Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 1238571, 8 pages
https://doi.org/10.1155/2022/1238571

Research Article

Count-Based Exploration via Embedded State Space for Deep Reinforcement Learning

Xinyue Liu,[1] **Qinghua Li**,[1] **and Yuangang Li**[2]

[1]*School of Software, Dalian University of Technology, Dalian 116620, China*
[2]*Faculty of Business Information, Shanghai Business School, Shanghai 200235, China*

Correspondence should be addressed to Yuangang Li; liyuangang76@163.com

Received 15 December 2021; Revised 17 March 2022; Accepted 19 April 2022; Published 17 May 2022

Academic Editor: Changqing Luo

Count-based exploration algorithms have shown to be effective in dealing with various deep reinforcement learning tasks. However, existing count-based exploration algorithms cannot work well in high-dimensional state space due to the complexity of state representation. In this paper, we propose a novel count-based exploration method, which can explore high-dimensional continuous state space and combine with any reinforcement learning algorithms. Specifically, by introducing the embedding network to encode the state space and to merge the states with similar key characteristics, we can compress the high-dimensional state space. By utilizing the state binary code to count the occurrence number of states, we generate additional rewards which can encourage the agent to explore the environment. Extensive experimental results on several commonly used environments show that our proposed method outperforms other strong baselines significantly.

1. Introduction

Reinforcement learning (RL), which was aimed at learning an optimal control strategy to maximize the reward from the environment, has achieved great success in various complex tasks, such as video games [1] and robot controlling [2]. One of the core problems of RL methods is how the agent should take trade-off decisions between the exploration of new actions and the selection of the best action based on existing knowledge. Although there are simple and theoretically guaranteed heuristic exploration methods for tabular RL algorithms, such as the ε-greedy strategy [3] and entropy regularization [4], these methods cannot be easily extended to a high-dimensional space environment due to the large state space and the complexity of state representation. Therefore, developing a common and simple exploration method is an important research direction.

In this paper, we propose a novel count-based exploration method via embedded state space for reinforcement learning. The core idea is to compress the high-dimensional state space by extracting the embedded representation of the state space and merging similar states. We use an action prediction model to train the state embedding network for obtaining a better state feature space. Take our human agent as an example. Assume someone is playing a racing game where the screen changes as the car goes ahead, as shown in Figure 1. The track changing in the yellow box will affect the next move, while the sky changing in the red box will not. We should focus on the state characteristics that affect the choice of actions. In summary, the contributions of this paper are as follows: (1) we propose a new count-based exploration method which is suitable for high-dimensional state space. And our method can be directly applied to most different RL algorithms. (2) We design a general mechanism for optimizing feature representations by introducing the embedding network and the action prediction model. (3) We conduct experiments on several games from the Atari [1] and achieve near state-of-the-art results, especially with fewer training epochs.

The rest of this article is organized as follows: in Section 2, we review the related work. In Section 3, we introduce some definitions and propose our novel exploration method. In Section 4, we show the experimental results on different kinds of environments. Finally in Section 5, we conclude the paper and point out the future work.

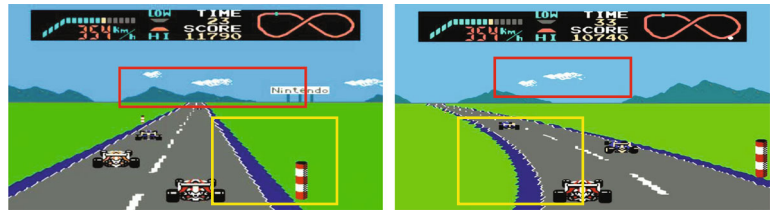

Figure 1: F1-race game screen. Red box line: state features that do not affect the choice of actions. Yellow box line: state features that affect the choice of actions.

2. Related Work

Many different approaches have been proposed in recent years to address the balance between exploration and exploitation. These methods can be divided into two types: count-based exploration methods and curiosity-based exploration methods. The former methods count the occurrence number of states and convert this number into a reward to encourage exploring states with higher rewards. One of the best-known approaches is the UCB bandit algorithm [5], which selects an action a_t at time t to maximize the upper confidence bound $\hat{r}(a_t) + \sqrt{2 \log t / n(a_t)}$, where $\hat{r}(a_t)$ is the estimated reward and $n(a_t)$ is the occurrence number of action a_t being previously chosen. Model-Based Interval Estimation-Exploration Bonus (MBIE-EB) of [6] has similar structure. It counts state-action pairs $n(s, a)$ with a table and adds a bonus reward of the form $\beta/\sqrt{n(s, a)}$ to encourage exploring less-visited pairs. It is proved by [7] that square root inverse correlation is optimal. Tang [8] uses hash functions to encode the state space, subsumes similar states into a single counter, and explores based on the counter's value. Martin et al. [9] generalize a probability density model by the characteristic representation of the state space and use this model to pseudocount. Curiosity-based exploration methods offer additional rewards based on the principle of optimism in the face of uncertainty [10]. These methods encourage the agent to choose actions that increase uncertainty about the value estimate. Classical examples utilize upper confidence bound [11] and Thompson sampling [12] for the stochastic sampling of actions. Recent algorithms combine these ideas with finer uncertainty, making them suitable for large state spaces that require deep exploration [13–15]. Dynamic auto-encoder (Dynamic-AE) [16] is proposed to compress the state space. The distance between predicted state and real state is computed in this compressed state space. And intrinsic rewards are defined by this distance. Pathak et al. [17] use a self-supervised inverse dynamics model to predict the next state based on the current state-action pair. Then, they use the error between prediction and reality to generate curiosity. Savinov et al. [18] propose a new curiosity definition that marks the novelty of states by reachability. The episodic curiosity module (ECO) uses an episodic memory pool to store part of visited states. To compute the state novelty, ECO compares each state with states in memory. If the current state is far from the states contained in memory, the agent is rewarded an intrinsic reward.

Several recent studies have discussed the generalization of reinforcement learning and designed procedurally generated environments to test the generalization of reinforcement learning [19–21]. More recent papers show that traditional exploration methods fall short in procedurally generated environments and address this issue with new exploration methods [22, 23]. [24] proposes a new perspective of exploration bonus in episode-level data and achieves significantly SOTA performance on procedurally generated benchmarks. In the field of multiagent reinforcement learning (MARL), the study on exploration is roughly at the preliminary stage. Most of these exploration methods extend the ideas in the single-agent setting and propose different mechanisms by integrating the characteristics of deep MARL. Compared to the RL exploration, the dimensions of the state-action space increase rapidly as the number of agents increases in MARL. Zhou et al. [25] propose to treat the Q-function as a high-order high-dimensional tensor. Then, they approximate the Q-function with factorized pairwise interactions. [26] adopts a similar factorization approach in the search space to solve this problem.

3. The Proposed Method

In this work, we propose a novel count-based exploration method via embedded state space for deep reinforcement learning. Our method can be divided into two major parts: (1) Embedding Network and Action Prediction Model and (2) Count-Based Extra Bonus Generator. For different RL tasks, we first collect state information by agents randomly interacting with the environment. These data will be used to train the embedding network that can represent state features better. Then, we count the occurrence number of states based on the embedded feature representation. Finally, we generate extra bonus and add it to RL algorithms for training the agent.

3.1. Notations. Reinforcement learning (RL) [3] addresses the task of learning from interactions to achieve goals. It is usually formulated as an MDP $\langle S, A, P, R, \gamma \rangle$, where S is the set of states of the environment, A is the set of available actions, $P : (S \times A) \times S \longrightarrow [0, 1]$ is the state transition distribution, $R : (S \times A) \times S \longrightarrow R$ is the reward function, and γ is the discount factor. The agent is formally a policy $\pi : S \longrightarrow A$ that maps a state to an action. At timestep t, the agent is in a state $s_t \in S$, receives a reward r_t, and takes an action

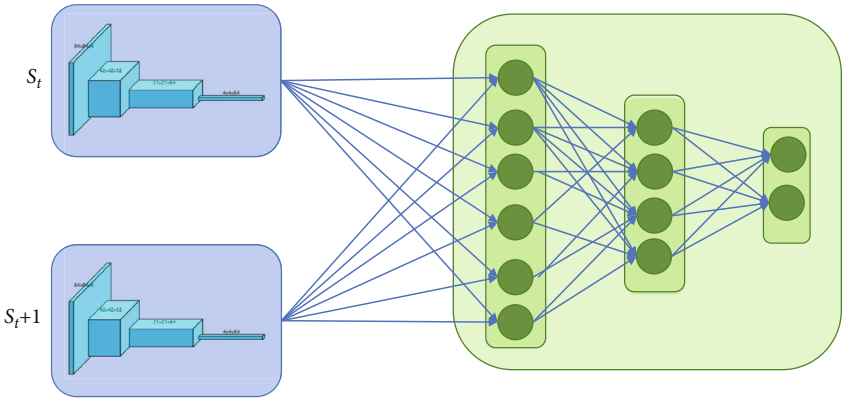

Figure 2: Left submodule encodes s_t into a feature $\phi(s_t)$. Right submodule predicts the action a_t.

$a_t \in A$. We seek a policy π that maximizes the expected sum of future rewards. The action-value $Q^\pi(s, a)$ of a state-action pair (s, a) under a policy π is the expected discounted sum of future rewards and follows π thereafter: $Q^\pi(s, a) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a]$.

3.2. Embedding Network and Action Prediction Model. When MDP states have complex structures, as in the case of image observations, directly measuring their similarities in pixel space does not provide effective metric. Previous works in computer vision [27–29] introduce manually designed feature representations of images. These representations are suitable for semantic tasks including detection and classification. More recent methods learn complex features directly from data by training convolutional neural networks [30–32]. Considering these researches, it may be difficult to combine similar states using raw pixels or the general state space.

As mentioned in the previous F1 racing game example, some features in the state space are invalid for the agent, so we need to extract the valid features in the state space. To achieve this, we first divide the features of the state into three parts: (1) something that can be controlled by the agent (e.g., the car in the game), (2) things that the agent cannot control but that can affect the agent (e.g., the track), and (3) things out of the agent's control and not affecting the agent (e.g., the sky). We need to find a good feature space that includes the features of (1) and (2) and excludes the features of (3).

Our goal is to come up with a general mechanism for learning feature representations rather than manually designing for each environment. We propose that such a feature space can be learned by training a deep neural network with two submodules: the first submodule (embedding network) encodes the raw state s_t into a feature vector $\varphi(s_t)$. The second submodule (prediction network) takes the feature encoding $\varphi(s_t)$, $\varphi(s_{t+1})$ of two consequent states as inputs and predicts the action a_t taken by the agent to move from s_t to s_{t+1}. The whole model is illustrated in Figure 2. Training this neural network is equivalent to learning function f defined as

$$\widehat{a}_t = f(s_t, s_{t+1}; \theta_e), \tag{1}$$

where $\widehat{a}_t$ is predicted estimate of the action a_t and the neural network parameters θ_e are trained to optimize

$$\min_{\theta_e} L(\widehat{a}_t, a_t), \tag{2}$$

where L is the loss function that measures the discrepancy between the predicted and actual actions. In order to facilitate the subsequent processing of the state encoding, the embedding network takes the state s as the input and contains one special dense layer comprised of D sigmoid functions. By rounding the sigmoid activation $b(s)$ of this layer to their closest binary number $\lfloor b(s) \rceil \in \{0, 1\}^D$, any state s can be binarized. A problem with this architecture is that if $b(s_t)$ near 0.5 at a particular dimension, the error will increase while rounding. A solution is forcing the binary code layer to take on binary values. Therefore, we add another loss term into L, and the complete loss function is defined as

$$L(s_t, a_t, s_{t+1}) = \left[NLLLoss(\widehat{a}_t, a_t) + \lambda \sum_{i=1}^{D} \begin{pmatrix} \min\left\{(1 - b_i(s_t))^2, b_i(s_t)^2\right\} + \\ \min\left\{(1 - b_i(s_{t+1}))^2, b_i(s_{t+1})^2\right\} \end{pmatrix} \right]. \tag{3}$$

The tuple (s_t, a_t, s_{t+1}) is obtained while the agent interacts with the environment using its current policy $\pi(s)$.

3.3. Count-Based Extra Bonus Generator. We get embedded state space $\phi(s)$ through the embedding network. An exploration bonus $r^c : S \longrightarrow \mathbb{R}$ is added to the reward function, defined as

$$r^c(s) = \frac{\beta}{\sqrt{n(\phi(s))}}, \tag{4}$$

where $\beta \in \mathbb{R}_{\geq 0}$ is the bonus coefficient. Initially, the counts $n(\bullet)$ are set to 0 for the whole range of ϕ. For every state s_t encountered at time step t, $n(\phi(s_t))$ is increased by 1. The agent is trained with rewards $(r + r^c)$, while per performance is evaluated as the sum of rewards without bonuses.

We represent the policy $\pi(s_t; \theta_p)$ by a deep neural network with parameters θ_p. Given the agent in state s_t, it

Initialize $A \in R^{k \times D}$ with entries drawn i.i.d. from the standard Gaussian distribution $\mathrm{N}(0, 1)$;
Initialize a hash table with values $n(\cdot) \equiv 0$;
Initialize policy network with parameter θ_p and embedding network with parameter θ_c;
for each iteration j **do** {
  Collect a set of state-action samples $(s_t, a_t, s_{t+1})_{m=0}^{M}$ with policy π;
  Add the state samples to replay buffer;
  if $j \bmod j_{update} = 0$ **then** {
    Update the embedding network with loss function in Eq.(3). using samples drawn from the replay buffer;
  }
  Compute $\phi(s_t) = \lfloor b(s_t) \rceil$, the D-dim rounded hash code for s_t learned by the embedding network;
  Update the hash table counts $\forall m : 0 \le m \le M$ as $n(\phi'(s_m)) \longleftarrow n(\phi'(s_m)) + 1$;
  Update the policy π using rewards $\{r(s_m, a_m) + (\beta / \sqrt{n(\phi'(s_m))})\}_{m=0}^{M}$ with any RL algorithm;
}

Algorithm 1: Count-based exploration via embedded state space.

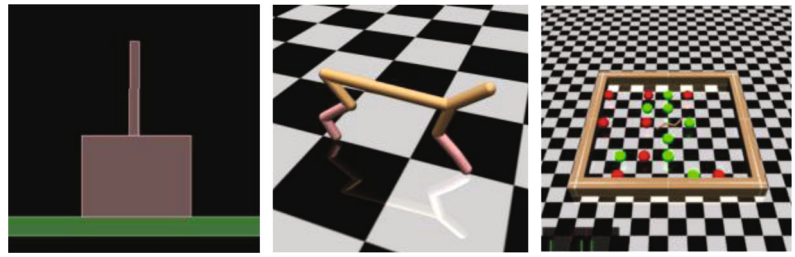

Figure 3: Illustrations of the Rllab task used in continuous control experiments, namely, CartPoleSwingup, HalfCheetah, and SwimmerGather.

executes the action $a_t \sim \pi(s_t; \theta_P)$ sampled from the policy. θ_P is optimized to maximize the expected sum of rewards:

$$\max_{\theta_p} \mathbb{E}_{\pi(s_t;\theta_p)} \left[\sum_t \left(r_t + r_t^c \right) \right]. \tag{5}$$

Because the code dimension often needs to be large for correctly predicting the action, we apply a downsampling procedure to the resulting binary code $\lfloor b(s) \rceil$, which can be done through random projection to a lower-dimensional space via SimHash:

$$\phi'(s) = \mathrm{sgn}\left(A\phi(s)\right) \in \{-1, 1\}^k, \tag{6}$$

where A is a $k \times D$ matrix drawn from a standard Gaussian distribution $\mathcal{N}(0, 1)$. The value for k controls the granularity: higher values lead to fewer collisions and clearly distinguish states. Algorithm 1 summarizes our method.

4. Experiments

4.1. Experimental Setup. We test our method in multiple environments from Rllab to Arcade Learning Environment (ALE) [19]. The experiments in Rllab verify that our method can be used in continuous control tasks. The experiments in ALE have recently become a standard high-dimensional benchmark for RL. The reward signal is computed from the game score. The raw state is a frame of video (a 160×210 array of 7-bit pixels). There are 18 available actions. The ALE is a particularly interesting testbed in our context, because the difficulty of exploration varies greatly among games. We choose six of these games where exploration is hard. Trust Region Policy Optimization is chosen as the RL algorithm for all experiments, because this algorithm can handle both discrete and continuous action spaces and is relatively insensitive to hyperparameter changes. The hyperparameter is $\beta = 0.01$ and $\lambda = 0.1$. All image curves are smoothed.

4.2. Rllab Environment. The Rllab benchmark consists of various control tasks to test deep RL algorithms. We selected several variants of the basic and locomotion tasks that use sparse rewards, as shown in Figure 3. These tasks are all highly difficult to solve with naive exploration strategies, such as adding Gaussian noise to the actions.

Figure 4 shows the results of TRPO (baseline), TRPO-SimHash [8], VIME [34], and our method on the classic tasks CartPoleSwingup, the locomotion task HalfCheetah, and the hierarchical task SwimmerGather. Using count-based exploration with embedded state space is capable of reaching the goal in all environments (which corresponds to a nonzero return), while baseline TRPO with Gaussian n control noise fails completely. Although our method picks up the sparse reward on HalfCheetah and receives better reward than other count-based exploration algorithm, it does not perform as well as VIME. In contrast, the

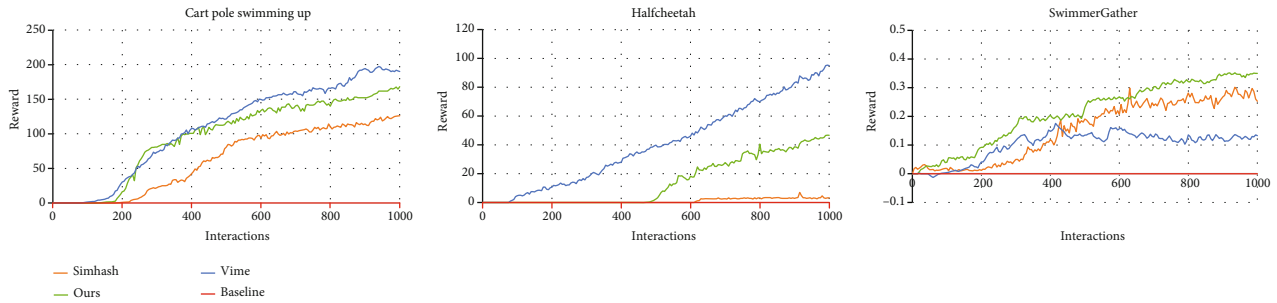

Figure 4: Mean average return of different algorithms on Rllab tasks with sparse rewards.

Table 1: Atari 2600: average total reward after training for 50 M time steps. Boldface numbers indicate best results. Italic numbers are the best among count-based exploration methods.

	Freeway	Frostbite	Gravitar	Montezuma	Solaris	Venture
TRPO (baseline)	16.5	2869	486	0	2758	121
Double-DQN	33.3	1683	412	0	3068	98
Dueling network	0	4672	588	0	2251	497
Gorila	11.7	605	**1054**	4	N/A	**1245**
DQN Pop-Art	33.4	3469	483	0	4544	1172
A3C+	27.3	507	246	142	2175	0
TRPO+AE	33.5	5214	482	75	4467	445
TRPO+BASS	28.4	3150	604	***238***	1201	616
TRPO+OURS	***34***	***5537***	*712*	196	***4860***	*983*

performance of ours is comparable with VIME on CartPoleSwingup, while it outperforms VIME on SwimmerGather.

4.3. Arcade Learning Environment. The Arcade Learning Environment (ALE) [33], which consists of Atari 2600 video games, is an important benchmark for deep RL due to its high-dimensional state space and wide variety of games. In order to demonstrate the effectiveness of the proposed exploration strategy, six games are selected featuring long horizons while requiring significant exploration: Freeway, Frostbite, Gravitar, Montezuma's Revenge, Solaris, and Venture. The agent is trained for 500 iterations in all experiments, with each iteration consisting of 0.1 M steps (the TRPO batch size corresponds to 0.4 M frames).

We compare our results to double DQN [35], dueling network [36], A3C+ [37], double DQN with pseudocounts [37], Gorila [38], DQN Pop-Art [39], and TRPO-SimHash [8] the "null op" metric. We summarize all results in Table 1.

As observed in Table 1, our approach has performed better on most of the games compared to similar count-based exploration methods. It means the embedded state space after feature extraction by using action predict network can select the part which has more important influence on the decision-making of the agent. Our method achieves near state-of-the-art performance on Freeway, Frostbite, and Solaris. A reason why TRPO+BASS is better than ours on Montezuma is that BASS is a hand-designed feature transformation for images in Atari 2600 games. The hash codes generated by our method distinguish between visually different states but fail to emphasize that the agent needs to explore different rooms. But the hand-designed feature transformation can clearly describe this key information.

4.4. Downsampling. We apply a downsampling process to the generated binary code in equation (6), which can be done using SimHash's random projection to a lower-dimensional space. However, there may be states that are distinct but fall into the same group after downsampling. Moreover, different downsampling dimensions have different effects on the final experimental results. We conduct more experiments in different game environments. Figure 5 and Table 2 show an overview of the results.

As observed in the results, in the case of low-latitude downsampling, the agent reaches the plateau fastest, but the reward is relatively lower. Conversely, in the case of high-latitude downsampling, the speed of convergence is slower, but the reward obtained is higher. This phenomenon is well explained because downsampling at low latitudes greatly compresses the state space, and the novelty to the agent disappears quickly. And because many different states are assigned to the same state, the agent's exploration ability is greatly affected, and more effective rewards cannot be obtained. And we found that there is a certain correlation between the complexity of the environment and the optimal dimension of downsampling. For complex games such as Montezuma, a higher-dimensional downsampling representation can better distinguish different states and achieve excellent returns. For games with relatively low complexity, such as Freeway, high-dimensional downsampling will cause overfitting. Overdistinguishing similar states will lead to lower overall rewards for the agent.

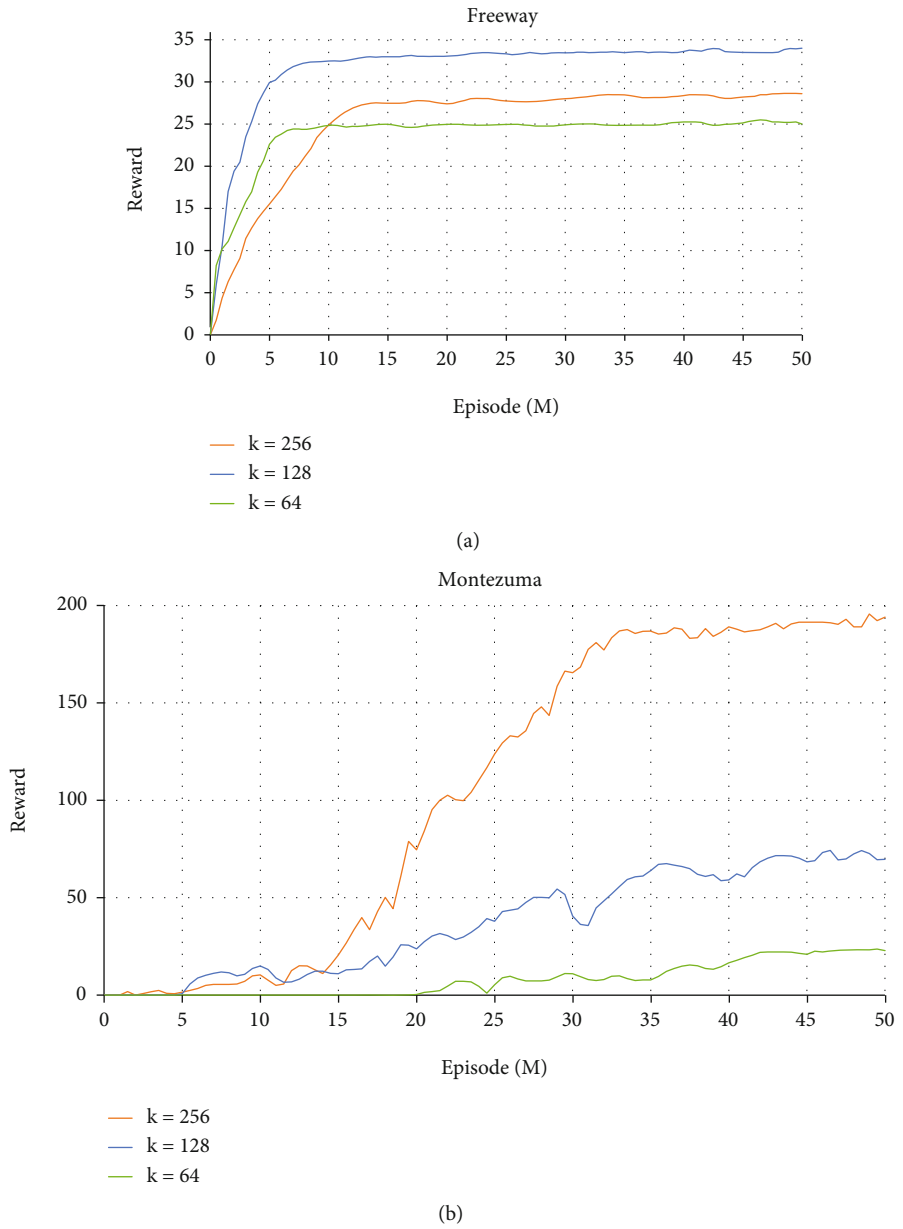

Figure 5: The influence of different downsampling dimensions on experimental results.

Table 2: Downsampling experiments in Atari 2600. K represents the dimension of downsampling. Boldface numbers indicate best results.

	$K = 64$	$K = 128$	$K = 256$
Freeway	25.1	**34**	28.3
Venture	479	677	**983**
Montezuma	24	74	**196**
Frostbite	3043	**5537**	4983

5. Conclusion

In this paper, we propose a novel count-based exploration method for deep reinforcement learning. By introducing the embedding network and the action prediction model, the proposed method tends to extract the state features that have positive impacts on the agent and encourage the agent to explore states with higher rewards. Extensive experiments demonstrate that our proposed method can achieve promising performance on different tasks. In future work, we plan to optimize the representation of state features and attempt to apply the state feature extraction framework to other kinds of reinforcement learning exploration methods.

Data Availability

Previously reported environment data were used to support this study and are available at 10.1613/jair.3912. These prior studies (and datasets) are cited at relevant places within the text as references. The experiment data used to support the

findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (No. 61972065).

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, 2015.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction[M]*, MIT press, 2018.

[4] V. Mnih, A. P. Badia, M. Mirza et al., "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, 2016.

[5] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[6] A. L. Strehl and M. L. Littman, "An analysis of model-based interval estimation for Markov decision processes," *Journal of Computer and System Sciences*, vol. 74, no. 8, pp. 1309–1331, 2008.

[7] J. Z. Kolter and A. Y. Ng, "Near-Bayesian exploration in polynomial time," in *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pp. 513–520, 2009.

[8] H. Tang, *Towards Informed Exploration for Deep Reinforcement Learning*, University of California, Berkeley, 2019.

[9] J. Martin, S. N. Sasikumar, T. Everitt, and M. Hutter, "Count-based exploration in feature space for reinforcement learning," 2017, http://arxiv.org/abs/1706.08090.

[10] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, 1991.

[11] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2/3, pp. 235–256, 2002.

[12] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3-4, pp. 285–294, 1933.

[13] I. Osband, C. Blundell, A. Pritzel, and B. V. Roy, "Deep exploration via bootstrapped DQN," *Advances in neural information processing systems*, vol. 29, 2016.

[14] B. O'Donoghue, I. Osband, R. Munos, and V. Mnih, "The uncertainty bellman equation and exploration," in *International Conference on Machine Learning*, pp. 3836–3845, 2018.

[15] M. Fortunato, M. G. Azar, B. Piot et al., "Noisy networks for exploration," in *ICLR*, 2018.

[16] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," 2015, http://arxiv.org/abs/1507.00814.

[17] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16-17, 2017.

[18] N. Savinov, A. Raichuk, D. Vincent, M. Pollefeys, T. Lillicrap, and S. Gelly, "Episodic curiosity through reachability," in *International Conference on Learning Representations*, 2018.

[19] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade, "Towards generalization and simplicity in continuous control," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[20] A. Zhang, N. Ballas, and J. Pineau, "A dissection of overfitting and generalization in continuous reinforcement learning," 2018, http://arxiv.org/abs/1806.07937.

[21] J. Choi, Y. Guo, M. Moczulski et al., "Contingency-aware exploration in reinforcement learning," in *International Conference on Learning Representations*, 2018.

[22] R. Raileanu and T. Rocktaschel, "Ride: rewarding impact-driven exploration for procedurally-generated environments," in *International Conference on Learning Representations*, 2020.

[23] A. Campero, R. Raileanu, H. Küttler, J. B. Tenenbaum, T. Rocktäschel, and E. Grefenstette, "Learning with amigo: Adversarially motivated intrinsic goals," 2020, http://arxiv.org/abs/2006.12122.

[24] D. Zha, W. Ma, L. Yuan, X. Hu, and J. Liu, "Rank the episodes: a simple approach for exploration in procedurally-generated environments," 2021, http://arxiv.org/abs/2101.08152.

[25] M. Zhou, Y. Chen, Y. Wen et al., "Factorized q-learning for large-scale multi-agent systems," in *Proceedings of the First International Conference on Distributed Artificial Intelligence*, Beijing China, 2019.

[26] B. H. Abed-Alguni, S. K. Chalup, F. A. Henskens, and D. J. Paul, "A multi-agent cooperative reinforcement learning model using a hierarchy of consultants, tutors and workers," *Vietnam Journal of Computer Science*, vol. 2, no. 4, pp. 213–226, 2015.

[27] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[28] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, IEEE, 1999.

[29] E. Tola, V. Lepetit, and P. Fua, "Daisy: an efficient dense descriptor applied to wide-baseline stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 815–830, 2010.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, http://arxiv.org/abs/1409.1556.

[33] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: an evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[34] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "Vime: variational information maximizing exploration," 2016, http://arxiv.org/abs/1605.09674.

[35] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, Phoenix, Arizona USA, 2016.

[36] Z. Wang, T. Schaul, M. Hessel, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*, pp. 1995–2003, PMLR, 2016.

[37] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[38] A. Nair, P. Srinivasan, S. Blackwell et al., "Massively parallel methods for deep reinforcement learning," 2015, http://arxiv.org/abs/1507.04296.

[39] H. P. van Hasselt, A. Guez, M. Hessel, V. Mnih, and D. Silver, "Learning values across many orders of magnitude," *Advances in Neural Information Processing Systems*, vol. 29, 2016.