DIALOGUE ENVIRONMENTS ARE DIFFERENT FROM GAMES: INVESTIGATING VARIANTS OF DEEP Q-NETWORKS FOR DIALOGUE POLICY

Yu-An Wang and Yun-Nung Chen

National Taiwan University, Taipei, Taiwan
`b04902004@csie.ntu.edu.tw` `y.v.chen@ieee.org`

ABSTRACT

The dialogue manager is an important component in a task-oriented dialogue system, which focuses on deciding dialogue policy given the dialogue state in order to fulfill the user goal. Learning dialogue policy is usually framed as a reinforcement learning (RL) problem, where the objective is to maximize the reward indicating whether the conversation is successful and how efficient it is. However, even there are many variants of deep Q-networks (DQN) achieving better performance on game playing scenarios, no prior work analyzed the performance of dialogue policy learning using these improved versions. Considering that dialogue interactions differ a lot from game playing, this paper investigates variants of DQN models together with different exploration strategies in a benchmark experimental setup, and then we examine which RL methods are more suitable for task-completion dialogue policy learning[1].

Index Terms— dialogue policy, reinforcement learning, deep Q-Networks, exploration

1. INTRODUCTION

Dialogue systems are a very popular topic in natural language processing (NLP) field in recent years. A task-completion dialogue system is usually built by three main modules: natural language understanding (NLU), dialogue manager (DM), and natural language generation (NLG). Each module can be built disjointly or trained in an end-to-end fashion [1]. The goal of a dialogue manager is to learn the dialogue policy, which can be viewed as a partially observable Markov decision process (POMDP), and often formulated as a reinforcement learning (RL) problem. [2–5]

One of the most successful RL algorithms is deep Q-networks (DQN) [6], which is widely used in many sequential decision-making problems, such as game playing [7, 8]. Numerous extensions of DQN have further been proposed, such as Double DQN [9] and Dueling DQN [10], which are simply modified from nature DQN and shown better in some tasks. Prioritized DQN uses a prioritized experience replay [11] to improve data efficiency. Distributional Q-learning [12], also known as categorical DQN, is an alternative solution to model the value function of value-based RL. Rainbow [13] integrates many variants of DQN and was performed on Atari 2600 games. However, the results showed that not all extensions of DQN improve performance on all tasks. Hence, this paper focuses on investigating which type of DQN fits better with the task about dialogue policy learning and discusses the potential direction for future work.

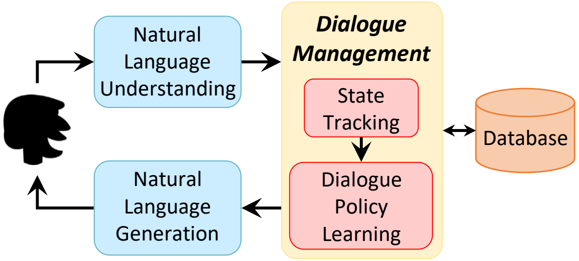


Fig. 1: Illustration of a neural dialogue system framework.

In RL algorithms, the exploration strategy focuses on encouraging the model to choose diverse actions in order to discover potentially good paths, which plays an important role in the performance. The conventional DQN often uses ϵ-greedy [14] and entropy regularization [15] as its exploration strategy. Another general approach is the noisy network [16], which directly adds parametric noise in the hidden layers of a network.

In addition to extensions of DQN, model-based RL has recently drawn some attention for solving RL problems. The prior work about dialogue policy learning designed model-based methods with domain knowledge about dialogues [17, 18], which achieved improved performance due to better interactive scenarios. Furthermore, more general model-based methods that are not restricted to dialogue policy perform extraordinarily in most RL problems. For instance, curiosity-based exploration [19, 20] is a general model-based exploration that can be applied to dialogue policy learning and performs better than conventional exploration strategies such as ϵ-greedy.

Even though there are many extensions of RL algorithms, there is no prior work that investigates such variants in the dialogue area. Therefore, this paper compares variants of DQN and different exploration strategies in order to discuss which algorithms are the most suitable for task-completion dialogue policy.

2. NEURAL DIALOGUE SYSTEM FRAMEWORK

The framework is illustrated in Figure 1 [1]. In a neural dialogue system, an input sentence (recognized utterance or text input) passes through a natural language understanding (NLU) module and becomes a corresponding semantic frame, and a dialogue manager (DM), which includes a state tracker and policy learner, is to accumulate the semantics from each utterance, robustly tracks the dialogue states during the conversation, and generates the next system action. This paper focuses on different reinforcement learning variants for modeling DM. Therefore, we detail the notation formulation of components within DM below.

[1] The code is available at `https://github.com/MiuLab/DialogDQN-Variants`

```
state = {
 'user_action': {'diaact': 'request', ...},
 'agent_action': {'diaact': 'request', ...},
 'current_slots': {'inform_slots': {}, ...},
 'kb_results': {'slot': 'value', ...},
 'history': [...],
 'turn': 2
 }
```

Fig. 2: Dialogue state representation.

2.1. Dialogue State Tracking

Given the symbolic semantics outputted by NLU, such as "`request( moviename; genre=action; date=this weekend )`", three major functions are performed by the state tracker:

1. A symbolic query is formed to interact with the database to retrieve the available results.
2. The state tracker will be updated based on the available results from the database and the latest user dialogue action.
3. The state tracker will prepare the state representation for policy learning.

In practice, the dialogue state can be represented as a dictionary. The format is presented in Figure 2. `user_action` and `agent_action` indicate the semantic dialogue action of the user and the agent respectively. `current_slots` includes slots that have been informed or requested so far. `kb_result` includes available slot values retrieved from the database, and `history` is a list containing all history actions. If deep RL is applied to dialogue policy learning, this state representation will be transformed into a vector s by encoding every dictionary value into one-hot or n-hot representations.

2.2. Dialogue Policy Learning

The state representation s for the policy learning includes the latest user action (e.g., `request( moviename; genre=action; date=this weekend )`), the latest agent action (`request( location )`), the available database results, turn information, and history dialogue turns, etc. By conditioning on the state representation s from the state tracker, the policy π is to generate the next available system action $a = \pi(s)$. Either supervised learning or reinforcement learning can be used to optimize the policy π. This paper focuses on investigating variants of RL-based policy learning detailed in Section 3, where all RL approaches are based on DQN and policy-based RL models are not considered.

3. VARIANTS OF DEEP-Q-NETWORKS

The basic idea of DQN [6] is to learn an optimal Q-value function obeying a Bellman equation:

$$Q^*(s,a) = \mathbb{E}_{s'}[r + \gamma Q^*(s',a') \mid s',a'], \tag{1}$$

where (s,a) is a state-action pair in the current step, and (s',a') is the pair of the next step.

At each step, the agent takes the state s as its input and evaluates the expected return values of each possible action a, greedily selects the action with the maximum value, then receives the next state s' and the reward r from the environment. In the training phase, the agent stores the transition tuple (s,a,s',r) in a replay memory buffer. The Q-value function is approximated by a neural network with parameters θ and optimized by mean square error (MSE) loss,

$$\mathcal{L}_\theta = \mathbb{E}[(r + \gamma \max_{a'} Q(s',a',\theta') - Q(s,a,\theta))^2], \tag{2}$$

where θ' denotes parameters of the target network, and γ is a tunable discount factor. The training tuple is randomly selected from the replay buffer, and the objective can be optimized by gradient descent. Due to the limitation of nature DQN, different variants of DQN and exploration strategies were proposed. We briefly summerize the variants below.

3.1. Double DQN

Double DQN [9] tries to solve the problem about the overestimated bias in the conventional DQN, where the model decouples selection and evaluation in DQN and rewrites the objective as

$$\mathcal{L}_\theta = \mathbb{E}[(r + \gamma Q(s', \arg\max_{a'} Q(s',a',\theta),\theta') - Q(s,a,\theta))^2]. \tag{3}$$

The change is to reduce the overestimation in Q-learning.

3.2. Dueling DQN

Dueling DQN [10] splits Q-networks into two separate estimators, advantage function $A(s,a)$ and value function $V(s)$, and factorizes the Q-function into the following form:

$$Q(s,a) = A(s,a) + V(s) - \frac{1}{N_{actions}} \sum_i A(s,a_i). \tag{4}$$

It claims that the values can be estimated more accurate.

3.3. Distributional DQN

Distributional RL learns the distribution of returns instead of expected values. Bellemare et al. [12] proposed categorical DQN to model the value distribution by a set of atoms $\{z_i = V_{min} + i(\frac{V_{max}-V_{min}}{N-1}) \mid 0 \leq i \leq N\}$, then approximate the probability mass of each atom $p^i_\theta(s,a)$. The approximated distributional value function can be denoted by

$$Z_\theta(s,a) = z_i \quad \text{w.p.} \quad p^i_\theta(s,a) = \frac{\sum \exp(\theta_i(s,a))}{\sum_j \exp(\theta_j(s,a))}. \tag{5}$$

To learn a discrete distribution as the target, we project the sampled target distribution onto the discrete support of Z_θ to derive the continuous distribution to multiclass classification, and then minimize the Kullbeck-Leibler divergence

$$L(\theta) = D_{KL}(\Phi \hat{\mathcal{T}} Z_{\theta'}(s,a) \parallel Z_\theta(s,a)), \tag{6}$$

where $\hat{\mathcal{T}} Z_\theta$ is the sampled Bellman update, so the distributional value function can be obtained.

3.4. Noisy Network

NoisyNet [16] is an exploration strategy aiming at inducing stochasticity of the agent's policy, which can be utilized in any deep RL algorithm. NoisyNet simply adds parametric noise in the linear layer of neural models.

$$y = (\mu^w + \sigma^w \odot \varepsilon^w) \cdot x + (\mu^b + \sigma^b \odot \varepsilon^b), \tag{7}$$

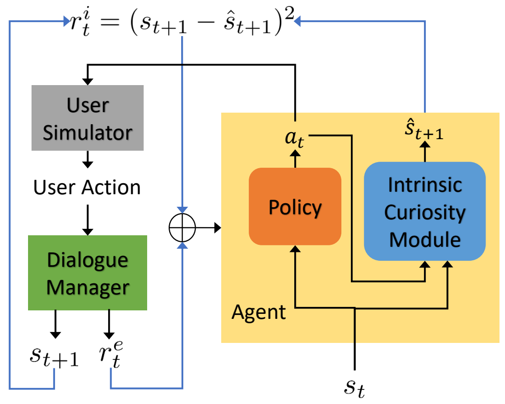


Fig. 3: Intrinsic curiosity module (ICM) in the dialogue system. ICM takes the current state and the action as its input for predicting the next state, and then calculates the intrinsic reward by mean square error.

where $\odot$ indicates the element-wise product, μ and σ are trainable parameters, and ε is the random noise. NoisyNet can be viewed as a normal linear layer $y = w \cdot x + b$ by combining $\mu^w + \sigma^w \odot \varepsilon^w = w$ and $\mu^b + \sigma^b \odot \varepsilon^b = b$. The agent is encouraged to act more randomly and explore more unvisited states with this special network.

3.5. Curiosity-Based Exploration

Curiosity-based exploration is motivated by human infants, who often tend to explore *novel* things in the environment. Pathak et al. [19] proposed a curiosity-based algorithm, which determines how novel the state is by predicting the next environment state, and then guide the exploration based on the curiosity.

Figure 3 illustrates how to apply this technique to the dialogue system framework. The agent shown in the block contains two modules, one for the policy that takes an action a_t to reply to the user and one for an additional intrinsic curiosity module (ICM) that focuses on predicting the next state. Then we can obtain the extrinsic next state s_{t+1} and the reward r_t^e from the dialogue manager. ICM takes a_t and s_t as its input and predicts $\hat{s}_{t+1}$, which can be optimized in a supervised manner. We can calculate mean square error $r_t^i = (s_{t+1} - \hat{s}_{t+1})^2$ as the intrinsic curiosity reward, so the final reward for RL training would be $r_t = r_t^i + r_t^e$.

In practice, we normalize the intrinsic rewards to $\mu = 0$ and $\sigma = 1$ before adding to final rewards. Because ICM would not converge to zero loss, this trick can reduce unnecessary bias caused by intrinsic rewards.

3.6. Prioritized Experience Replay

Off-policy RL algorithms store experience transitions in a replay memory [21] and reuse the experiences to improve data efficiency. Conventional DQN samples transitions from the replay memory uniformly. However, prioritized experience replay [11] assigns each transition a priority according to the last encountered absolute TD error:

$$p_i = |r + \gamma \max_{a'_i} Q(s'_i, a'_i, \theta') - Q(s_i, a_i, \theta)|^\alpha, \quad (8)$$

and samples from the replay with a multinomial distribution $P(i) = \frac{p_i}{\sum_j p_j}$, where α is a hyperparameter that determines the shape of distribution. Theoretically, transitions with higher TD error are more valuable for agent training, so applying a prioritized replay can further improve data efficiency.

Table 1: Number of intents, slots and dialogues in three dataset.

Task	Intents	Slots	Dialogues
Movie-Ticket Booking	11	29	2890
Restaurant Reservation	11	30	4103
Taxi Ordering	11	29	3094

This paper focuses on applying the above variants of DQN to goal-oriented dialogue system scenarios and investigates the performance difference and the suitability of each model.

4. EXPERIMENTS

In order to evaluate the performance of RL agents in dialogue scenarios, we conduct the experiments by only manipulating dialogue policy learning algorithms.

4.1. Benchmark Dialogue System Data

We use the benchmark task-completion dialogue environments provided by Microsoft Dialogue Challenge [22, 23]. The environments include pre-trained NLU and NLG models and rule-based user simulators, so the experiments are conducted by only switching the dialogue policy for fair comparison.

The experiments are performed in three domains: **movie-ticket booking**, **restaurant reservation** and **taxi ordering**. Each domain has its domain-specific intents and slots, and the statistics is shown in Table 1. The goal of each agent is to interact with the user in order to help them achieve specific goals.

4.2. Setup

All model for RL agents are 2-layers perceptrons with hidden sizes of 80, and optimized by `RMSprop` with 0.001 learning rate. The size of experience replay is 10,000. The target network of DQN uses an exponentially moving average soft-update with $\tau = 0.01$. ϵ annealing is employed in ϵ-greedy, and ϵ starts from 0.2 and decays every episode with a decay rate of 0.95. In distributional DQN, we set $N_{atoms} = 51$ (C51 Algorithm), which is the best setting in the original paper, and min/max values are -40 and 80. All learning curves in the results are the average values with 5 different random seeds, and the colored areas between curves are 0.5 times standard deviation in each episode.

A reward function is required for training RL policy. In this task, the agent receives $2 \cdot turns_{max}$ reward when a dialogue successes and $-turns_{max}$ when it fails. To encourage the policy to reach the goal more efficiently, the agent receives -1 penalty every turn. In other words, more turns the agent spends achieving the goal, a lower reward it would get. $turns_{max}$ is set to 40 in the movie-ticket booking domain and 30 in the other two domains.

4.3. Analysis of Different DQN Algorithms

Figure 4 shows the learning curves of 5 variants of DQN in three domains. The curves in the first row plot the success rate over simulation epoch, and the second row plot the average turns.

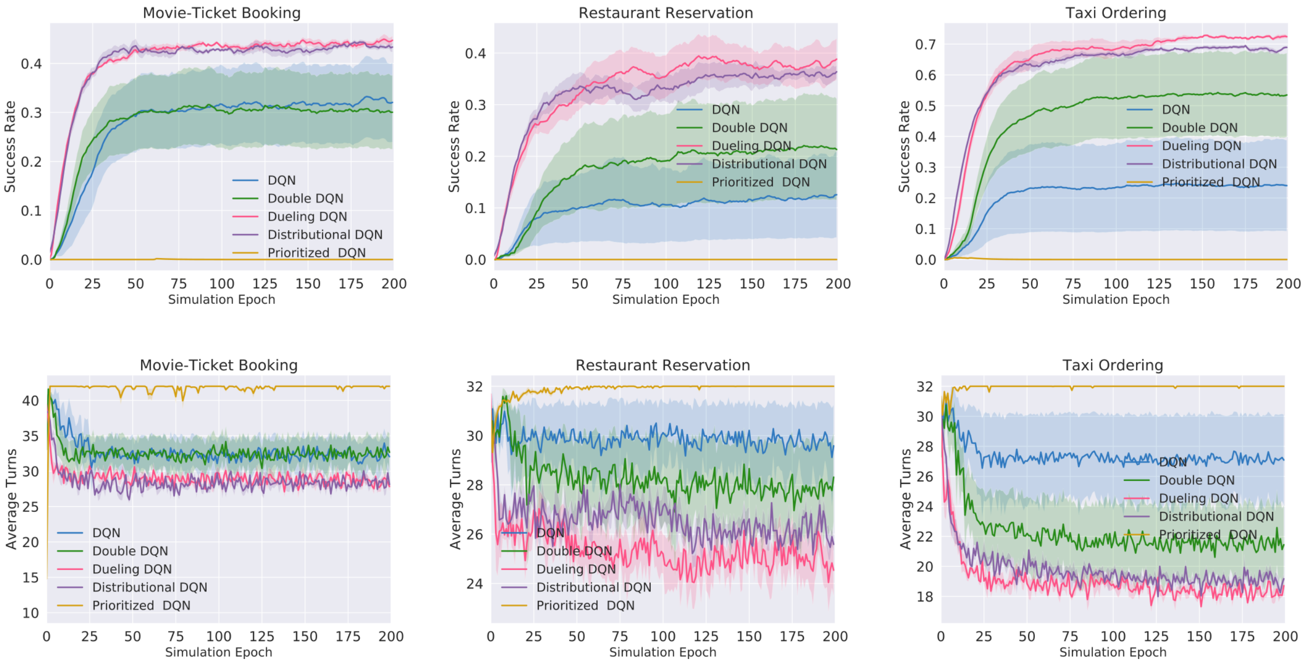


Fig. 4: Learning curves of variants of DQN in three domains. Dueling DQN and Distributional DQN are more stable, especially Dueling perform best. Double DQN only improve a little, and prioritized DQN fails in all tasks.

4.3.1. *Double DQN*

Previous work [13] has experimented these variants of DQN on many of Atari games. Their results show that Double DQN improves very little in most environments, and even harms the performance sometimes. Thus, we examine how this component affects task-completion dialogue policy.

In this task, Double DQN performs better in restaurant and taxi domains, while it seems not to affect significantly in the movie domain. We observe that both conventional DQN and Double DQN have very high standard deviations of success rate and average turns, because some training processes among 5 random seeds got very poor results, where their success rate is close to 0%. This issue does not come from bad random seeds, because increasing the number of random seeds does not help. Hence, the higher standard deviation tells that both DQN and Double DQN are unstable algorithms in this task.

To further analyze the results, we plot the error histogram of the predicted returns for both DQN and Double DQN in Figure 5, and then find that predicted returns of Double DQN only higher about 2 than DQN in average, and the predicted returns of conventional DQN is already very close to actual returns in most dialogue turns. Therefore, we can conclude that DQN does not suffer a lot from the problem of overestimation in this task, so the improvement of Double DQN is limited.

4.3.2. *Dueling DQN*

From Figure 4, Dueling DQN performs best over all other algorithms in three domains. Its performance is very close to Distributional DQN in the movie-ticket domain and outperforms all others in restaurant reservation and taxi ordering. In addition, we observe that Dueling DQN has more stable training processes than DQN and Double DQN, where the standard deviation of both success rate and average turns is much lower. Moreover, it also converges to a better optimal than the other. It seems that the network design of conventional DQN cannot model the Q-function perfectly, while Dueling DQN considering a extra value function can perform better in a dialogue system. According to the above results, we consider Dueling DQN an appropriate algorithm for this task.

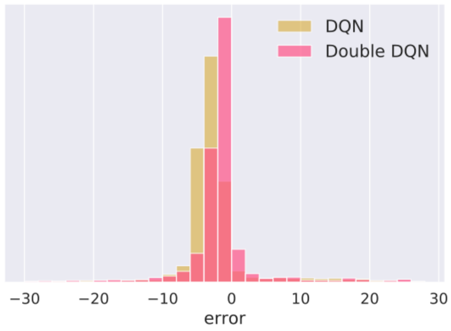


Fig. 5: Histogram of the difference between actual and predicted returns for DQN and Double DQN. We sample transitions from 300 dialogue episodes, then count the error in every interval of 2.

4.3.3. *Distributional DQN*

Utilizing a distributional value function hugely affects the performance of this task. Distributional DQN also produces more stable training processes than DQN and Double DQN, but its success rate and average turns are slightly worse than Dueling DQN. We find that it has lower standard deviation than Dueling DQN in restaurant and taxi domains. The original Distributional DQN [12] claimed that the distributional value function can reduce chattering, leading to instability in Bellman optimality operator. By reducing chattering, the policy can converge more stable and produce less standard deviation

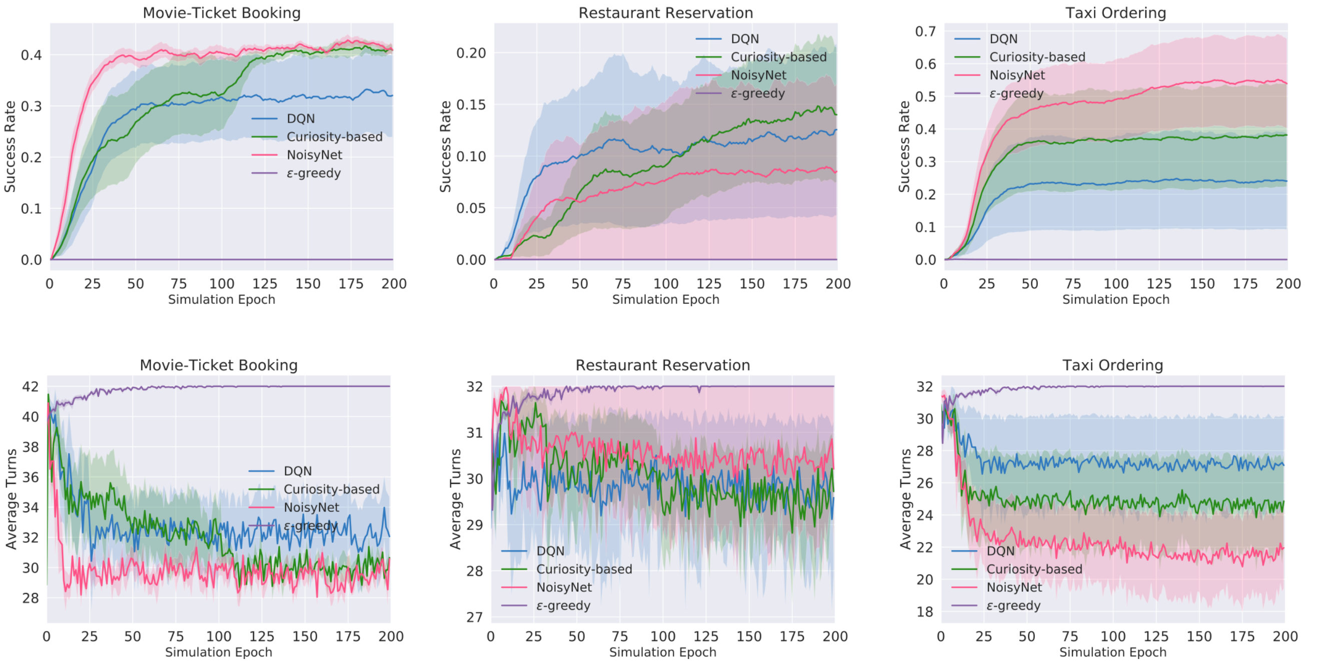


Fig. 6: Learning curves of different exploration strategies in three domains

in different random seeds. The results imply that the contribution of the distributional value function is orthogonal to Dueling DQN.

4.3.4. *Prioritized DQN*

DQN with a prioritized experience replay fails to learn the good policy and gets 0% success rate in all domains. Here the Bellman error in DQN with the prioritized experience replay would converge very soon then gets stuck in a local minimum. Tuning the hyperparameter α lower in prioritized experience replay may make it sample transitions more uniformly, but it still fails. Hence, the experimental results tell that prioritized DQN is not suitable for our target environments.

4.4. Comparison of Exploration Strategies

We compare four exploration strategies on DQN: ϵ-greedy, NoisyNet, curiosity-based and no strategy. The results are shown in Figure 6.

4.4.1. *Curiosity-based Exploration*

In task-completion dialogues, the agent can get positive rewards only when reaching the goal. That means, the agent cannot receive any reward if it makes a wrong decision leading to failure, even though it acts perfectly before. Once the agent makes too many wrong decisions in early training time, the policy may easily get stuck in a local minimum where the agent cannot get any positive reward. In contrast, curiosity-based exploration can always produce meaningful gradient by the dynamic intrinsic rewards to escape the local minimum.

We plot the curves of intrinsic and extrinsic rewards during the training processes in Figure 7. The standard deviation of rewards is higher in early training episodes, because the agent sometimes get stuck in a local minimum, but the intrinsic reward can always help the agent to escape. In conclusion, curiosity exploration does not suffer from the local minimum issue, so it is more suitable for dialogue policy learning scenarios.

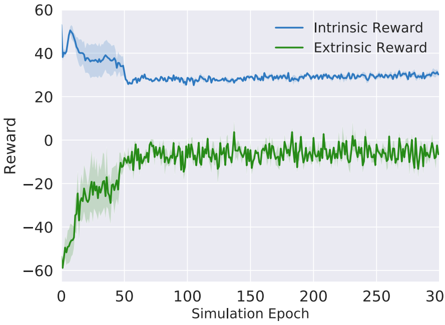


Fig. 7: The rewards of curiosity-based exploration, where intrinsic rewards and extrinsic rewards converge at the same time.

4.4.2. *Noisy Network*

Noisy Network actually improves DQN and beats the other three strategies in movie and taxi domains. However, the performance is very poor in the restaurant domain. Considering that the improvement of curiosity-based is subtle in the restaurant domain, we hypothesize that this domain may not need an efficient exploration strategy. Compare with movie-ticket booking, this domain has a much lower average success rate, but the average turns are also lower. That means that it has less tolerant of wrong actions, and the agent cannot take too many unnecessary explorations, especially just adding noise to make the agent act more randomly.

4.4.3. *ϵ-greedy*

ϵ-greedy fails for policy learning in all domains. Prior results tell that just letting the agent act randomly is not appropriate in every task. In contrast to Noisy Network, ϵ-greedy does not learn with parametric noise, but samples random action directly. This makes the agent harder to reach the successful dialogues, and the agent easily gets stuck in a local minimum.

4.5. Effectiveness of Additity

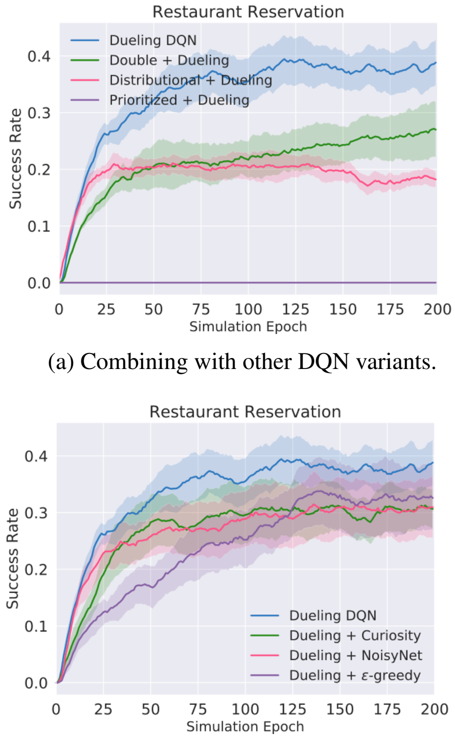
In our task, simply integrating all DQN algorithms do not perform better. To further analyze the effectiveness of each algorithm and whether they can be additive, we integrate different DQN algorithms and see whether the performance is further improved. We test on the combination of Dueling DQN with other four algorithms considering that Dueling DQN performs best in the previous experiments. We conduct the effectiveness test on movie-ticket booking and restaurant reservation domains. However, we find that combining other algorithms with Dueling DQN affect very little, and cannot observe any important finding in movie-ticket booking domain. Thus, we only plot the result of the restaurant reservation domain in Figure 8 for analysis.

4.5.1. *Combining with Different Variants of DQN*

Figure 8a shows the learning curves of Dueling DQN combined with other variants of DQN. We observe that applying Double Q-learning gives a negative effect on Dueling DQN. In Sec. 4.3.1, we have illustrated that Double DQN affects slightly, since DQN does not significantly overestimate the Q-function in this task. On the other hand, the Q-function can be estimated more precisely with Dueling DQN; therefore Double DQN can even allow Dueling DQN to underestimate the Q-function, and further leads to a worse result. In term of the distributional value function, it actually reduces the standard deviation of success rate but performs worse in average. It seems that although it has an orthogonal improvement to Dueling DQN, it harms the optimal policy with dueling network in this task. Last but not least, we observe that prioritized experience replay still fails with Dueling DQN, aligning with the conclusion we have discussed in Sec. 4.3.4.

4.5.2. *Combining with Different Exploration Strategies*

We also combine Dueling DQN with different exploration strategies shown in Figure 8b. Surprisingly, all exploration strategies perform worse than Dueling DQN without any strategy. Furthermore, this only happens in the restaurant reservation domain. In the movie domain, except for Noisy Network which also causes a poor result with Dueling DQN, other two exploration strategies take almost no effect on Dueling DQN. This finding corresponds to the conclusion in Sec. 4.4 saying that the restaurant reservation domain does not need too many exploration actions. Another finding is that ϵ-greedy which fails on conventional DQN in all domains, but works with Dueling DQN.



(a) Combining with other DQN variants.

(b) Combining with exploration stratigies.

Fig. 8: Effectiveness test results in the restaurant reservation domain.

Table 2: Human evaluation results.

Agent	Success Rate (%)	Rating (0-5)
Rule-based	6.42	1.78
DQN	30.8	2.62
Double Dueling DQN	31.1	2.65

4.6. Human Evaluation

To evaluate the system performance from a human perspective, we participate the Microsoft Dialogue Challenge and submit a Double Dueling DQN agent for human evaluation[2]. The results are shown in Table 2, where the DQN and rule-based agents are the baselines provided by the challenge. It is obvious that the Double Dueling DQN agent achieves better success rate and receives slightly better average ratings from the human perspective. The results also demonstrate that Double Dueling DQN is suitable for dialogue systems.

5. CONCLUSION

This paper investigates several variants of DQN applied to goal-oriented dialogue settings, and finds which extension of DQN is suitable for the target scenarios. Among variants of DQN, Dueling DQN and Distributional DQN improve significantly in our benchmark experiments. However, they are not additive. In term of exploration strategies, the experiments show that curiosity-based exploration and NoisyNet improve DQN in most settings of dialogue policy learning. The investigation and analysis from this paper guide the future directions of applying value-based RL for dialogue policy learning. We also found that some common improvement of RL algorithms such as double DQN, ϵ-greedy, do not help dialogue policy learning due to the property difference between dialogues and other RL environments such as Atari games. As a result, we conclude that although it is not naive to choose the best RL algorithm for a specific task, analyzing the characteristics of the task helps choose a suitable algorithm[3].

[2] Dueling DQN, the best model from the above experiments, is not submitted for human evaluation during the competition.

[3] This work was financially supported from the Young Scholar Fellowship Program by MOST in Taiwan, under Grant 108-2636-E002-003

6. REFERENCES

[1] Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz, "End-to-end task-completion neural dialogue systems," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2017, vol. 1, pp. 733–743.

[2] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams, "Pomdp-based statistical spoken dialog systems: A review," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1160–1179, 2013.

[3] Jason D Williams and Steve Young, "Partially observable markov decision processes for spoken dialog systems," *Computer Speech & Language*, vol. 21, no. 2, pp. 393–422, 2007.

[4] James Henderson, Oliver Lemon, and Kallirroi Georgila, "Hybrid reinforcement/supervised learning for dialogue policies from communicator data," in *IJCAI workshop on knowledge and reasoning in practical dialogue systems*. Citeseer, 2005, pp. 68–75.

[5] Milica Gašić and Steve Young, "Gaussian processes for pomdp-based dialogue manager optimization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 28–40, 2014.

[6] 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*, vol. 518, no. 7540, pp. 529, 2015.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[8] 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*, vol. 529, no. 7587, pp. 484, 2016.

[9] Hado Van Hasselt, Arthur Guez, and David Silver, "Deep reinforcement learning with double q-learning.," in *AAAI*. Phoenix, AZ, 2016, vol. 2, p. 5.

[10] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[11] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[12] Marc G Bellemare, Will Dabney, and Rémi Munos, "A distributional perspective on reinforcement learning," *arXiv preprint arXiv:1707.06887*, 2017.

[13] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver, "Rainbow: Combining improvements in deep reinforcement learning," 2017.

[14] Richard S Sutton and Andrew G Barto, *Introduction to reinforcement learning*, vol. 135, MIT press Cambridge, 1998.

[15] Ronald J Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[16] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al., "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[17] Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong, "Deep dyna-q: Integrating planning for task-completion dialogue policy learning," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, vol. 1, pp. 2182–2192.

[18] Shang-Yu Su, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Yun-Nung Chen, "Discriminative deep dyna-q: Robust planning for dialogue policy learning," *arXiv preprint arXiv:1808.09442*, 2018.

[19] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International Conference on Machine Learning (ICML)*, 2017, vol. 2017.

[20] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros, "Large-scale study of curiosity-driven learning," *arXiv preprint arXiv:1808.04355*, 2018.

[21] Long-Ji Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[22] Xiujun Li, Sarah Panda, Jingjing Liu, and Jianfeng Gao, "Microsoft dialogue challenge: Building end-to-end task-completion dialogue systems," *arXiv preprint arXiv:1807.11125*, 2018.

[23] Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen, "A user simulator for task-completion dialogues," *arXiv preprint arXiv:1612.05688*, 2016.