AFDQN & CANDQN: Wining Uno with Reinforcement Learning

Siyuan Li [1] **Han Yan** [2] **Ziyuan Li** [3] **Zhesheng Xu** [4]

Abstract

Uno is a complex card game, and it's difficult to explore its environment with reinforcement learning. The difficulty is from litter legal actions, imperfect information, etc. For these problems, we propose Action-Focused DQN (AFDQN) and Color-and-Number DQN (CANDQN) to train our agents to play Uno on RLCard. Our approaches significantly outperform DQN baseline in individual tournaments against other agent competitors, which demonstrates that our approaches effectively optimize our agent's policy. The source code is available at `https://github.com/wolfball/AF-CAN-DQN-For-Uno`.

1. Introduction

Uno is a popular card game played with its own unique deck, card with four colors and some special card. In the common version of Uno, the first player to have no cards remaining is the winner. Since other additional rules are too complex and have inconsistent standards, our works focus on the common version of the game, and we aim to train an agent that optimally plays.

With the popularity of reinforcement learning(RL), previous work has shown that card games have ideal structure for RL including large state spaces, competitive agents with partial information sets, large action sets, and small or delayed returns. As a result, many approaches have been applied to card games. Approaches include Deep Q Networks (DQN)(Mnih et al., 2015), Advantage-Actor Critic methods (Mnih et al., 2016), and Counterfactual Regret Minimization (CFR)(Brown et al., 2018).

Unlike chess games (Go for example), Uno is an Imperfect Information Game(IIG) where a player can not see the opponent's private information. Unfortunately, CFR can not apply to Uno for the reason that the state space and action space of Uno are pretty large.

While DQN can train an agent for Uno, there are still some problems remaining. For one thing, the legal actions set are small compared with action space. For example, the legal actions are constrained by the target card, and most of time you have only one or two choices. To tackle this problem, we propose an Action Focused DQN that can learn from legal actions and focus more on these actions, . For another, the state contains hybrid information which may not be clear for agent to learn. Thus, we present Color-and-Number DQN that disentangles the color and number information in the state which betters the understanding of agent.

Our main contributions are listed as bellow:

1. We propose AFDQN that can learn from legal actions, which outperforms baseline DQN algorithm.

2. We present CANDQN that can extract the color and number information in the state, which outperforms baseline DQN algorithm.

3. We prove the performance of our methods by many experiments against Random and Human agents.

The remainder of this report is organized as follows: Sec.2 describes the Uno game in detail, Sec.3 introduce the environment used for our project, Sec.4 formulizes the Uno game problem, Sec.5 presents our proposed algorithms in detail, Sec.6 records and visualizes the experimental results and makes some analysis, Sec.7 draws a conclusion and Sec.8 shows the main contributions of each member.

2. Game Description

2.1. Set Up

Uno is a card game with the following set up: (1) To reduce the complexity of our project, we consider games with only two players; (2) Each player is dealt seven cards from a deck. You can only see your own cards; (3) The remaining cards, which are also not visible, are placed in a draw pile; (4) The top card from the draw pile becomes the first card in the discard pile.

The set-up determines that Uno is an imperfect information game.

[1]ID:519030910344 [2]ID:519030910404 [3]ID:519030910346 [4]ID:519030910353.
Shanghai Jiao Tong University AI3617 Final Project.

2.2. Deck

The Uno deck contains numbered cards, action cards, and wild cards, 108 cards in total. The numbered cards are each one of four different colors (blue, green, red, and yellow) and have a number between 0-9. The action cards each have one of the four colors and one of three actions (skip, reverse, and draw 2 cards). The wild cards(Wild and Wild Draw 4) allow you to change the current color of gameplay and some(Wild Draw 4) also include an action to draw 4 cards. There is a list in Table.1.

Table 1. Cards of Uno

Value	Count
Number 0	1 of each color
Number 1-9	2 of each color
Reverse	2 of each color
Skip	2 of each color
Draw 2	2 of each color
Wild	4
Wild Draw 4	4

2.3. Game Play

During a player's turn, their goal is to place one of their cards in the discard pile. To do so, the player must place a card from their hand that matches the card on top of the discard pile by number, color, and/or action. If they do not have a card that matches the top of the discard pile, they must draw a card from the draw pile. If they still do not have a card that matches the top of the discard pile, the game moves on to the next player. Alternatively, a player can play a "wild card" which allows them to reset the color on top of the draw pile. Certain cards can alter the game play. If a player plays a reverse card, the direction of gameplay is reversed. If a player plays a skip card, the next player is skipped. If a player plays a draw card, the next player must draw a card from the draw pile. The objective of the game is to be the first player to play all of your cards. The game ends once any player has played all their cards and thus has no cards remaining in their hand.

3. Environment

Thanks for RLCard (Zha et al., 2020), an open source library, we can continue our work based on a well-defined environment.

3.1. State

In RLCard, the state is encoded as 4 planes, where each plane is a 4×15 matrix. The 4 rows correspond to each of the 4 colors available, while the 15 columns correspond to the different values a card can have. Each of the entries is either 1 or 0 to indicate having or not. See Table.2 for specific definitions. Now we represent the state as a $4 \times 4 \times 15$ matrix. And the state space $\mathcal{S}$ has size $2^{4\times4\times15} = 2^240 (\approx 10^{72})$.

Table 2. The State of Each Player. Notice that a player can have at most 2 of any card with the exception of the wild cards.

Plane 1	Have zero cards in Hand
Plane 2	Have one cards in Hand
Plane 3	Have two cards in Hand
Plane 4	Target card

3.2. Action and Reward

Each possible action is given an identifier. In total, there are 61 different actions a player can take, which are playing each of the 60 different cards and drawing a card from the deck. Our action space $\mathcal{A}$ has size 61.

The reward is defined as +1 for winning, -1 for losing and 0 for all intermediate states.

4. Notations and Background

4.1. Imperfect-Information Games and Nash Equilibrium

We use extensive-form game (EFG) to formalize our imperfect information game (IIG). A node (also called **history**) $h \in \mathcal{H}$ in the tree represents all information of the current situation. For each history h, there is a player $p \in \mathcal{P}$ or a chance player c that act at h. Define **player function** $P : \mathcal{H} \to \mathcal{P} \cup \{c\}$. For $P(h) \in \mathcal{P}$, player $P(h)$ should take an action $a \in A(h)$ which is the set of legal actions in h. The chance player is responsible for taking actions for random events, for example, dealing the first seven cards for each player. Define $\mathcal{Z}$ as the set of terminal nodes. For each player $p \in \mathcal{P}$, the payoff function (**utility function**) is $u_p : \mathcal{Z} \to \mathbb{R}$. In Uno, a 2-player zero-sum game, we have $\mathcal{P} = \{0, 1\}$, $u_0(z) + u_1(z) = 0$, for each $z \in \mathcal{Z}$.

For each player p, the set of histories $\mathcal{H}$ is partitioned into **information sets** (infosets). We denote the set of infosets for player p by $\mathcal{I}_p$ and an infoset in $\mathcal{I}_p$ by I_p. Two histories are in the same infoset iff they are indistinguishable from the perspective of player p. Therefore, player p's policy π_p is defined as a function that maps an infoset to a probability distribution over legal actions. We can further define $A(I_p) = A(h)$ and $P(I_p) = P(h)$ for any $h \in I_p$. A **policy profile** is $\pi = (\pi_p, \pi_{-p})$ where π_{-p} is the player p's opponent policy. The expected payoff for p under π is $u_p(\pi_{p,-p})$. We use $\Delta(I)$ to denote the range of payoffs reachable from a history h in infoset I. Let $\Delta = \max_{I \in \mathcal{I}_p, p \in \mathcal{P}} \Delta(I)$. Let $f^{\pi}(h)$ be the joint probability of reaching h under π. $f_p^{\pi}(h)$ is the contribution of player p to $f^\pi(h)$. So $f^\pi(h) = f_p^\pi(h) f_{-p}^\pi(h)$. We focus on the perfect-recall setting, where each player recalls the sequence of their own infosets reached. We define $f_p^\pi(I_p) = f_p^\pi(h), \forall h \in I_p$ and $f_{-p}^\pi(I_p) = \sum_{h \in I_p} f_{-p}^\pi(h)$.

A **best response** $BR(\pi_{-p})$ should satisfy that $u_p(BR(\pi_{-p}), \pi_{-p}) = \max_{\pi_p'} u_p(\pi_p', \pi_{-p})$. A **Nash Equilibrium** π^* satisfies that $u_p(\pi_p^*, \pi_{-p}^*) = \max_{\pi_p'} u_p(\pi_p', \pi_{-p}^*), \forall p \in \mathcal{P}$. The **exploitability** of a player's policy is $e(\pi_p) = u_p(\pi_p^*, \pi_{-p}^*) - u_p(\pi_p, BR(\pi_p))$. The exploitability of π is $\epsilon(\pi) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} e(\pi_p)$.

4.2. Reinforcement Learning and Policy Gradient Methods

RL usually assumes a Markov Decision Process (MDP), where the agent selects an action a_i from the legal action set $A(s_i)$ in state $s_i \in \mathcal{S}$ at each time step i. The agent then receives a reward r_i from the environment and transitions to a new state s_{i+1}. The objective in RL is to learn a policy that maximizes the expected discounted returns, i.e. the state value $V^\pi(s_i) = \mathbb{E}^\pi[\sum_{j=i}^{\infty} \gamma^{j-i} r_j]$, with the discount factor $\gamma \in [0, 1]$.

5. Method

When fixing the opponent's policy, the problem is reduced to a single-agent task. So we employ DQN as baseline (See Figure.1(a)). However, two problems remain: (1) There are only few legal actions $\mathcal{A}(s)$ compared to the action space $\mathcal{A}$. But the agent has to consider every action when making decision, thus causing disturbance ;(2) From experience, we have to choose whether the same number or the same color when facing multiple legal actions. But the baseline ignores these information.

Therefore, in this section, we first briefly review the DQN algorithm baseline(Sec.5.1). We then introduce the Action Focused DQN, which is aimed at solving problem (1) (Sec.5.2). Furthermore, we present the Color-and-Number DQN, which tries to deal with problem (2) (Sec.5.3).

5.1. Baseline: DQN

While Q-learning uses Q-Table to store Q values for each state-action pair, which may fail in curse of dimension, DQN utilize a parameterized function (neural network) to approximate value function $Q(s, a)$. In Q-learning, we update $Q(s_t, a_t)$ by $r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$. Therefore, suppose that the network is parameterized by θ, we can write it as:

$$r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta) - Q(s_t, a_t; \theta) \tag{1}$$

5.2. AFDQN: Action Focused DQN

Due to the rule of Uno, the available actions are strictly controlled by the target card, thus resulting in litter legal actions in every round (See Figure.2). In the baseline, however, the agent does not know which action is legal and just struggles to increase the value of potential actions. This will limit the learning performance of agent.

In order to incorporate the prior knowledge of legal actions, we propose a new algorithm called **Action Focused DQN** (AFDQN). AFDQN has two outputs: one for Q values of the state ($Q(s, a)$), and the other for a vector named **focused action**, whose values indicate the probability of being legal.

Input a state $s \in \mathbb{R}^{240}$, AFDQN will output Q values $Q \in \mathcal{R}^{61}$ and focused action $F \in \mathbb{R}^{61}$:

$$x(s) = M_{\phi_1}(s) \in \mathbb{R}^{D_x} \tag{2}$$

$$Q(s) = M_{\phi_2}(x(s)) \in \mathbb{R}^{61} \tag{3}$$

$$F(s) = M_{\phi_3}(x(s)) \in \mathbb{R}^{61} \tag{4}$$

where $M_{\phi_1}, M_{\phi_1}, M_{\phi_1}$ are three MLPs, and D_x is the dimension of feature vector x.For legal action set $A(s)$, we use one-hot encoding ξ to generate legal action vector $l(s) = \xi(A(s)) \in \mathbb{R}^{61}, l_i \in \{0, 1\}, 0 \le i \le 60$, where 1 for legal and 0 for illegal.

Now, we can define our loss function $\mathcal{L}$:

$$\mathcal{L} = L_{mse}(Q(s_t)_{a_t}, r_{t+1} \gamma \max_a \hat{Q}(s_{t+1})_a) \tag{5}$$

$$+ \lambda L_f(F(s_t), l(s_t)) \tag{6}$$

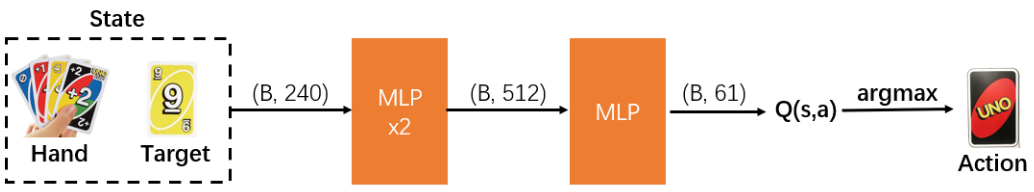
where λ is used for balance, $\hat{(\cdot)}$ means no gradient and L_f can be MSE or BCE loss function. Here is a pseudocode for AFDQN in Alg.1.

In addition to DQN baseline, the learning of focused action vector can in return to refine $\mathcal{M}_{\phi_1}$ to extract more robust features from the state.

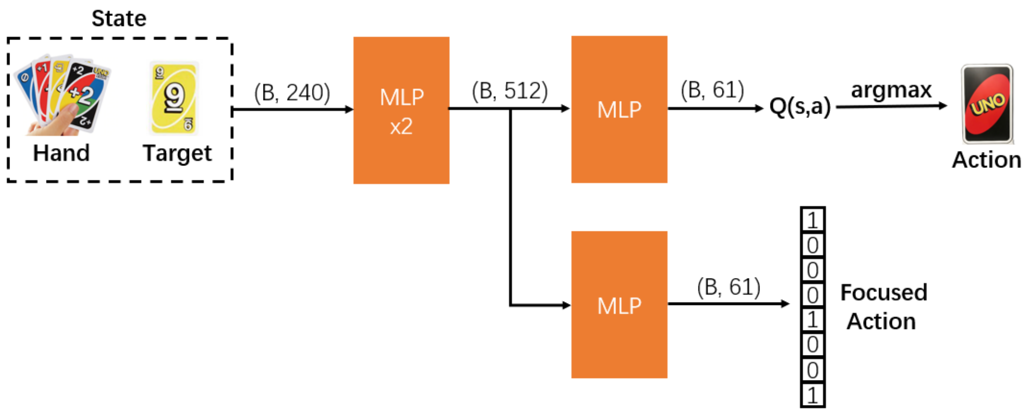
5.3. CANDQN: Color-and-Number DQN

Empirically, our actions are according to the color and the number of the target card. So we incorporate a new feature extractor method into DQN, which can potentially pay attention to the color and number information. We name this module as Color-and-Number DQN (CANDQN).

The pipeline is in Figure.3. For each state $s \in \mathbb{R}^{240} = (s_{hand} \in \mathbb{R}^{180}, s_{target} \in \mathbb{R}^{60})$, We first process s_{target} to produce two new targets $s_{target}^c, t_{target}^n$ which only contains color and number information respectively. Then concatenated with s_{hand}, these two new targets are fed into two networks to produce feature vectors $x_c \in \mathbb{R}^{D_c}, x_n \in \mathbb{R}^{D_n}$. Finally, to remind the network of the target, we concatenate x_c, x_n and s_{target} together as the feature vector. The following stages are the same as the baseline.



(a) Baseline



(b) AFDQN

Figure 1. Pipeline.

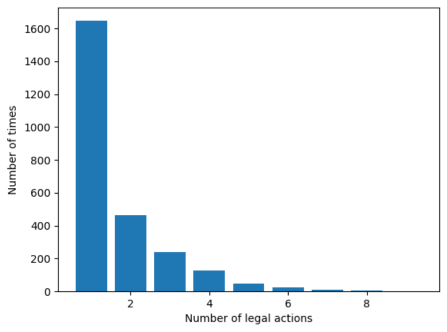


Figure 2. The frequency of the number of legal actions within 100 games. In most of rounds, the number of legal actions will not be more than 4.

Algorithm 1 AFDQN

```
Initialize $M_{\phi_1}, M_{\phi_2}, M_{\phi_3}$, set $\lambda$
**repeat**
  Run game to get trajectory $T$ and store it to memory $D$
  **if** $D$ has enough samples **then**
    Sample $(s_t, a_t, r_t, s_{t+1}, d_t, A(s_t))$ from $D$
    $Q(s_t) \rightarrow M_{\phi_2}(M_{\phi_1(s_t)})$
    $Q(s_{t+1}) \rightarrow M_{\phi_2}(M_{\phi_1(s_{t+1})})$
    $F(s_t) \rightarrow M_{\phi_3}(M_{\phi_1(s_t)})$
    $\mathcal{L}_q \rightarrow MSE(Q(s_t)_{a_t}, r_{t+1} + \gamma \max_a \hat{Q}(s_{t+1})_a)$
    $\mathcal{L}_f \rightarrow L_f(F(s_t), l(s_t))$
    $\mathcal{L} = \mathcal{L}_q + \lambda \mathcal{L}_f$
    Use $\mathcal{L}$ to update $M_{\phi_1}, M_{\phi_2}, M_{\phi_3}$,
  **end if**
**until** The training episodes are reached
```

6. Experimental Results

Here we introduce two baselines for experiments, both are trained with DQN but different in opponents' policy. One is Random agent (Baseline-R) who uniformly chooses a legal action to perform. The other is Human agent (Baseline-H) who will play ”wild-4” if he can and change to the color with the most color in his hand, and play as Baseline-R if without ”wild-4”.

6.1. AFDQN

From the reward curve (See Figure.4(a)(b)) and the average reward within another tournament (See Table.3,Table.6, Table.4, Table.7), we find that our AFDQN outperforms the baseline DQN. When the opponent is Random Agent, the best results are around $\lambda = 1.5$, which shows that paying more attention to legal actions will guide the agent to learn better. Intuitively, knowing what actions to be updated and

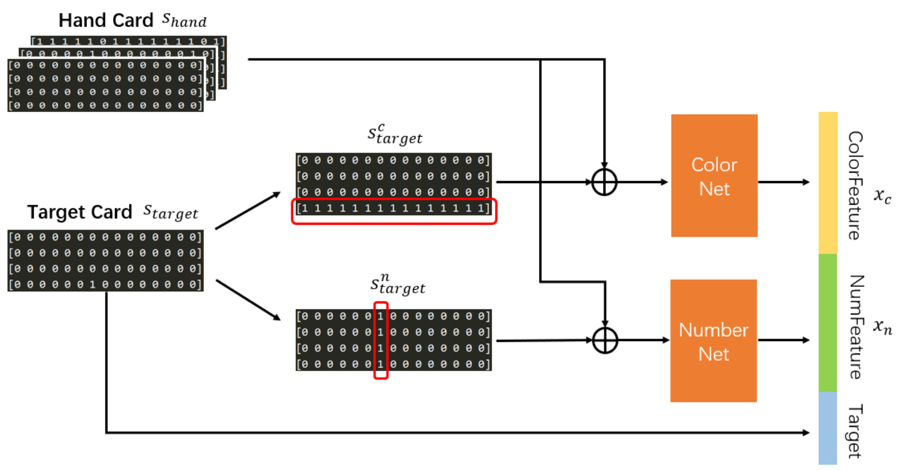


Figure 3. CANDQN

Table 3. AFDQN with different L_f and λ within 1000 games against Random Agent.

Experiment	Result	Experiment	Result
Baseline-R	0.024	Baseline-R	0.024
BCE-0.1	0.084	MSE-0.1	0.048
BCE-0.5	0.088	MSE-0.5	0.024
BCE-1.0	0.018	MSE-1.0	0.030
BCE-1.5	0.068	MSE-1.5	**0.136**
BCE-2.0	0.014	MSE-2.0	0.046

Table 4. AFDQN with different L_f and λ within 1000 games against Human Agent.

Experiment	Result	Experiment	Result
Baseline-H	-0.036	Baseline-H	-0.036
BCE-0.1	0.056	MSE-0.1	0.056
BCE-0.5	0.036	MSE-0.5	0.066
BCE-1.0	-	MSE-1.0	0.086
BCE-1.5	0.058	MSE-1.5	0.048
BCE-2.0	0.040	MSE-2.0	**0.112**

what actions are useless are important in learning. When facing Human Agent, higher λ is still satisfying.

6.2. CANDQN

From the reward curve (See Figure.4(c)(d)) and the average reward within another tournament (See Table.5,Table.8), we find that our CANDQN outperforms the baseline DQN. Moreover, $(D_c = 256, D_n = 128)$ performs the best which indicates that color information should be paid more attention to than number information.

Table 5. CANDQN with different D_c, D_n within 1000 games. The first number represents D_c and the second one represents D_n.

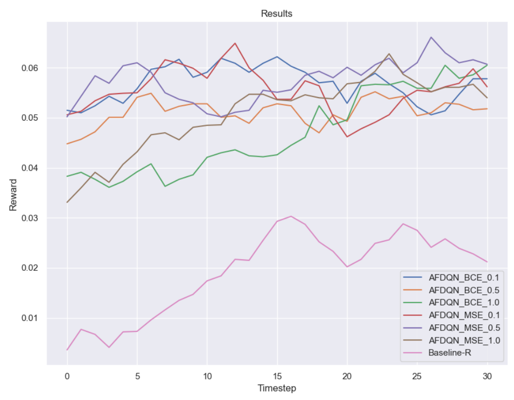
Experiment	Result	Experiment	Result
Baseline-R	0.024	Baseline-H	-0.036
128-128	0.028	256-256	-0.0364
128-256	0.04	512-512	0.004
256-128	**0.062**	CANDQN-H	0.016

Table 6. AFDQN with different L_f and λ within 2000 games against Random Agent.

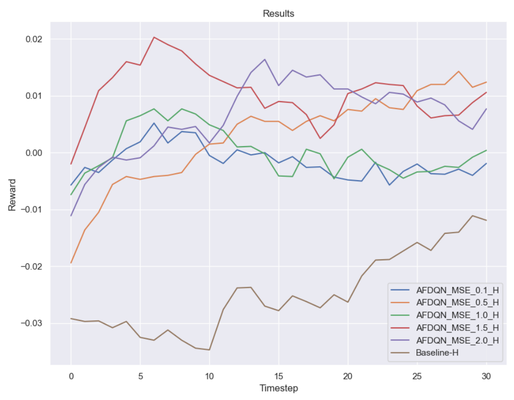
Experiment	Result	Experiment	Result
Baseline-R	0.024	Baseline-R	0.024
BCE-0.1	0.090	MSE-0.1	0.058
BCE-0.5	0.109	MSE-0.5	0.042
BCE-1.0	0.043	MSE-1.0	0.0334
BCE-1.5	0.069	MSE-1.5	**0.109**
BCE-2.0	0.019	MSE-2.0	0.028

Table 7. AFDQN with different L_f and λ within 2000 games against Human Agent.

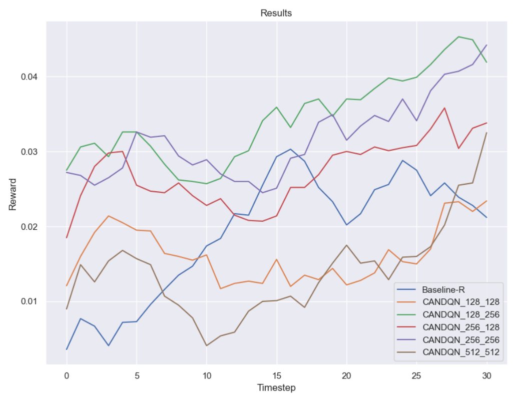
Experiment	Result	Experiment	Result
Baseline-H	-0.036	Baseline-H	-0.036
BCE-0.1	0.076	MSE-0.1	0.046
BCE-0.5	0.080	MSE-0.5	0.075
BCE-1.0	-	MSE-1.0	**0.089**
BCE-1.5	0.053	MSE-1.5	0.065
BCE-2.0	0.059	MSE-2.0	0.086



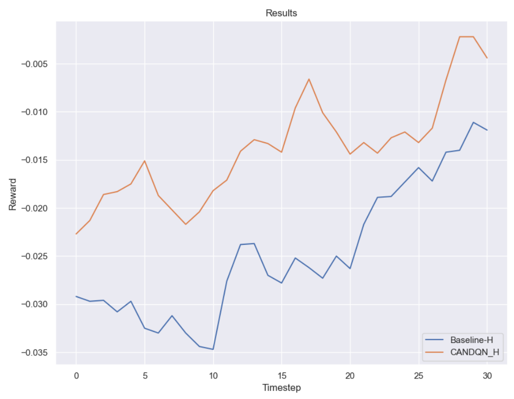
(a) AFDQN vs Baseline-R



(b) AFDQN-H vs Baseline-H



(c) CANDQN vs Baseline-R



(d) CANDQN-H vs Baseline-H

Figure 4.

Table 8. CANDQN with different D_c, D_n within 2000 games. The first number represents D_c and the second one represents D_n.

Experiment	Result	Experiment	Result
Baseline-R	0.024	Baseline-H	-0.036
128-128	0.044	256-256	0.01
128-256	0.031	512-512	0.0034
256-128	**0.067**	CANDQN-H	0.051

7. Conclusion

We propose AFDQN and CANDQN to train the agent to play Uno. For AFDQN, it utilizes the additional information of legal actions to improve the learning. For CANDQN, it decoupling the color and number information in the state, which betters the learning. Both proposed algorithms outperform the baseline DQN algorithm under many experiments.

In addition to the above attempts, we also reproduce the code of (Fu et al., 2022), which modifies the policy optimization objective from originally maximizing the discounted returns to minimizing a type of weighted cumulative counterfactual regret. Unfortunately, we could not find a set of proper hyper-parameters to normal training. The modified code is uploaded to Github as well. Besides, we will try to combine both AFDQN and CANDQN as one algorithm in the future.

8. Author Contribution Statement

Siyuan Li: try to implement different algorithms such as DDPG, PPO but failed.

Han Yan: investigate recent literature, propose AFDQN and CANDQN, implement the methods, analyze results, write the manuscript, visualize results.

Ziyuan Li: propose a rule-based agent as another baseline(does not appear in minipaper), collect related work, conduct experiments.

Zhesheng Xu:Try to implement AFDQN but failed. Run some experiments, visualize results, test models.

Acknowledgements

We would like to thank Professor, Ying Wen, and TAs for their wise guidance and patient help, including computing resources from SJTU Siyuan Yi Hao super computer and SJTU Cross Media Language Intelligence Lab.

References

Brown, N., Lerer, A., Gross, S., and Sandholm, T. Deep counterfactual regret minimization. *CoRR*, abs/1811.00164, 2018.

Fu, H., Liu, W., Wu, S., Wang, Y., Yang, T., Li, K., Xing, J., Li, B., Ma, B., FU, Q., and Wei, Y. Actor-critic policy optimization in a large-scale imperfect-information game. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=DTXZqTNV5nW.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2016.

Zha, D., Lai, K.-H., Huang, S., Cao, Y., Reddy, K., Vargas, J., Nguyen, A., Wei, R., Guo, J., and Hu, X. Rlcard: A platform for reinforcement learning in card games. In *IJCAI*, 2020.