Appendix

Anonymous Author(s)
Affiliation
Address
email

A Preliminaries

Markov Decision Process (MDP). A standard MDP can be represented as a tuple: $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, T)$, where $\mathcal{S}$ denotes the state set, $\mathcal{A}$ denotes an action set, $\mathcal{P}$ is the transition function: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ and $\mathcal{R}$ is the reward function: $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$. $\gamma \in [0, 1)$ is a discount factor and T is the decision horizon. The target of the agent is to optimize its policy to maximize the expected discounted cumulative reward.

Frameskip. Frame-skipping may be viewed as an instance of (partial) open-loop control, under which a predetermined sequence of (possibly different) actions is executed without heed to intermediate states. Aiming to minimize sensing, **?** proposes a framework for incorporating variable-length open-loop action sequences in regular (closed-loop) control. The primary challenge in general open-loop control is that the number of action sequences of some given length d is exponential in d. Consequently, the main focus in the area is on policies to prune corresponding data structures (**?**). Since action repetition restricts itself to a set of actions with size linear in d, it allows for d itself to be set much higher in practice. With frame-skipping, the agent is only allowed to sense every d state: that is, if the agent has sensed a state s_t at time step $t >= 0$, it is oblivious to states$s_{t+1}, s_{t+2}, ..., s_{t+d-1}$, and next only observes s_{t+d}.

Variational Auto-encoder. The variational auto-encoder (VAE) is a directed graphical model with certain types of latent variables, such as Gaussian latent variables. A generative process of the VAE is as follows: a set of latent variable z is generated from the prior distribution $p_\theta(z)$ and the data x is generated by the generative distribution $p_\theta(x|z)$ conditioned on $z : z \sim p_\theta(z), x \sim p_\theta(x|z)$. In general, parameter estimation of directed graphical models is often challenging due to intractable posterior inference. However, the parameters of the VAE can be estimated efficiently in the stochastic gradient variational Bayes (SGVB) framework, where the variational lower bound of the log-likelihood is used as a surrogate objective function. In this framework, a proposal distribution $q_\theta(x|z)$, which is also known as a "recognition" model, is introduced to approximate the true posterior $p_\theta(x|z)$. The multilayer perceptrons (MLPs) are used to model the recognition and the generation models. Assuming Gaussian latent variables, the first term of Equation A can be marginalized, while the second term is not. Instead, the second term can be approximated by drawing samples $z^{(l)}(l = 1, ..., L)$ by the recognition distribution $q_\theta(x|z)$, and the empirical objective of the VAE with Gaussian latent variables is written as follows:

$$L_{VAE}(\phi, \psi) = \frac{1}{L} \sum_{\theta} (x|z^{(l)}) - KL\big(q_\phi(z|x)||N(0, I)\big) \tag{1}$$

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

Layer	Actor Network	Critic Network
Fully Connected	(state dim, 256)	(statedim + υ dim + latent space dim, 128)
Activation	ReLU	ReLU
Fully Connected	(256, 128)	(256, 128)
Activation	ReLU	ReLU
Fully Connected	(128,latent space dim) and υ dim	(128, 1)
Activation	Tanh	None

Table 1: Network Structures for DRL Methods

B Experimental Details

B.1 NETWORK STRUCTURE

Our codes are implemented with Python 3.7.9 and Torch 1.7.1. All experiments were run on a single NVIDIA GeForce GTX 2080Ti GPU. Each single training trial ranges from 4 hours to 17 hours, depending on the algorithms and environments. We will open source code in the near future.

Our TD3 is implemented with reference to `github.com/sfujim/TD3` (TD3 source-code). DDPG and PPO are implemented with reference to `https://github.com/sweetice/Deep-reinforcement-learning-with-pytorch`. For a fair comparison, all the baseline methods have the same network structure (except for the specific components of each algorithm) as our MARS-TD3 implementation. As shown in Tab.1, we use a two-layer feed-forward neural network of

Model Component	layer	dimension
Conditional Encoder Network	Fully Connected (encoding)	$(\mathbb{R}^x, 256)$
	Fully Connected (condition)	(stae dim + υ dim, 256)
	Element-wise Product	ReLU (encoding), ReLU(condition)
	Fully Connected	$(256, 256)$
	Activation	ReLU
	Fully Connected (mean)	(256, latent space dim)
	Activation	None
	Fully Connected (log std)	(256, latent space dim)
	Activation	None
Conditional Decoder, Prediction Network	Fully Connected (latent)	(latent space dim, 256)
	Fully Connected (condition)	(stae dim +υ dim, 256)
	Element-wise Product	ReLU (encoding), ReLU(condition)
	Fully Connected	(256, 256)
	Activation	ReLU
	Fully Connected (υ)	(256, action dynamic transition)
	Activation	None
	Fully Connected (reconstruction)	(256, multi-step action dim)
	Activation	None
	Fully Connected	$(256, 256)$
	Activation	ReLU
	Fully Connected (prediction)	(256, state dim)
	Activation	None

Table 2: Network structures for the Multi-step action representation (MARS).

256 and 256 hidden units with ReLU activation (except for the output layer) for the actor network for all algorithms. For DDPG the critic denotes the Q-network. For PPO, the critic denotes the V-network. All algorithms (TD3, DDPG, PPO) output two heads at the last layer of the actor network, one for latent action and another for dynamic transition potential.

The structure of MARS is shown in Tab.2. We use element-wise product operation (**?**) and cascaded head structure (**?**) to our model.

B.2 Hyperparameter

For all experiments, we use the raw state and reward from the environment, and no normalization or scaling is used. No regularization is used for the actor and the critic in all algorithms. An exploration noise sampled from N(0, 0.1) (**?**) is added to all baseline methods when selecting an action. The discounted factor is 0.99 and we use Adam Optimizer (**?**) for all algorithms. Tab.3 shows the common hyperparameters of algorithms used in all our experiments.

Hyperparameter	Frameskip-TD3	Multistep-TD3	MARS-PPO	MARS-TD3	MARS-DDPG	
Actor Learning Rate	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$
Critic Learning Rate	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$3e^{-4}$	$3e^{-4}$	$1e^{-3}$
Representation Model Learning Rate	None	None	None	$1e^{-4}$	$5e^{-3}$	$5e^{-3}$
Discount Factor	0.99	0.99	0.99	0.99	0.99	0.99
Batch Size	128	128	128	128	128	128
Buffer Size	$1e5$	$1e5$	$1e5$	$1e5$	$1e5$	$1e5$

Table 3: A comparison of common hyperparameter choices of algorithms. We use 'None' to denote the 'not applicable' situation.

B.3 Additional Implementation Details

For PPO, the actor network and the critic network are updated every 2 and 10 episode respectively for all environments. The clip range of the PPO algorithm is set to 0.2 and we use GAE (**?**) for a stable policy gradient. For DDPG, the actor network and the critic network is updated at every 1 environment step. For TD3, the critic network is updated every 1 environment step and the actor network is updated every 2 environment steps.

The default latent action dim is 8, we set the KL weight in representation loss L_{MARS} as 0.5. Environment dynamic prediction loss weight β is 5 (default).

C Additional experiment

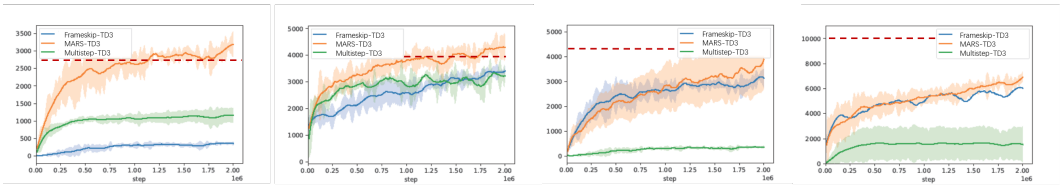
C.1 Performance of model-based reinforcement learning algorithms on Intermittent-MDP tasks.

Methods	Ant (fixed-Intermittent)	Hopper (fixed-Intermittent)	Ant (random-Intermittent)	Hopper (random-Intermittent)
TD-MPC	1795.4 ± 375.6	1795.4 ± 214.8	1447.2 ± 694.8	1073.6 ± 157.1
Dreamer-v2	1648.2 ± 417.5	788.1 ± 116.4	1064.7 ± 694.8	974.7 ± 201.8
TD3-multistep	2673.6 ± 316.8	1359.7 ± 258.3	2795.4 ± 264.1	1211.6 ± 169.5
MARS-TD3	2572.9 ± 248.1	3762.7 ± 371.4	3105.7 ± 412.6	2647.9 ± 204.8

Table 4: Comparison between MBRL and MFRL in intermittent control tasks, average of 3 runs.

The results in Table 4 show that the model-based reinforcement learning approach is significantly lower than our approach in all four scenarios, and even slightly worse than using TD3 for direct multi-step decision making. We analyze that this is because the errors caused by the mismatch (sub-optimal) of the dynamic model will accumulate due to multi-step decision-making, resulting in sub-optimal policy. Unfortunately, the training of dynamic models is data hungry (high cost), that is, a large amount of high-quality expert data is required to ensure the accuracy of the model shop, which is difficult to obtain, especially in real-world scenarios.

C.2 Validation of the combination of MARS and online methods



(a) Hopper (b) Ant (c) Walker (d) HalfCheetah

Figure 1: The performance of the methods on four simulated tasks. The curve and shade denote the mean and a standard deviation over 5 runs.

We use the mainstream online reinforcement learning algorithm TD3 in combination with MARS and compare it with the baseline mentioned in Sec.**??** in four tasks. We set the interval to 10 time steps, requiring the policy to generate an effective action sequence $a_{t:t+9}$ based on the received state s_t.

For all tasks, we set the dimension of z_t to 12 and the scaling parameter β to 4. We set the warm-up (stage 1) step to 300000 and 100000 for the Mujoco tasks and the navigation task respectively. The results in Figure 1 show that MARS-TD3 outperforms the other baselines in all fixed Intermittent-MDP tasks and achieve comparable performance with perfect-TD3 in most tasks. **This further proves that MARS can effectively improve the effectiveness of Online DRL on fixed Intermi tasks.**

C.3 Generalization of MARS

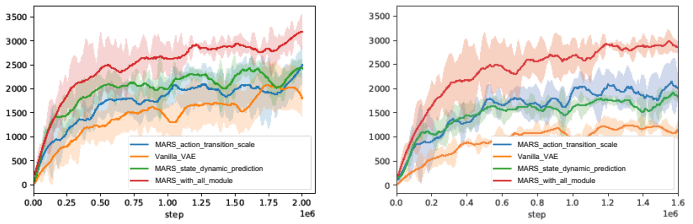
We test MARS with popular RL methods on three tasks: Hopper, Walker, and hardMaze. To make the experiment fair, we used the same parameters for all methods and implemented them based on public code. We use each RL algorithm to train on three tasks under the ideal setting and compare them with their corresponding improvement methods. To show the optimal score after the algorithm convergence, we train all the algorithm's 2000000 time steps. The results in Tab.5 show that all methods can learn effective policies with the help of MARS and perform similarly to their ideal settings. The differences in scores are mainly due to the variation in performance of the RL algorithms. In summary, MARS can be combined with different methods to provide a reliable action space for solving Intermittent-MDP as normal MDP with RL.

Benchmarks	MARS-PPO	MARS-DDPG	MARS-TD3
Maze hard	256 | 0.7 ↑	243 | 2.5 ↑	311 | 16.3 ↑
Hopper	2851.4 | 13.5 ↓	1815.6 | 184.3 ↑	3384 | 53.1 ↑
Walker	3831.2 | 285.1 ↓	1032.7 | 201.9 ↓	4821.6 | 427.6 ↑

Table 5: The parameters of all methods are optimized by grid search. The results of applying MARS to popular RL algorithms on three random interaction interval tasks. The maximum interaction interval is set to 8. Each data in the table is in the following format: MARS-RL score | the score difference compared to the perfect dense interaction baseline. ↓ denotes the score of MARS lower than the dense interaction baseline. ↑ denotes the score of MARS is higher. All scores are averaged over 5 runs.

C.4 Details of Ablation study

We conducted two experiments to show how well the two mechanisms of MARS work together. Although the results of randomized Intermittent-MDP and fixed Intermittent-MDP are slightly different, the same conclusion can be derived: The green curves in Figure 2 demonstrate that the representation model with increased action transition scale is much better than the original VAE. This means that dynamic transition potential can create an latent action space by explicitly modeling the dependence between multi-step actions. The blue curves also show that VAE with state dynamic prediction is better than the original VAE because it can represent action sequences that have similar environmental effects at close locations. Finally, the red curves show that the two mechanisms work well together in MARS, and combining them improves representation ability.

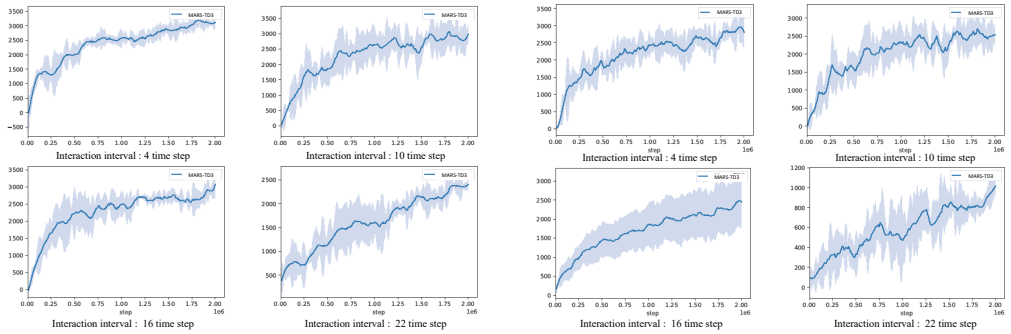


(a) Fix Intermittent-MDP Walker (b) Random Intermittent-MDP Walker

Figure 2: Details of ablation study. The curve and shade denote the mean and a standard deviation over 5 runs.

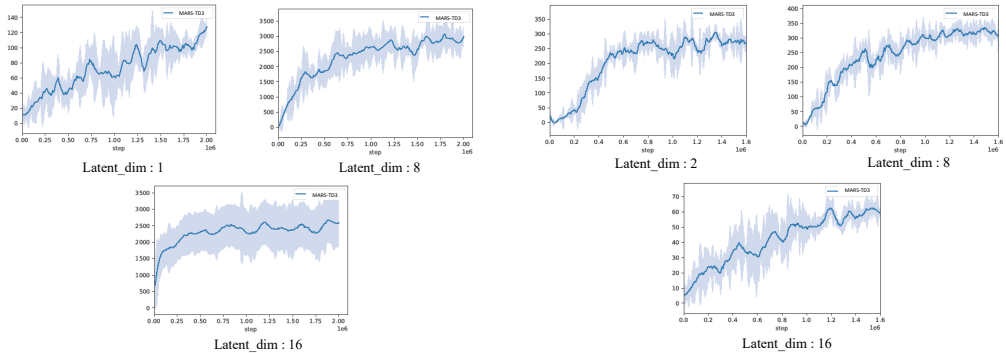
C.5 Validity verification of multi-style interaction intervals

To further demonstrate the effectiveness of MARS in diverse intermittent control scenarios. For fixed interaction control tasks, we uniformly set the forbidden interaction duration and conducted four experiments on Hopper. The results in Figure 3(a) show that MARS can solve most tasks effectively and still guarantee good scores at long intervals, but the effectiveness of MARS decreases significantly when the interval is too long (which is not common in real-world scenarios). We believe



(a) Fixed Intermittent-MDP scenarios (b) Random Intermittent-MDP scenarios

Figure 3: The curve and shade denote the mean and a standard deviation over 5 runs.



(a) Results on Hopper (b) Results on MAZE

Figure 4: The curve and shade denote the mean and a standard deviation over 5 runs.

that this is because VAE is unable to effectively characterize excessively long sequences, leading to the failure of multi-step action space modeling.

In addition, to observe the sensitivity of MARS to interaction intervals on random Intermittent-MDP tasks, we uniformly set the forbidden interaction duration and conducted four experiments on Hopper. The results in Figure 3(b) show that in random Intermittent-MDP scenarios, MARS performs well in both short and medium-interval scenarios. However, convergence changes slowly in the very long interval scenario, and the score is only half that of the medium interval task. Because MARS's representational capabilities are not perfect for modeling long action sequences for extremely long-spaced tasks (even if this setting rarely occurs in real-world scenarios). Therefore, in the future, we hope to find more suitable representation models to overcome this problem.

C.6 The influence of Latent action space dimension on algorithm effect

The representation space dimension of VAE is an important hyperparameter. If the latent space dimension is too low, a large amount of original data information will be lost, resulting in invalid representation space. On the contrary, when the latent space dimension is too large, the calculation amount of the model will be increased. To verify the sensitivity of MARS to latent space dimensions, we test it on two tasks with different original action dimensions. We set up four sets of latent space dimensions for fixed Intermittent-MDP Hopper (interaction interval time step: 8, original action dimension: 3, so the action sequence dimension to be modeled is 24). The learning curve in Figure 4(a) shows that for raw data of such high dimensions, when the latent space dimension is set

too low, the latent space information will be lost, resulting in the convergence failure of reinforcement learning policies. On the contrary, too high a latent space dimension increases the complexity of reinforcement learning policy exploration.

In addition, we set up four comparison experiments on the 2dmaze task with a lower dimension of the original action sequence (interaction interval time step: 4, original action dimension: 2, so the action sequence dimension to be modeled is 8). The experimental results in Figure 4(b) show that the suboptimal policy can be learned when the latent space dimension is low, because the original data dimension is low. So the low-dimensional latent space loses less information. The score increases as the latent space dimension increases. However, when the latent space dimension is too high, the score will drop significantly, which is because of the exploration difficulties brought by high-dimensional latent space.

C.7 The influence of environment steps of warmup stage

In this section, we conduct some additional experimental results for a further study of MARS from different perspectives: We provide the exact number of samples used in the warm-up stage (i.e., stage 1 in Algorithm **??** in each environment in Tab.6. The number of warm-up environment steps is about $5\% \sim 10\%$ of the total environment steps in our original experiments. Moreover, we also conducted some experiments to further reduce the number of samples used in the warm-up stage (at most 80% off). See the colored results in Tab.6. MARS can achieve comparable performance with $< 3\%$ samples of the total environment steps.

Conclusion: The number of warm-up environment steps is about $5\% \sim 10\%$ of the total environment steps in our original experiments. The number of warmup environment steps can be further reduced by at most 80% off (thus leading to $< 3\%$ of the total environment steps) while the comparable performance of our algorithm remains.

Environment	Warm-up steps (original)	Warm-up steps (new)	Total Env. Steps
Hopper	400000(0.08|3219.1)	100000(0.02|3086.4)	5000000
Ant	400000(0.08|4305.7)	100000(0.02|4025.6)	5000000
Walker	400000(0.08|4961.3)	100000(0.02|4792.6)	5000000
HalfCheetah	400000(0.08|6593.2)	100000(0.02|6071.2)	5000000
2dmaze-medium	100000(0.083|127.8)	30000(0.025|118.5)	1200000
2dmaze-hard	100000(0.083|327.6)	35000(0.0292|296.1)	1200000

Table 6: The exact number of samples used in warm-up stage training in different environments. The column of 'original' denotes what is done in our experiments; the column of 'new' denotes additional experiments we conduct with fewer warm-up samples (and proportionally fewer warm-up training). For each entry $x(y|z)$, x is the number of samples (environment steps), y denotes the percentage number of $\frac{warm-up\ environment\ steps}{number\ of\ total\ environment\ steps\ during\ the\ training\ process}$, and z denotes the corresponding performance of MARS-TD3.