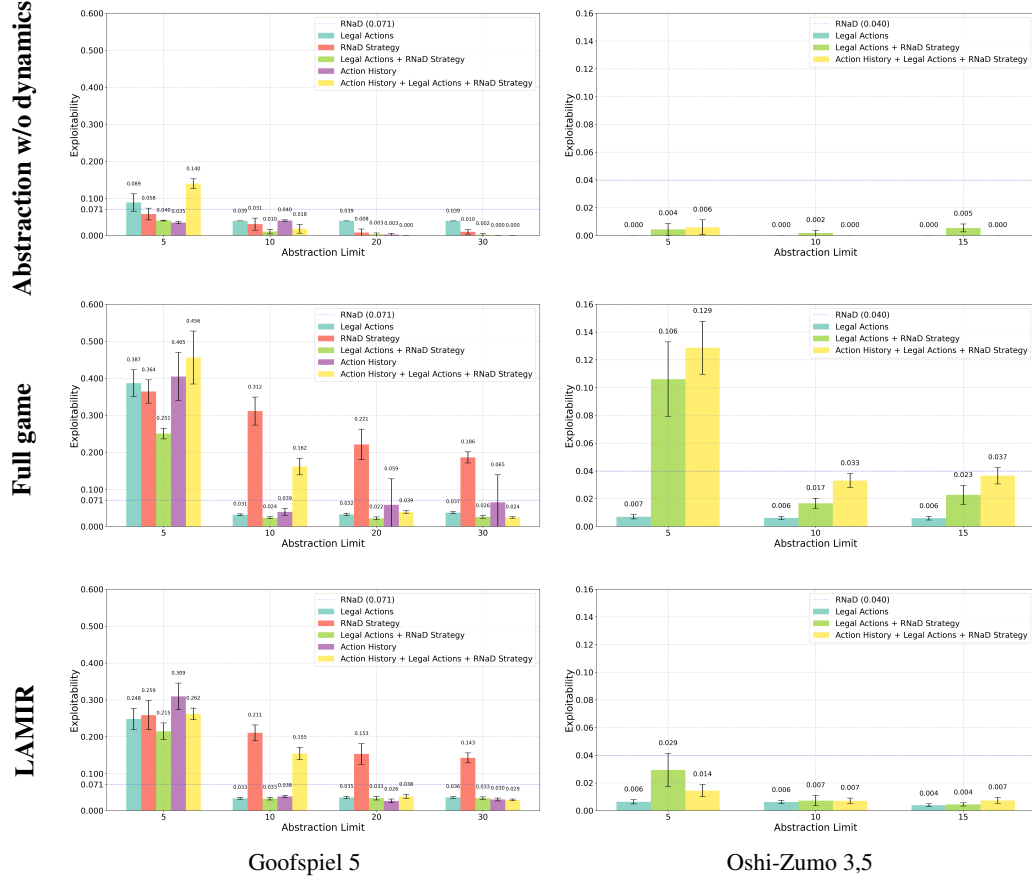


Figure 4: Exploitability of different LAMIR runs with different κ either by mapping the information abstraction onto the original game tree, or by constructing the whole game tree from the dynamics network or by using the LAMIR with depth-limit 1

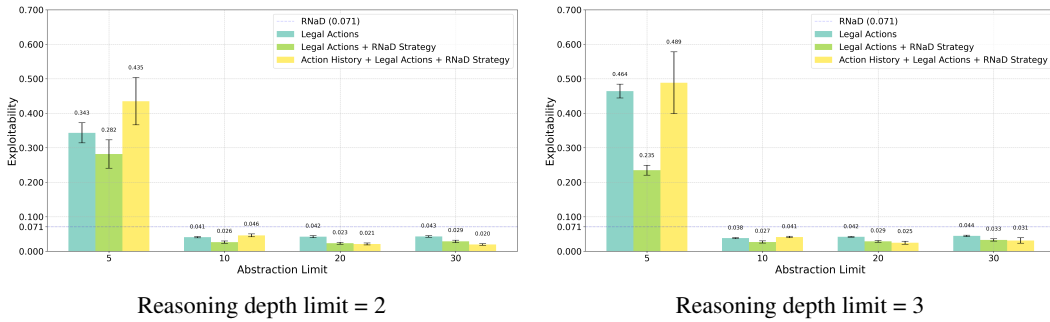


Figure 5: Exploitability of different LAMIR runs based on the look-ahead reasoning depth-limit

Algorithm	RNaD	$L = 5$	$L = 10$	$L = 15$	$L = 20$	$L = 25$	$L = 30$
Time	48s	98s	99s	104s	110s	117s	132s

Table 4: Time of 1000 training steps of RNaD and LAMIR with varying abstraction size L in Goofspiel 5

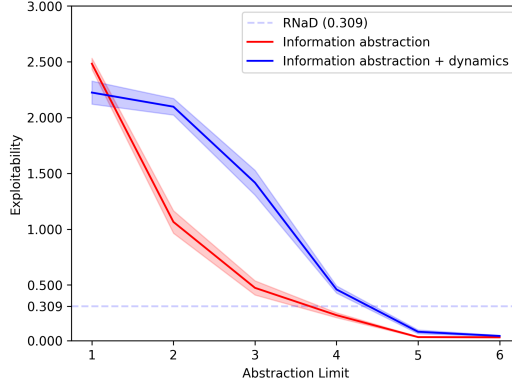


Figure 6: Exploitability in Leduc Hold'em either by constructing the subgame from the rules of the game, or from the dynamics network

lambda returns, which sequentially traverses the trajectory from back to front. On the other hand, the targets of LAMIR can be computed in parallel. We have used JAX with JIT, which optimizes parallel computations, and as a result, even with $L = 30$, the LAMIR is only at most 3 times slower.

Computational resources: Training of each seed, abstraction limit L , clustering property κ and game was ran on a single GPU Nvidia Tesla A100 for less than an hour. Then each checkpoint made during this training was evaluated sequentially on a single CPU AMD EPYC 7543 for each ablation study. So in total this experiment cost 390 GPU hours and 6240 CPU hours.

B.3 LEDUC HOLD'EM

We have also evaluated LAMIR in a small version of Poker, Leduc Hold'em, which is a popular benchmark. LAMIR cannot model the chance nodes and is not intended to be used for such games. However, with some domain-specific workarounds, we were able to use LAMIR even for Leduc. These workarounds were only used in the test part to unroll the chance nodes out of game rules of Leduc. We have used the same training setting as in Section 6.1 with 100,000 episodes and evaluated the last 21 checkpoints each 1000 episodes. We evaluated the same trained models in two settings. In both we solved separately part of the game before dealing a public card, by using a value function at the depth-limit and after dealing a public card. In one setting, we have created the subgame from the rules of the game and only mapped the information structure from the abstraction; in the second, we only used the rules of the game to construct states after the chance nodes and then used the dynamics network to construct the rest of the subgames. The results are displayed in Figure 6.

In Leduc Hold'em, each public state has 6 information sets for each player. So for $L \geq 6$, LAMIR learns the underlying game. However, even with $L = 5$, it is still capable of outperforming RNaD. Decreasing the size further degrades the performance, even if the abstraction $L = 3$ should be sufficient due to the invariance in the card suits. This suggests that improving κ and the clustering may produce better results.

Computational resources: We have again used for a training of a single seed with abstraction limit just a single GPU Nvidia Tesla A100 for at most hour. Each checkpoint was then evaluated on a single CPU AMD EPYC 7543 for each of those 2 ablation studies in less than 2 hours. In total, the computational cost was 60 GPU hours and 120 CPU hours.

Algorithm	II Goofspiel 10	II Goofspiel 13	II Goofspiel 15
LAMIR $\kappa = \text{Legal actions}$	$47.07 \pm 1.27 \%$	$54.14 \pm 1.16 \%$	$62.85 \pm 1.12 \%$

Table 5: Average win rate with 2-sigma error bars of LAMIR against RNaD in different games under the same training time.

B.4 HEAD-TO-HEAD IN LARGER GAMES

In Section 6.2, we have compared LAMIR’s win-rate over RNaD in head-to-head matches under the same number of training steps. However, since LAMIR’s training step takes more time than that of RNaD, we have also compared an earlier checkpoint of LAMIR, which was trained roughly the same time as $3 \cdot 10^6$ training steps of RNaD, against the same trained network as in Section 6.2. We show the average win rates in Table 5. As expected, the win-rate of LAMIR decreased, as it had fewer training steps. Still, it manages to outperform RNaD in the larger instances.

Computational resources: The training of each seed and clustering function κ for both RNaD and LAMIR was done in parallel each on a single GPU Nvidia Tesla A100. The training took 24 hours. Heads-to-heads was done performed in parallel for each final saved model from training. The evaluation of each pair took 192 hours on a CPU AMD EPYC 7543. The total computational cost was 864 GPU hours and 20736 CPU hours.

C GAME RULES

In this section, we provide the game rules for all the games used in the main body of the paper. Moreover, in Table 6, we provide size estimations of all the game instances used in experiments. The amount of histories $|\mathcal{H}|$ in Goofspiel N was computed as $|\mathcal{H}| = \sum_{n=0}^{N-1} \frac{N!^2}{(N-n)!^2}$. The number of information sets $|\mathcal{S}_1|$ was estimated using a dynamic program, which computed the number of information sets after $n - 1$ actions, considering that each action can produce two different observations. This is a lower bound, as in the original game, some actions can produce three different observations (win, draw, loss). Similarly, the number of public states $|\mathcal{S}_0|$ was estimated using dynamic programming, where each round can produce $N + 2$ unique public observations, where N corresponds to drawing a given card, and the other two consider loss or win. The largest public state in the game occurs after an alternating sequence of wins and losses. So we computed $\max_{s_0} |\mathcal{H}(s_0)|$ by finding the number of sequences consistent with those observations using dynamic programming. The amount of information sets in the largest public state was simply estimated as $\max_{s_0} |\mathcal{S}_1(s_0)| = \sqrt{\max_{s_0} |\mathcal{H}(s_0)|}$. Similarly to public states, we assumed the largest information set to consist of an alternating sequence of losses and wins after playing either the highest or lowest available action. We again computed this size $\max_{s_1} |\mathcal{H}(s_1)|$ using dynamic programming.

Game	Oshi zumo 3,5	Goofspiel 5	Goofspiel 10	Goofspiel 13	Goofspiel 15
$ \mathcal{H} $	1250837	4026	$\approx 1.6 \cdot 10^{13}$	$\approx 4.9 \cdot 10^{19}$	$\approx 1.7 \cdot 10^{24}$
$ \mathcal{S}_1 $	54761	1062	$\approx 5.0 \cdot 10^9$	$\approx 1.9 \cdot 10^{14}$	$\approx 3.3 \cdot 10^{17}$
$ \mathcal{S}_0 $	3509	300	$\approx 6.0 \cdot 10^7$	$\approx 2.5 \cdot 10^{11}$	$\approx 9.6 \cdot 10^{13}$
$\max_{s_0} \mathcal{H}(s_0) $	50072	290	$\approx 2.0 \cdot 10^8$	$\approx 4.1 \cdot 10^{13}$	$\approx 3.1 \cdot 10^{17}$
$\max_{s_0} \mathcal{S}_1(s_0) $	625	30	$\approx 1.4 \cdot 10^4$	$\approx 6.4 \cdot 10^6$	$\approx 5.6 \cdot 10^8$
$\max_{s_1} \mathcal{H}(s_1) $	625	23	$\approx 3.6 \cdot 10^6$	$\approx 4.7 \cdot 10^8$	$\approx 8.7 \cdot 10^{10}$

Table 6: Size estimations of different games used in Section 6.2. The traditional lookahead reasoning techniques are limited by amount of unique histories in the largest public state $\max_{s_0} |\mathcal{H}(s_0)|$

C.1 IMPERFECT INFORMATION GOOFSPIEL

Imperfect information Goofspiel N is a game played by two players, where each player receives cards valued from 1 to N . The dealer has the same cards. Then each turn, the dealer reveals a single card from its deck to both players. Each player then secretly places one of its cards as a bet. The

dealer looks at both cards and gives the points corresponding to the public card to the player that had the highest bid. In case of a draw, the dealer discards its card. The players can only play the same card once. We have used a version where the dealer has predefined order of cards, so it always shows from the highest card to the lowest one.

C.2 IMPERFECT INFORMATION OSHI-ZUMO

Imperfect information Oshi-Zumo K, N is played by two players on a board of size $2K + 1$ with a fighter in the middle of the board. Each player starts with N coins. Then each turn, players secretly place bids from 0 to the amount of coins they are still holding. The player that had the higher bet pushes the fighter closer to the opponents edge. The game ends either when neither of the players has any coins, when the fighter is on the edge of the board (positions 0 and $2K$) or after the maximum number of rounds. We have used N as the number of rounds. The reward of player 1 is then $\mathcal{R}_1 = \frac{P-K}{K}$, where P is the position of the fighter at the end of the game.

C.3 LEDUC HOLD’EM

Leduc Hold’em is a simplified version of Texas Hold’em poker. It is played by two players with a deck of 6 cards in two suits: Spades and Hearts. Each suit has three ranks: Jack, Queen, and King. At the beginning of the game, each player performs a mandatory bet of 1 coin to the pot, and the dealer deals privately one card to each player. The game then proceeds to the first betting round, where players take turns by either folding, calling, or raising. If some player folds, the game immediately ends, and the other player receives all the coins in the pot. If any player calls, it puts as many coins into the pot so that the total amount of coins put in by both players is the same, and the game proceeds to the next round. If a player raises, it puts the same amount of coins and two more coins into the pot as during the call. The players can raise only twice during a single round. After the first round ends, one of the remaining four cards is revealed as public. Then, the players proceed with the betting round. The only difference is that the raise now adds four more coins to the pot. At the end of this betting round, both players reveal the following rule that decides their card and the winner: If a player’s private card matches the rank of the public card (e.g., the player has a King and the public card is a King), they have a pair and win. Otherwise, the player holding a higher value card wins, and the order from the highest rank is King, Queen, Jack. If both players hold the same rank, it is a draw, and they will split the money in the pot.

D PSEUDOCODES AND VISUALIZATIONS

D.1 TRAINING

We provide illustration of LAMIR’s abstraction training process in Figure 7. Each iteration, LAMIR samples a trajectory, the visualization shows single timestep in single iteration. The visualization omits training of the legal actions function, which is straight-forward as it uses the legal actions from the environment as a target.

D.2 LOOK-AHEAD REASONING

In this section, we provide high-level pseudocode in Algorithm 1 that provides details about the usage of individual components within LAMIR in the look-ahead reasoning. We also provide an illustration of LAMIR’s construction of the look-ahead tree in the first two moves in Figure 8. At the beginning of the game, LAMIR constructs the subgame root using the $\Lambda_{i,\theta}^I$ and $\Lambda_{i,\theta}$. In each subsequent move, the root is taken from the previous subgame. As a result, every move beyond the first one uses the same procedure shown in the left part of Figure 8

E LARGE LANGUAGE MODELS USAGE

During the writing of this paper, the Large Language Models (LLMs) were used to refine the writing, both by polishing the text and to better communicate the main contributions of the paper. Specifically, we have used Gemini 2.0 Flash, ChatGPT, and Claude Sonnet 4.

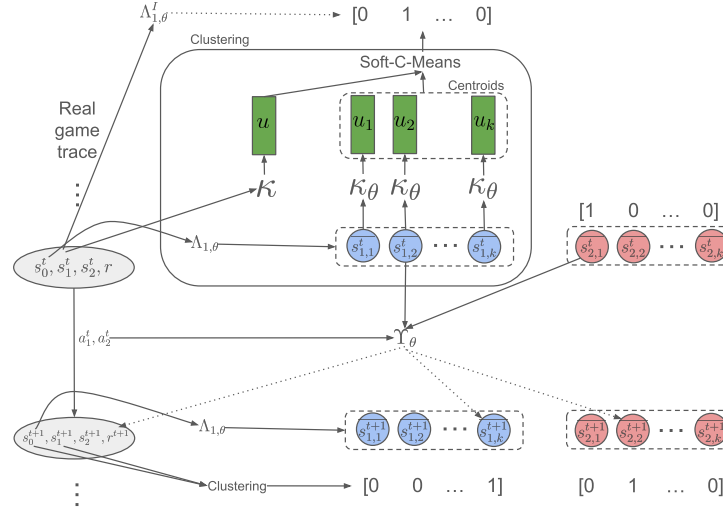


Figure 7: Illustration of computing a targets for abstraction networks of player 1 and dynamics network in a single trajectory step. At timestep t , the $\Lambda_{1,\theta}$ maps s_0^t to all possible abstract information sets $\overline{s_{1,j}^t}$, where $j \in [1, \dots, k]$. Each of those is through κ_θ mapped to the clustering space. Similarly the real information set s_1^t is mapped to the same space. Then the soft clustering loss is computed for $\Lambda_{1,\theta}, \kappa_\theta$. The closest representative $\overline{s_{1,2}^t}$ is used as a target for $\Lambda_{1,\theta}$. Same procedure happens for player 2, which finds closest representative $\overline{s_{2,1}^t}$. Those two along with taken actions a_1^t, a_2^t are mapped through dynamics function, which uses observed reward r^{t+1} and the closest representatives from the next timestep $\overline{s_{1,k}^{t+1}}, \overline{s_{2,2}^{t+1}}$ as a target.

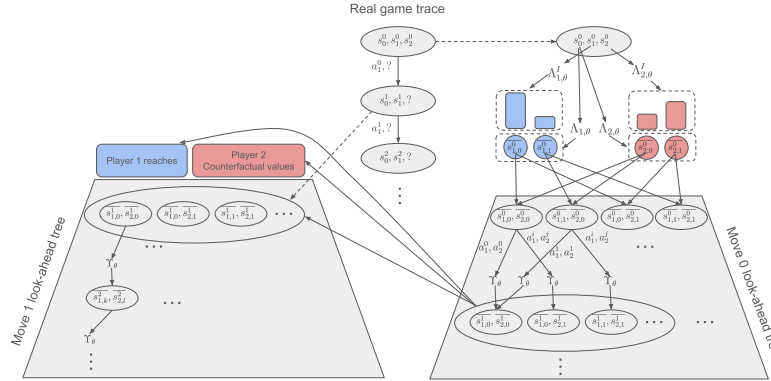


Figure 8: Illustration of the first two moves of LAMIR. At the beginning of the game, Player 1 knows the initial history h^{INIT} with corresponding public state and information set representations s_0^0, s_1^0, s_2^0 . It uses $\Lambda_{i,\theta}$ of each player i to get all clustered information sets (shown as circles) and $\Lambda_{i,\theta}$ to get the probability distribution over those abstractions. These are then used to create the root of the subgame, along with their corresponding reaches. For each joint abstract information set $(\overline{s_1}, \overline{s_2})$ and corresponding legal actions $\Gamma_\theta(\overline{s_1}), \Gamma_\theta(\overline{s_2})$ the Υ_θ generates next joint abstract information set up until the depth-limit. This subgame is solved using the CFR, which results in strategy for each abstract information set. The player then samples an action a_1^0 from the strategy of the most likely abstraction to transition the game into a new state, where it only observes its information set s_1^1 and the public state s_0^1 . All of the consistent histories with s_0^1 are used to create the root of a gadget subgame, where reaches of player 1 and counterfactual values of player 2 are reused from the previous game tree. The rest of the subgame is again constructed using Υ_θ .

Algorithm 1: LAMIR Gameplay

Input: depth limit d , trained networks $\Lambda_\theta, \Lambda_\theta^I, \Upsilon_\theta, \Gamma_\theta, v_\theta$, initial history h^{INIT} , player i

```

1  $s_0, s_1, s_2 \leftarrow \text{GenerateInitialInformation}(h^{\text{INIT}})$ ; /* Initial history is assumed
   to be observable */
2  $R \leftarrow \text{GenerateInitialRoot}(\Lambda_\theta, \Lambda_\theta^I, s_0, s_1, s_2)$ ;
3  $S \leftarrow \text{GenerateSubgame}(\Upsilon_\theta, \Gamma_\theta, v_\theta, R)$ ;
4 while NotTerminal( $s_0$ ) do
5    $\pi_i \leftarrow \text{SolveSubgame}(S)$ ;
6    $\bar{s}_i \leftarrow \text{MapRealInfoSetToAbstraction}(\Lambda_\theta, \Lambda_\theta^I, s_0, s_i)$ ;
7    $a_i \leftarrow \text{SampleAction}(\pi_i, \bar{s}_i)$ ;
8    $s_0, s_i \leftarrow \text{ActInRealGame}(a_i)$ ;
9    $R, P_i, v_{-i} \leftarrow \text{ExtractNewRootWithStatistics}(S, s_0)$ ;
10   $S \leftarrow \text{GenerateGadgetSubgame}(\Upsilon_\theta, \Gamma_\theta, v_\theta, R, P_i, v_{-i})$ ;
11 end

```

We have also used the LLM coding assistant Cursor with Claude Sonnet 3.5 and later Claude Sonnet 4 as the underlying model for the experimental evaluation.

The authors always double-checked all of the LLM outputs to ensure their correctness.