Received: 5 January 2024 | Revised: 28 February 2024 | Accepted: 9 March 2024 | Published online: 13 March 2024

RESEARCH ARTICLE

Journal of Data Science and Intelligent Systems
2025, Vol. 3(4) 271–281
DOI: 10.47852/bonviewJDSIS42022432

BON VIEW PUBLISHING

# A Model-Based Reinforcement Learning Method with Conditional Variational Auto-Encoder

**Ting Zhu[1], Ruibin Ren[1], Yukai Li[1] and Wenbin Liu[1,2,*]**

[1]*School of Mathematics, Southwest Jiaotong University, China*

[2]*The 30th Research Institute of China Electronics Technology Group Corporation, China*

**Abstract:** Model-based reinforcement learning can effectively improve the sample efficiency of reinforcement learning, but the environment model in this method has errors. The model errors can mislead the policy optimization, leading to suboptimal policy. To improve the generalization ability of the environment model, existing methods often use ensemble models or Bayesian models to build the environment model. However, these methods are computationally intensive and complex to update. Since the generated model can describe the stochastic nature of the environment, this paper proposes a model-based reinforcement learning method based on a conditional variational auto-encoder (CVAE). In this paper, we use a CVAE to learn task-related representations and apply the generative model to predict environmental changes. Considering the problem of multi-step error accumulation, model adaptation is utilized to minimize the difference between simulated and real data distributions. Furthermore, the experiments verified that the proposed method can learn task-relevant representations and accelerate policy learning.

**Keywords:** model-based reinforcement learning, conditional variational auto-encoder, task-relevant representations

## 1. Introduction

Reinforcement learning enables learning by interacting with the environment and mainly solves the problem of sequential decision-making. Deep reinforcement learning, which is said to be the key to general artificial intelligence, has been widely used in robot control, strategic gaming, autonomous driving, and other fields [1–3]. However, reinforcement learning training requires many samples and sample collection is expensive in practical applications. Therefore, improving the sample efficiency is a key issue in reinforcement learning research [4]. Model-based reinforcement learning provides additional information for training by modeling the environment, which can effectively improve the sample efficiency. However, the environment model differs from the real environment in model-based reinforcement learning. Its prediction errors can further mislead the policy evaluation and affect policy optimization. Therefore, it is important to establish an environment model that can accurately depict the features of the environment.

Learning an accurate environment model is challenging due to random noise and limited diversity in training data [5]. Effective representation and model uncertainty measurement can mitigate the negative impact of model errors. On the one hand, appropriate data representation can help the model extract pivotal information about the environment for many high-dimensional tasks, while ignoring much irrelevant background information in the data. On the other hand, the uncertainty of the model reflects its knowledge about the natural environment. Adaptively applying the uncertainty-aware model can minimize the accumulation of errors.

In model-based reinforcement learning, state representations are usually learned by reconstruction or contrastive learning [6–8]. When the observation is complex, the variational auto-encoder (VAE) is often used to project the observation into a low-dimensional latent space [9, 10]. The encoder is often further optimized using contrastive learning to improve the learning of representations. However, the classic VAE usually assumes that the prior distribution of the latent variables is a standard Gaussian, which only guarantees that the model can extract useful information from some of the modes. Considering the uncertainty of the environment model, existing methods often use probabilistic and ensemble models to construct the environment model [11, 12]. The difference among ensemble models reflects the model's cognitive bias toward environmental transition. Both are practical tools for modeling uncertainty.

In order to learn effective representations and capture environmental uncertainty, this paper proposes a model-based reinforcement learning method based on a conditional variational auto-encoders (CVAE) because the generative model can better characterize environmental stochasticity than the discriminative model [13–15]. First, the task-related environmental information is extracted by a prior network. The decoder is constructed using a probabilistic model. Next, the prior network is applied to learn the

*Corresponding author: Wenbin Liu, School of Mathematics, Southwest Jiaotong University and the 30th Research Institute of China Electronics Technology Group Corporation, China. Email: WenbinLiu@my.swjtu.edu.cn

knowledge of the encoder, and it is used for environment model construction. Then, the simulated data features are learned adaptively from the real data features. Finally, the trained environment model is used for policy learning in conjunction with a model-free reinforcement learning approach.

The comprehensive evaluation results show that the method proposed in this paper can effectively improve the sample efficiency, and its convergence performance can reach or even exceed the state-of-the-art algorithm. In a nutshell, our contributions are as follows:

1) We propose a CVAE-based feature extraction model with expressive ability and generalization and can learn task-relevant information.
2) We reduce the instability of the environment model by applying a prior model to learn the encoder using a method of knowledge distillation.
3) We mitigate the bias of the simulated data distribution by reducing the difference between the real and simulated data feature distributions.

This paper is structured as follows: Section 2 introduces methods related to model-based reinforcement learning. In Section 3, the basics required for this paper are presented. A model-based reinforcement learning method with a CVAE is presented in Section 4. Comparative experiments verify the effectiveness of the proposed method in Section 5. In Section 6, we include the limitations and future scope of the work. Finally, the paper is concluded.

## 2. Literature Review

Due to the discrepancy between the environment model in model-based reinforcement learning and the real environment, the convergence performance of model-based reinforcement learning is often lower than that of model-free reinforcement learning. Existing work typically improves the performance of model-based reinforcement learning algorithms by modeling the environment with sufficient accuracy.

With the development of deep learning, neural networks have demonstrated powerful representation capabilities [16]. Therefore, neural networks are often used to model the environment. Ha and Schmidhuber [17] proposed the World Model, which uses a VAE to learn abstract compressed environmental representations and performs model learning and policy optimization based on the latent space. In addition, recurrent neural networks are applied to build a model that combines historical information to predict future state representations. The model based on a VAE [18] can transform the high-dimensional state space into a low-dimensional latent space, but the low-dimensional representation may lose information [15]. Considering the difference between simulated and real data distribution, some works [19, 20] have reduced the bias of the simulated data distribution by minimizing the difference between the feature distributions. Similarly, conditional generative adversarial networks are commonly used to constrain the bias of the data distribution [15, 21]. The difference between simulated and real data is reduced by discriminator training.

The environment may be stochastic and multimodal. However, the approach of directly combining the prior model with the decoder as an environmental model exhibits instability [22]. Gal et al. [23] proposed the DEEP PILCO, incorporating the Bayesian approach to estimate the model uncertainty. Using Bayesian neural networks to model the environment, the posterior distribution of parameters captures the model uncertainty. However, this method could be more precise in estimating the model uncertainty and is complicated in updating the model. The model uncertainty can be categorized into stochastic uncertainty and cognitive uncertainty. Chua et al. [11] proposed an ensemble probabilistic model approach, PETS. The probabilistic neural network model is mainly used to estimate the stochasticity, and the ensemble model is used to estimate the cognitive uncertainty. In addition, some works [24, 25] reduced the impact of model prediction errors on model application through uncertainty perception.

Existing model-based reinforcement learning methods commonly use ensemble models and Bayesian models to construct the environment model [11, 19, 26, 27], but the models of such methods are complex and have high computational resource requirements. The model proposed in this paper uses only a single generative model to learn the environment, which can be generalized to unseen state data.

## 3. Preliminary

### 3.1. Summary of notations

Table 1 shows the notations used in this paper.

**Table 1**
**Summary of notations**

| Notation | Meaning |
|---|---|
| $S$ | The state space |
| $A$ | The action space |
| $s_t$ | State at time $t$ |
| $a_t$ | Action at time $t$ |
| $\Delta s_{t+1}$ | State change at time $t+1$, $\Delta s_{t+1} = s_{t+1} - s_t$ |
| $\mathbf{R}$ | Set of real numbers |
| $r(s,a)$ | Immediate reward from state $s$ after action $a$ |
| $T(s,a)$ | Transition to next state from state $s$ taking action $a$ |
| $\pi(a\|s)$ | Probability of taking action $a$ in state $s$ |
| $\rho_0$ | The distribution of the initial state |
| $\gamma$ | The discount factor |
| $Q(s,a)$ | Value of taking action $a$ in state $s$ |
| $\theta$ | The parameters of the encoder |
| $\phi$ | The parameters of the decoder |
| $\hat{\theta}$ | The parameters of the prior model |
| $p_\theta(z\|x)$ | The variational distribution of $z$ conditioned on $x$ |
| $q_\phi(x\|z)$ | The generative distribution of $x$ conditioned on $z$ |
| $p(z)$ | The prior distribution of $z$ |
| $p_{\hat{\theta}}(z\|y)$ | The prior distribution of $z$ conditioned on $y$ |
| $N(\mu, \sigma^2 \mathrm{I})$ | Gaussian distribution with the mean $\mu$ and the variance $\sigma^2$ |

### 3.2. Reinforcement learning

Reinforcement learning [28] mainly learns the optimal policy by interacting with the environment and the interaction process is usually built as a Markov decision process, denoted as $\{S, A, r, T, \rho_0, \gamma\}$. $S, A$ denotes the state space and action space,

respectively, and $\rho_0$ denotes the initial state distribution. $r: S \times A \to \mathbf{R}$ is the reward function, and $T: S \times A \to S$ is the environmental transition kernel, which is used to represent the state $s_{t+1}$ resulting from the environmental change after the action $a_t$ is performed at the state $s_t$. $\gamma \in (0,1)$ is the discount factor used to balance the weights between long-term benefits and short-term rewards. The policy function $\pi(a|s)$ usually represents the probability of choosing an action $a$ given state $s$. The purpose of reinforcement learning is to find an optimal policy that maximizes the expected cumulative reward, denoted as,

$$\pi^* = \arg\max_{\pi} \mathbf{E}_{\pi}\left[\sum_{t=1}^{H} \gamma^{t-1} r(s_t, a_t) | s_0 \sim \rho_0\right], \tag{1}$$

where $s_{t+1} = T(s_t, a_t)$, $a_t \sim \pi(a|s_t)$ and $H$ is the length of the episode. The environment model in model-based reinforcement learning usually refers to the reward function and the transition kernel.

After each update of the policy function, the expected cumulative reward can be calculated according to Equation (1), which can be used to evaluate the policy. However, this process requires a large amount of data collection, which seriously reduces the efficiency of policy evaluation and updates. Therefore, the state-action value function $Q(s,a)$ is often used to evaluate the value of a policy decision [29], which is expressed as the expected cumulative reward that the agent receives in the future after executing an action $a$.

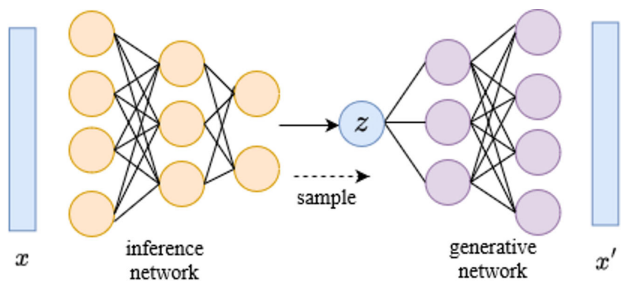
$$Q_{\pi}(s,a) = E_{\pi}\left[\sum_{k=t}^{H} \gamma^{k-1} r(s_k, a_k) |\, s_t = s, a_t = a\right]. \tag{2}$$

### 3.3. Conditional variational auto-encoder

VAE is a deep generative model with latent variables, which mainly applies neural networks to model two complex conditional probability functions [30]. As shown in Figure 1, VAE mainly consists of an inference network and a generative network. The inference network is mainly used to estimate the variational distribution $p_\theta(z|x)$, and the generative network is used to estimate the probability distribution $q_\phi(x|z)$. The inference network can be regarded as an encoder that maps the observed variables $x$ into latent variables $z$. The generative network is a decoder that decodes the sampled latent variables $z$. It is usually assumed that the distribution $p_\theta(z|x)$ follows a Gaussian distribution with diagonalized covariance, whose mean and variance are predicted by the encoder.

The optimization objective of the VAE is to maximize the evidence lower bound (ELBO) [30], which can be regarded as the E-step and M-step in the EM algorithm.

**Figure 1**
**A network framework for variational auto-encoder**




$$\begin{aligned}&\max_{\theta,\phi} \mathrm{ELBO}(q, x; \theta, \phi)\\ &= \max_{\theta,\phi}\left(\mathrm{E}_{z\sim p_\theta(z|x)}[\log q_\phi(x|z)] - \mathrm{KL}(p_\theta(z|x), p(z))\right)\end{aligned} \tag{3}$$

where $p(z)$ is the prior distribution of the latent variables, usually assumed to be isotropic standard Gaussian distribution and $\theta, \phi$ are the parameters of the encoder and decoder, respectively. VAE is often used to extract the features of high-dimensional data, and the size of the control parameters can be reduced by projecting the observation to a low-dimensional state through the encoder [9].

Unlike VAE, the input of the CVAE also contains label information $y$, and the corresponding distribution of latent variables also depends on $y$ [31]. CVAE mainly consists of three parts: an encoder, a decoder, and a prior network, where the prior network $p(z|y)$ represents the prior distribution of the latent variables. The outputs of the encoder, the mean $\mu$ and the variance $\sigma^2$, represent the parameters of the distribution of the latent variable $z$. Therefore, the latent variable distribution can be expressed as, $p_\theta(z|x,y) = \mathcal{N}(z; \mu_\theta, \sigma_\theta^2 \mathrm{I})$. After sampling $z$ from the latent variable distribution, the labeling information $y$ is concatenated with it and inputted into the decoder to obtain $x$.

Using variational inference, the CVAE is optimized by maximizing ELBO [31],

$$\begin{aligned}\max_{\theta,\phi} \mathrm{ELBO}(q, x; \theta, \phi) = \max_{\theta,\phi}(\{&\mathrm{E}_{p_\theta(z|x,y)}[\log q_\phi(x|, y, z)]\\ &- KL(p_\theta(z|x,y)||p(z|y))\})\end{aligned} \tag{4}$$

where $p(z|y)$ is the prior distribution of the latent variable $z$, which is usually obtained through a prior network. The architecture of the prior network is similar to that of the variational encoder except for the inputs.

## 4. Model Learning Based on CVAE

### 4.1. Learning potential transition information

As a common model for feature learning, the VAE can learn essential representations of the data through feature encoding and data reconstruction. However, the features sampled in VAE are not controllable and may lose much information. Thus, this paper utilizes the CVAE to learn environmental transfer features.

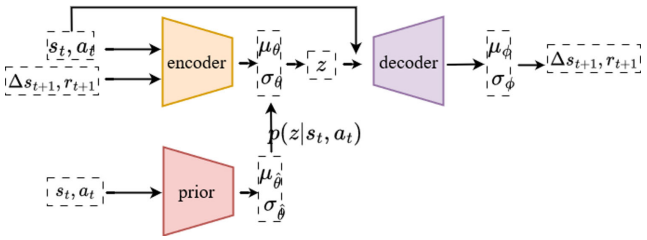
The feature learning framework of this paper is shown in Figure 2, where $(s_t, a_t)$ represents the condition $y$, and $\Delta s_{t+1}, r_{t+1}$ represent the original data $x$. With $p_\theta(z|x,y)$ denoting the latent variable distribution, the latent features can be obtained through reparametrized sampling, $z = \mu_\theta + \grave{o} \cdot \sigma_\theta, \grave{o} \sim N(z; 0, \mathrm{I})$. In order to learn the transfer features corresponding to a specific state-action pair $(s_t, a_t)$, the decoder's input contains the condition labels $(s_t, a_t)$ in addition to the latent features $z$. The decoder is trained to reconstruct $\Delta s_{t+1}, r_{t+1}$, making the encoder learn pivotal features of the transition.

Since the deterministic model has the problem of overfitting, a probabilistic model $q_\phi(\Delta s_{t+1}, r_{t+1}|z_t, s_t, a_t)$ is used to construct the decoder.

$$q_\phi(\Delta s_{t+1}, r_{t+1} | z_t, s_t, a_t) = \mathcal{N}(\Delta s_{t+1}, r_{t+1}; \mu_\phi, \sigma_{\phi^2} \mathrm{I}). \tag{5}$$

Similarly, the output of the encoder $\mu_\phi, \sigma_\phi^2$ is parameterized to represent a Gaussian distribution. For different $(s_t, a_t)$, the decoder $q_\phi$ learns different state change distributions. Sampling $\Delta s_{t+1}$ from $q_\phi$, the next state can be represented as,

**Figure 2**
**The feature learning framework based on CVAE**




$$s_{t+1} = s_t + \Delta s_{t+1}. \tag{6}$$

The loss of CVAE consists of two main components, the KL divergence term and the reconstruction loss,

$$\begin{aligned} &L_{cvae}(\Delta s_{t+1}, r_{t+1}, s_t, a_t; \theta, \phi) \\ &= KL(p_\theta(z|\,\Delta s_{t+1}, r_{t+1}, s_t, a_t)||p(z|\,s_t, a_t)) \\ &- \frac{1}{n}\sum_{i=1}^{n} \log q_\phi(\Delta s_{t+1}, r_{t+1}|s_t, a_t, z_i) \end{aligned} \tag{7}$$

where $z_i$ is sampled from the distribution of the latent variable and $n$ is the number of samples.

## 4.2. Updates of the prior network

To enable environment models to take full advantage of feature models, model-based RL usually constructs the environment model by directly combining the prior model with the decoder [14]. However, due to the difference between the prior and the posterior of the latent variables, the method can lead to model instability [15]. Therefore, after the training of CVAE, this paper applies the prior model to distill the knowledge of the encoder and updates the prior model with the encoder output. The prior model, parametrized to represent the distribution of transfer features $p_{\hat{\theta}}(z_t|s_t, a_t)$, is consistent with the encoder model except for the different inputs. Its inputs and outputs are $(s_t, a_t)$ and $\mu_{\hat{\theta}}, \sigma^2_{\hat{\theta}}$, respectively.

Specifically, the distribution $p_\theta$ represented by the encoder is used as the target value and the distribution $p_{\hat{\theta}}$ represented by the prior model is made close to it. The KL divergence is used to measure the distance between the prior distribution and the posterior distribution, and the prior model is trained by minimizing the KL divergence.

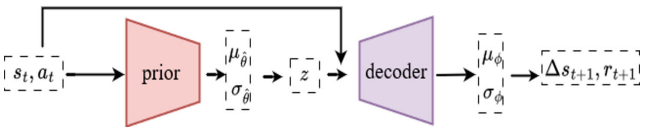
$$L_{prior}(s_t, a_t; \hat{\theta}) = KL\big(p_\theta(z|\,s_t, a_t, \Delta s_{t+1}, r_{t+1})||p_{\hat{\theta}}(z|\,s_t, a_t)\big) \tag{8}$$

## 4.3. The environment model

The feature $z$ obtained by the prior model is similar to the latent feature obtained by the encoder, and the decoder can decode $z$ to obtain $\Delta s_{t+1}, r_{t+1}$. Therefore, as shown in Figure 3, this paper combines the prior model with the decoder to construct the environment model. The prior model learns the distribution of environmental transfer features, while the decoder predicts the state changes and rewards, $q_\phi(\Delta s_{t+1}, r_{t+1}|z_t, s_t, a_t)$.

To optimize the environment model, the model is trained by minimizing the negative log-likelihood of the predicted data.

**Figure 3**
**The environment model**

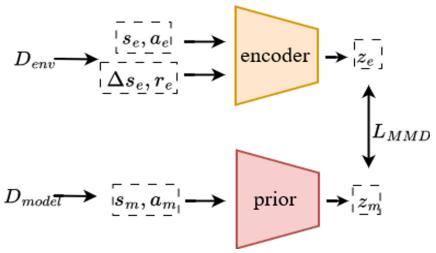



$$L_{env}(s_t, a_t; \hat{\theta}, \phi) = -\frac{1}{n}\sum_{i=1}^{n} \log q_\phi(\Delta s_{t+1}, r_{t+1}|\,s_t, a_t, z^i_{\hat{\theta}}). \tag{9}$$

## 4.4. Model adaptation

Generative models can mitigate the overfitting problem of one-step prediction of environment models, but the problem of multi-step prediction still exists [32]. Prediction errors accumulate with multi-step predictions, resulting in deviations between the simulated trajectory and the real trajectory. This bias will mislead the assessment of the value of the policy. Thus, this paper uses the model adaptation method to mitigate the problem. Learning the invariant features between simulation data and real data can reduce the offset of simulation data distribution [19]. The model framework is shown in Figure 4.

**Figure 4**
**The feature adaptation model**




When the distance between the simulated and real data distribution is small, the corresponding feature distributions should be similar. Therefore, the environment model is optimized by minimizing the distance between the feature distributions. The encoder and prior model extracts the features of real data and simulated data, respectively. In this paper, the maximum mean discrepancy (MMD) is used to measure the distance between the real and simulated feature distributions. The deviation of the simulated data distribution is reduced by minimizing the MMD loss,

$$\begin{aligned} L_{MMD} &= \left\| \frac{1}{n}\sum_{i=1}^{n} f(z_{ei}) - \frac{1}{m}\sum_{j=1}^{m} f(z_{mj}) \right\|^2_H \\ &= \left\| \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i'=1}^{n} k(z_{ei}, z'_{ei}) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} k(z_{ei}, z_{mj}) + \frac{1}{m^2}\sum_{j=1}^{m}\sum_{j'=1}^{m} k\left(z_{mj}, z'_{mj}\right) \right\| \end{aligned} \tag{10}$$

where $f$ is the mapping function on feature $z$. The MMD is usually computed implicitly using the kernel function $k(x, y)$ with the kernel trick.

After the environment model is built, it can be applied to the learning of the controller in model-free reinforcement learning. As shown in Algorithm 1, with the SAC method, this paper uses the environment model as a data augmenter for the branch rollout [27]. The generated simulation data are mixed with real data for value and policy learning.

**Algorithm 1:** CVAE-SAC

**Input:** the environment dataset $D_{env}$.
**Output:** the policy $\pi$, the value function $Q$.
1 Initialization: the model dataset $D_{model}$, the policy $\pi$, the value function $Q$, the environment model $M$.
2 **For** episode $i = 1$ to $N$ **do**
3 Train the CVAE model and build the environment model $M$.
4 **For** step $j = 1$ to $H$ **do**
5 Interact with the environment using the policy $\pi$. Add data $(s_j, a_j, s_{j+1}, r_j)$ to $D_{env}$.
6 Model rollouts: Randomly sample the start state $s$ from $D_{env}$. Perform the k-step rollout from $s$ in $M$. Add data $(s, a, s^{'}, r)$ to $D_{model}$.
7 **End**
8 Sample data from $D_{env}$ and $D_{model}$ to update policy $\pi$ and the value function $Q$ by applying SAC method.
9 **End**

## 5. Experiments and Results

### 5.1. Experiment settings

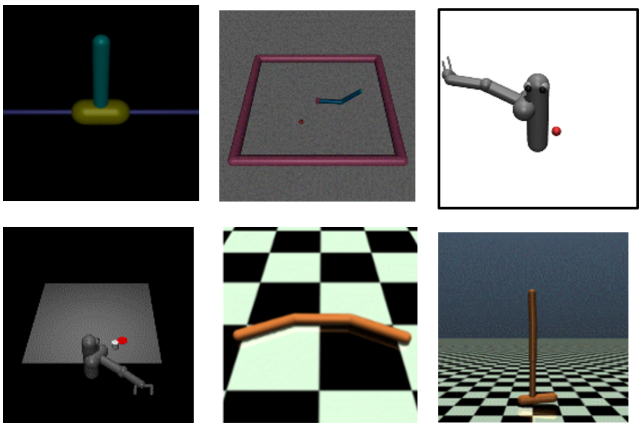
This paper evaluated the proposed methods and other baselines on several MuJoCo continuous control tasks from OpenAI Gym with a maximum horizon of 150. The experiments in this paper were conducted under five different random seeds, and the curve results were averaged under random seeds. The following are the experimental details involved in the CVAE experiment. The environment model consists of a prior network and a decoder. The prior network is a multilayer perceptron (MLP) with three hidden layers of size 200 and the decoder is a MLP with two hidden layers of size 200. The encoder and the prior network have the same architecture except for the different input sizes. The dimensions of the encoded features are the sum of the dimensions of states and actions. The length of the rollout in the experiment is 1.

#### *5.1.1. The experimental scenarios*

The experiments in this paper mainly use the Mujoco scenarios of Inverted Pendulum, Reacher, Reacher3D, Pusher, and Swimmers, as shown in Figure 5.

1) Inverted Pendulum: consists of a cart that moves linearly, with a pole fixed at one end and free at the other. The cart can be pushed to the left or right, and the goal is to balance the bar on top of the cart by applying force to the cart.
2) Reacher: a robotic arm with two joints. The goal is to move the tip of the robot near a randomly generated target. The action is the torque applied at the two hinges, represented as a two-dimensional vector. The reward consists of the distance from the tip to the target position and the action penalty. Reacher3D is identical to the Reacher target, but its robot arm is three-dimensional.
3) Pusher: a multi-jointed robotic arm, consisting of shoulder, elbow, forearm, and wrist joints. The goal is to move the target cylinder to the target position using the robot's end-effector. The action is to apply torques to the hinges of the shoulder and elbow, expressed as a seven-dimensional vector. The reward is composed of fingertip-to-object position distance, object-to-target position distance, and action penalty.
4) Swimmer: consists of three links and two joints. The goal is to move the object to the right as fast as possible by applying torque to the joints. The action is the torque applied to the two joints, represented as a two-dimensional vector and the reward consists of a forward reward and a penalty for the action.
5) Hopper: a two-dimensional one-legged figure. The goal is to make hops that move in the forward (right) direction. The action is applying torques on the three hinges connecting the four body parts. The reward consists of a healthy reward, a forward reward, and a penalty for the action.

**Figure 5**
**The experiment scenarios: Inverted Pendulum, Reacher, Reacher3D, Pusher, Swimmer, and Hopper**



#### *5.1.2. The baseline algorithms*

The comparison algorithms used for the experiments in this paper are MBPO [27], AMPO [19], SAC [33], and DDPG [34].

1) MBPO utilizes an ensemble probabilistic model to learn the environment, mainly considering when to trust the environmental model. In order to minimize the impact of model errors on policy evaluation, the planning length is determined according to the impact of model errors.
2) AMPO is an improved algorithm based on MBPO, which mainly studies the problem of simulation data bias. Combined with the domain adaption method, it enhances the optimization of the model by minimizing the distance between the feature distributions of the simulated data and the real data.
3) SAC is an offline learning method that mainly introduces entropy regularization to achieve the exploration of the policy, which can prevent the policy from falling into the local optimum, and speed up the training.
4) DDPG combines DQN and AC methods, utilizes neural networks to fit the value function and policy function, and performs well in many continuous control problems.

### 5.2. Feature visualization

This paper records the data feature of the proposed method during the training process in the Mujoco environment to test whether the feature model can learn the task-relevant features. Figure 6 shows the representation of the features handled by t-SNE 2-dimensions reduction. The color of the scatter indicates the value of the state, the lighter the color, the higher the value. As shown in the figure, neighboring points in the latent feature

**Figure 6**
**A visualization of the latent features of the training data**

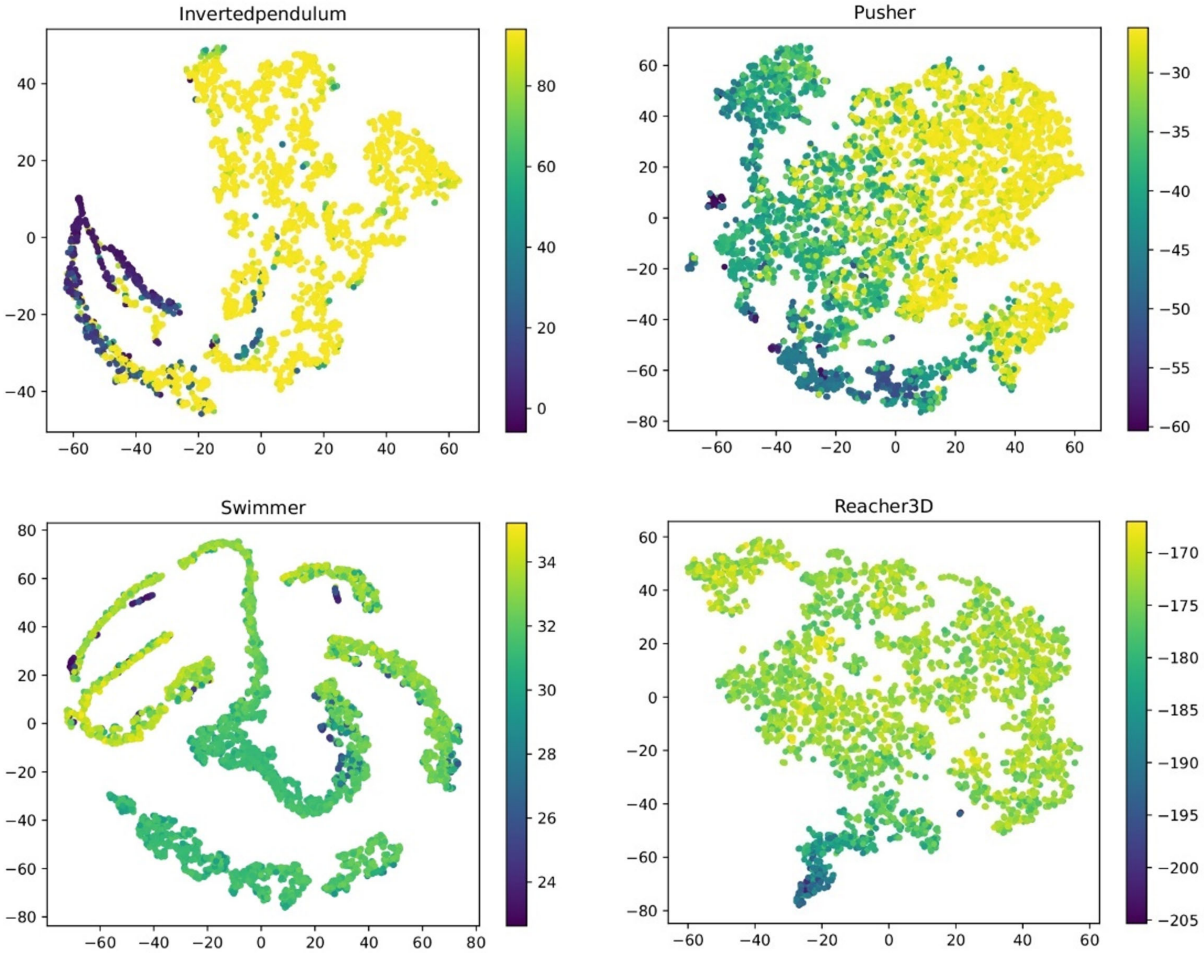



**Figure 7**
**Training trajectories of robot arm fingertips under MBPO and CVAE algorithms**

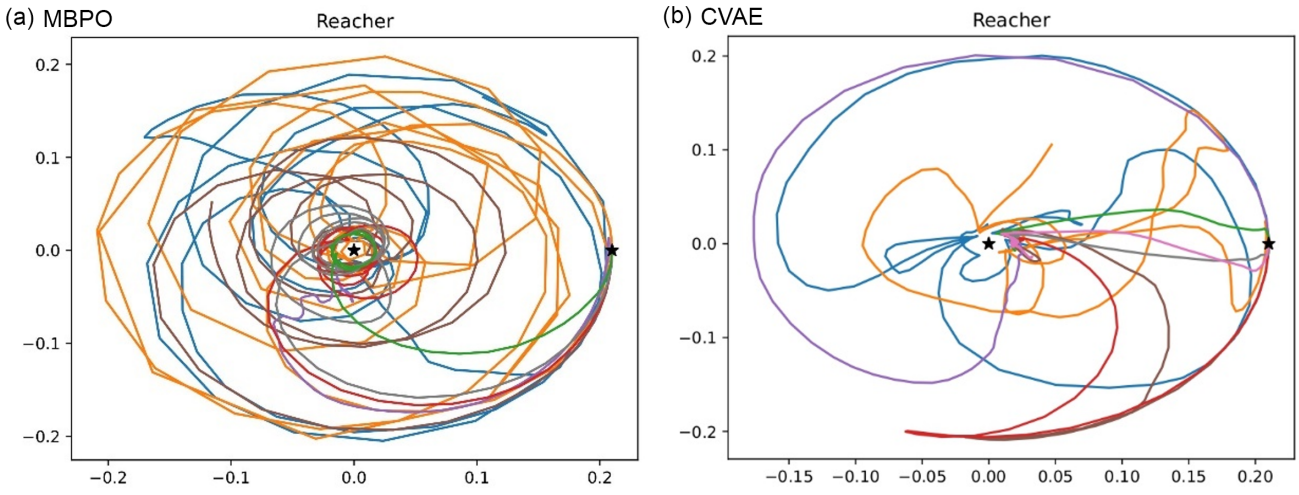


**Figure 8**
**Distribution of training data and visualization of latent features using the CVAE algorithm**

(a) Distribution of training data, color shades indicate value amounts

(b) Latent feature visualization of training data

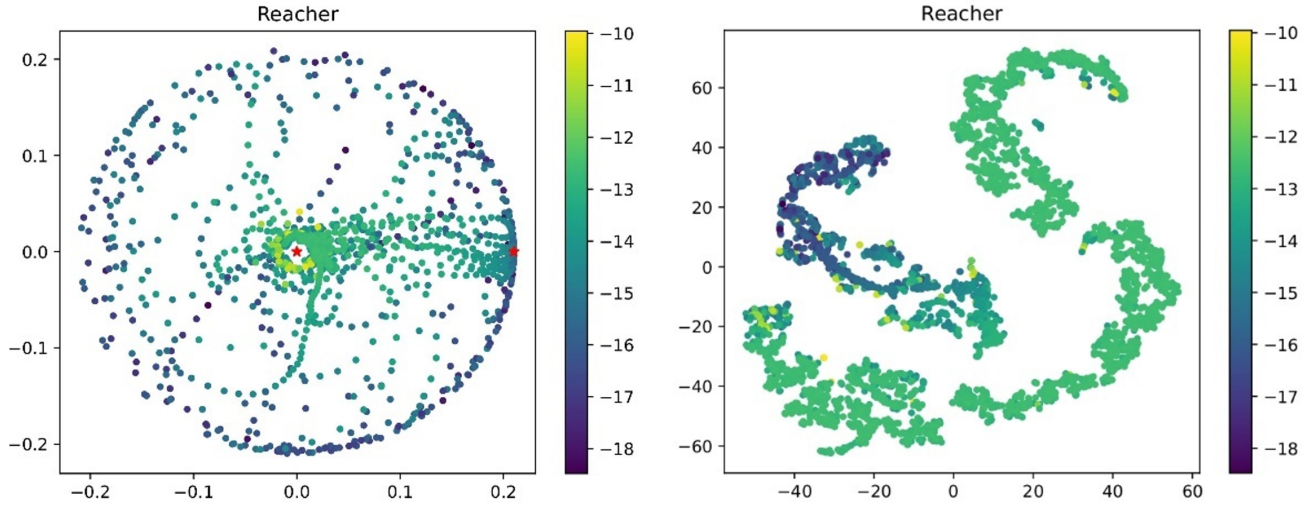
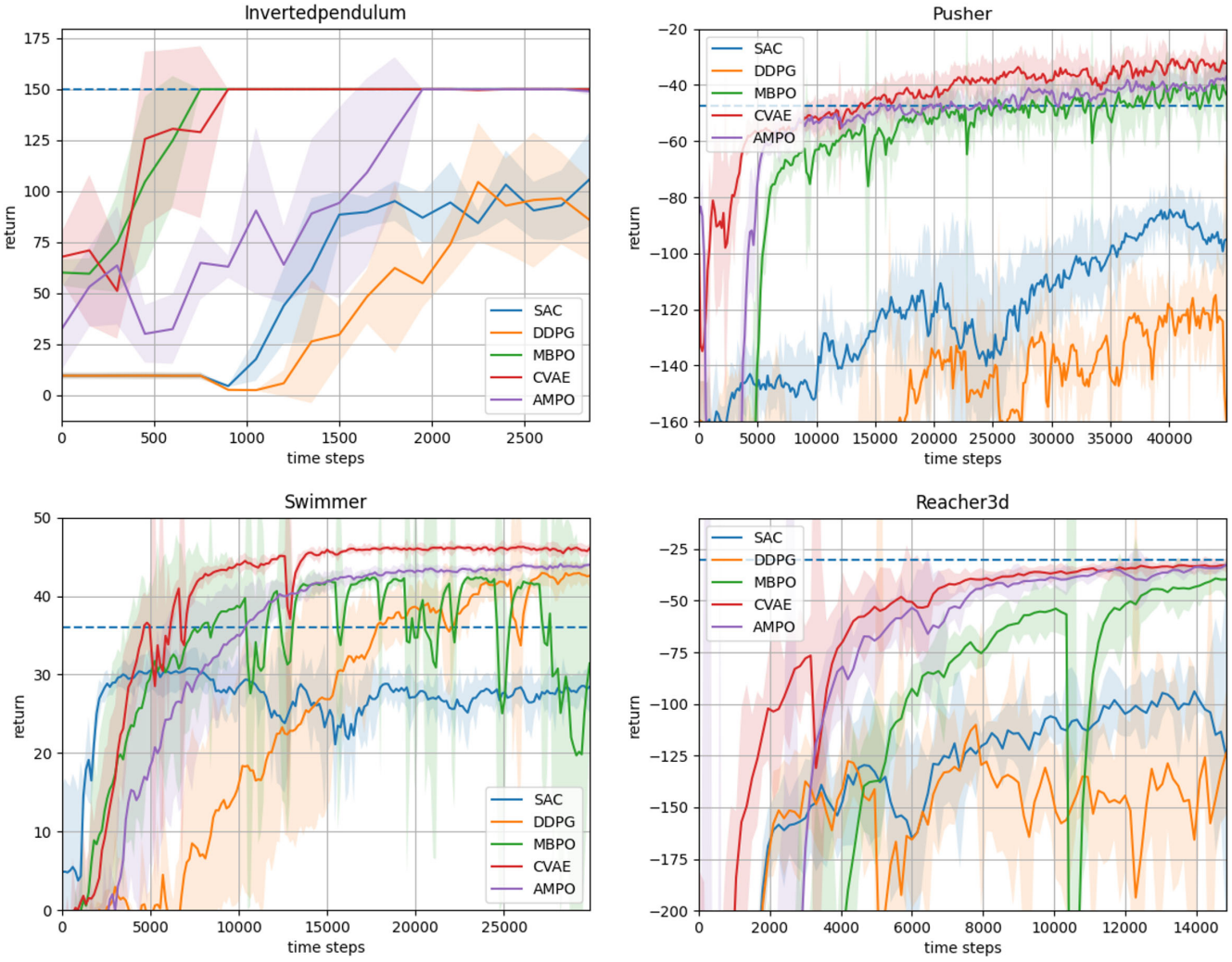



**Figure 9**
**Comparison of the convergence performance of the benchmark algorithms on continuous tasks such as Inverted Pendulum**



space have similar values, which indicates that the prior network can project states with similar values to a similar latent space. The region division between different value features is evident in Inverted Pendulum, Reacher3D, and Swimmer scenarios. This implies that the bias of state values generated by randomness in the feature sampling process is not too large. In conclusion, the prior model can learn task-relevant information.

## 5.3. Training trajectories

In the Reacher environment, the agent's task is to make the fingertips of the robot arm reach the target position as soon as possible. A coordinate system is established to visualize the trajectory of the fingertip of the robot arm and use the target position as the origin. Figure 7(a) and Figure 7(b) individually present the fingertip trajectories navigating from the starting point (0.209,0) to the designated target position when employing the MBPO and CVAE methodologies, respectively. Trajectories of different colors in the same figure indicate the test trajectories after different training times, and trajectories of the same color in both figures indicate the same training time. As shown in the figure, MBPO takes more time to reach the target position in the pre-training period, while CVAE reaches the target position faster with the same training time. The controllers of MBPO and CVAE are the same, the only difference is the source of simulation data. This indicates that the simulation data generated by CVAE are more accurate and can provide effective information for policy learning.

Figure 8(a) shows the allocation of training data for CVAE, and the scatter color's darkness shows the corresponding data's value. CVAE has less data in the low-value region, and the training data are mainly distributed in the high-value region. Figure 8(b) shows the visualization of the latent layer features after t-SNE dimensionality reduction. The figure shows that CVAE can separate the high-value and low-value data. It shows that the environment model of CVAE combines the advantages of generative modeling, which can learn the nature of the surrounding data with less data.

## 5.4. Performance testing of the algorithm

Figure 9 records the cumulative reward changes of various algorithms during the training process in continuous control tasks such as the Inverted Pendulum. The CVAE represents the algorithm generated by combining the environment model proposed in this paper with the controller in MBPO. As shown in the figure, the final cumulative rewards of CVAE can be optimal and converge relatively quickly. The environment model in CVAE enables rapid learning of effective task information to assist in policy learning. However, the proposed methods are only suitable for simple continuous control tasks. Figure 10(a) records the reward curves of each method, while Figure 10(b) shows the return at convergence in the Hopper. As shown in Figure 10(b), the convergence performance in the proposed method is lower than the model-free method under the complex continuous control task. This indicates that the model architecture of the proposed method is relatively simple and cannot accurately describe the changes in the complex environment.

The solid line represents the mean of the cumulative rewards under the 5 random seeds, the shaded portion indicates the range of variation, and the dashed line is the rewards when SAC converges.

## 5.5. Effectiveness of model adaptation

In order to test the effect of the model adaptation on the environment model, this paper records the performance of the removed optimization module, denoted as CVAE_no_adapt, in the Swimmer scenario.

**Figure 10**
**Comparison of the convergence performance of the benchmark algorithms in Hopper**

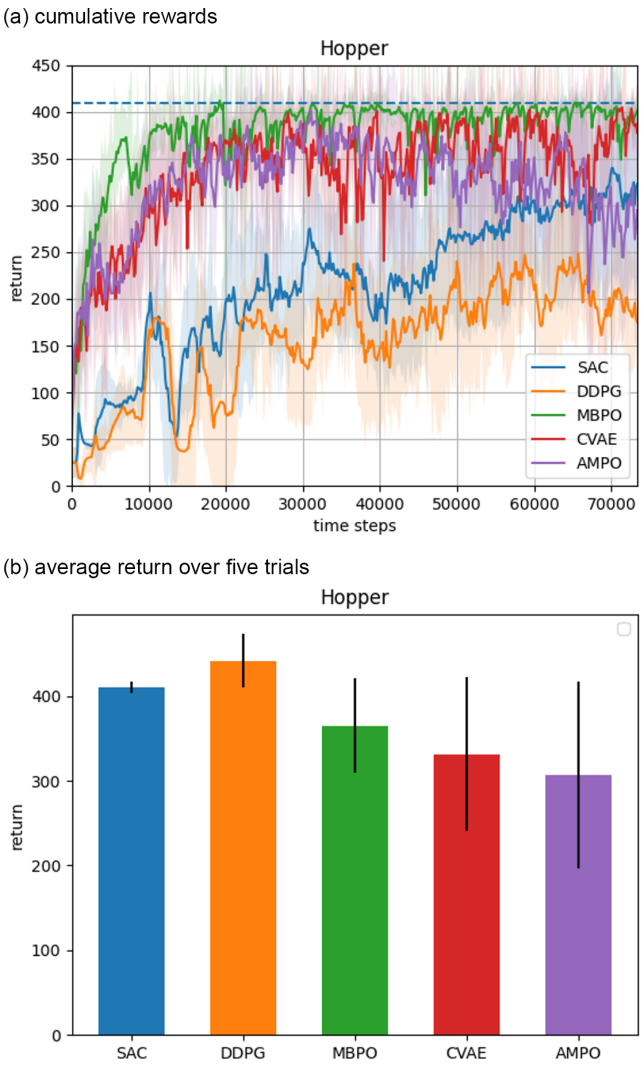



Figure 11(a) records the cumulative reward changes over the training process for multiple model-based algorithms. Figure 11(b) records the KL loss for the prior model, reflecting the difference between the prior model and the encoder. As shown in the figure, CVAE accelerates the algorithm's convergence compared to other algorithms. Figure 12(a) and Figure 12(b) respectively record the distribution of the hidden layer features of CVAE and CVAE_no_adapt, and the distribution of the features learned by CVAE_no_adapt is scattered. This indicates that the model adaptation module can optimize model learning and reduce the bias of simulated data.

## 6. Discussion

This paper proposes a model-based reinforcement learning method based on CVAE. Considering the stochastic nature of the environment and the uncertainty of model prediction, this paper applies a CVAE to learn the transfer features of environmental changes. It uses knowledge distillation to learn feature encoding and combines the decoder to construct the environment model. In addition, to mitigate the accumulation of multi-step prediction errors, a model adaptive method is applied to optimize the

**Figure 11**
**Cumulative rewards variation in model-based algorithms and KL loss for the prior model training**

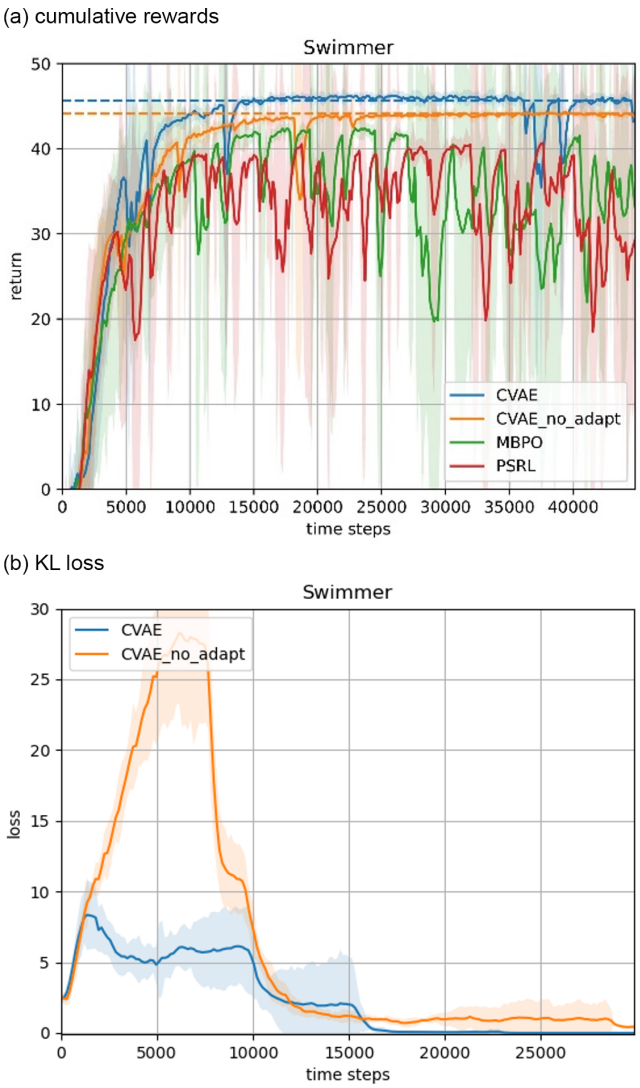
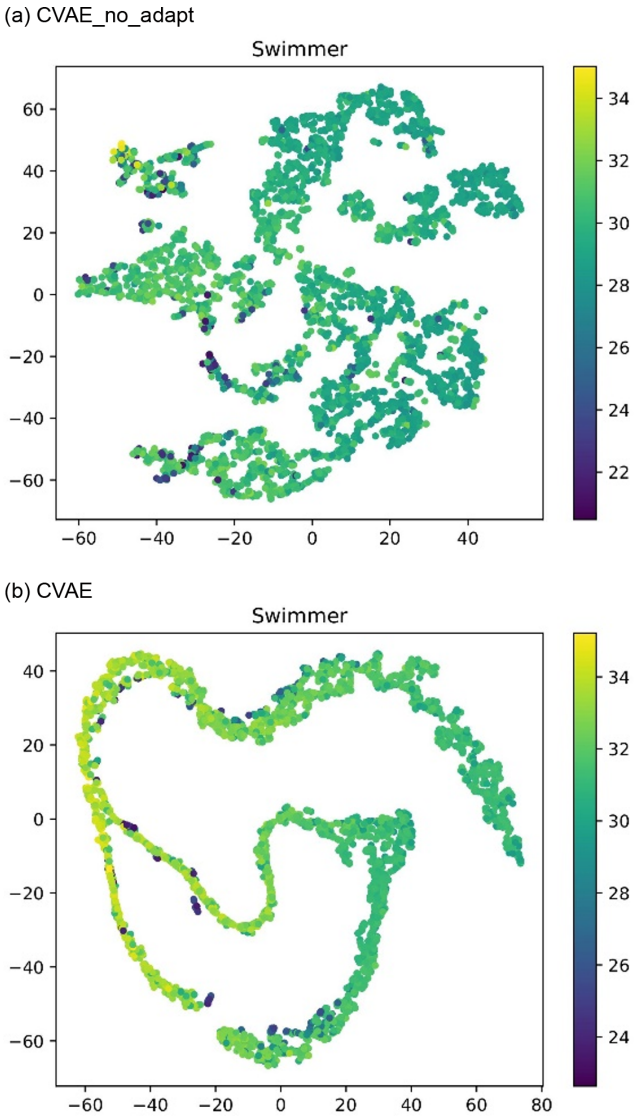



**Figure 12**
**Latent feature visualization image of the training data**




environment model. Experiments show that the proposed method can effectively learn effective features of the environment in the simple continuous control task, which can assist the policy learning and enable the return and sample efficiency to exceed that of the benchmark methods. Section 5.3 shows that the proposed method can learn the task information faster and accelerate the policy to explore the high-value region compared with other model-based methods.

However, since the network architecture of the environment model is relatively simple, the model cannot accurately capture the information of the complex environment. Section 5.4 shows that for complex tasks, the model prediction errors of the proposed method mislead the policy optimization, which makes the final return lower than that of the model-free reinforcement learning method. In addition, the proposed method only focuses on reducing model prediction errors in terms of model construction, while the effect of model uncertainty on policy optimization still exists. In future work, it can be suggested to apply complex neural networks to construct the model framework and consider the effect of historical information to extend the proposed method to complex and high-dimensional tasks. In addition, it is also possible to adaptively apply the environment model in terms of model application, reducing the impact of model errors.

## 7. Conclusion

The prediction errors of the environment model in model-based reinforcement learning can accumulate with the planning steps and affect the policy evaluation. In order to mitigate the problem, this paper proposes an environment model based on CVAE. Specifically, CVAE is applied to learn the potential information of the data, and the stochasticity of the generative model is exploited to improve the generalization of the environment model. Through comprehensive experimental evaluation, it is verified that the model in this paper can effectively learn the features related to the task, accelerate the learning of the policy, and effectively improve the sample efficiency. However, this paper does not consider the application of the model based on the effect of error and leaves the adaptive application of the model for future work.

## Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

## Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

## Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

## Author Contribution Statement

**Ting Zhu:** Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing. **Ruibin Ren:** Conceptualization, Methodology, Formal analysis, Investigation, Resources, Writing – review & editing, Visualization, Supervision, Project administration. **Yukai Li:** Software, Validation, Formal analysis, Data curation, Writing – review & editing. **Wenbin Liu:** Software, Validation, Resources, Writing – review & editing, Project administration.

## References


[1] Wang, L., Yang, S., Yuan, K., Huang, Y., & Chen, H. (2023). A combined reinforcement learning and model predictive control for car-following maneuver of autonomous vehicles. *Chinese Journal of Mechanical Engineering*, *36*(1), 80. https://doi.org/10.1186/s10033-023-00904-7

[2] Wu, J., Huang, Z., & Lv, C. (2023). Uncertainty-aware model-based reinforcement learning: Methodology and application in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, *8*(1), 194–203. https://doi.org/10.1109/TIV.2022.3185159

[3] Wu, J., Huang, C., He, H., & Huang, H. (2024). Confidence-aware reinforcement learning for energy management of electrified vehicles. *Renewable and Sustainable Energy Reviews*, *191*, 114154. https://doi.org/10.1016/j.rser.2023.114154

[4] Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E. D., . . . , & Ba, J. (2019). Benchmarking model-based reinforcement learning. *arXiv Preprint: 1907.02057*. https://doi.org/10.48550/arXiv.1907.02057

[5] Luo, F.-M., Xu, T., Lai, H., Chen, X.-H., Zhang, W., & Yu, Y. (2024). A survey on model-based reinforcement learning. *Science China Information Sciences*, *67*(2), 121101. https://doi.org/10.1007/s11432-022-3696-5

[6] Ho, J., & Ermon, S. (2016). Generative adversarial imitation learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 4572–4580.

[7] Nabati, O., Tennenholtz, G., & Mannor, S. (2023). Representation-driven reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning, 202*, 25588–25603.

[8] Niv, Y. (2019). Learning task-state representations. *Nature Neuroscience*, *22*(10), 1544–1553. https://doi.org/10.1038/s41593-019-0470-8

[9] Liang, X.-X., Feng, Y.-H., Huang, J.-C., Wang, Q., Ma, Y., & Liu, Z. (2020). Novel deep reinforcement learning algorithm based on attention-based value function and autoregressive environment model. *Journal of Software*, *31*(4), 948–966. http://dx.doi.org/10.13328/j.cnki.jos.005930

[10] Rafailov, R., Yu, T., Rajeswaran, A., & Finn, C. (2021). Offline reinforcement learning from images with latent space models. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control, 144*, 1154–1168.

[11] Chua, K., Calandra, R., McAllister, R., & Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 4759–4770.

[12] Deisenroth, M., & Rasmussen, C. E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning,* 465–472.

[13] Moerland, T. M., Broekens, J., Plaat, A., & Jonker, C. M. (2023). Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, *16*(1), 1–118. http://dx.doi.org/10.1561/2200000086

[14] Moerland, T. M., Broekens, J., & Jonker, C. M. (2017). Learning multimodal transition dynamics for model-based reinforcement learning. In *1st Scaling-Up Reinforcement Learning (SURL) Workshop*.

[15] Zhao, T., Wang, Y., Li, G., Kong, L., Chen, Y., Wang, Y., . . . , & Yang, J. (2021). A model-based reinforcement learning method based on conditional generative adversarial networks. *Pattern Recognition Letters*, *152*, 18–25. https://doi.org/10.1016/j.patrec.2021.08.019

[16] Selvaraj, J., & Jayanthy, A. K. (2023). Automatic polyp semantic segmentation using wireless capsule endoscopy images with various convolutional neural network and optimization techniques: A comparison and performance evaluation. *Biomedical Engineering: Applications, Basis and Communications*, *35*(06), 2350026. https://doi.org/10.4015/S1016237223500266

[17] Ha, D., & Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2455–2467.

[18] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational Bayes. *arXiv Preprint: 1312.6114*. https://doi.org/10.48550/arXiv.1312.6114

[19] Shen, J., Zhao, H., Zhang, W., & Yu, Y. (2020). Model-based policy optimization with unsupervised model adaptation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2823–2834.

[20] Shen, J., Lai, H., Liu, M., Zhao, H., Yu, Y., & Zhang, W. (2023). Adaptation augmented model-based policy optimization. *Journal of Machine Learning Research*, *24*(218), 1–35.

[21] Dong, W., Liu, S., & Sun, S. (2023). Safe batch constrained deep reinforcement learning with generative adversarial network. *Information Sciences*, *634*, 259–270. https://doi.org/10.1016/j.ins.2023.03.108

[22] Rempe, D., Birdal, T., Hertzmann, A., Yang, J., Sridhar, S., & Guibas, L. J. (2021). HuMoR: 3D human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision,* 11488–11499. https://doi.org/10.1109/ICCV48922.2021.01129

[23] Gal, Y., McAllister, R., & Rasmussen, C. E. (2016). Improving PILCO with Bayesian neural network dynamics models. In *Data-Efficient Machine Learning Workshop*.

[24] Malekzadeh, P., Hou, M., & Plataniotis, K. N. (2023). Uncertainty-aware transfer across tasks using hybrid model-based successor feature reinforcement learning☆.

*Neurocomputing*, *530*, 165–187. https://doi.org/10.1016/j.neucom.2023.01.076

[25] Wu, Z., Yu, C., Chen, C., Hao, J., & Zhuo, H. H. (2022). Plan to predict: Learning an uncertainty-foreseeing model for model-based reinforcement learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 15849–15861.

[26] Fan, Y., & Ming, Y. (2021). Model-based reinforcement learning for continuous control with posterior sampling. In *Proceedings of the 38th International Conference on Machine Learning*, 3078–3087.

[27] Janner, M., Fu, J., Zhang, M., & Levine, S. (2019). When to trust your model: Model-based policy optimization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 12519–12530.

[28] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. USA: MIT Press.

[29] Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292. https://doi.org/10.1007/BF00992698

[30] Nielsen, M. A. (2015). *Neural networks and deep learning*. http://neuralnetworksanddeeplearning.com/

[31] Sohn, K., Yan, X., & Lee, H. (2015). Learning structured output representation using deep conditional generative models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems, 2*, 3483–3491.

[32] Marino, D. L., & Manic, M. (2019). Modeling and planning under uncertainty using deep neural networks. *IEEE Transactions on Industrial Informatics*, *15*(8), 4442–4454. https://doi.org/10.1109/TII.2019.2917520

[33] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, 1861–1870.

[34] Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., Dhruva, T., ..., & Lillicrap, T. P. (2018). Distributed distributional deterministic policy gradients. *arXiv Preprint: 1804.08617*. https://doi.org/10.48550/arXiv.1804.08617

**How to Cite:** Zhu, T., Ren, R., Li, Y., & Liu, W. (2025). A Model-Based Reinforcement Learning Method with Conditional Variational Auto-Encoder. *Journal of Data Science and Intelligent Systems*, *3*(4), 271–281. https://doi.org/10.47852/bonviewJDSIS42022432