Hindawi
Journal of Robotics
Volume 2022, Article ID 7169594, 14 pages
https://doi.org/10.1155/2022/7169594

Research Article

Improving Model-Based Deep Reinforcement Learning with Learning Degree Networks and Its Application in Robot Control

Guoqing Ma,[1] **Zhifu Wang**,[2] **Xianfeng Yuan**,[1] **and Fengyu Zhou**[2]

[1]*School of Mechanical, Electrical & Information Engineering, Shandong University, Weihai 264209, China*
[2]*Control Science and Engineering, Shandong University, Jinan 250061, China*

Correspondence should be addressed to Xianfeng Yuan; yuanxianfeng@sdu.edu.cn

Received 4 November 2021; Revised 15 January 2022; Accepted 2 February 2022; Published 4 March 2022

Academic Editor: Weitian Wang

Deep reinforcement learning is the technology of artificial neural networks in the field of decision-making and control. The traditional model-free reinforcement learning algorithm requires a large amount of environment interactive data to iterate the algorithm. This model's performance also suffers due to low utilization of training data, while the model-based reinforcement learning (MBRL) algorithm improves the efficiency of the data, MBRL locks into low prediction accuracy. Although MBRL can utilize the additional data generated by the dynamic model, a system dynamics model with low prediction accuracy will provide low-quality data and affect the algorithm's final result. In this paper, based on the A3C (Asynchronous Advantage Actor-Critic) algorithm, an improved model-based deep reinforcement learning algorithm using a learning degree network (MBRL-LDN) is presented. By comparing the differences between the predicted states outputted by the proposed multidynamic model and the original predicted states, the learning degree of the system dynamics model is calculated. The learning degree represents the quality of the data generated by the dynamic model and is used to decide whether to continue to interact with the dynamic model during a particular episode. Thus, low-quality data will be discarded. The superiority of the proposed method is verified by conducting extensive contrast experiments.

1. Introduction

A machine learning method for solving sequential decision-making problems, reinforcement learning algorithm learns strategies through continuous interactions with the environment and produces the highest cumulative reward. Its basic principle is to facilitate interaction between the agent and the environment and optimize the action policy using a reward system. If the model receives a positive reward, it is more likely to repeat the rewarded action, and vice versa.

Deep reinforcement learning (DRL) [1, 2], which has gradually emerged in recent years, has realized end-to-end learning using the powerful nonlinear representation capabilities of deep neural networks and has made breakthroughs in various fields, such as gaming and robot control. However, the problem of low learning efficiency due to trial and error still exists. A method for guiding the agent to explore unknown space efficiently as well as a method for finding an appropriate balance between exploration and exploitation given limited computing resources is a key problem that reinforcement learning is currently facing.

Model-free RL algorithms [3–5] have rapidly developed and been applied in the fields of video games, computer vision, self-driving automobiles, and robotics in recent years. The classification of model-free reinforcement learning is shown in Figure 1. According to the policy update and learning methods, they can be divided into value-based methods and policy-based methods. The former primarily focus on updating and training the DNN related to the value function, while the latter optimize the parameterized policy directly.

The Policy Gradient (PG) algorithm [6] is a classic policy-based method in which the loss function is the method's output and the learning rate depends on the expected total reward. Therefore, if the model obtains a positive reward, the probability of the model repeating this

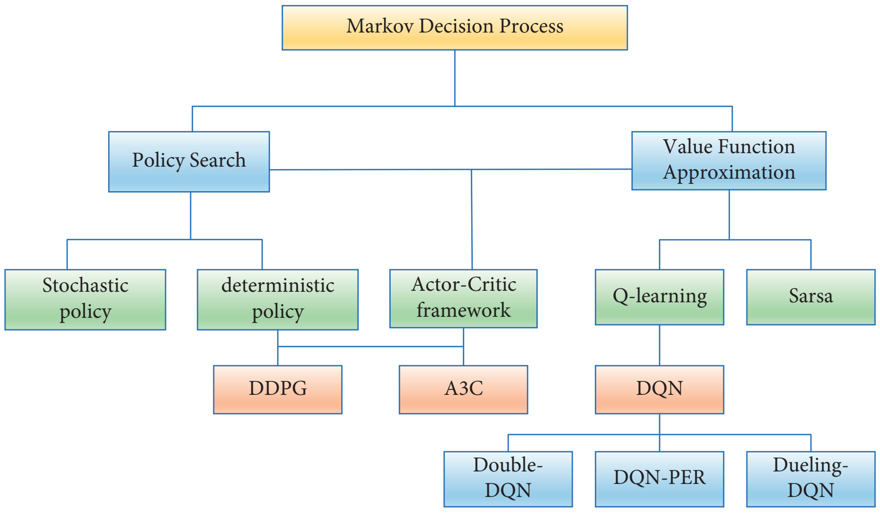


Figure 1: The development of model-free reinforcement learning.

action increases, and vice versa. More specifically, the policy is updated by continuously calculating the gradient of the expected total reward with respect to the policy parameters until an optimal policy is found. These kinds of policy search methods are similar to biological neural networks where the value functions are unnecessary. In addition, the optimization of network parameters with respect to an action's value is similar to the biological neural network learning process. Compared with deep Q-networks (DQNs) [7] and their variants, policy-based algorithms generally have a wider range of applications and produce better results. However, the original PG algorithm is easily trapped in local optima. To solve this problem, the Asynchronous Advantage Actor-Critic (A3C) [8–11] algorithm was proposed, which utilizes distributed computing resources and effectively increases the convergence speed.

However, A3C still has an obvious drawback, in that it requires a large amount of environmental interaction data to iterate the algorithm. It is convenient to generate a large amount of interaction data in simulations. In contrast, implementation in the real world has to consider security and cost. As a result, although reinforcement learning has achieved more satisfactory results in the simulation environment, it has not made many breakthroughs in real-world tasks.

Model-based reinforcement learning [12–14] uses a system dynamics model to improve the efficiency of data use and reduces the interaction times between environments. Model-based reinforcement learning is not as well developed as model-free reinforcement learning, but it has its own theoretical advantages. In addition, model-based reinforcement learning is more promising for solving real-world learning tasks by virtue of its efficient utilization of samples.

The Dyna algorithm is a simple model-based reinforcement learning framework. In the framework of the Dyna algorithm, training is carried out using two iterative steps: first, the algorithm collects interaction data from the real environment and trains the dynamic environmental model; then, the policy is updated with respect to the interaction data generated by the learned dynamic model. Nagabandi et al. [15] proposed neural network dynamics for model-based DRL with model-free fine-tuning (MBMF) [16] by establishing a neural network dynamic model $\mathbf{f}$, fitting the changes between adjacent states when performing an action $\mathbf{a}$:

$$f_{\theta}(s_t, a_t) = s_{t+1} - s_t. \quad (1)$$

Embed to control (E2C) [17] is used to address high-dimensional data flow problems. E2C contains a locally linear latent dynamic model for controlling raw images. This process uses an encoder to converge the input into a low-dimensional hidden space and then considers the dynamic environment as a local linear model in the hidden space and calculates the (Kullback-Leibler) KL divergence for model updates.

World models [18] use RNNs to establish a system dynamics model. In this method, the predicted state $\mathbf{h_t}$ and the current state $\mathbf{s_t}$ are merged into one state, which is fed to the agent for decision-making.

These MBRL models all have the problems of low prediction accuracy and cumulative errors, which affect the final training results. In [19], K critic models were applied, and the probability of obtaining data from the dynamic model was determined by the variance of the Q-value. Reference [20] uses a set of dynamic models to determine whether to continue iterating according to the number of dynamic models that achieved better performance. This article combines the advantages of the above two methods and proposes a method based on using the learning degree to determine the probability of using the dynamic model to solve the problems stated above.

2. Improved MBRL with Learning Degree Network

The system dynamics model is a neural network trained with data generated from interactions with the environment. When using the Dyna algorithm framework to update the policy, it is necessary to use the dynamic model to interact with the agent and update the policy (hereinafter referred to as imaginary learning), and the interaction process requires multiple iterations. During this process, the dynamic model produces errors in each iteration, and the total error will be accumulated and amplified in the next iteration. Furthermore, the accumulated error will cause deviation between the final state and the real state and reduce the agent's learning ability. This problem reflects the impact of insufficient learning on the prediction results. In this paper, we define the accuracy of the predicted dynamic model, named the learning degree (LD). A set of neural networks, called the learning degree network (LDN), is employed to evaluate the accuracy.

The cumulative error problem is especially aggravated when optimizing long sequence tasks, which is very common in model-based reinforcement learning. Analyses of the source of error are needed to overcome the impact of the accumulated error. First, underfitting of the dynamic model trained with limited environmental data will cause bias at the beginning. In addition, the model's predicted imaginary learning state has never been sampled from the perspective of the agent, meaning that the overfitting of the dynamic model trained with the partial environmental data also influences the final results.

Here, we use the human decision-making process as an example. Suppose that a person has been living in a house and has never left that house. Then, when he is thirsty, he imagines the entire water-drinking process, from obtaining to drinking the water, and, in this case, the result of drinking water is very certain. Next, this person carries out the imagined process and is happy. However, he has not experienced the outdoor environment before and is therefore unable to imagine the outdoor environment. When this person leaves the house, he does not know where to find water and cannot use his imagination to help him achieve this goal. He therefore needs to explore how to get water by himself. In this paper, the "degree of certainty of the results of the imagination" is defined as the parameter "degree of learning," and a neural network model is applied to evaluate the degree of learning. Reference [21] emphasizes the role imagination plays in our decisions. When we decide between two possible actions, we imagine ourselves in each situation, imagine the outcome of these two actions, and then compare these two imagined scenarios. Therefore, imagination must play a role in the human decision-making process.

We propose a method (see Algorithm 1) to evaluate the learning degree (LD) of the dynamic models and determine whether to proceed with the next iteration. The A3C algorithm is used as an example to explain the reinforcement learning exploration strategy based on the learning degree dynamics model. The A3C algorithm first creates the actor and critic network as well as two dynamic model neural networks M^0 and M^1 with the same initial parameters and defines **R** and **I** to store the data from interacting with the real environment and the data from interacting with the dynamics model, respectively. In Algorithm 1, steps 6 to 16 are taken from the classic A3C algorithm framework. Steps 17 and 18 use data **R** to train the M^1 network, and the reinitialized environment state is obtained. After Step 19, the M^1 network is used to train the agent. There are two termination conditions for this training. The first is the state that triggers the end of the episode; the second is that the LD predicted by the dynamic model is the probability of continuing iteration; that is, every step must have a probability of 1 – **LD** to terminate the iteration. This makes it possible to "stop imagining" when the network is not fully trained or using a state that has not been experienced in the "imaginary learning" process. In this way, the impact of low-quality data on the algorithm can be reduced. In Algorithm 1, E represents the number of episodes of the whole learning process, and T is the maximum number of steps in one episode, used for both the process of iterating with the real environment and the process of iterating with the system dynamics model. The Boolean "done" is returned by the real environment or system dynamics model and its value is true when the state triggers the end of the episode.

The system dynamics model training uses the BP neural network supervised learning algorithm. Combine s_{t+1} and r_t as the target vector y_t. The loss function is defined as

$$L\left(y, M^1\left(s, a|\beta^{M^1}\right)\right) = \frac{1}{N}\sum_t\left(y_t - M^1\left(s_t, a_t|\beta^{M^1}\right)\right)^2, \quad (2)$$

where the dimensions of vector y_t are l and $N = T \times l$.

The loss function is optimized using the mean square error gradient descent method, which is described by the following equation:

$$\overline{\sigma}_t^2 = \frac{1}{2}\left(M^1\left(s_t, a_t|\beta^{M^1}\right) - M^0\left(s_t, a_t|\beta^{M^0}\right)\right)^2, \quad (3)$$

where $\overline{\sigma}_t^2$ is the mean square deviation of the prediction results from the system dynamics model network M^1 and the original neural network M^0.

In this paper, the LD of the dynamics models is defined as

$$\mathrm{LD}_t = 2 \times \frac{1}{1 + e^{-\overline{\sigma}_t^2}} - 1, \quad (4)$$

where LD_t uses the sigmoid function to limit the result to [0, 1].

In the algorithm, LD_t is used as the probability of continuing iteration; that is, every step must have a probability of $1 - \mathrm{LD}_t$ to terminate iteration. The probability of termination $P_t^{\mathrm{terminate}}$ is defined as

$$P_t^{\mathrm{terminate}} = 1 - \mathrm{LD}_t. \quad (5)$$

The MBRL-LDN architecture is shown in Figure 2. In the first iteration, every worker interacts with Env (environment) using transfer observation and action. Each worker contains an actor and critic network. The actor network uses a three-layer fully connected network with 200 hidden layer neurons. The critic network is also a three-layer fully connected network with 100 hidden layer neurons. The activation function of the hidden layer is the ReLU function, and the output layer of the actor network uses the softmax function as the activation function. In the interaction period, interactive data will be generated and stored in R (the real

```
1 **Input**: Interactive data generated by interaction with the environment
2 **Output**: Policy $\mu$
3 Initialize critic network parameters $Q$, actor network $\mu$'s parameters $\theta$
4 Initialize real memory space $R$ and imaginary memory space $I$
5 Initialize LD network $M^0$ and $M^1$'s parameters $\beta^{M^0}$ and $\beta^{M^1}$
6 for $e = 1\ldots E$ do
7   Get the initialized state $s_1$
8   for $t = 1\ldots T$ do
9     Get the action corresponding to the current state
        $a_t = \mu(s_t|\theta^\mu) + \xi$ (Search by adding noise)
10    Get the next state $s_{t+1}$ and reward $r_t$
11    Store $(s_t, a_t, s^1_{t+1}, r^1_t)$ in Real Memory $R$
12    If done:
13      Train $Q$ and $\theta$ with batch data in $R$
14      Break
15 end for
16 Train $\beta^{M^1}$ of $M^1$ with batch data in $R$
17 Initialize the state $s_1$
18 for $t = 1\ldots T$ do
19   Get the action corresponding to the current state
       $a_t = \mu(s_t|\theta^\mu) + \xi$
20   Get the next state and reward
       $s^0_{t+1}, r^0_t = M^0(s_t, a_t|\beta^{M^0})$
       $s^1_{t+1}, r^1_t = M^1(s_t, a_t|\beta^{M^1})$
21   Store $(s_t, a_t, s^1_{t+1}, r^1_t)$ in Imaginary Memory $I$
22   Calculate learning degree $\mathrm{LD}_t$ (equation (3))
23   Generate random numbers rand $\in [0, 1)$
24   Calculate termination probability $P_t^{\text{terminate}}$ (equation (5))
25   if rand $< P_t^{\text{terminate}}$ or done
26     Train $Q$ and $\theta$ with batch data in $I$
27     Break
28   end for
29 end for
```

Algorithm 1: Model-based Deep Reinforcement Learning using an LD Network (MBRL-LDN)

memory space). In this iteration, R was used to train the global network by updating the gradient. In the second iteration, R was used to train the M^1 network. In the third iteration, workers interact with M^1. At the same time, s_t and a_t were fed into M^1 and M^0. The LD networks M^0 and M^1 have the same network structure and the same initial parameters. First, state s_t and action a_t are used to predict the next state s_{t+1} utilizing a three-layer fully connected neural network, and then the predicted state s_{t+1} is used to predict reward r for this action through another three-layer neural network. Neither network has an activation function. Then, LD will be calculated using $s^0_{t+1}, r^0_t, s^1_{t+1}, r^1_t$ (equations (2) and (3)). The LD will decide whether to break during the third iteration using equation (4). This paper uses the gradient descent optimizer to target the network, and the input size is 10 batches.

Compared with the traditional RL algorithm, the proposed method applies multiple neural networks (LD networks) to judge the dynamic model, and these networks are employed by the agents and use sampled data in the memory buffer to conduct supervised learning. The key to this algorithm is that the LD network evaluates the LD of its own estimation results by comparing the predicted difference between the dynamic model network M^1 and the initial network M^0. It can be seen from equations (3) and (4) that the higher the learning degree is, the more obvious the difference between the outputs of the two dynamic model networks is, meaning that the dynamic network has extracted the environment features effectively. Conversely, it is difficult for the system dynamics model learning network to describe the real environment accurately when it accumulates error. It is appropriate to stop training while the learning degree is low.

3. Improved A3C-Model Experiment in Gym

3.1. Experimental Settings. This experiment is based on CartPole-v0 and CartPloe-v1 from Gym as well as a customized CartPole-v2. CartPole-v2's task is the same as CartPole-v0 and CartPole-v1. In the experiment, forces are applied to the movable bottom plate in different directions to keep the straight pole deflection angle within ±12°, and the maximum displacement of the plate does not exceed ±2.4. For every step, if the iterations are not terminated, the environment rewards 1 score. The difference between CartPole-v2 and the others is that the two unmodified versions have only 2 discrete actions in the action space, and the maximum step size per episode is 200 and 500,

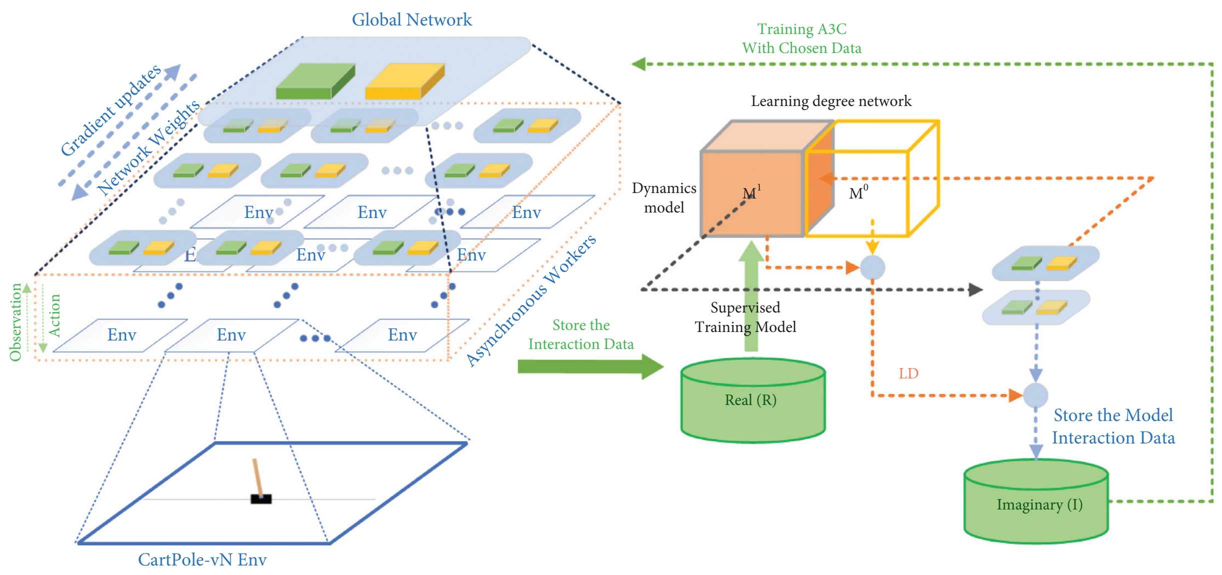


Figure 2: MBRL-LDN architecture. The classic A3C algorithm framework is depicted on the left and carries out its normal learning episode. Other structures, such as the LDN, are depicted on the right. These structures assist in the imaginary learning process.

respectively, while the customized CartPole-v2 has 5 discrete actions in the action space, and the maximum step size is 1000. The difficulty of using CartPole-v2 is higher than that of the former two versions, which is expressed by the abundant action choices and long horizons. The environmental parameters are shown in Table 1. Figure 3 shows the stable process of CartPole; in the beginning, CartPole shifts between a and b. The pole is unstable, and the plate moves left and right with large fluctuations. After the agent learned part of the control strategy, the pole would be situated as it is in c, where the angle of the pole is close to 0°; nevertheless, the plate is not stable and keeps moving until it falls. Finally, when the agent can control the environment, such as in d, the angle of the pole is close to 0°, and the position of the plate can be stable at 0.

3.2. Results and Discussion. In this paper, the traditional A3C, A3C-Model, and the proposed MBRL-LDN are compared by training three Gym environments, CartPole-v0, CartPole-v1, and CartPole-v2. The learning rate, mini-batch, and other hyperparameters of the three algorithms are the same. Each algorithm is independently trained in three environments for 250–300 episodes, and each episode is trained for 200, 500, or 1000 steps (decided by the environment). The cumulative reward of each episode is used as an evaluation indicator for the current policy. In the A3C-Model and MBRL-LDN algorithms, each training episode is divided into three stages: in the first stage, the agent interacts with the real environment to update the actor and critic network; in the second stage, the dynamics model uses the data in real memory to train M^1 100 times; and, in the third stage, the agent interacts with M^1 and updates the actor and critic networks at the same time. The cumulative rewards of the three algorithms in each Gym environment are shown in Figure 4, and the confidence interval is 95%.

In each episode, agent will collect the same amount of data, so the axis of episode can be regarded as "data usage." Figure 4 compares the rewards of three algorithm with the same usage of data. It can be seen from the experimental results that both the A3C-Model and MBRL-LDN produce better results when compared with the model-free reinforcement learning A3C algorithm in terms of the growth rate, initial growth time, and training episode required for convergence. In the CartPole-v0 experiment, the traditional A3C algorithm converges after 400 training episodes, and the model-based A3C-Model algorithm tends to converge at approximately 330 episodes, while the MBRL-LDN converges within 250 episodes and the cumulative reward after convergence is better than that of the A3C-Model and A3C algorithms. For the CartPole-v1 experiment, the A3C-Model grew earlier than A3C and MBRL-LDN. However, due to the large amount of low-quality data it learned, the A3C-Model did not utilize the advantages of model-based algorithms in later periods, which hinders later learning in comparison to the proposed MBRL-LDN. This reflects the advantages of collecting high-quality data and discarding low-quality data based on the learning degree. The CartPole-v2 experiment has more action space and requires the environmental model to calculate higher prediction accuracies. The cumulative reward curve in the CartPole-v2 experiment shows that the proposed MBRL-LDN demonstrates better performance than the A3C and A3C-Model algorithms.

The learning degree distribution in Figure 5 intuitively illustrates the variation trend of the learned model. First, in the CartPole-v2 experimental environment, the LD of the learned model within 10,000 steps clearly increases. However, the probability distribution is random. The fluctuation represents the inaccuracy of the predicted data. In other words, there is a recognizable prediction bias. During the subsequent 40,000 steps, the LD of the environmental model is generally higher. When the LD increases, the predicted data become increasingly accurate. Second, in general, the

Table 1: State space and action space of the OpenAI Gym.

Environment	State space	Action space	Max step size
CartPole-v0	4D continuous space displacement: $x \in [-2.4, 2.4]$	1D discrete action: action $\in \{0, 1\}$	200
CartPole-v1	Speed: v	Corresponding to the actual forces in the environment: $F \in \{-10, 10\}$	500
CartPole-v2	Angle: $\theta \in [-12 \cdot \pi/180, 12 \cdot \pi/180]$ Angular velocity: θ'	1D discrete action: action $\in \{0, 1, 2, 3, 4\}$ Corresponding to the actual forces in the environment: $F \in \{-10, -5, 0, 5, 10\}$	1000

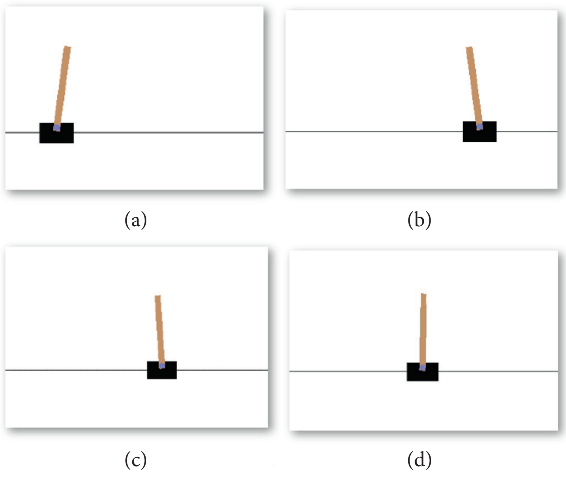


Figure 3: Stabilization process of the CartPole.

dense locations of peak show that the LD is distributed close to 1 in most steps. A high LD means that the state has been learned in that step. The peak distribution means that most of the states have been experienced, while few states have not been experienced.

The ablation experiments cover all combinations of algorithm and environments. The experiments are repeated 10 times. The maximum, minimum, average, and standard deviation of the episode with the same threshold reward are recorded for analysis. The reward threshold is 150. Since the CartPole-v2 environment is complex and the cumulative reward is generally low, the threshold reward is set to 100. The experimental results are shown in Table 2.

Table 2 shows that the average number of iterations required by the MBRL-LDN algorithm is generally lower than that of the A3C and the A3C-Model algorithms. In addition, the standard deviation of the MBRL-LDN algorithm is slightly better than that of the A3C algorithm, while the standard deviation performance of the A3C-Model algorithm is the worst. The box plot of the experimental results is shown in Figure 6.

Figure 6 indicates that the median and the quartile of the convergence episodes of the proposed MBRL-LDN are smaller than those of the competitors. In addition, from the CartPole-v2 experiment, the proposed MBRL-LDN algorithm performs better in a more complex environment.

In summary, reinforcement learning with an LD network has a great advantage in terms of convergence speed. The detailed comparison shows that MBRL-LDN greatly improves upon the A3C-Model and A3C. In the three environments, performance was improved by 31.1% and 14.6%, and the standard deviation was reduced by 6.6% and 29.0%, on average. The proposed MBRL-LDN can effectively reduce the number of interactions between the agent and the environment. In these experiments, the utilization of sample data was improved.

3.3. *Computing Time Analysis.*

Regarding the computing time, we deployed a computing time ablation experiment based on the Gym environment. Table 3 lists the computing time with the same reward. Theoretically, the model-free algorithm A3C should take less time than the model-based algorithms to achieve the same reward. Meanwhile, the time cost should be nearly the same between two model-based algorithm, A3C-Model and MBRL-LDN. The ablation experiments cover all three algorithms and environments 10 times, and the maximum, minimum, average, and standard deviation of the computing time for the reward threshold experiments are recorded for analysis. The reward threshold is 150. Since the CartPole-v2 environment is complex and the cumulative reward is generally low, the time when the cumulative reward of this environment reaches 100 is counted. The time is measured in seconds.

Table 3 shows that the average computing time required by the MBRL-LDN and A3C-Model is generally longer than that of the A3C. More specifically, MBRL-LDN is the same as the A3C-Model. MBRL-LDN is a little faster than A3C-Model, because MBRL-LDN will terminate in advance in some early imaginary learning steps. In addition, the standard deviation of the three is generally the same. The standard deviation of model-based algorithm MBRL-LDN and A3C-Model is slightly higher than that of the model-based algorithm A3C, because the dynamics model will bring randomity.

However, the major advantage of the model-based algorithm is that it has a higher efficiency of data usage, which enables model-based algorithms to perform better in real-world applications. This manuscript also tries to demonstrate that when getting the same number of data, MBRL-LDN will perform better than the competitors. The main idea of the proposed MBRL-LDN is to improve the efficiency of data usage. The cost of getting plenty of training data is very high; therefore, an RL algorithm that has a higher rate of data usage is required.

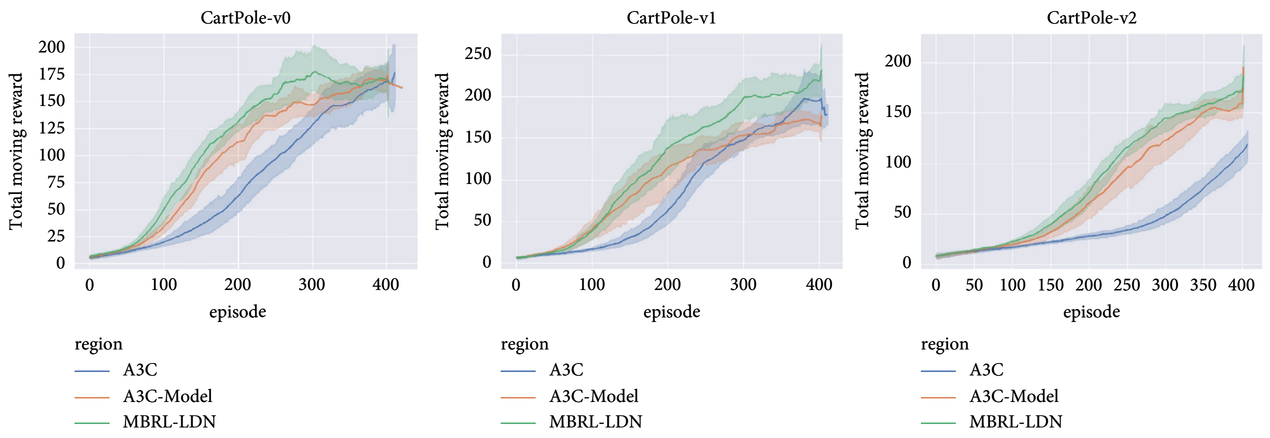


Figure 4: Learning curves of our method versus state-of-the-art methods. The horizontal axis indicates the number of steps in the real-world data. It can be also regarded as data usage. The vertical axis denotes the average return. These figures clearly demonstrate that our proposed method significantly outperforms other methods (best viewed in color).

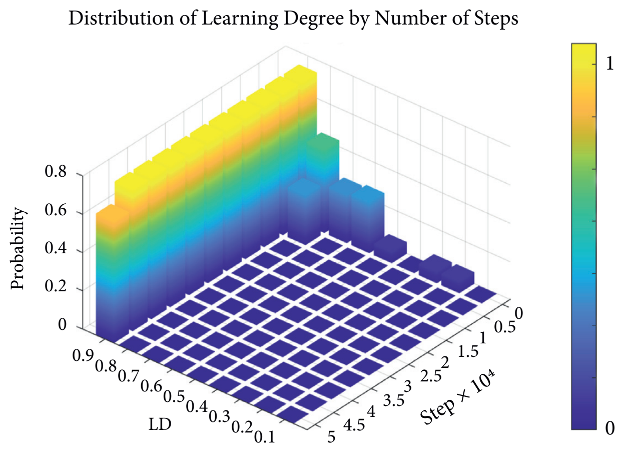


Figure 5: Distribution of learning degree in CartPole-v2 Training. The step-axis represents the total number of steps when using the dynamic model. The steps are calculated during the entire training process. The LD-axis represents the LD in each step. The step-axis and LD-axis form a set of square areas, each of which has an area measuring 500 steps and 0.1 LD. Each step locates a point in a certain area. The height of each square area represents the frequency of a point falling in this area. The frequency is normalized.

Table 2: Iterative episode with the same reward.

Env	A3C			A3C-model			MBRL-LDN		
	V0	V1	V2	V0	V1	V2	V0	V1	V2
1	321	280	371	264	205	197	184	179	243
2	293	320	407	253	218	242	221	179	265
3	299	320	376	376	333	301	288	157	228
4	411	400	327	209	195	237	207	236	212
5	310	275	381	362	299	323	298	263	241
6	304	245	400	266	329	301	229	230	211
7	377	280	377	310	286	245	260	182	239
8	348	320	367	233	255	259	236	297	197
9	398	304	395	254	400	249	284	313	232
10	403	221	407	334	286	244	239	279	227
Min	293	221	327	209	195	197	184	157	197
Max	411	400	407	376	400	323	298	313	265
AVG	346.4	296.5	380.8	286.1	280.6	259.8	244.6	231.5	229.5
S.D.	44.5	46.7	22.7	53.4	61.1	35.7	35.4	52.6	18.4

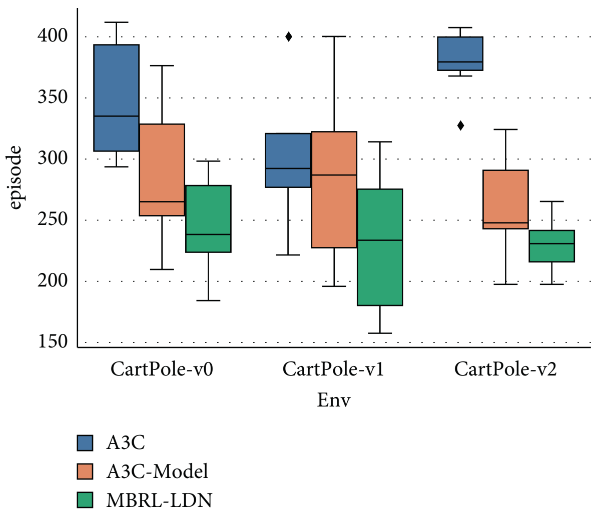


FIGURE 6: Iterative episode with the same reward.

TABLE 3: Computing time with the same reward.

Env/s	A3C			A3C-model			MBRL-LDN		
	V0	V1	V2	V0	V1	V2	V0	V1	V2
1	658	625	749	964	963	1574	916	895	1521
2	628	590	698	923	928	1467	932	854	1515
3	652	616	723	966	953	1573	894	897	1486
4	623	636	742	1019	998	1584	943	932	1494
5	616	589	697	962	953	1494	932	914	1496
6	583	614	658	986	951	1496	889	941	1573
7	627	642	752	970	976	1535	950	952	1489
8	593	580	725	984	965	1583	956	977	1538
9	651	642	753	996	1012	1548	964	1003	1584
10	624	598	734	974	956	1527	939	968	1528
Min	583	580	658	923	928	1467	889	854	1486
Max	658	642	753	1019	1012	1584	964	1003	1584
AVG	625.5	613.2	723.1	974.4	965.5	1538.1	931.5	933.3	1522.4
S.D.	23.2	21.9	29.0	23.8	23.1	39.5	23.7	42.2	32.6

4. MuJoCo Experiments and Ablation Study

This paper uses the MuJoCo environment in OpenAI Gym as the experimental environment to prove the improvement in data efficiency. MuJoCo is a general-purpose physics engine that aims to facilitate research and development in robotics, machine learning, and decision-making. It is widely used in reinforcement learning for algorithm evaluation. Ant-v2, HalfCheetah-v2, Hopper-v2, Reacher-v2, Walker2D-v2, and InvertedPendulum-v2 are the most classic games used in MuJoCo. Besides, there are some differences in reward mechanism, action, and state space between games. Compared with the classic control environment like CartPole, MuJoCo simulated high-dimensional continuous action space, which is more suitable for real robotics tasks. We chose 6 MuJoCo games to test the performance of the A3C-Model and MBRL-LDN. The MuJoCo games are similar to those in [22]. MuJoCo games are widely used as benchmarks in the field of reinforcement learning [23–25]. The A3C-Model uses the same algorithm and network structure as the model-based actor-critic learning in [26]. To evaluate the influence of the choice of equation (4), comparative experiments are also conducted where equation (4) has been modified as below:

$$\mathrm{LD}_t = 2 \times \frac{1}{1 + e^{-K\times\overline{\sigma}_t^2}} - 1. \tag{6}$$

Each game has a series of comparative experiments, each of which has a single K. From equations (4) and (5), the monotonicity between K and $P_t^{\text{terminate}}$ is confirmed. Therefore, the probability of iteration termination can be changed, and the impact of LDN can be changed. There are three candidate K values in MBRL-LDN: 0.5, 1, or 2. Finally, we analyzed the stability and performance of the proposed MBRL-LDN algorithm in detail.

The experiment aims to solve the following problems:

(1) Compared with the A3C-Model, can the MBRL-LDN improve the performance of learning policies?

(2) Does the choice of equation (4) affect the performance of the MBRL-LDN?

4.1. Experimental Environment and Setup. The purpose of our experimental evaluation is to understand how the performance and stability of the learning policy in our method compare with the previous A3C-Model algorithm, especially in more complex and continuous control tasks. Six MuJoCo games are conducted: Ant-v2, HalfCheetah-v2, Hopper-v2, Reacher-v2, Walker2D-v2, and InvertedPendulum-v2. The aim of Walker2D-v2, Ant-v2, Hopper-v2, and HalfCheetah-v2 is to walk as far as possible while expending minimal energy. The aim of Reacher-v2 is to control the two sticks and reach the goal point. The aim of InvertedPendulum-v2 is to keep the pole up and the chassis stable by applying force to the chassis. These games are shown in Figure 7. Although a variety of different algorithms can be used to solve simpler tasks [27], it is sometimes difficult to achieve stable performance in some tasks using model-free algorithms, such as 8-dimensional Ant. We aim to show how LD affects the performance of the A3C-Model and how K affects the performance of MBRL-LDN.

4.2. Experimental Results of MBRL-LDN and the A3C-Model. We tested the performance of MBRL-LDN in the game tasks. We chose 6 MuJoCo games: Ant, HalfCheetah, Hopper, Reacher, Walker2D, and InvertedPendulum. During training, the performances of MBRL-LDN and A3C-Model are compared. The results are shown in Figure 8. These results show that, generally, MBRL-LDN for all 6 MuJoCo games is better than the A3C-Model. In the MuJoCo environment, each episode includes 1000 steps. First, the performance of MBRL-LDN is shown in red and compared to the A3C-Model, shown in blue. For all game environments, the hyperparameters of MBRL-LDN and A3C-Model are the same as those in Experiment 1. Figure 8 illustrates that MBRL-LDN consistently outperforms the A3C-Model in all environments and all training stages. More specifically, in Ant-v2, MBRL-LDN's total reward in each episode is always higher than its competitor. It reaches the top at 4178 episodes, which is earlier than A3C-Model at 5123 episodes. At 4178 episodes, the performance of MBRL-LDN is 1.4 times that of the A3C-Model; in HalfCheetah-v2, using high-quality data from the dynamic model, the total reward value of MBRL-LDN increases faster than A3C-Model and it reaches a score of more than 8000 at episode 753, while the A3C-Model reaches a score of just 6624; in Hopper-v2, the total reward value of MBRL-LDN reaches a score of 4000, which is 1.5 times faster than the A3C-Model, and the total performance of MBRL-LDN is higher than that of the A3C-Model; in Reacher-v2, the A3C-Model goes faster than MBRL-LDN, but this result may be because the random data used by A3C-Model improves its exploration space and, therefore, the A3C-Model achieves better generalization. Nevertheless, MBRL-LDN's total reward in each episode reaches its maximum at episode 6592 with a score of 8835, which is 2 times that of the A3C-Model; in Walker2D-v2, MBRL-LDN's accumulated reward in each episode reaches its maximum at episode 4742 with a score of 8035, which is 1.5 times that of the A3C-Model; in InvertedPendulum-v2, MBRL-LDN's total reward in each episode reaches its maximum at episode 2500 with a score of 10935, which is 1.2 times that of A3C-Model. However, MBRL-LDN is less stable than A3C-Model during training.

4.3. Ablation Study of K Values. Figure 9 shows the performance of MBRL-LDN in 6 different MuJoCo game environments with different K values. When the K value is 1, the corresponding MBRL-LDN algorithm is better than it with other K values. However, there is little difference in the performance with various K values, especially in the Walker2D and InvertedPendulum games. As a result, we can draw the following conclusions: suitable K improves the ability of the agent, but the difference is not obvious, so the choice of equation (4) does not greatly impact the results (Figure 9).

In addition, to compare the performances of MBRL-LDN and the A3C-Model more intuitively, we use a table to list their total reward per episode. The results are shown in Table 4. Overall, the performance of MBRL-LDN is better than that of the A3C-Model.

5. Model Generalization Experiment with PPO

To verify the effectiveness of LDN in different reinforcement learning algorithms, additional PPO2 comparative experiments were carried out. The PPO solved the problem that the learning rate is hard to determine in the PG algorithm. PPO2 abandons the KL divergence loss function and introduces the clip loss function to improve the performance. PPO is widely used as benchmark in the field of reinforcement learning. It is also one of the baseline algorithms in OpenAI. In this experiment, we introduce the dynamics model into PPO2 and define it as PPO2-Model, which makes it a model-based algorithm. Besides, LDN is employed into PPO2 and the variant is named as PPO2-LDN. This experiment also uses the MuJoCo environment in OpenAI Gym as the experimental environment. Three MuJoCo games are selected to test the performances of the PPO2, PPO2-Model, and PPO2 -LDN. The MuJoCo games are similar to those in [22]. MuJoCo games are widely used as benchmarks in the field of reinforcement learning [23–25]. The PPO2-Model uses the same algorithm and network structure as the model-based actor-critic learning in [26].

5.1. Experimental Environment Setup. The purpose of the additional comparative experiments is to check if LDN keeps advantages over other methods without LDN. Three MuJoCo games are conducted: HalfCheetah-v2, Hopper-v2, and InvertedPendulum-v2. These games are shown in Figure 7. In this experiment, the model-free algorithm PPO2 was selected as the comparison algorithm. Although a variety of different algorithms can be used to solve simpler tasks [27], it is difficult to achieve stable performance in some tasks using model-free algorithms, such as the 8-dimensional Ant. Therefore, we chose these three games that are relatively easy to converge.

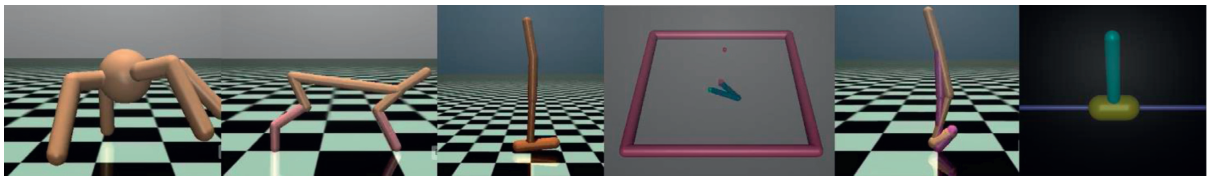
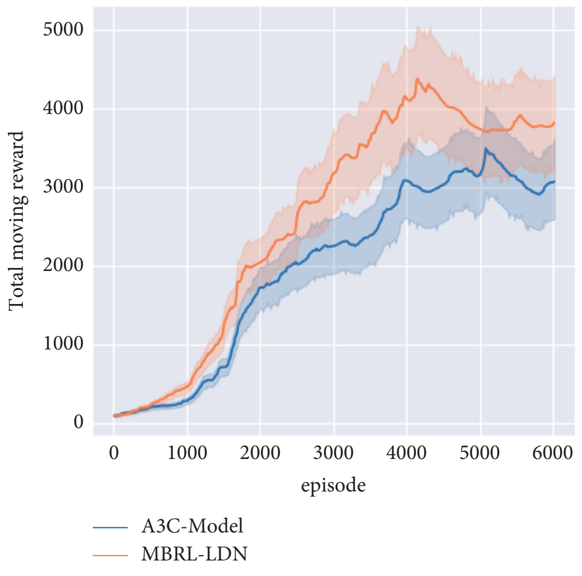
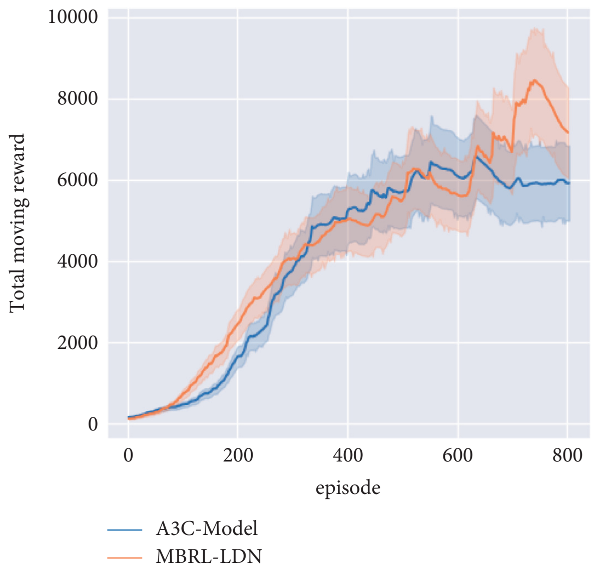


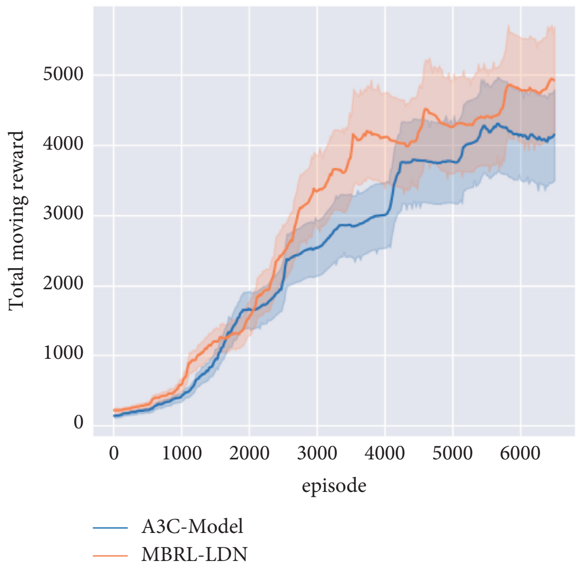
Figure 7: MuJoCo games: Ant-v2, HalfCheetah-v2, Hopper-v2, Reacher-v2, Walker2D-v2, and InvertedPendulum-v2.



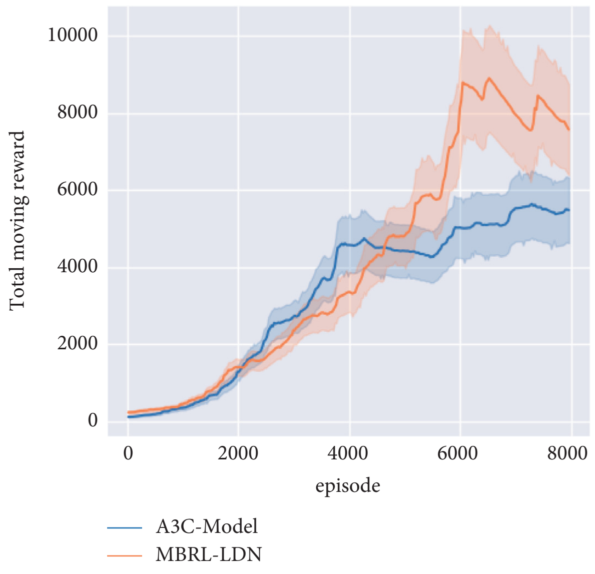
(a)



(b)

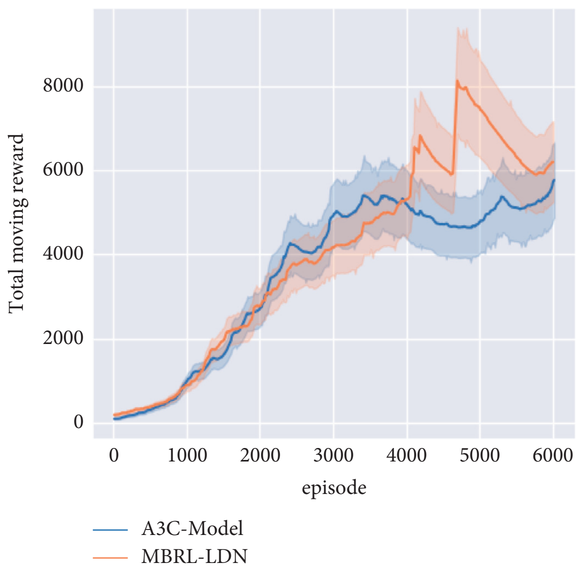


(c)

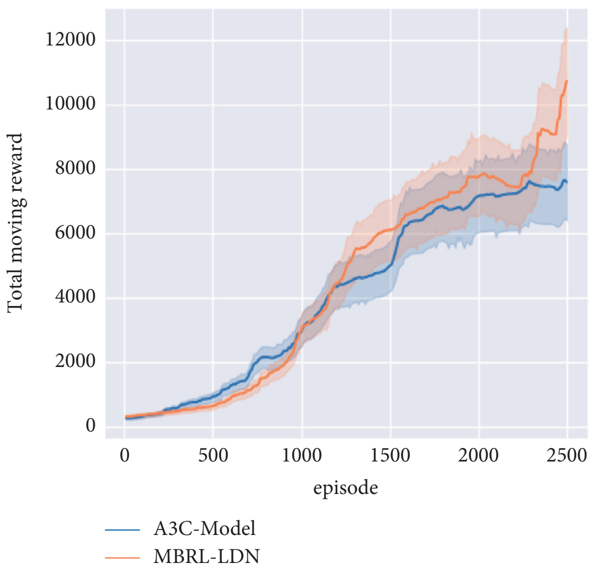


(d)

Figure 8: Continued.

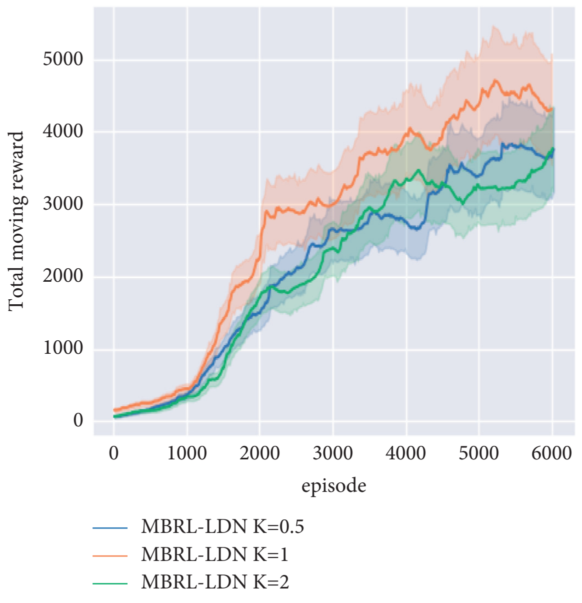


(e)

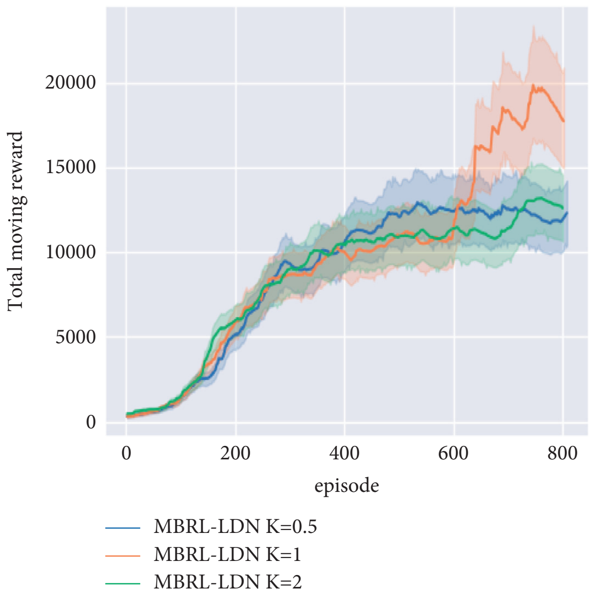


(f)

Figure 8: Experimental results of MBRL-LDN and the A3C-Model in 6 MuJoCo games. (a) Ant-v2. (b) HalfCheetah-v2. (c) Hopper-v2. (d) Reacher-v2. (e) Walker2D-v2. (f) InvertedPendulum-v2.

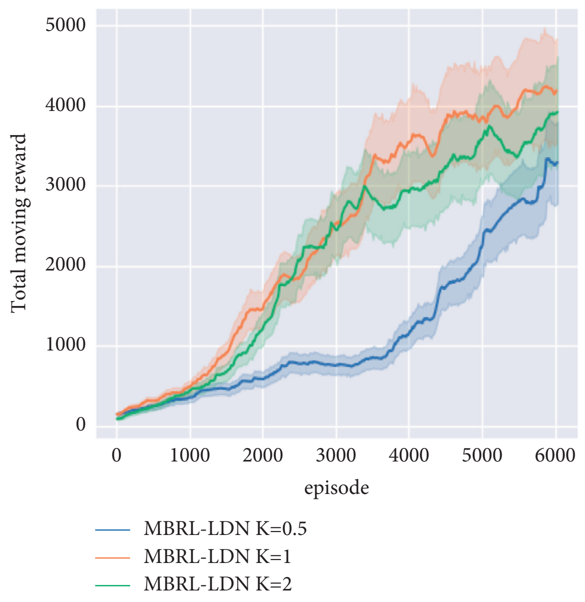


(a)

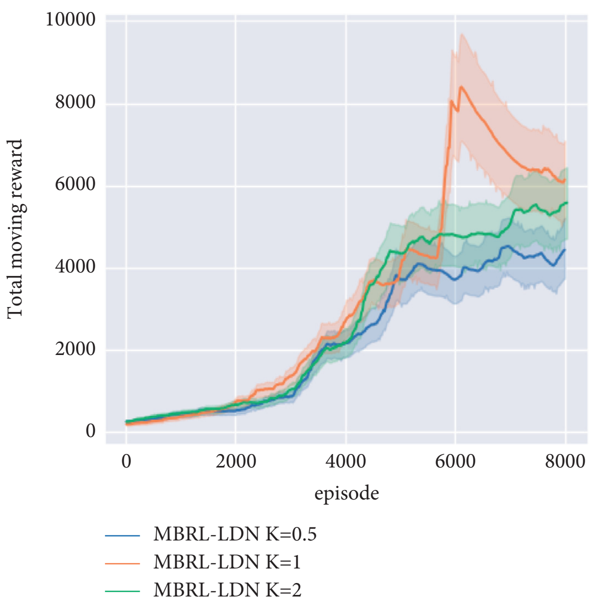


(b)

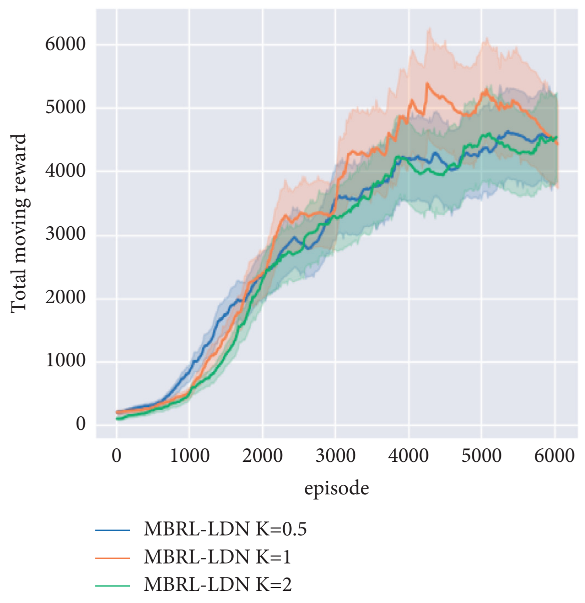
Figure 9: Continued.



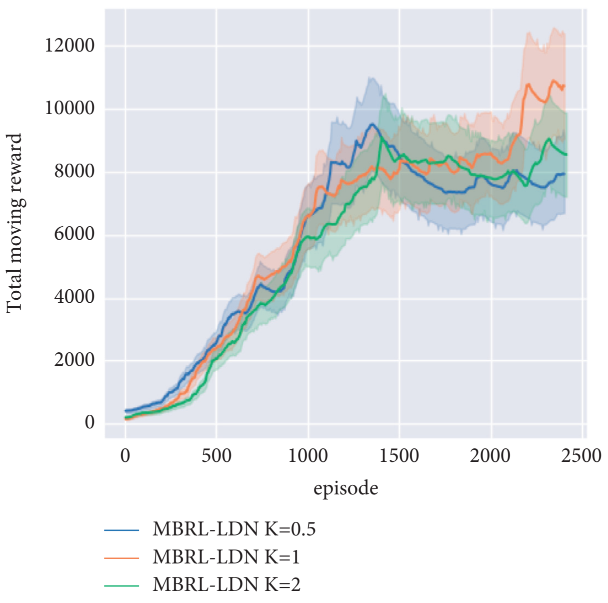
(c)



(d)



(e)



(f)

FIGURE 9: Results of MBRL-LDN with different K values in 6 MuJoCo games. (a) Ant-v2. (b) HalfCheetah-v2. (c) Hopper-v2. (d) Reacher-v2. (e) Walker2D-v2. (f) InvertedPendulum-v2.

5.2. Experimental Results of PPO2, PPO2-Model, and PPO2-LDN. During training, the performances of three algorithms are compared. The results are shown in Table 5 and Figure 10, where the green, orange, and blue lines represent the performances of PPO2-LDN, PPO2-Model, and PPO2. In the MuJoCo environment, each episode includes 1000 steps. For all game environments, the hyperparameters of the three methods are the same as those in Experiment 1. Figures 10(a) and 10(b) illustrate that PPO2-LDN consistently outperforms PPO2 in all environments and all training stages, which means it needs fewer data for learning. More specifically, in Figure 10(a), PPO2-LDN's total reward in each episode is always higher than its competitors. It reaches the top at 2175 episodes, which is earlier than PPO2-Model at 2465 episodes. At 2175 episodes, the performance of PPO2-LDN is 3.25 times that of PPO2-Model and PPO2; in Figure 10(b), using high-quality data from the dynamic model, the total reward value of PPO2-LDN increases faster than PPO2-Model and it reaches a score of more than 7000 at episode 1753, while PPO2-Model reaches 7000 at episode 2201 and PPO2 reaches 7000 at episode 2632. Affected by the inaccurate data, the reward of PPO2-Model has great fluctuation after reaching the top. PPO2-Model begins to go up at 1760 episodes which is earlier than PPO2's 1920, which

Table 4: The total moving rewards of MBRL-LDN and the A3C-Model in 6 MuJoCo games.

Environment	A3C-model	MBRL-LDN
Ant-v2	3124.7 ± 249.1	3882.4 ± 183.6
HalfCheetah-v2	5963.3 ± 524.7	6593.2 ± 624.8
Hopper-v2	4153.8 ± 362.9	4984.0 ± 423.1
Reacher-v2	5546.3 ± 382.6	7624.9 ± 375.6
Walker2D-v2	5923.6 ± 381.6	6124.7 ± 274.9
InvertedPendulum-v2	7842.1 ± 125.4	10864.3 ± 173.2

Table 5: The episodes of PPO2, PPO2-Model, and PPO2-LDN when reaching the top in 3 MuJoCo games.

Environment	PPO2	PPO2-model	PPO2-LDN
InvertedPendulum-v2	2463 ± 16	2465 ± 17	2175 ± 12
HalfCheetah-v2	2601 ± 32	2254 ± 31	1734 ± 22
Hopper-v2	2603 ± 56	2504 ± 68	2397 ± 63

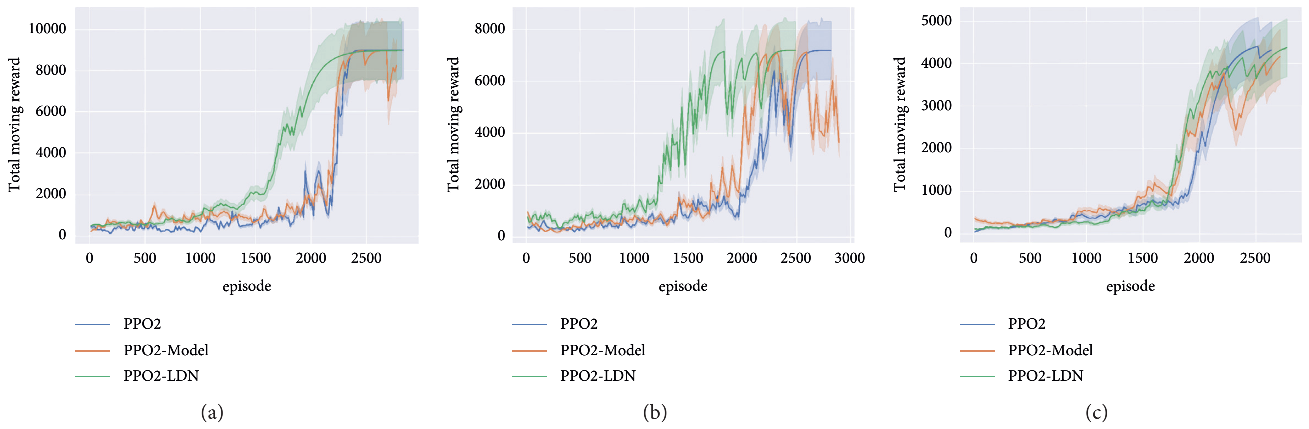


Figure 10: Additional comparative experiments with PPO2, PPO2-Model, and PPO2-LDN. The virtual environment is in MuJoCo. (a) InvertedPendulum-v2. (b) HalfCheetah-v2. (c) Hopper-v2.

means its performance is not much better than PPO2. In Figure 10(c), during 1795 to 2251 episodes, PPO2-LDN and PPO2-Model outperform PPO2 slightly. However, in general, the performance of PPO-LDN is similar to its competitors. The possible reason is that Hopper-v2 contains some inertial mechanisms; that is, the present state will affect the next several states. The dynamics model we used is not a recurrent neural network; it cannot handle time-series data effectively.

6. Conclusion

In this paper, we introduce a model-based reinforcement learning method with learning degree networks, an algorithm for managing imperfect system dynamics models using estimations from learning degree networks. The learning degree is defined and performs as the probability of continuing iteration in the model-based framework. Our approach provides improved sample complexity on a set of OpenAI Gym benchmark tasks, and the experimental results indicate that the model's learning degree gradually increases with training, which makes our method converge faster than traditional DYNA-like algorithms and is more efficient than model-free algorithms. In particular, the threshold reward test showed that the LDN-based method trained faster because low-quality data was discarded. Our work provided further exploration of improved accuracy model data for model-free sample complexity reduction. MBRL-LDN is also verified in MuJoCo games, and the results illustrate that MBRL-LDN performs better than model-based actor-critic learning. Future directions include designing an accumulated reward-based error degree estimation benchmark and deploying it on real-world robotic tasks.

Data Availability

The experiments were performed on six MuJoCo games. The MuJoCo games are commonly used public environments, which can be found at http://www.mujoco.org.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grants nos. 61803227, 61773242, and 61973184), National Key Research and Development Plan of China (2020AAA0108903), Independent Innovation Foundation of Shandong University (Grant no. 2018ZQXM005), and Young Scholars Program of Shandong University, Weihai (Grant no. 20820211010).

References

[1] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Deep deterministic policy gradient (DDPG)-Based energy harvesting wireless communications," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8577–8588, 2019.

[2] D. Silver, A. Huang, C. J. Maddison et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] G. Jing, H. Bai, J. George, and A. Chakrabortty, "Model-free reinforcement learning of minimal-cost variance control," *IEEE Control Systems Letters*, vol. 4, no. 4, pp. 916–921, 2020.

[4] F. Wang, H. Yu, H. Li, X. Li, J. Ye, and H. Yu, "Deterministic diagnostic pattern generation (DDPG) for compound defects," in *Proceedings of the 2008 IEEE International Test Conference*, pp. 1–10, IEEE, Santa Clara, CA, USA, October 2008.

[5] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," *Proceedings of the Learning for Dynamics and Control*, vol. 120, pp. 486–489, 2020.

[6] H. Chung, S. J. Lee, H. B. Jeon, and J. G. Park, "Semi-supervised speech recognition acoustic model training using policy gradient," *Applied Sciences*, vol. 10, no. 10, p. 3542, 2020.

[7] D. Lee, Y. G. Sun, S. H. Kim et al., "DQN-based adaptive modulation scheme over wireless communication channels," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1289–1293, 2020.

[8] V. Mnih, A. P. Badia, M. Mirza et al., "Asynchronous methods for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, pp. 1928–1937, New York, NY, USA, 2016.

[9] T. Tongloy, S. Chuwongin, K. Jaksukam, C. Chousangsuntorn, and S. Boonsang, "Asynchronous deep reinforcement learning for the mobile robot navigation with supervised auxiliary tasks," in *Proceedings of the 2017 2nd International Conference on Robotics and Automation Engineering*, pp. 68–72, IEEE, Shanghai, China, 2017.

[10] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: a brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

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

[12] G. Hartmann, Z. Shiller, and A. Azaria, "Model-based reinforcement learning for time-optimal velocity control," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6185–6192, 2020.

[13] A. Rajeswaran, I. Mordatch, and V. Kumar, "A game theoretic framework for model based reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, pp. 7953–7963, Vienna, Australia, 2020.

[14] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566, IEEE, Piscataway NJ, USA, May 2018.

[15] L. Zou, L. Xia, P. Du et al., "Pseudo dyna-Q: a reinforcement learning framework for interactive recommendation," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 816–824, Houston, TX, USA, February 2020.

[16] J. P. Corriou, "Model predictive control," in *Process Control, Springer, Cham*, pp. 631–677, 2018.

[17] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: a locally linear latent dynamics model for control from raw images," in *Proceedings of the Advances in neural Information Processing Systems. NIPS, Montreal, Quebec, Canada*, pp. 2746–2754, Massachusetts, USA, December 2015.

[18] D. Ha and J. Schmidhuber, "World models," 2018, https://arxiv.org/abs/1803.10122.

[19] G. Kalweit and J. Boedecker, "Uncertainty-driven imagination for continuous deep reinforcement learning," in *Proceedings of the Conference on Robot Learning*, pp. 195–206, Mountain View, CA, USA, 2017.

[20] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," in *Proceedings of the International Conference on Learning Representations*, Vancouver, Canada, 2018.

[21] B. Nanay, "The role of imagination in decision-making," *Mind & Language*, vol. 31, no. 1, pp. 127–143, 2016.

[22] F. Ding, G. Ma, Z. Chen, J. Gao, and P. Li, "Averaged soft actor-critic for deep reinforcement learning," *Complexity*, vol. 2021, no. 39, 16 pages, Article ID 6658724, 2021.

[23] L. Lü, S. Han, W. Zhou, and J. Zhang, "Recruitment-imitation mechanism for evolutionary reinforcement learning," *Information Sciences*, vol. 553, pp. 172–188, 2021.

[24] H. Zhan, F. Tao, and Y. Cao, "Human-guided robot behavior learning: a GAN-assisted reference-based reinforcement learning approach," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3545–3552, 2021.

[25] C. Yu and A. Rosendo, "Risk-aware model-based control," *Frontiers in Robotics and AI*, vol. 813 pages, 2021.

[26] X. Zhao, B. Tao, and L. Qian, "Model-based actor-critic learning for optimal tracking control of robots with input saturation," *IEEE Transactions on Industrial Electronics*, vol. 6, no. 68, pp. 5046–5056, 2020.

[27] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel DDPG method with prioritized experience replay," in *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 316–321, IEEE, Banff, AB, Canada, October 2017.