Applying Deep Reinforcement Learning to the Card Game Cego

Philipp Oeschger
Faculty of Computer Science
Furtwangen University
Furtwangen, Germany
philipp.oeschger@hs-furtwangen.de

***Abstract*—Reinforcement learning research is sparse in the German card game of Cego. This work applies deep reinforcement learning to sub-problems of the game. For this task, the game environment is implemented in the framework RLCard. A suitable action encoding and an encoding approach for game state observations are provided. The number of possible states is restricted by the implementation of heuristic rules. Benchmarks are defined for evaluating sub-problems. Models for a specific sub-game are trained with the algorithms deep Q-network (DQN), neural fictitious self-play (NFSP), and deep Monte Carlo (DMC). In direct comparison, DMC achieves the highest performance. When training models with DMC for other sub-problems, all models beat their benchmarks.**

***Index Terms*—artificial intelligence, card games, Cego, machine learning, reinforcement learning, RLCard**

I. Introduction

Cego is a card game that is mainly played in and around the German region *Baden* [1]. Despite its unique properties, Cego has sparsely been addressed in research on reinforcement learning (RL). Due to Cego's unique characteristics, this work aims to introduce deep RL methods to the game. First of all, Cego has the game mechanic of having a large blind card set, which has major ramifications on the outcome of the game [1], [2]. This mechanic results in Cego having a large randomness factor from a player's perspective. Secondly, Cego can be split into various sub-games, with sometimes variations within a single sub-game [2], [3]. These aspects make Cego an interesting test bed for game theory as well as real-life problems.

RL has seen a resurgence in recent years. In 2016, AlphaGo [4] achieved the goal of beating the European champion in the challenging game Go by a significant margin. The following project AlphaZero [5] has since then advanced the level of play further and has been implemented for the games Chess and Shogi [6]. These games all share the property of being games with complete information states [7]. However, card games, such as Cego, are games where the information on the current and previous game states is incomplete. This leads to card games typically having large game spaces and, therefore, many possible game states [8]. Due to this level of complexity, card games are difficult to solve.

RLCard [9] is an RL framework that focuses on the complexity of card games. It implements state-of-the-art RL algorithms to compare them over a standardized environment interface. The provided algorithms are mostly based on deep neural networks (DNNs). In other problem domains, DNNs have proven to be effective in solving complex problems [10]. Furthermore, DouZero [11] demonstrated that DNNs can be applied to the Chinese card game DouDizhu to remarkable effect.

This work applies deep RL algorithms to the card game of Cego. The final goal is to train models that effectively maximize the level of play. The game is split into different sub-problems to maximize each problem individually. Restrictions are defined to reduce the complexity of the game. To compare different algorithms in a generalized environment, the game is implemented in RLCard. By simulating scenarios for specific sub-problems beforehand, benchmarks are created. This work compares the algorithms deep Q-network (DQN) [12], neural fictitious self-play (NFSP) [13] and deep Monte Carlo (DMC) [11] with one another on a single sub-problem in order to identify the algorithm that performs best under the specified conditions. Hyperparameter optimization (HPO) is applied to certain algorithms to optimize their performance. After identifying the algorithm with the most successful results, this algorithm is applied to train models for other Cego sub-problems. An evaluation of the sub-game models by using the defined benchmarks as a base of comparison concludes this paper.

II. Related Work

In other imperfect information games, specifically *poker* variants, counterfactual regret minimization (CFR) has been popular [14]. Nevertheless, CFR is computationally expensive because for solving a game, the full game tree has to be traversed. Bowling et al. introduced CFR$^+$ [15], an improvement to the standard CFR algorithms that solved Heads-up limit Texas Hold'em poker. It has to be noted that Cego consists of more players and more cards, making the game tree potentially orders of magnitude larger. Therefore, other methods might be deemed more efficient in learning Cego.

Bridge is a trick-taking game similar to Cego. Ginsberg introduced an AI which uses Monte Carlo (MC) techniques and perfect information states [16]. This result proves that MC techniques can create an AI able to beat professional players. However, the question is whether an AI can achieve the same level of play with imperfect information states.

Another trick-taking game originating from Germany is the card game *Skat*, which also has blind cards as a game component. The mathematical and strategic background of Skat has been well researched [17]–[20].

In 2007, Kupferschmid et al. [21] applied MC simulation and alpha-beta search to Skat. With the addition of move ordering, quasi-symmetry reduction, and adversarial heuristics to improve tree searching algorithms, the final program processes the game-theoretical value of a Skat hand in about 10 milliseconds. Furtak [22] solved perfect game states in Skat for use in an MC-based AI by using search extensions that are symmetry-based. Both previous named research focused on perfect information states. In contrast, this work explores to what degree Cego can be solved without having all the information at hand.

An imperfect information approach for Skat was introduced by Edelkamp [23]. His AI applied expert rules, winning probability aggregation, and fast tree exploration. The final AI beat human players in some sub-games but was disadvantaged in others. A similar approach is feasible for Cego. However, this work deems other approaches as more effective.

The AI DouZero learned the Chinese game of DouDizuh [11]. This game is no trick-taking game but similar to Cego has two playerwise unevenly matched sides. In addition, DouDizuh has the obstacle of having a large action space. Despite this, the introduced DMC algorithm, which combines an MC approach with DNNs, was able to beat any other existing AI in the game. This work compares this algorithm with other deep RL algorithms on the game Cego.

III. Cego

Cego is a card game that is typically played with four players [2]. There are rule sets for different regions. This work focuses on the generalized rule book employed by the *Schwarzwaldmeisterschaft* [24] as well as *Cego Online* [25], the web-based game.

A. Cards

Cego uses Tarock cards [1]. A special card set consisting of the card suits trump, clubs, diamonds, hearts, and spades (see Figure 1) [1], [2]. The deck includes 54 unique cards. The cards can be divided into:

- 22 trump cards,
- Each eight,
  - Clubs cards,
  - Diamonds cards,
  - Hearts cards, and
  - Spades cards.

Among the trump cards, the *Gstieß* [2] is considered the highest card. The other trump cards are numbered. Other suits can be divided into *picture cards* and *empty cards*. Kings (Ger. *König*), queens (Ger. *Dame*), cavaliers (Ger. *Reiter*), and jacks (Ger. *Bube*) are the pictured cards. The empty cards are named by the number of suit symbols they have on them.

Each card has a value associated with it, which is used to determine points. The cards have the following values [25]:

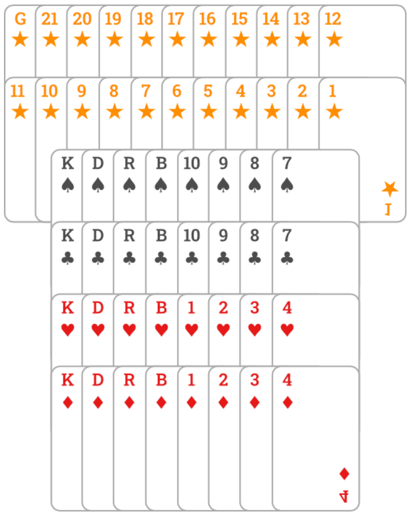

Fig. 1. Overview of Cego cards

- Gstieß, 21-trump, 1-trump, and kings: 4.5 points.
- Queens: 3.5 points.
- Cavaliers: 2.5 points.
- Jacks: 1.5 points.
- All other cards: 0.5 points.

B. Rules

At the start of the game, each player gets dealt 11 cards. The left 10 cards are the blind cards and are laid down in the middle of the table [2].

After the cards have been dealt, it is decided on what sub-game to play [2]. For this, a bidding phase, consisting of multiple rounds, takes place. The bidder tries to win the right to play against the other three players as the *single-player*. When playing as the single-player, the player takes a higher risk of losing [2], [3]. But as compensation, the number of points the player can receive may be higher.

When it has been decided on a sub-game, the players (usually) must win tricks [2], [3]. For that, each player plays a card in counterclockwise order. After every player has played a card, the player who played the highest card wins the trick and receives the points of the trick cards. The first suit that is played in the trick has to be served. When the suit can't be served, a trump card must be played when possible. Any card can be played when the suit cannot be served and there is no trump card at hand. This act is called *schmieren*. Tricks are played until no cards are left or a sub-game-specific winning condition is met.

Each card has a rank which can be viewed in Figure 1 in the reading direction from top left to bottom right descending. Trump cards beat other suits.

C. Sub-Games

Cego consists of the following six sub-games [2], [3]:

- ***Standard-Cego***: The single-player exchanges his hand cards with the blind cards. The discarded cards, called *Legage*, consist of hand cards. The Legage is added to the player's point score. In order to win, the single-player must receive at least *40* out of *79* possible points.

- ***Solo***: The Legage consists of the blind cards and is added to the single-player's point score. Out of *79* possible points, the single-player has to receive at least *40* points.
- ***Ultimo***: The single player must win the last played trick with the card *1-trump* (also called *Geiß*).
- ***Bettel***: The single-player is not allowed to win a trick.
- ***Piccolo***: The single-player must win exactly one trick.
- ***Räuber***: The player that receives the highest number of points loses.

The game mode Standard-Cego is commonly called *Cego* [2], but to minimize confusion between the game name, it is termed differently in the context of this work.

In all but Standard-Cego, the blind cards are discarded and, therefore, out of the game [2], [3].

D. Complexity of Cego

Cegos number of possible starting information states $|I_0|$ can be calculated by multiplying the binomial coefficients of each hand deck as well as the blind cards:

$$\begin{aligned} |I_0| &= \binom{54}{11} \cdot \binom{43}{11} \cdot \binom{32}{11} \cdot \binom{21}{11} \cdot \binom{10}{10} \\ &\approx 2.51 \cdot 10^{34} \end{aligned} \quad (1)$$

This considerably large number of starting states is further increased by the number of ways to call sub-games in the bidding phases and the number of possible card play orders. Therefore, Cego is considered a complex game.

E. Game Theoretic Classification

Cego is a game with imperfect information [7]. Players do not know the cards dealt to other players.

In the context of most sub-games, Cego is a two-player zero-sum game [7]. Despite being played by more than two players, these players can be divided into two directly opposing factions, the single-player and the non-single-players.

Because Cego is a sequential game, it can be described in extensive form [7].

IV. Game Environment Implementation

The game implementation defines four players, of which the first player is considered the single-player (outside of Räuber) and the other three players are ordered counterclockwise in regard to the single-player.

A. Limitations

The bidding phase vastly differs from the sub-games. In addition, the bidding phase-specific actions have huge ripple effects on how the rest of the game is played out and further increase the complexity. Therefore, the bidding phase is excluded from consideration.

All sub-games have varying winning conditions and priorities. To simplify the problem and take away the complexity of having to learn multiple sometimes opposing problems, every sub-game is implemented and trained separately.

Standard-Cego has different bidding levels, which in addition to playing a card add the actions of keeping, exchanging, and discarding cards. This expands the action space by a factor of four. Algorithms such as DQN have problems dealing with enlarged action spaces [11]. Therefore, this work does not consider these additional actions and generalizes the bidding levels of Standard-Cego into one sub-game.

Cego players usually follow specific rules based on heuristics when deciding which sub-game to play [2], [26]. For the implementation of sub-games, the decision was made to use these rules to further restrict the sub-games. This has the advantage of shrinking the number of possible game states. Aside from that, game states that are unlikely to be played in a game of Cego with human players are excluded in advance.

B. Action Encoding

Each action gets assigned an index (see Table I). Actions are cards that can be played. There are a total of 54 actions.

TABLE I
ACTION ENCODING

Actions/ Cards	Indices
Clubs	0 – 7
Spades	8 – 15
Hearts	16 – 23
Diamonds	24 – 31
Trumps	32 – 53

C. State Encoding

For an algorithm to be trainable in RLCard, the information state a player can observe at a certain time step has to be provided in a machine-readable numerical array [9].

Table II represents the state encoding in Cego. The one-dimensional observation array consists of 336 binary entries. Encodings (1)–(6) encode information about the card states. Each of these encodings has a length of 54 where every index indicates the existence (one) or the absence (zero) of a card. Encodings (7)–(9) represent player-specific information, and each has a length of four. Every index is representative of a player. For example, when player 2 started the trick, the encoding of (9) is $[0, 1, 0, 0]$.

TABLE II
INFORMATION STATE ENCODING

No.	Description	Indices
1	hand cards	0 – 53
2	playable cards by other players	54 – 107
3	card that wins the trick	108 – 161
4	first trick card	162 – 215
5	second trick card	216 – 269
6	third trick card	270 – 323
7	same team players	324 – 327
8	current trick winner	328 – 331
9	player that started the trick round	332 – 335

Along with the player's observation, the indices of legal actions at a certain time step also need to be included in the game state. The selection of legal actions is abbreviated from the game rules described in Subsection III-B.

D. Reward Handling

Two reward systems are considered:

- ***Variant 1***: Players receive one point for winning a game and zero points for losing a game.
- ***Variant 2***: The point system of Standard-Cego and Solo is used. Players can receive a maximum of *79* points and a minimum of *0* points.

Variant 2 has the advantage that the range of possible values is broader. Therefore, the reward system has more nuances. The only disadvantage is that this reward system can only be applied to the sub-games Standard-Cego and Solo. For this reason, Variant 2 is used for these two sub-games. Meanwhile, the other sub-games use Variant 1.

Note that for Standard-Cego and Solo, the reward system is designed in a way so that the single-player is maintaining a separate point score. The score of the other three players is mirrored. Considering, the point score started with $[17, 0, 0, 0]$, with the single-player earning 17 points from the Legage. Player 3 received 5 points by winning a trick, so the resulting point score is $[17, 5, 5, 5]$.

For both Variants, it should be noted that RLCard handles rewards sparsely [9]. Therefore, the full reward is always received at the end of the game.

E. High and Low Cards

Some heuristic rules for sub-games rely on the loosely defined concepts of high and low cards. In order to construct a definition, the generalized probability of a card i winning a trick $P(W_i)$ over all trick rounds is approximated over 10^6 simulation games for each card in games with *random agents* (players that select random actions) on the random seed of 12. Based on the approximations, low cards are defined as the biggest possible set of unique cards that, on average, account for less than 20% of all trick wins. The cards left out of this definition are considered the high cards.

Figure 2 illustrates the card win probabilities in descending order and the classification of high cards (violet) and low cards (blue). Note that i has no relation to the action indices in Table I. The 24 high cards account for around 81.72% and the 30 low cards for around 18.28% of all trick wins.

F. Sub-Game Restrictions

For each sub-game, the following restrictions, based on heuristics, are defined:

- ***Cego-Standard***: The single-player has to have at least 15 combined card points in hand [26].
- ***Solo***: The single-player has to have either at least eight trump cards or seven trump cards, of which at least two are ranked higher or equal to *17-trump* and not more than two suits outside of trump [2].
- ***Ultimo***: The single-player has to have at least eight trump cards, of which at least two are ranked higher or equal to *17-trump* and of which one is the card *1-trump*.

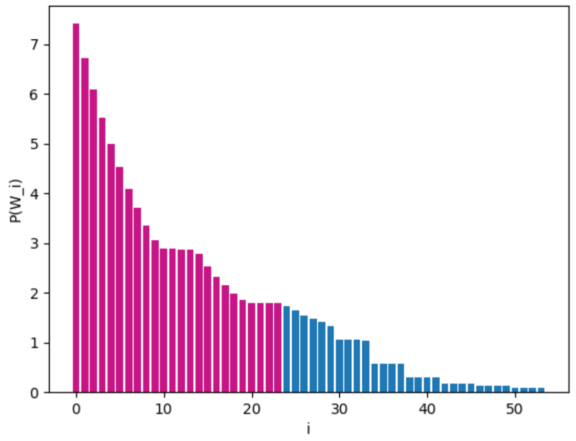

Fig. 2. Distribution of High and Low Cards

- ***Bettel***: All cards of the single-player must be low cards.
- ***Piccolo***: 10 cards of the single-player must be low cards and one card has to be a high card.
- ***Räuber***: No restrictions are made.

V. Defining Benchmarks

This section defines benchmarks for each sub-game considered for evaluation and comparison.

A. Metrics

To evaluate the sub-games, the following metrics are defined:

- ***Win percentage (WP)***: The number of games won by the player divided by the total number of conducted games [27].
- ***Average points per game (APPG)***: The number of total points received by a player divided by the number of games conducted. This average is in the range of $0 \leq x \leq 79$.

B. Method

For each sub-game, 10^6 games are conducted with random agents and the defined game selection rules. Only in Standard-Cego and Solo can the APPG be determined. For all other sub-games, the WP is calculated.

C. Results

Table III visualizes the resulting benchmarks. The on-average winning side is marked in bold. It can be observed that the balance between player sides varies between sub-games. In the context of random agents, the sides in the sub-games Standard-Cego and Räuber are most evenly balanced. The biggest discrepancy can be seen in Ultimo, with a large disadvantage for the single-player.

VI. Comparison of Algorithms

The following three algorithms are considered for the training of a Cego AI.

TABLE III
BENCHMARKS

Player Position	Standard-Cego: *APPG*	Solo: *APPG*	Ultimo: *WP*
1	**40.795**	**48.996**	8.42
2	38.205	30.004	**91.58**
3	38.205	30.004	**91.58**
4	38.205	30.004	**91.58**

Player Position	Bettel: *WP*	Piccolo: *WP*	Räuber: *WP*
1	**73.081**	39.275	73.848
2	26.919	**60.725**	**74.839**
3	26.919	**60.725**	**75.425**
4	26.919	**60.725**	**75.887**

Results rounded to three decimal places.

1) DQN: DQN is an algorithm that combines the concept of Q-Learning [28] with DNNs. In Q-learning, at each step of a game episode, the current state-action pair's value in the Q-Table is adjusted by observing the action, reward, and follow-up state. The Q-value of the current state-action pair is adjusted with the current reward and the discounted highest neighboring Q-value. The following update rule describes this process [29]:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \quad (2)$$

In DQN, a DNN minimizes the difference between the expected target Q-Value and the approximated Q-Value of the DNN with gradient descent and backpropagation [12]. Moves are randomly selected from a memory of experiences. This memory is filled by sampling actions with an ϵ-greedy policy.

In addition, RLCard implements a double DQN [30], where there is a separate online and target network. The target network copies the weights of the online network every r steps [9].

2) NFSP: NFSP is an implementation of fictitious self-play (FSP) [31] utilizing DNNs for the approximation of the best response policy as well as the average response policy [13]. Two separate memories are managed. The first memory, M_{RL}, is filled with player experiences and is used to train the best response network. The second memory M_{SL} relies on reservoir sampling and, consequently, only is filled when the experience follows the best response policy. This memory trains the average response network. Furthermore, NFSP uses anticipatory dynamics, which allows an NFSP agent to select actions from the best response network as well as the average response network. The weighting of action selection between the two networks is regulated by the anticipatory parameter η. This parameter defines the probability at which actions are selected based on the best policy network weights.

RLCard implements the best response network as a DQN [9].

3) DMC: DMC approximates every-visit MC with DNNs [11]. MC Methods are RL methods where returns of full game episodes are sampled, and a value table is updated with the average of (discounted) returns that were received visiting a state [29]. In every-visit MC, multiple visits to a state in the same episodes are considered when calculating the average return.

The loss function of DMC consists of the mean square error (MSE) between the calculated target state value and the DNN approximation of the state value [11]. As an innovation to speed up training, DMC parallelizes the sampling of game episodes. Actions are sampled by multiple actors with an ϵ-greedy policy.

A. *Method of Comparison*

Throughout all phases, the focus is on Standard-Cego. An additional constraint is that all algorithms train a model for the single-player position. This restricts the comparison to a manageable scope.

1) Phase 1: In the first phase, all models are compared against random agents. The models play a total of 50,000 games with the random seeds 12, 17, 20, 30, and 33, resulting in a total of 250,000 games.

2) Phase 2: In the second phase, the models are compared with each other. It starts with 250,000 games split into 50,000 games with the random seeds 12, 17, 20, 30, and 33 in the following constellation:

1) Model A
2) Model B
3) random agent
4) random agent

After this, Model A and Model B swap places, and another 250,000 games are played with the same random seeds. This ensures that both models are an even number of games in both advantaged and disadvantaged positions.

B. *HPO*

In RLCard, DQN and NFSP hyperparameters are lightly tuned [8]. Therefore, the decision was made to optimize the hyperparameters of both DQN and NFSP for better training results. For DMC, the HPO is excluded, and the hyperparameters tuned on DouDizuh [11] are used in the comparison.

Random search is utilized as HPO method [32]. Searching spaces are created for each algorithm. 20 unique hyperparameter sets are sampled for each reward system. The models are trained for 50,000 episodes with random seed 12. Then 1000 games, each with the random seeds 12, 17, 20, 30, and 33, are conducted, and the APPG is compared. In addition, the training slopes created by performing a linear regression are considered in the comparison. Rankings are created for both APPG and training slopes. The model that receives the highest combined average ranking is selected for the training of a final model.

1) DQN: For DQN the *replay memory init size* is fixed to *100*, the *epsilon decay steps* to 50,000, and the *epsilon start* to *1*. Outside these parameters the following searching space is defined:

- *Replay memory size*: 50000, 100000, 200000
- *Update target estimator every*: 1000, 2000, 10000

- *Discount factor*: 0.75, 0.8, 0.95, 0.99
- *Epsilon end*: 0.1, 0.05, 0.01
- *Batch size*: 32, 64
- *Multilayer perceptron (MLP) layers*: [512, 512, 512], [512, 256, 128], [512, 512]
- *Learning rate*: $1 \cdot 10^{-4}$, $5 \cdot 10^{-5}$, $1 \cdot 10^{-5}$, $5 \cdot 10^{-6}$

2) NFSP: Because the DQN hyperparameters were optimized beforehand, these parameters are taken as a base for the in NFSP utilized DQN to counteract the higher number of hyperparameters to optimize in NFSP. The *min buffer size to learn* is fixed to 100. In addition, the following searching space is defined:

- *Hidden layers sizes*: [128, 128], [256, 256], [512, 512], [512, 512, 512]
- *Reservoir buffer capacity*: 20000, 50000, 100000, 200000
- *Anticipatory param*: 0.1, 0.2, 0.25, 0.35, 0.5
- *Batch size*: 128, 64, 32
- *Supervised learning (SL) learning rate*: $1 \cdot 10^{-3}$, $1 \cdot 10^{-4}$, $5 \cdot 10^{-5}$, $1 \cdot 10^{-5}$, $5 \cdot 10^{-6}$

A single hyperparameter set has been determined for DQN and NFSP each. The final hyperparameters are presented in the next subsection.

C. Training

The hyperparameters used for training are displayed in Table IV. All algorithms use the same random seed of 20 for training. DMC was trained for a total of $2.5 * 10^9$ environment frames. DMC trained four agents in parallel, where only the single-player model is considered in the comparison.

DQN and NFSP train in an environment, with the first player being the algorithm and all other players being random agents. DQN and NFSP are each trained in 50,000 episode steps. The *epsilon decay steps* are adjusted to 100,000 to account for the longer training duration. Every 1000 episodes, 1000 evaluation games are conducted against random agents to calculate the current APPG as training progress. After 50,000 episodes, the APPG of the last training progress points of these episodes is determined. When the APPG is lower than the average of the 50,000 episodes before, the training is terminated. DQN terminated after 450,000 episodes and NFSP after 250,000 episodes. For comparison, the model checkpoint of the 50,000 previous episodes is used, where no training regression has occurred.

D. Discussion of Results

1) Phase 1: The results of Phase 1 are displayed in Table V. DMC overall performs the best against random agents, with an average margin of 2.619 points over the next best algorithm, DQN. NFSP comes in last with a margin of -1.334 points to DQN. All models are able to beat the benchmarks.

2) Phase 2: In Table VI, the results of Phase 2 are represented. The highest scores are achieved by DMC against both DQN and NFSP. When combining and calculating the results of all rounds for each algorithm. DMC achieves the highest APPG with 42.105 and the highest WP with 58.001%. The second-best averages are recorded by DQN, with an APPG of 38.777 and a WP of 48.136%. NFSP takes the last place with an APPG of 37.618 and a WP of 43.864%.

In both phases, DMC performs the best in the single-player position.

It should be noted that these results depend on the setting of Standard-Cego. In order to prove the superiority of DMC in the full game of Cego. This experiment would have to be repeated for all sub-games and players. In addition, the choice of random seeds can largely affect the results of training in RL algorithms [33]. Therefore, these results should be viewed in the context of the stated restrictions and random seeds.

VII. Training and Evaluation of Sub-Games

After identifying DMC as the algorithm that performs the best in the conducted comparison, models for all left sub-games are trained with DMC and subsequently evaluated.

A. Training

Because of long training times and DMC converging earlier than expected in the training of Standard-Cego, the number of total environment frames is reduced to $1.5 \cdot 10^9$ for the training of the left sub-games. Other than this adjustment, the hyperparameters are the same as in Table IV.

B. Evaluation Method

After training, a model for every player and sub-game, these models are compared in various tournament setups. Each tournament consists of 50,000 games with the random seeds 31, 43, 67, 78, and 112 each, resulting in a total of 250,000 games per setup. The following setups are considered:

1) Player 1 = DMC model; Other players = random agents
2) Player 2 = DMC model; Other players = random agents
3) Player 1 = DMC model; Other team = 1 DMC model + 2 random agents
4) All players = DMC models

In the game mode Räuber, where the team dynamics are different compared to the other sub-games, only the following three setups are evaluated:

1) Player 1 = DMC model; Other player = random agents
2) Player 1 and 2 = DMC models; Player 3 and 4 = random agents
3) All players = DMC models

C. Discussion of Results

Table VII presents the results of the evaluation. The on-average winning side is marked in bold. As can be perceived, single-player and non-single-player models improve on the specified benchmarks.

In both Standard-Cego and Solo, the single-player model improves on similar levels.

The highest improvement over the benchmarks achieves the single-player model of Ultimo with an increase of more than 1000%. A reason for this high discrepancy between benchmark and evaluation reward could be that there is an underlining logic that is trivial for the AI to learn. The key component

TABLE IV
HYPERPARAMETERS

DQN		NFSP		DMC	
Hyperparameter	*Value*	*Hyperparameter*	*Value*	*Hyperparameter*	*Value*
Replay memory size	100,000	SL hidden layers sizes	[512, 512]	Num actor devices	1
Replay memory init size	100	SL reservoir buffer capacity	100,000	Num actors	5
Update target estimator every	10,000	Anticipatory param	0.5	MLP size	[512, 512, 512, 512, 512]
discount factor	0.95	SL batch size	32	Exp epsilon	0.01
Epsilon start	1	SL learning rate	10^{-4}	Batch size	32
Epsilon end	0.1	SL min buffer size to learn	100	Unroll length	100
Epsilon decay steps	100,000	RL replay memory size	100,000	Num buffers	50
Batch size	32	RL replay memory init size	100	Num threads	4
MLP layers	[512, 512]	RL update target estimator every	10,000	Max grad norm	40
Learning rate	10^{-5}	RL discount factor	0.95	Learning rate	10^{-4}
		RL epsilon start	1	Alpha	0.99
		RL epsilon end	0.1	Momentum	0
		RL epsilon decay steps	100,000	Epsilon	10^{-5}
		RL batch size	32		
		RL MLP layer	[512, 512]		
		RL learning rate	10^{-5}		

TABLE V
COMPARISON PHASE 1 – RESULTS

Algorithm	APPG	WP
DMC	46.462	69.958
DQN	43.843	62.761
NFSP	42.509	57.658

All values rounded to three decimal places.

TABLE VI
COMPARISON PHASE 2 – RESULTS

B / *A*	DQN		DMC		NFSP	
	APPG	*WP*	*APPG*	*WP*	*APPG*	*WP*
DQN	-	-	42.133	57.115	44.033	63.506
DMC	46.54	70.139	-	-	46.787	70.85
NFSP	42.520	57.94	40.771	51.871	-	-

All values rounded to three decimal places.

most likely is the *1-trump* card that has to be played at a specific point in time to have a winning chance.

Small improvements can be seen from the single-player AI in Bettel. Sub-game-specific HPO might improve this model further.

The single-player model of Piccolo improves on the benchmarks by a solid margin but achieves less than a WP of 50% in a full setup against random agents. More improvements are needed in the sub-game Piccolo to push the single-player AI to a higher WP.

In all sub-games, it can be observed that the single-player AI can achieve higher reward increases than a non-single-player AI against random agents. However, it can be observed that this advantage shifts when combining non-single-player AIs in a team. This observation is visible in Standard-Cego, where in Setup 1 the single-player is advantaged, but in Setup 4 this advantage shifts to the non-single-players side. It appears that non-single-players can achieve at least some degree of cooperative play.

VIII. CONCLUSION

This work applied deep RL to the card game Cego. The game environment was implemented in RLCard and divided into sub-games to maximize each problem separately. An information state encoding was provided to format the information observed by players into by DNNs interpretable numerical arrays. For the encoding of actions, all cards are encoded separately. To simplify each sub-game, the number of possible game states is restricted by rules based on heuristic knowledge.

Simulations were conducted to identify benchmarks for each sub-game in order to evaluate the models.

Models were trained and compared on the sub-game of Standard-Cego using the algorithms DQN, NFSP, and DMC. Hyperparameters of DQN and NFSP were optimized. In two separate experiments, it was concluded that DMC performed the best under the defined conditions.

Afterward, models for all players and sub-games were trained and evaluated based on the previously defined benchmarks. All models improve on their benchmarks. High improvements can be observed in the single-player position of Ultimo. Solid improvements are examined in the first player positions of the sub-games Standard-Cego, Solo, Räuber, and Piccolo. Bettel's single-player AI has improved slightly. The combination of non-single-player models leads to higher average rewards, which indicates that models learn to play cooperatively.

A. Future Work

The single-player AI in Piccolo improves, but wins less than half of the games. It is proposed to restrict the number of game-states by further game selection rules in order to increase performance.

With the goal of further maximizing the rewards of each player in all sub-games, it is proposed to hyperparameter optimize each sub-game individually. Hyperparameter tuning

TABLE VII
SUB-GAME EVALUATION RESULTS

Player Position	**Standard-Cego**: *APPG*			
	Setups:			
	1.	2.	3.	4.
1	**46.48**	38.02	**43.2**	37.98
2	32.52	**40.98**	35.8	**41.03**
3	32.52	**40.98**	35.8	**41.03**
4	32.52	**40.98**	35.8	**41.03**

Player Position	**Solo**: *APPG*			
	Setups:			
	1.	2.	3.	4.
1	**54.26**	**46.01**	**51.42**	**44.93**
2	24.74	32.99	27.58	34.07
3	24.74	32.99	27.58	34.07
4	24.74	32.99	27.58	34.07

Player Position	**Ultimo**: *WP*			
	Setups:			
	1.	2.	3.	4.
1	**94.74**	7.61	**88.06**	**69.42**
2	5.26	**92.39**	11.94	30.58
3	5.26	**92.39**	11.94	30.58
4	5.26	**92.39**	11.94	30.58

Player Position	**Bettel**: *WP*			
	Setups:			
	1.	2.	3.	4.
1	**78.94**	**67.3**	**75.79**	**60.78**
2	21.06	32.7	24.21	39.22
3	21.06	32.7	24.21	39.22
4	21.06	32.7	24.21	39.22

Player Position	**Piccolo**: *WP*			
	Setups:			
	1.	2.	3.	4.
1	47.65	34.32	43.76	28.76
2	**52.35**	**65.68**	**56.24**	**71.24**
3	**52.35**	**65.68**	**56.24**	**71.24**
4	**52.35**	**65.68**	**56.24**	**71.24**

Player Position	**Räuber**: *WP*		
	Setups:		
	1.	2.	3.
1	**85.70**	**82.94**	**74.46**
2	**71.52**	**83.85**	74.26
3	71.36	**66.97**	**75.02**
4	**71.41**	66.25	**76.27**

Results rounded to two decimal places.

methods, such as Bayesian optimization [34] could potentially lead to better results.

The bidding phase is something that hasn't been addressed in this work. In the context of future work, it is proposed to implement this aspect of the game with a classification model. Here, the problem definition lies in classifying information states to the sub-game that potentially leads to the highest reward.

The final models have not been tested to see how they compare with professional Cego players. Outside of this work, the AI models have been made available for the relaunch of the web-based game Cego Online. The relaunch is planned to be available to all in the near future. Tests are planned and the first results against real players may follow soon.

REFERENCES

[1] G. Blümle, "Das badische Nationalspiel Cego," p. 16, 2016. [Online]. Available: https://www.cego-online.de/tl_files/documents/Zegolang160414.pdf

[2] G. Baumann and G. Blümle, "DAS CEGOSPIEL," 2013. [Online]. Available: https://www.cego-online.de/tl_files/documents/CegoSpielregelndef1301224mit%20Anhang.pdf

[3] A. Laber, "Cego," 2011. [Online]. Available: http://www.cego.de/

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016. [Online]. Available: http://www.nature.com/articles/nature16961

[5] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017, number: 7676 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nature24270

[6] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," 2017, publisher: arXiv Version Number: 1. [Online]. Available: https://arxiv.org/abs/1712.01815

[7] M. J. Holler, G. Illing, and S. Napel, *Einführung in die Spieltheorie*, 8th ed., ser. Lehrbuch. Berlin: Springer Gabler, 2019.

[8] D. Zha, K.-H. Lai, Y. Cao, S. Huang, R. Wei, J. Guo, and X. Hu, "RLCard: A Toolkit for Reinforcement Learning in Card Games," *arXiv:1910.04376 [cs]*, Feb. 2020, arXiv: 1910.04376. [Online]. Available: http://arxiv.org/abs/1910.04376

[9] D. Zha, K.-H. Lai, S. Huang, Y. Cao, K. Reddy, J. Vargas, A. Nguyen, R. Wei, J. Guo, and X. Hu, "RLCard: A Platform for Reinforcement Learning in Card Games," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, Jul. 2020, pp. 5264–5266. [Online]. Available: https://doi.org/10.24963/ijcai.2020/764

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, ser. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016.

[11] D. Zha, J. Xie, W. Ma, S. Zhang, X. Lian, X. Hu, and J. Liu, "DouZero: Mastering DouDizhu with Self-Play Deep Reinforcement Learning," *arXiv:2106.06135 [cs]*, Jun. 2021, arXiv: 2106.06135. [Online]. Available: http://arxiv.org/abs/2106.06135

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv:1312.5602 [cs]*, Dec. 2013, arXiv: 1312.5602. [Online]. Available: http://arxiv.org/abs/1312.5602

[13] J. Heinrich and D. Silver, "Deep Reinforcement Learning from Self-Play in Imperfect-Information Games," *arXiv:1603.01121 [cs]*, Jun. 2016, arXiv: 1603.01121. [Online]. Available: http://arxiv.org/abs/1603.01121

[14] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret Minimization in Games with Incomplete Information," in *Advances in Neural Information Processing Systems*, vol. 20. Curran Associates, Inc., 2007. [Online]. Available: https://papers.nips.cc/paper/2007/hash/08d98638c6fcd194a4b1e6992063e944-Abstract.html

[15] M. Bowling, N. Burch, M. Johanson, and O. Tammelin, "Heads-up limit hold'em poker is solved," *Science*, vol. 347, no. 6218, pp. 145–149, Jan. 2015, publisher: American Association for the Advancement of Science. [Online]. Available: https://www.science.org/doi/10.1126/science.1259433

[16] M. Ginsberg, "GIB: Steps Toward an Expert-Level Bridge-Playing Program," in *IJCAI*, 1999.

[17] M. Quambusch, *Gewinnen beim Skat gläserne Karten*, 1990, oCLC: 75456320.

[18] P. Lincoln, *Skat lernen*, 1st ed. Stuttgart: Urania, Jun. 2004.

[19] S. Grandmontagne, *Meisterhaft Skat spielen. 1: Allgemeine Spiellehre*. Saarbrücken: Grandmontagne, 2005.

[20] D. S. Harmel, *Skat-Zahlen*. Dr. Siegfried Harmel, Jan. 2016.

[21] S. Kupferschmid and M. Helmert, "A Skat Player Based on Monte-Carlo Simulation," in *Computers and Games*, ser. Lecture Notes in Computer Science, H. J. van den Herik, P. Ciancarini, and H. H. L. M. J. Donkers, Eds. Berlin, Heidelberg: Springer, 2007, pp. 135–147.

[22] T. M. Furtak, "Symmetries and Search in Trick-Taking Card

Games," 2013. [Online]. Available: https://era.library.ualberta.ca/items/df9f1055-df09-43ed-910b-38da4e9d0792

[23] S. Edelkamp, "Challenging Human Supremacy in Skat," *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, no. 1, pp. 52–60, 2019, number: 1. [Online]. Available: https://ojs.aaai.org/index.php/SOCS/article/view/18502

[24] W. Fürderer and L. Gerda, "Über das Spiel Cego," 2019. [Online]. Available: http://cego-schwarzwald.com/Ueber-das-Spiel-Cego/desktop/

[25] J. Dell'Oro-Friedl, M. Wangler, N. Dretvic, F. Gruner, C. Kluge, R. Kocher, and F. Reuting, "Cego-Online: Startseite - Cego online," 2013. [Online]. Available: https://www.cego-online.de/startseite.html

[26] Cegofreunde St. Georgen, "Taktik und Tipps," 2012. [Online]. Available: http://cegofreunde.jimdofree.com/taktik-und-tipps/

[27] Q. Jiang, K. Li, B. Du, H. Chen, and H. Fang, "DeltaDou: Expert-level Doudizhu AI through Self-play," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. Macao, China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 1265–1271. [Online]. Available: https://www.ijcai.org/proceedings/2019/176

[28] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992698

[29] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, second edition ed., ser. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018.

[30] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," 2015, publisher: arXiv Version Number: 3. [Online]. Available: https://arxiv.org/abs/1509.06461

[31] J. Heinrich, M. Lanctot, and D. Silver, "Fictitious self-play in extensive-form games," in *International conference on machine learning*. PMLR, 2015, pp. 805–813.

[32] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012. [Online]. Available: http://jmlr.org/papers/v13/bergstra12a.html

[33] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep Reinforcement Learning That Matters," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018, event-place: New Orleans, Louisiana, USA.

[34] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms," in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html