Reinforcement Learning Agents for the Game of Differenzler Jass

Bernard Jaquet
bernard.jaquet@students.unibe.ch

Matej Kutirov
matej.kutirov@students.unibe.ch

https://github.com/Huo12345/reinforcment-learing-differenzler-jass

1 Abstract

Games with incomplete information are a difficult problem to solve within the field of reinforcement learning. This project aims to investigate if learning-based techniques can utilise information inferred from past actions of opponents with different observations of the game state to reduce the uncertainty about the unobserved part of the game state and therefore improve their performance. The game Differenzler Jass was chosen for this investigation because the rules of the game force players to reveal partial information about the cards they hold, but the game doesn't include player cooperation. The experiments run show that the impact of providing additional inferred information about the game state doesn't prove majorly beneficial. The biggest impact observed came from a reduction in the dimensionality of the state space, even if omitting less crucial information, and using proven techniques like look-ahead. This might partially be caused by the fact that due to the complexity of the game, the reinforcement learning agents couldn't be trained to reach their full capacity within the given timeframe.

2 Introduction

Significant progress in artificial intelligence has been achieved using supervised learning, but these systems are often limited by the availability and quality of expert data. In contrast, reinforcement learning (RL) systems learn from their own experiences, allowing them to surpass human capabilities and perform well in areas without human expertise. Recent RL advancements, particularly with deep neural networks, have shown success in surpassing human performance in various domains, including complex games.

This research applies RL to Differenzler Jass[1], a strategic card game with unique prediction and scoring mechanisms. The goal is to determine if using game history and rules to infer hidden game states can enhance an RL agent's decision-making under uncertainty. We hypothesise that such an agent will outperform those without this capability. Differenzler Jass, with its complexity and lack of player cooperation, provides an ideal environment to test this hypothesis.

3 Methodology

To examine the validity of the hypothesis, this project will be implemented in four distinct phases. These guarantee a systematic approach to developing, training, and evaluating reinforcement learning (RL) agents for the game Differenzler Jass[1]. Each phase builds upon the previous one to enhance the agents' capabilities and to address the research hypothesis effectively.

Phase	**Objective**	**Description**
1	Implement the Differenzler Jass[1] in RLCard[2]	The game mechanics, rules, and scoring system of Differenzler Jass are implemented within the RLCard framework. This setup provides a standardised environment where various RL agents can be trained and tested.
2	Implement, train and evaluate different agent types (Random, Greedy, DeepQ) on a fixed target point system.	Three types of agents—Random, Greedy and DeepQ—are implemented and trained to reach a fixed number of points. The performance is measured to understand basic capabilities and limitations. In this and the following phases, the impact of additional statistics on the agent's performance will be evaluated.
3	Test the adaptability of agents to different target points.	Agents are trained with randomly assigned target points to assess their ability to adapt strategies based on varying goals. This phase focuses on evaluating the flexibility and strategic adjustments made by learning-based agents.
4	Enhanced strategies and prediction	The final phase aims to complete the agent's capabilities and boost their performance. Possible enhancements are the implementation of a learning-based approach to make the point predictions and using look-ahead strategies to improve the performance of existing agents.,

Table 1. Description of each of the project phases.

The agents are evaluated using several metrics to comprehensively assess their performance:

- Win Rates: The primary metric to determine overall effectiveness in achieving the game's objectives.
- Prediction Accuracy: Measures how accurately agents predict the points they aim to achieve.
- Learning Improvement: Assesses the improvement in agents' performance over time and across different training phases.

4 Game

4.1 Game Sequence

Differenzler Jass is a strategic card game with a unique prediction and scoring mechanism. After the cards are dealt, each player must predict the points they want to achieve in this round. These predictions are made publicly, allowing successive players to adjust their prediction based on the other players' predictions. The last card dealt selects the trump suit of the round dealt, shown to all players. This means all players know one card of the last player in the round.
After the prediction round the first player plays one of their cards. This starts a pile on which each player must add a card. The player having played the strongest card takes the pile and all points of the cards in that pile and starts the next pile.
After all cards are played, each player sums up the points on the piles taken and calculates the difference between the scored points and their prediction. This concludes the round.
The final score is accumulated across all the rounds of a match. The player with the lowest total difference points at the end is declared the winner.

4.2 Rules of the Game

Differenzler Jass follows specific rules regarding the play of suits and the use of trump cards. The fundamental rule is that players must follow suit whenever possible. Here are the detailed rules:

1. Following suit:
   Players must play a card of the suit that led the pile if possible. This obligation to follow suit ensures strategic depth and fairness. Alternatively, they can play a trump card that is higher than all trump cards on the current pile.
2. Exceptions to following suit:
   If a player doesn’t hold any card of the suit that led the pile, they may play any other card. When playing a trump card, it must still beat all trump cards on the pile.
3. Following suit when playing trump cards:
   If the first card on the pile is of trump suit, players must follow suit if possible. The rule enforcing the play of stronger trump cards is inactive in this case. If the only trump card the player holds is the jack, they are freed of the obligation to follow suit.
4. Catch all exceptions:
   If the previous rules prevent the player from playing any card, they may play any card they currently hold. This tends to happen towards the end of a round.

4.3 Counting and Card Strength

Each card has a specific point value, which differs between standard suits and the trump suit. The Table 1 shows the points for each card rank:

Rank	Points (Normal Suit)	Points (Trump Suit)
A	11	11
K	4	4
Q	3	3
J	2	20
10	10	10
9	0	14

Table 2. Card strength with its point value

Additionally, the player who captures the last pile of the round receives an extra 5 points. This bonus can be strategically significant, influencing the overall score and the outcome of the match. In total, there are 157 points available in each round.

For taking a pile the strongest card on the pile needs to be determined. For this only cards on the same suit as the card that led the pile and trump cards are considered. Trump cards always beat cards of other suits. Within suits, the card's strength is given by the following order from strongest to weakest: A, K, Q, J, 10, 9, 8, 7, 6. For the trump suit the order is different: J, 9, A, K, Q, 10, 8, 7, 6.

5 Challenges in Learning a RL Agent for Differenzler Jass

Training a reinforcement learning (RL) agent to effectively play Differenzler Jass involves addressing several unique and complex challenges inherent to the game's mechanics and strategic requirements. Below, we explore these challenges in detail:

Partial Observability
In Differenzler Jass, each player has incomplete information. They only know their own hand. This partial observability means that the RL agent must make decisions without knowing the hands of the other players. This drastically increases the complexity of the decision-making process as the agent has to infer or estimate the full state of the game from limited information.

Strategic Depth
The game is deeply strategic, requiring players to predict their points and play cards to match their predictions while accounting for the actions of other players. An RL agent must develop

sophisticated strategies that not only consider immediate actions but also plan several moves to meet its predictions. This involves learning to balance the expending of resources to take or pass up the current and future piles.

Random Elements

Each round of Differenzler Jass begins with the random selection of a trump suit and randomly distributed cards. This element of randomness introduces variability that the RL agent must learn to handle. The agent must adapt its strategy based on the current trump suit, which can significantly alter the value and effectiveness of certain cards and strategies. This requires the agent to be flexible and responsive to changing game conditions.

Scoring Mechanism

The game's scoring system is based on the point difference, which is the difference between the predicted and actual points achieved in each round. This difference is accumulated throughout multiple rounds of a match, and the player with the lowest total difference points wins. This scoring is complex since the same card might have different points in different situations. Additionally, depending on the prediction a card might be highly desirable or hindering in reaching the target point total. This mechanic requires the RL agent to focus on minimising cumulative differences rather than maximising immediate rewards. Such a focus on long-term outcomes adds complexity to the learning objective.

Rule Constraints

Differenzler Jass has specific rules regarding the play of suits and the use of trump cards. Players must follow the suit that was led if they can, and they have the option to play trump cards strategically. These rules introduce constraints that the RL agent must learn to navigate effectively. Holding back cards might turn out to be a waste of that card's potential. The agent must internalise these rules and incorporate them into its decision-making process, which requires understanding and applying complex game mechanics.

Long-Term Planning

Success in Differenzler Jass requires long-term planning and the ability to predict future outcomes. The agent must consider the implications of its current actions on future rounds and the overall match. When losing it might be beneficial to take bigger risks in later rounds. This involves advanced planning capabilities and the ability to adjust strategies based on new information as the game progresses. Long-term planning is a critical skill for the agent to develop to be successful in this game.

Sparse Rewards

The feedback mechanism in Differenzler Jass is characterised by sparse and delayed rewards. Players only receive meaningful feedback in the form of difference points at the end of each round. This sparse reward structure makes it difficult for the RL agent to learn from its actions, as there is a delay between the actions taken and the resulting feedback. The agent must learn to associate its actions with outcomes over longer timescales, which is a challenging aspect of the learning process.

6 Environment

6.1 Initialization and Configuration

We initialised the environment with flexible configurations that accommodate various game setups, including a customizable number of players (default is four) and a set number of rounds. We also implemented a fixed prediction strategy by default, guiding players in making their point predictions at the beginning of each round. This structured setup forms the foundation for the agent's learning process.

6.2 State Representation

To test the impact of the state representation, we implemented different ways to encode the state of the game as a vector. This includes the following representations:

- Full state (default): Contains the full history of all previously played cards and who played them along the current pile, the player's hand, the scores and predictions for each player and the trump suit of the current round. The final tensor has a size of 42x40.

- Compressed state (compressed): Contains a compressed state of the round history, only showing which cards have already been played. Additionally, the state representation contains the current pile, the player's hand, the scores and predictions for each player and the trump suit of the current round. The final tensor has a size of 4x36.

- Simple enhanced (enhanced_small): Contains everything from the compressed representation, but adds which player has followed suit for all the suits in the past. The final tensor still has a size of 4x36.

- Advanced enhanced (enhanced_large): Contains everything from the enhanced_small representation, but adds a list of the cards on the player's hand that are strong (can only be taken by a trump card) and a list of the weak card on the player’s hand (must be taken by another card). The final tensor still has a size of 6x36.

6.3 Prediction Strategies

The handling of the prediction round was implemented using prediction strategies. These are callables that take in a state observation and need to return a prediction. To support learning-based approaches, the strategy is provided with the reached score at the end of the round. The following strategies were implemented:

- Fixed strategy: Always returns the same prediction regardless of the observation. Can be used to incentivise agents to collect the maximal number of points by setting the value to 157 and the minimal number of points by setting it to 0.

- Random strategy: Chooses a random value between 0 and 157 for the player to reach. Incentivizes the agent to learn reaching a specific value provided in the state observation.

6.4 Action Space

The action space in the environment is defined to encompass all cards in the deck. The illegal moves are filtered out before by masking the q values in the DQN agents and are programmatically enforced in the other approaches.

6.5 Reward System

The simplest target for the agent is to win the game, meaning to be the player with the least difference in predicted to scored points, characterised by the winner-takes-all strategy. But this is a difficult reward to learn based on. A proxy target is to reward agents for having the least difference in points, giving the agent more feedback on its performance. This reward structure can lead to agents cooperating to all reach 0 points in difference if not prevented by the prediction strategy. This reward structure was implemented as the default reward.

7 Agents

As mentioned before to benchmark the performance of learning based agents, this project implements multiple kinds of agents. These algorithms range from simple heuristic-based approaches to advanced reinforcement learning (RL) techniques. Here, we outline the key algorithms used and their respective roles in the project.

7.1 Random Agent

The Random agent serves as a baseline for comparison. It selects a random legal action at each turn, without any strategic consideration. This agent is expected to perform the worst among all implemented agents, providing a benchmark against which more sophisticated strategies can be measured.

7.2 Greedy Agent

The Greedy agent uses a heuristic approach to maximise immediate gains. It calculates the expected value of the pile given each legal action and takes the action that minimises the difference between the prediction and the round score based on this expected value. While this strategy is expected to be more effective than the random approach, it lacks the foresight and adaptability of learning-based agents. Learning-based agents that have developed a more sophisticated strategy planning ahead should easily beat a greedy agent. There are two flavours of the greedy agent, the fully greedy agent which always picks the action with the highest expected payout and the semi-greedy agent which samples the soft-max distribution of the expected payout for each action.

7.3 DeepQ Agent

The DeepQ agent employs the Deep Q-Network (DQN) algorithm, a popular RL method. DQNs are trained against random agents and use deep neural networks to approximate the Q-value function, which estimates the expected future rewards for each action given the current state. This agent makes decisions based on the current game state but does not attempt to infer additional information about unobservable parts of the game. The implementation leverages the RLCard framework's default DeepQ learning agent, allowing us to benchmark against a standard RL approach.

7.4 Lookahead DeepQ Agent

This class of agents uses a pre-trained DeepQ agent and simulates one or multiple steps in the game to evaluate the best action. Since Differenzler Jass is an incomplete information game, the lookahead agent needs to make assumptions about the full game state and calculate future states based on it. Agents looking just a few steps ahead usually outperform classical q agents without this capability.

8 Experiments

To evaluate the performance of different agents playing Differenzler Jass multiple experiments were conducted. These were designed to test the efficacy of different state representations and agent strategies. Below is an outline of the experiments conducted as well as a discussion of the corresponding results.

8.1 Experiment Setups

The learning-based agents were trained with different state representations to evaluate how different formats of input impact the agent’s performance:

- Full State: This state representation includes a complete history of every card played, consisting of 42x40 values.

- Compressed State: This representation captures only the cards that have been previously played, resulting in a 36x4 values matrix.

- Augmented Small State: Similar to the compressed state, but includes additional information about which players followed suit.

- Augmented Large State: Builds on the augmented small state by adding flags for strong cards that can only be taken by a trump card and weak cards that must be taken, resulting in a 36x6 values matrix.

8.2 Results

8.2.1 Greedy Agent

The first experiment compares the performance of the Random agent with two types of Greedy agents - Full Greedy and Semi Greedy, yielding the results shown in Table 2.

	Full Greedy	Semi Greedy
Average points off	52,4	53.8
Win rate	27%	28%

Table 3. Results for winning rate and deviation of predicted points between two different Greedy agents against Random agents

8.2.2 DQN Agent random prediction

This section compares the performance of the different state representations on the DeepQ learning agents. The evaluation metrics are the winrate of the agent and the average points by which the agents are off the prediction.

Full State

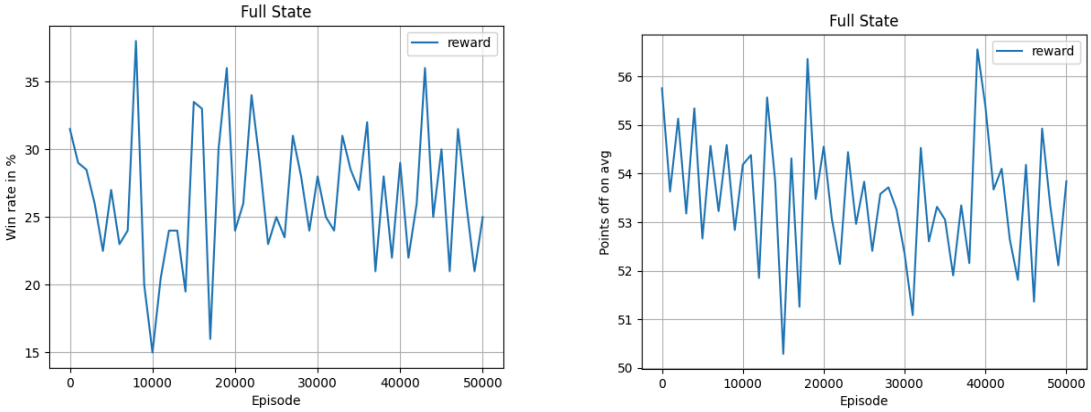

Figure 1. Visualisation of Win rate (left) and points off (right) for the DQN agent with random prediction with full state space.

Compressed State

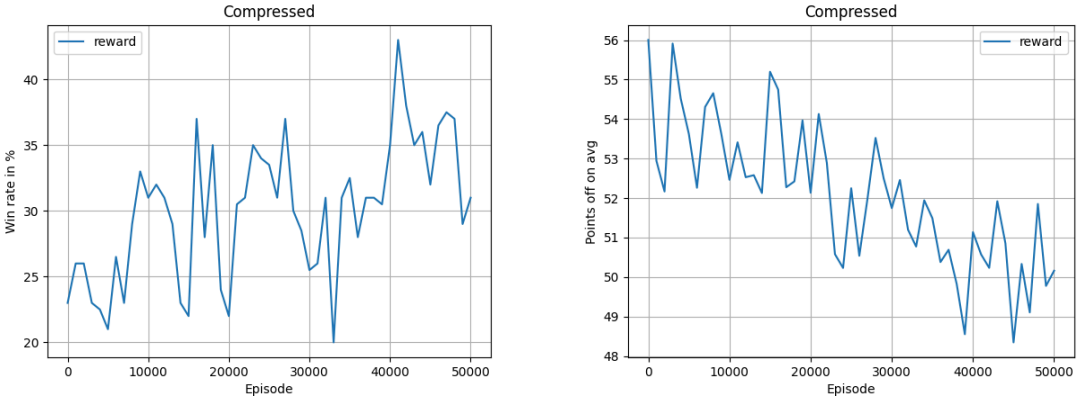

Figure 2. Visualisation of Win rate (left) and points off (right) for the DQN agent with random prediction with compressed state space.

Augmented State small

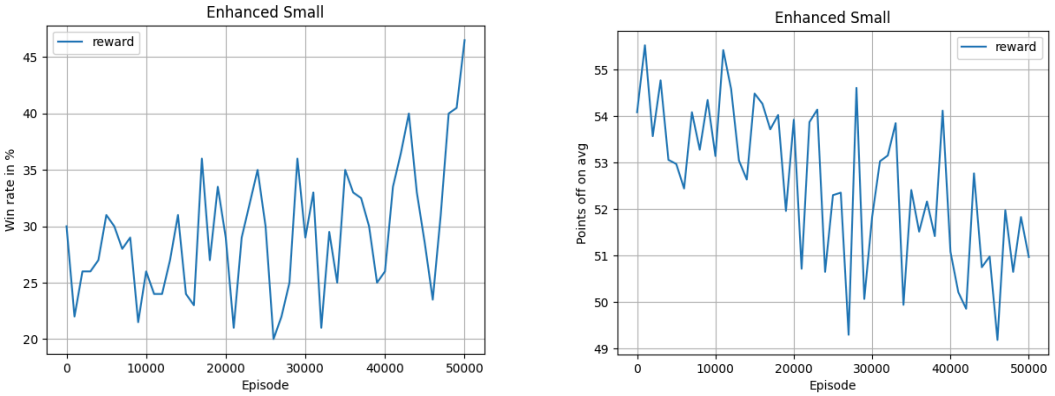

Figure 3. Visualisation of Win rate (left) and points off (right) for the DQN agent with random prediction with augmented small state space.

Augmented State large

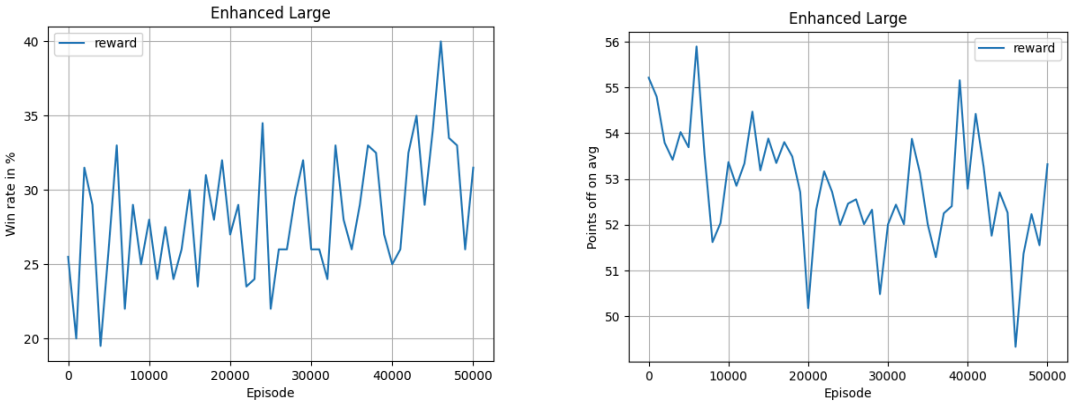

Figure 4. Visualisation of Win rate (left) and points off (right) for the DQN agent with random prediction with augmented large state space.

8.2.3 DQN Agent minimum points

Since the training of the DQN agents didn't seem to affect the win rate a lot, an agent was trained on the simplified goal of reaching the minimum points possible as a comparison. This experiment was only done for the compressed state representation.

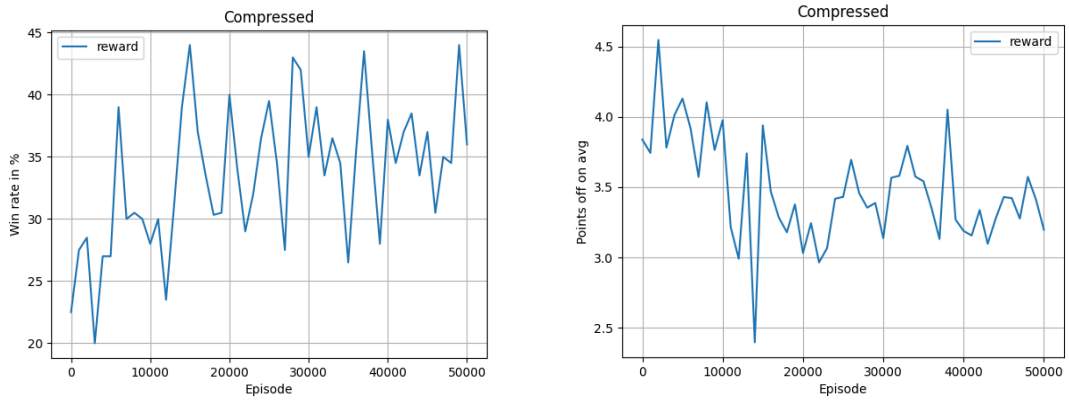

Figure 5. Visualisation of Win rate (left) and points off (right) for the DQN agent with the goal of reaching minimum points with compressed state space.

8.2.4 Lookahead DQN

For the final experiment, a trained DQN for the compressed state was run on a look-ahead strategy to measure the effect of looking ahead. The experiment looked ahead one step and calculated 10 possible next states.

	Lookahead on compressed state
Average points off	43.7
Win rate	47.5%

Table 4. Results for winning rate and deviation of predicted points for the look-ahead strategy

9 Conclusion and Further Steps

The conducted experiments seem to indicate that Differenzler Jass is a difficult game to master for DQNs. While most training shows an improvement in points off on average, this difference is usually not enough to significantly improve the win rate of the agent. This conclusion is further supported by the results for the simplified goal of reaching the minimum number of points, where the win rate goes up significantly with training.

Contrary to the hypothesis, adding additional information to the state of the DQN agent doesn't significantly boost that agent's performance. It seems that adding information to the state representation requires effort on the side of the agent to sort out the meaning of the input, negating the benefit that information might have provided. But if the additional information doesn't significantly increase the size of the state space, the impact on the learning process seems small. Additionally, the agents didn't seem to reach peak performance during training, therefore it might be possible that the additional information in the state representation could lead to a higher performance when reaching the performance plateau. Regardless, compressing the state spaces seems to be a winning strategy.

The other results are in line with what was expected. The Greedy agent does outperform the random agents, even if the measured performance is smaller than expected. Under the right circumstances, learning-based agents outperform greedy agents easily. And finally, the look-ahead agent gave a significant boost to already trained DQNs. Look-ahead agents could probably benefit more from enhanced states, especially from the information of which player has followed suit in the past since this information allows them to make a better guess on how the remaining cards are distributed.

Overall, these findings highlight the necessity for longer and more intensive training sessions to fully explore and optimise the potential of reinforcement learning agents in this game. Future research should focus on addressing the state space size problem and refining learning strategies to achieve more significant performance improvements.

10 References

[1] "gratis online Differenzler jassen." *gratis online Differenzler jassen*, https://www.differenzler.ch/drules.php. Accessed 17 March 2024.

[2] "datamllab/rlcard: Reinforcement Learning / AI Bots in Card (Poker) Games - Blackjack, Leduc, Texas, DouDizhu, Mahjong, UNO." *GitHub*, https://github.com/datamllab/rlcard. Accessed 17 March 2024.