Curiosity-Driven Variational Autoencoder for Deep Q Network

Gao-Jie Han, Xiao-Fang Zhang(✉), Hao Wang, and Chen-Guang Mao

School of Computer Science and Technology, Soochow University, Suzhou, China
xfzhang@suda.edu.cn

Abstract. In recent years, deep reinforcement learning (DRL) has achieved tremendous success in high-dimensional and large-scale space control and sequential decision-making tasks. However, the current model-free DRL methods suffer from low sample efficiency, which is a bottleneck that limits their performance. To alleviate this problem, some researchers used the generative model for modeling the environment. But the generative model may become inaccurate or even collapse if the state has not been sufficiently explored. In this paper, we introduce a model called Curiosity-driven Variational Autoencoder (CVAE), which combines variational autoencoder and curiosity-driven exploration. During the training process, the CVAE model can improve sample efficiency while curiosity-driven exploration can make sufficient exploration in a complex environment. Then, a CVAE-based algorithm is proposed, namely DQN-CVAE, that scales CVAE to higher dimensional environments. Finally, the performance of our algorithm is evaluated through several Atari 2600 games, and the experimental results show that the DQN-CVAE achieves better performance in terms of average reward per episode on these games.

Keywords: Reinforcement learning · Deep Q learning · Exploration · Variational autoencoder

1 Introduction

Reinforcement learning (RL) [17] is a popular area of current research across various fields. The goal of the RL algorithm is achieving the target task by maximizing the expected rewards provided by the environment. Recently, Mnih et al. proposed Deep Q learning (DQN) [3,13,14], which combines deep learning (DL) [12] and RL, achieving a remarkable result in classic games such as Atari 2600 games.

Although DQN and its extensions have tremendous success in Atari 2600 environment, at the beginning of the training process, the current DQN algorithms require millions of training samples based on the random policy before any optimal policy is trained, and insufficient sample diversity will result in the slow training speed. However, in many scenarios, the training sample may be difficult or time-consuming to obtain. Thus, some researchers attempt to represent

H. W. Lauw et al. (Eds.): PAKDD 2020, LNAI 12084, pp. 764–775, 2020.
https://doi.org/10.1007/978-3-030-47426-3_59

the actual environment by using a generative model [1,6,8] to improve sample efficiency. When the generative model is sufficiently trained, the DRL algorithm can be trained without interacts with the actual environment. In [1,2,6,8], it is confirmed that the agent can learn the optimal policy only use generate training samples. However, these generative models may become inaccurate and even collapse where the state-action pair insufficient explored [1,2,6].

Moreover, inadequate exploration of the environment may also result in slow learning speed. In traditional model-free DRL algorithms, they rely on simple heuristics exploration strategies such as ϵ-greedy. However, these exploration strategies are often trapped in local minima of the state space, which leads to the state space may be partially observed in the high-dimensional environment. Curiosity-driven exploration uses an extra reward signal that inspired the agent to explore the state that has not been sufficiently explored before. It tends to seek out the unexplored regions more efficiently in the same amount of time.

In this paper, we propose a new algorithm called Curiosity-driven Variational Autoencoder (CVAE), which uses a CVAE to model the environment in latent space to improve sample efficacy while curiosity-driven exploration to make a sufficient exploration. Then we apply the CVAE to DQN and its variants denoted as DQN-CVAE. In addition, we provide experimental results on several Atari 2600 games. Experimental results show that the DQN-CVAE algorithm can improve the exploration and performance of the agent.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 elaborates on the DQN and VAE algorithms. Section 4 offers an overview of our approach, then describes DQN-CVAE algorithm in detail. Section 5 provides our experimental setup and results. Section 6 concludes.

2 Related Work

Recently, some researchers attempt to model the environment by using a generative model to improve sample efficiency. The notion of modeling the environment in latent space may trace back to [8], which proposed DARLA, an architecture for modeling the environment with β-Variational Autoencoder, and have applied the latent features for transfer learning across multiple environments. In [6], Ha et al. proposed World Model, an architecture for modeling the environment using a VAE model and a recurrent neural network (RNN) model, which shows that the agent can learn the optimal policy only use generate training samples. Similarly, Anderson et al. [1] proposed Dreaming Variational Autoencoder, an architecture for modeling the environment using VAE and RNN, which uses the real trajectories from the actual environment to imitate the behavior of the actual environment. Conversely, Anderson et al. [2] found that in high-dimensional tasks, simple heuristics exploration are often trapped in local minima of the state space, which may cause the generative model to become inaccurate or even collapse.

Previous research on exploration technique may solve the problem that the agent achieves a sufficient exploration in the high-dimensional task. Many researchers focus on using the intrinsic reward to drive the agent to make an efficient exploration. Kulkarni et al. [11] suggested a hierarchical DRL model in which the agent receives the extrinsic reward and the intrinsic reward at different temporal scales. Stadie et al. [16] introduced incentivizing exploration, which use extra reward signal to encourage the agent to visit the state-action pairs that it has not sufficiently explored. Pathak et al. [5,15] proposed an exploration method called curiosity-driven exploration method. The main idea of curiosity-driven exploration is to attempt to use the intrinsic reward to drive an agent to explore trajectories that it has not visited frequently. Hoothooft et al. [9] suggested a curiosity-driven based method called Variational Information Maximizing Exploration, which uses the information gain as an intrinsic reward and achieves a better performance than heuristic exploration methods across various continuous control tasks.

3 Background

3.1 Deep Q Network

DQN combines Q learning and DL, which use the experience replay mechanism and target network mechanism are used to alleviate learning instability [13,14]. The experience replay mechanism is sampling a fixed number of training samples from experience replay pool D uniformly at random. At each discrete time step t, agent receives a state s_t, and selects an action a_t based on ϵ-greedy policy with respect to the action values. As a feedback, agent gets a reward r_t and receives next state s_{t+1}, then (s_t, a_t, r_t, s_{t+1}) is stored as a sequence in experience replay pool D, and a fixed number of samples are taken from the training process as a network input.

DQN uses two independent deep networks, the current value network $Q(s, a; \theta)$ with parameters θ and the target value network $Q(s, a; \theta^-)$ with parameters θ^-, where DQN learns the parameters of the network $Q(s, a; \theta)$ online, and the parameters θ^- is periodically copied by θ. The loss function is determined by the mean square error of the target value function and the current value function. The corresponding formula is shown in Eq. (1):

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}[(r + \gamma\, max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2] \tag{1}$$

In order to solve the minimized loss function, the parameter θ is derived in Eq. (1). The gradient update is shown in Eq. (2):

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

3.2 Variational Autoencoder

VAE is a generative model capable of learning unsupervised latent representations of complex high-dimensional data [10]. The VAE model consists of two parts: encoder $q_\phi(z|x)$ and decoder $p_\theta(x|z)$. The encoder consumes the sample x, yielding the input in latent space z, then z is fed into decoder to predict sample x. The key idea of the VAE is to learn the marginal likelihood of a sample

x from a distribution parametrized by generative factors z. Thus, $q_\phi(z|x)$ is the variational approximation for the true posterior $p_\theta(z|x)$. The marginal likelihood of a data point x can take following form:

$$\log p_\theta(x) = \mathcal{L}(x;\theta,\phi) + D_{KL}(q_\phi(z|x)||p_\theta(z|x)) \tag{3}$$

Since the true data likelihood is usually intractable, instead, the VAE optimizes an evidence lower bound (ELBO) which is a valid lower bound of the true data log likelihood, denoted as:

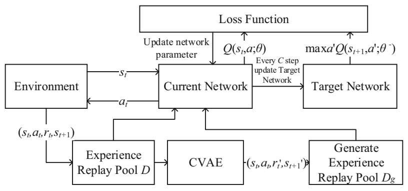
$$\mathcal{L}(x;\theta,\phi) = \mathbb{E}_{q_\phi}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)) \tag{4}$$

$\mathcal{L}(x;\theta,\phi)$ consists of two terms: the first term can be considered as reconstruction loss, and the second term is approximated posterior $q_\phi(z|x)$ from prior $p(z)$ via KL-divergence. In practice, q_ϕ and p_θ are implemented via deep neural networks, and prior $p(z)$ usually sets to follow Gaussian distribution $N(0,1)$.

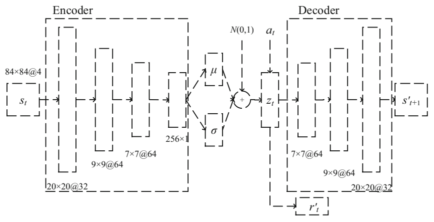
4 Curiosity-Driven Variational Autoencoder

We propose the Curiosity-driven Variational Autoencoder (CVAE), which combines curiosity-driven exploration with the VAE model. The CVAE model uses the prediction error as an intrinsic reward to drive the agent to make a sufficient exploration, which can improve the quality of the generate training samples.

The DQN-CVAE model is consists of two components: the DQN reinforcement learning method and the CVAE model. The structure of DQN-CVAE is presented in Fig. 1(a). Since we use a CVAE model to generate training samples, an additional experience replay pool D_g is used to store up the training samples generated by the CVAE model.



(a) DQN-CVAE



(b) CVAE

Fig. 1. Illustration of the DQN-CVAE model.

The structure of CVAE model is shown in Fig. 1(b). During the training process, the model consumes state s_t, yielding the input encoded in latent space $h_t = [\mu, \sigma]$, which represents a concatenated form of the mean μ and the standard deviation σ, then the h_t is reparametrized into a posterior variable z_t. Then inject the action a_t to the z_t, latent parameter z_t element-wise addition with a_t is fed into the decoder network to predict the next state s'_{t+1}. The predict next state s'_{t+1} is compared with the real state s_{t+1} given by the environment after the action a_t is taken.

To improve the generate sample quality. At time step t we consider a training sample $x_t = (s_t, a_t, r_t, s_{t+1})$ from D. We feed the state s_t and action a_t into encoder as an input, and predict the reward r_t and next state s_{t+1}. In this case, we divided the training sample x_t into two pairs as follows:

$$x_t = [(s_t, a_t), (s_{t+1}, r_t)] \tag{5}$$

where (s_t, a_t) represents the current state-action pair, and (s_{t+1}, r_t) represents the next state pair, which obtains from the agent interacts with the environment. Then, we focused on (s_{t+1}, r_t), we use the KL divergence as prediction error as follows:

$$e_t = D_{KL}((s_{t+1}, r_t)||(s'_{t+1}, r'_t)) \tag{6}$$

Moreover, we use curiosity-driven exploration to improve the efficiency of exploration. An intrinsic reward associated with e_t drives the agent to make a sufficient exploration, the reward function is modified as follows:

$$r'_t = r'_t + \beta e_t \tag{7}$$

where β is the weighted variables.

According to Eq. (4), the loss function of the VAE model consists of two parts: reconstruction loss and latent space loss. Thus, different from the Eq. (4), we use the prediction error e_t as reconstruction error, the loss function is computed by the following formula:

$$\mathcal{L}_{cvae} = e_t - D_{KL}(q_\phi(z_t|s_t)||N(0,1)) \tag{8}$$

The DQN-CVAE algorithm is presented in Algorithm (1). During the learning process, the agent collects the training samples (s_t, a_t, r_t, s_{t+1}) from many episode, and accumulates it as a experience replay pool D. The VAE model is trained using the real training sample in D and generates a new training sample $(s_t, a_t, r'_t, s'_{t+1})$. At the same time, the prediction error e_t is used to predict the intrinsic reward. Then, add the generate sample to D_g, which is an experience replay pool follow the First-in First-out principle to store generate training samples. Next, turn to the DQN part, a fixed number of samples from D and D_g are selected as a minibatch according to the proportion factor g and provided to the agent for training the action-value function and learning the optimal strategy. Besides, during the training process, the VAE model continues to generate training samples and add them to D_g to speed up the learning speed. Although the CVAE model increases the size of the parameter of the neural network, the CVAE model is in parallel with the agent, which does not significantly increase the time complexity of the algorithm, but speed up the learning speed.

Algorithm 1. DQN-CVAE

```
Initialize replay memory $D$ with capacity $N$, generate replay memory $D_g$ with capacity $N_g$, minibatch size $M$, proportion factor $g$
Initialize the action-value function $Q$ with random weight $\theta$
Initialize the target action-value function $Q$ with weight $\theta^-$
for episode=1, $I$ do
    Observe state $s_0$
    for t=1,$T$ do
        Choose an action $a_t$ based on $\epsilon$-greedy policy
        Observe transition$(s_t, a_t, r_t, s_{t+1})$
        Store transition$(s_t, a_t, r_t, s_{t+1})$ in $D$
        /*CVAE part*/
        Sample random minibatch of transition $(s_t, a_t, r_t, s_{t+1})$ from $D$
        Generate transition $(s_t, a_t, r'_t, s'_{t+1})$
        Compute the prediction error $e_t$
        Store transition $(s_t, a_t, r'_t + \beta e_t, s'_{t+1})$ in $D_g$
        /*DQN part*/
        Random sample $M \times (1-g)$ of transition $(s_j, a_j, r_j, s_{j+1})$ from $D$
        Random sample $M \times g$ of transition $(s_j, a_j, r_j, s_{j+1})$ from $D_g$
        Set
            $y_j = \begin{cases} r_j & \text{if episode terminates at step } j{+}1 \\ r_j + \gamma max_{a'} Q(s_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
        Perform a gradient descent step on $(y - Q(s, a; \theta))^2$ with respect to the network parameters $\theta$
        Every $C$ step update $\theta^- = \theta$
    end for
end for
```

5 Experiments

5.1 Research Questions

In these experiments, there are several research questions (RQ) that we consider. For starters, we wish to know whether our algorithm leads to improved DQN. Then, we also want to know whether our algorithm can apply in other DQN extensions. Finally, we investigate how the proportion factor of g affects the performance of our algorithm.

RQ1: Does the DQN-CVAE improve the performance of the DQN?

RQ2: Does the DQN-CVAE improve the performance of other DQN variants?

RQ3: How does the proportion factor g affect the performance of the DQN-CVAE?

5.2 Experimental Environment and Setup

Experimental Environment. We use the Atari 2600 game environment in the OpenAI gym [4] as the experimental environment to evaluate the performance of our proposed algorithm. OpenAI gym is an open-source toolkit that provides a wide variety of Atari 2600 game interfaces. Five games were used in our experiments. A brief introduction to these games is presented in Table 1.

Table 1. A brief introduction of some Atari games

Game	Action number	Introduction
Alien	18	Agent avoids enmenies and reach the target point
BeamRider	9	Agent avoids bullets and hits moving enemies
Kangroo	18	Agent climbs through stairs and avoids obstacles
Seaquest	18	Agent evades obstacles and attacks enemies under water
SpaceInvaders	6	Agent evades and attacks the enemies

Experimental Setup. In order to compare the performance of different algorithms, all algorithms use the same network architecture and hyperparameters settings. The main hyperparameter settings are shown in Table 2.

Table 2. Main hyperparameters and their values

Hyperparameter	Value
Minibatch size	32
Discount factor	0.99
Learning rate	2.5×10^{-4}
Initial exploration factor	0.96
Final exploration factor	0.1
Replay start size	500000
Target network update frequency	1×10^4
Experience replay pool size	1×10^6
Generate experience replay pool size	1×10^5
RMSprop momentum coefficient	0.95
Frame skip rate	4

The DQN-CVAE algorithm consists of two parts: the DQN model and the CVAE model. The network architecture used in DQN and DDQN is the same as the study of Mnih et al. [14] and Hasselt et al. [7]. There are 3 convolutional layers, 2 full-connected layers and 3 deconvolutional layers in CVAE model. The structure used in CVAE is shown in Fig. 1(b).

Evaluation Criteria And Comparison Algorithms. In the Atari environment, we use the average rewards per episode as the evaluation criteria and uses 200 epoch as the training periods, in which 50,000 steps were used to train the network parameters, a total of 100,000,000 steps are trained.

In these experiments, we compare the training performance of two original network (DQN and DDQN) and three networks with DQN-CVAE(DQN-CVAE, DQN-VAE, CDQN), we denote these algorithms as follows:

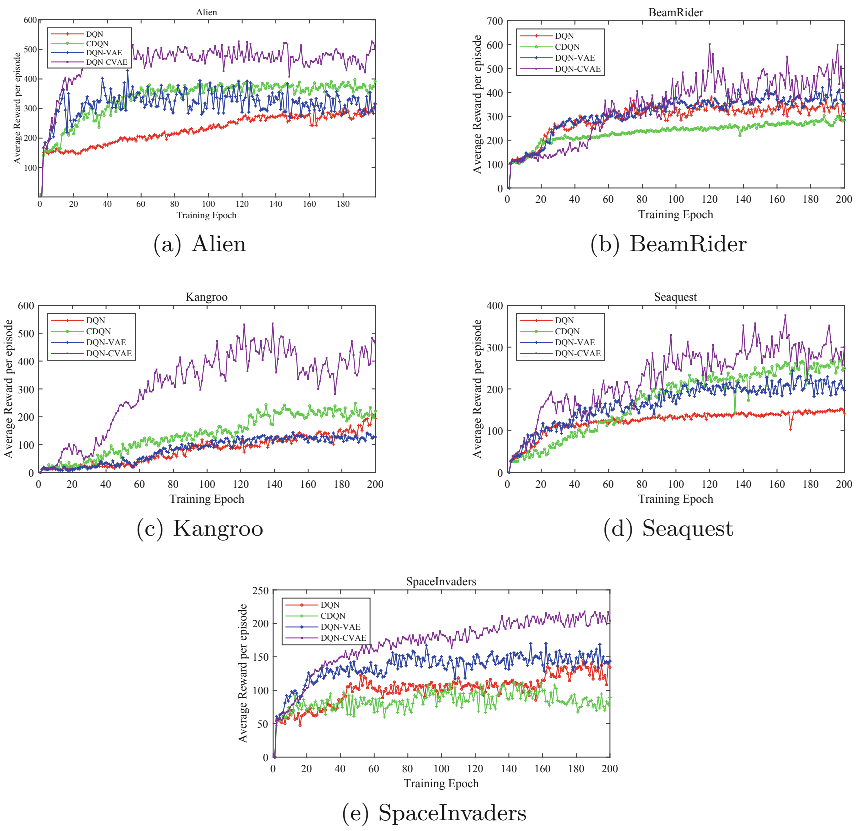
(1) DQN and DDQN are deep Q learning [13,14] and double deep Q learning [7], which are benchmark comparison algorithms;
(2) CDQN and CDDQN are DQN or DDQN based on curiosity-driven exploration [5,15];
(3) DQN-VAE and DDQN-VAE add a VAE structure to DQN or DDQN, which only uses the VAE model to alleviate insufficient sample diversity;
(4) DQN-CVAE and DDQN-CVAE are our proposed algorithms that combine (2) and (3). It was different from (3) in that we use curiosity-driven exploration to improve the efficiency of exploration.

5.3 Experimental Result

RQ1 asks whether the DQN-CVAE algorithm can improve the performance of DQN. To answer this question, we first compared the performance of DQN, CDQN, DQN-VAE, and DQN-CVAE during each epoch of training. The results are presented in Fig. 2. The x-axis represents the training epoch, and the y-axis represents the average rewards per episode.

As expected, it can be observed that the average rewards per episode of DQN-CVAE are obviously higher than other algorithms. For models with curiosity-driven exploration (DQN-CVAE, CDQN), the performance is significantly improved than DQN. It is indicated that insufficient exploration has existed in some Atari games, which confirms the contribution of the curiosity-driven exploration. For model with VAE (DQN-CVAE, DQN-VAE), we have found that the DQN-CVAE has a better performance than DQN-VAE. It is illustrated that CVAE improves the sample efficiency can improve the generative model performance. However, in some scenarios, it can be seen that the performance of DQN-VAE has dropped below the DQN. It is not surprising, as DQN-VAE would be inaccurate if the state space is sufficiently explored.

To confirm that DQN-CVAE can perform well after training, we compared the performance of DQN, CDQN, DQN-VAE, and DQN-CVAE on five games after training. For each game, the training completed model will be tested 100 times. Each test will receive a score that represents the average reward per episode. The result in terms of the average reward per episode is reported in Table 3. The result demonstrated that the performance of DQN-CVAE is more effective than DQN in the testing process. It is indicated that DQN-CVAE can improve the performance of DQN.



(a) Alien (b) BeamRider

(c) Kangroo (d) Seaquest

(e) SpaceInvaders

Fig. 2. Comparisons of DQN, CDQN, DQN-VAE and DQN-CVAE

RQ2 asks whether the DQN-CVAE algorithm can improve the performance of other DQN variants. In these experiments, we keep all these settings as in RQ1, but adopt DDQN [7] instead of DQN as the original network, and compares the performance of DDQN, CDDQN, DDQN-VAE and DDQN-CVAE on 5 Atari 2600 games. Figure 3 shows the performance of each algorithm. The x-axis represents the training epoch, and the y-axis represents the average reward per episode.

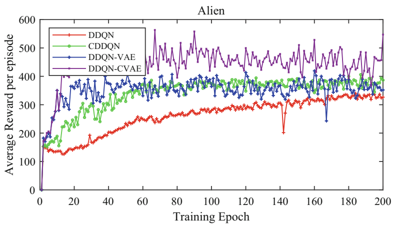
As shown in Fig. 3, similar to the result of Fig. 2, the performance of the DDQN-CVAE is better than other algorithms. So, we can confirm that DQN-CVAE can perform well in the training process when it applies to DQN and its extensions. Then, we also compared the performance of DDQN, CDDQN, DDQN-VAE, and DDQN-CVAE on 5 Atari 2600 games after training. Table 3 lists the results of these four algorithms for DDQN.

Overall, these results indicated that, in the Atari environment, DQN-CVAE outperforms than DQN in the training and testing process. It is indicated that DQN-CVAE uses the CVAE to model the environment that can improve the performance of the DQN. Besides, DQN-CVAE can successfully apply to other DQN variants, which demonstrates that the DQN-CVAE is a simple extension that can be easily integrated with other DQN variants.

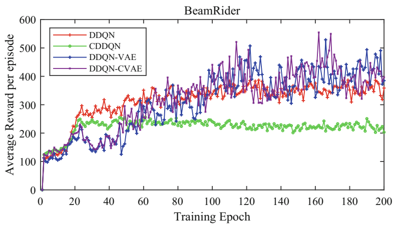
Table 3. Average score after training

Game	DQN		DQN-VAE		CDQN		DQN-CVAE	
	DQN	DDQN	DQN	DDQN	DQN	DDQN	DQN	DDQN
Alien	896	965	945	1104	1052	1142	1087	**1164**
BeamRider	1435	1642	1726	1942	1652	1820	1955	**1974**
Kangroo	2744	2582	2862	2647	2665	2674	**3353**	3287
Seaquest	1215	1285	1326	1426	1462	1508	1527	**1683**
SpaceInvaders	457	559	463	482	342	361	563	**575**

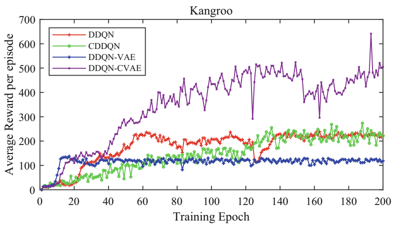
* the best results are highlighted in bold.



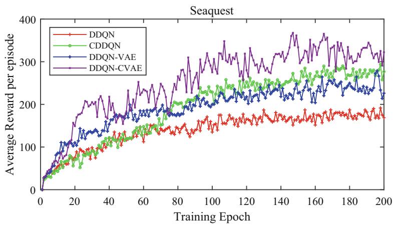
(a) Alien



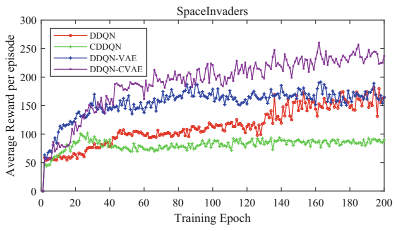
(b) BeamRider



(c) Kangroo



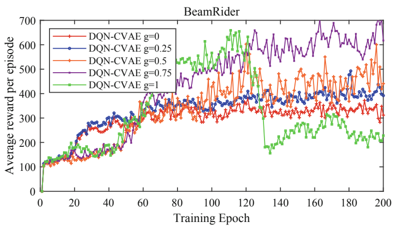
(d) Seaquest



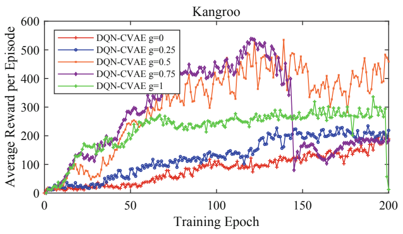
(e) SpaceInvaders

Fig. 3. Comparisons of DDQN, CDDQN, DDQN-VAE and DDQN-CVAE

RQ3 asks how the proportion factor g affects the performance of the DQN-CVAE algorithm. We investigate the performance of DQN-CVAE with different values of g, which is $g = 0, 0.25, 0.5, 0.75, 1$, respectively. Figure 4 presents the result of DQN-CVAE with various g. It can be seen that the performance of DQN-CVAE becomes better with the increasing of the value of g. However, we also can observe that, in some scenarios like Fig. 4(b), the average reward per episode has a large fluctuation while g is greater than 0.75. So, in these experiments, it is recommended that $g = 0.5$.



(a) BeamRider



(b) Kangroo

Fig. 4. Comparisons of various g value for Atari games

After conducting three sets of experiments, we confirm that DQN-CVAE can achieve better performance than DQN in the Atari environment during the training and testing process. Besides, we confirm that CVAE model can be easily applied in other model-free DRL algorithms. Moreover, the performance of DQN-CVAE is affected by the value of the proportion factor of g. With the increasing value of g, the performance of DQN-CVAE become better in general.

6 Conclusion

In this paper, we introduce the CVAE algorithm, which combines the VAE model and curiosity-driven exploration. The VAE model can improve sample efficiency, and curiosity-driven exploration can make a sufficient exploration to improve the accuracy of the VAE model. CVAE algorithm can be applied in the traditional model-free DRL algorithm, such as DQN and DDQN. The experiment results show that the DQN-CVAE algorithm can improve the exploration and performance of the agent, and we also confirm that the CVAE algorithm is flexible since it can be easily integrated with other DQN variants.

In future work, more experiments can be conducted on other Atari 2600 games to conform to the generalization of our CVAE algorithm. Besides, a priority can be used to select to generate samples from D_g based on intrinsic reward. Another direction is to make g become a dynamic learnable parameter with the use of neural networks.

Acknowledgments. This work was supported in part of the National Natural Science Foundation of China (61772355, 61876119), Suzhou Technology Development Plan (SYG201807), and the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

1. Andersen, P.-A., Goodwin, M., Granmo, O.-C.: The dreaming variational autoencoder for reinforcement learning environments. In: Bramer, M., Petridis, M. (eds.) SGAI 2018. LNCS (LNAI), vol. 11311, pp. 143–155. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04191-5_11
2. Andersen, P.A., Goodwin, M., Granmo, O.C.: Towards model-based reinforcement learning for industry-near environments. arXiv preprint arXiv:1907.11971 (2019)
3. Arulkumaran, K., Deisenroth, P.M., Brundage, M., Bharath, A.A.: Deep reinforcement learning: a brief survey. IEEE Signal Process. Mag. **34**(6), 26–38 (2017)
4. Brockman, G., Cheung, V., Pettersson, L., et al.: Openai gym. arXiv preprint arXiv:1606.01540 (2016)
5. Burda, Y., Edwards, H., Pathak, D., et al.: Large-scale study of curiosity-driven learning. arXiv preprint arXiv:1808.04355 (2018)
6. Ha, D., Schmidhuber, J.: World models. arXiv preprint arXiv:1803.10122 (2018)
7. Hasselt, V.H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16), pp. 2094–2100 (2016)
8. Higgins, I., Pal, A., Rusu, A.A., et al.: DARLA: improving zero-shot transfer in reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 1480–1490. JMLR. org (2017)
9. Houthooft, R., Chen, X., Duan, Y., et al.: VIME: variational information maximizing exploration. In: Advances in Neural Information Processing Systems, vol. 29, pp. 1109–1117. Curran Associates, Inc. (2016)
10. Kingma, P.D., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
11. Kulkarni, D.T., Narasimhan, R.K., Saeedi, A., Joshua, T.B.: Hierarchical deep reinforcement learning: integrating temporal abstraction and intrinsic motivation. In: Advances in Neural Information Processing Systems, pp. 3675–3683 (2016)
12. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
13. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Playing atari with deep reinforcement learning. In: NIPS Deep Learning Workshop (2013)
14. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
15. Pathak, D., Agrawal, P., Efros, A.A., Darrell, T.: Curiosity-driven exploration by self-supervised prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 16–17 (2017)
16. Stadie, C.B., Levine, S., Abbeel, P.: Incentivizing exploration in reinforcement learning with deep predictive models. arXiv preprint arXiv:1507.00814 (2015)
17. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)