ROBUST MEMORY AUGMENTATION BY CONSTRAINED LATENT IMAGINATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The latent dynamics model summarizes an agent's high dimensional experiences in a compact way. While learning from imagined trajectories by the latent model is confirmed to have great potential to facilitate behavior learning, the lack of memory diversity limits generalization capability. Inspired by a neuroscience experiment of "forming artificial memories during sleep", we propose a robust memory augmentation method with Constrained Latent ImaginatiON (CLION) under a novel actor-critic framework, which aims to speed up the learning of the optimal policy with virtual episodic. Various experiments on high-dimensional visual control tasks with arbitrary image uncertainty demonstrate that CLION outperforms existing approaches in terms of data-efficiency, robustness to uncertainty, and final performance.

1 INTRODUCTION

Model-based Reinforcement Learning (MBRL) approaches benefit from the knowledge of a model, allowing them to summarize the agent's past experiences and foresight future outcomes into a compact latent space. For complex high-dimensional visual domain tasks, the latent dynamic model has shown great potential to enable efficient long-sighted behavior learning (Hafner et al. (2019b), Hafner et al. (2019a)). Recurrent Stochastic State Model (RSSM) (Hafner et al. (2019b)) used in the PlaNet (Hafner et al. (2019b)) and Dreamer (Hafner et al. (2019a)) is the state-of-the-art solution for building latent dynamics in MBRL, in which there are both a stochastic path and a deterministic path to summarize the experiences. The historical information, which we called memory, is mainly embedded through the deterministic path to reliably remember the experiences for multiple time steps. However, such deterministic embedding is a trade-off between the model prediction error and its efficiency to generate useful trajectories. In prior works, imagined trajectories branch off from the states of sequence sampled from the agent's past experiences with a same deterministic memory embedding. The efficiency for generating useful trajectories is limited by the interaction between the agent and the environment, which leads to the lack of policy robustness to the noise on the deterministic memory vector caused by observation uncertainty and model error.

Inspired by a neuroscience experiment in inducing sleeping mice's artificial memories of a specific reward situation (Gundersen (2015)), which shows for the first time that artificial memories can be implanted into the brains of sleeping animals, we investigate the integration of memory augmentation techniques with model-based reinforcement learning to enable agents learning behaviors from trajectories they have never really experienced. As shown in Figure 1, performing augmentation can improve the data-efficiency significantly without any interaction because every augmented memory vector corresponds to a state transition sequence. We propose a robust memory augmentation approach by Constrained Latent ImaginatiON (CLION) that aims to improve the policy robustness by effectively enhancing memory diversity. Different arbitrary transforms are implanted on the deterministic hidden states of the latent trajectories sampled from the dataset. Every augmented state is chosen as the initial states of the imagined trajectories, which enable the agent to learn how to obtain more rewards from the trajectories it has not experienced before. The optimal policy is learned with the memory augmented latent trajectories by a novel actor-critic algorithm, in which the state values optimize Bellman consistency for imagined rewards, and the policy maximizes the state values by propagating their analytic gradients back through the dynamics.

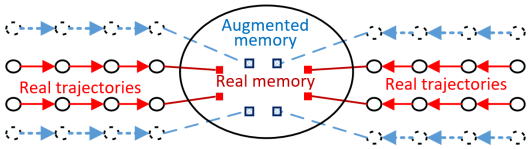


Figure 1: Memory augmentation in latent space.

In this paper, we aim to improve the policy robustness by effectively enlarge the model's latent space with the enhancement of agent's memory diversity by performing augmentation on the deterministic memory vectors. Our main contributions are:

(1) We improve the latent model used in Dreamer and PlaNet by adding two constraints to guide the augmentation in latent space to meet a basic condition that the latent states near the embedded states from real data should also be mapped to real physical states.

(2) We proposed a latent trajectory augmentation framework to improve the policy robustness to the latent noise and data-efficiency, giving the confidence of every augmented trajectory simultaneously expected to filter out the augmented data in the area with poor model prediction and make full use of the reliable augmented trajectories.

2 Constrained Latent Stochastic State Model

2.1 Recurrent Stochastic State Model

Learning the optimal policy by latent imagination requires an accurate dynamics model without loss of generality. RSSM can facilitate long-term predictions and enable the agent to imagine thousands of trajectories in parallel. It represents both the historical memory and the dynamic uncertainty by a recurrent neural network. Denote the sequence of observations $\tilde{o} = \{o_0, o_1, \ldots, o_T\}$, actions $\tilde{a} = \{a_0, a_1, \ldots, a_T\}$ and rewards $\tilde{r} = \{r_0, r_1, \ldots, r_T\}$, with corresponding stochastic latent states $\tilde{s} = \{s_0, s_1, \ldots, s_T\}$ and memories $\tilde{h} = \{h_0, h_1, \ldots, h_T\}$. The RSSM consists of four key components: (1) the representation model $p(s_t|h_t, o_t)$, which encodes observations and actions to create continuous vector-valued latent states with Markovian transitions and embeds the long-term historical information h_t recurrently; (2) the transition model $q(s_t|s_{t-1}, a_{t-1}, h_{t-1})$, which is built by a recurrent neural network, predicts future latent states with historical memory and actions; (3) the observation model $q(o_t|h_t, s_t)$, which is used to provide a reconstruction learning signal; and (4) The reward model $q(r_t|s_t, h_t)$ predicts the rewards given to the latent states.

The components of the RSSM are optimized jointly to increase the variational lower bound (Jordan et al. (1999)) of $\log p(\tilde{o}, \tilde{r} \mid \tilde{u})$, which includes reconstruction terms for observations and rewards and a KL regularizer. The expectation is taken under the dataset and representation model,

$$\begin{aligned} \mathcal{J}_M &= \mathbb{E}_p \left(\sum_t \left(J_O^t + J_R^t + J_D^t \right) \right) \\ J_O^t &= \log q(o_t|s_t, h_t) \qquad\qquad J_R^t = \log q(r_t|s_t, h_t) \\ J_D^t &= \mathrm{KL}\left[p\left(s_t|s_{t-1}, a_{t-1}, o_t\right) \| q\left(s_t|s_{t-1}, a_{t-1}, h_{t-1}\right) \right] \end{aligned} \tag{1}$$

Here, we implement the transition model as a recurrent neural network, the representation model as a convolutional neural network (CNN), the observation model as a transposed convolutional neural network , and the reward model as a dense network.

2.2 Constrained RSSM for Memory Augmentation

The RSSM embeds the long-term memory by the recurrent neural network and outperforms the non-recurrent network in many model-based tasks. However, the memory embedded in the deterministic hidden state results in the lack of memory diversity, which limits the generality of the latent dynamics model. Memory augmentation is an emerging technology in neuroscience whereby either virtual or

real memories can be implanted into an agent's brain. One benefit is to enable the agent to remember things they may not otherwise be able to experience.

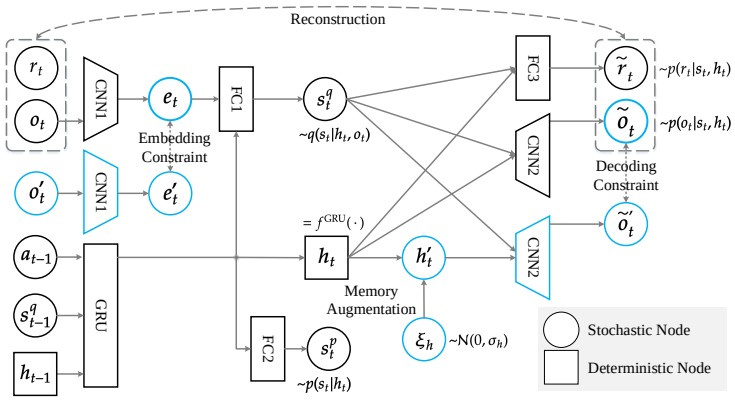
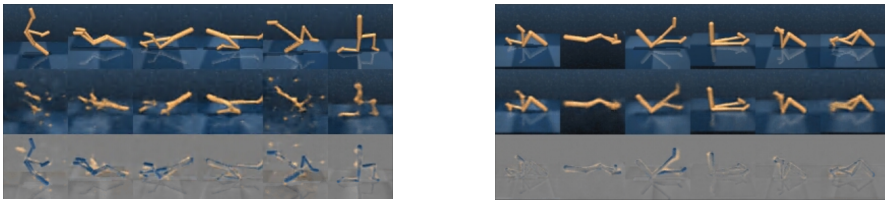


Figure 2: Constrained RSSM framework. CNN, GRU and FC represent a convolutional neural network, a GRU-cell, and a fully-connected layer, respectively.

However, the augmented memory obtained by intuitively adding random transforms to the memory vector could lead to mismatching between augmented trajectories and real physical states. As shown in Figure 3a, if we add an arbitrary noise to the memory embedding h_t, the augmented latent state (h'_t, s'_t) in RSSM results in low-fidelity reconstructed images that couldn't be mapped to real existed physical states. Such a phenomenon suggests that RSSM is sensitive to the memory noise. It will damage the policy because both the model prediction error during the imagination and the observation uncertainty in the testing process will introduce noise on the memory vector. In an attempt to ensure



(a) RSSM (without constraints)

(b) Ours (with constraints)

Figure 3: Reconstruction with augmented memory. The first row is the original image input, the second row is the image reconstructed by (h'_t, s'_t), and the third row is reconstruction error.

the latent model is robust to the added memory noise, we propose a Constrained RSSM (CRSSM) framework for memory augmentation with two constraints: (1) embedding constraint, which ensures the embedding consistency between the original input o_t and transformed input o'_t (we perform (-5, 5) degree random rotation on the original image o_t), and (2) decoding constraint, which ensures the consistency between augmented latent states and the real physical states. As shown in Figure 2, o'_t is an arbitrary transformed image from the original image input o_t, e_t is the encoding feature of original input o_t, e'_t is the encoding feature of augmented input o'_t, ξ_h is a noise that obeys $N(0, \sigma_h)$ distribution, h'_t is the augmented memory which adds the noise ξ_h to the hidden state h_t, and $\tilde{o}_t$ is the reconstructed image from (h'_t, s_t). The CRSSM is to minimize $\mathcal{J}_M$ in equation 1 with two constraints:

$$\begin{aligned} &\min \mathcal{J}_M \\ s.t. &||e'_t - e_t||_2 \leq \delta_1 \\ &||\tilde{o}_t - o_t||_2 \leq \delta_2 \end{aligned} \tag{2}$$

As demonstrated in Figure 3b, our proposed CRSSM generates high-fidelity reconstructions, which suggests the enriched dynamics representation is capable of allowing agents to learn behaviors by latent imaginations with leveraging diverse memory. Note that we leave margin δ_1 and δ_2 between

the augmented branch's output and the non-augmented branch's output instead of requiring them to be entirely consistent. It can be seen from Figure 3b that the reconstructed graph is still different from the ground truth in detail. For instance, in the third column, the robot's leg which close to the ground is straight in the ground truth, while in the reconstructed graph of the latent augmentation, the leg is bent. In summary, the two constraints used in CRSSM aims to guide the augmentation in latent space to meet a basic condition that the latent states near the embedded states from real data should also be mapped to real physical states.

3 MEMORY AUGMENTATION AND CONSTRAINED LATENT IMAGINATION

3.1 LATENT TRAJECTORY AUGMENTATION WITH DIVERSE MEMORY

By leveraging CRSSM, we propose a Constrained Latent ImaginatiON (CLION) framework that aims to enable the agent to learn the policy in a larger latent space to improve the policy robustness by enlarging the capacity and diversity of the deterministic part that deployed to store memory. We add several zero-mean noises $\xi_h \sim N(0, \sigma_h^2)$ on the memory vector h_t in the replay buffer to extend the agent's memory which includes diverse embedded historical information. Therefore, the agent could learn behaviors with the trajectories in a larger latent space, rather than just the agent's real experiences. The latent trajectories are imagined from both original states and the augmented states by the learned transition model. The CLION framework can be summarized as equation 3, for the i-th augmented trajectory, the augmentation process contains two key steps: (1) start point resampling with diverse memory and (2) trajectory imagination (model-based rollout).

$$\begin{aligned} &\text{Resample start point:} \\ &\quad h_t^i \sim N(h_t, \sigma_h^2), \qquad s_t^i \sim N(m_t, \sigma_t^2) \\ &\text{Trajectory imagine:} \\ &\quad h_t^i, s_t^i \sim p(h_t^i, s_t^i | h_{t-1}^i, s_{t-1}^i, a_{t-1}^i), \qquad a_t^i \sim \pi(h_t^i, s_t^i) \end{aligned} \tag{3}$$

In order to further mitigate the model error's impact, we also consider the model's reliability of the augmented trajectories by adding a trajectory evaluator. We evaluate every imagined trajectory predicted by the model by an evaluator and use the trajectories with different weights which depend on its reliability. In the area where the model has reliable predict accuracy, the augmented data should be able to improve the policy robustness and data efficiency significantly and we only filtered out the augmented data in the area with poor model prediction.

Trajectory evaluation is widely used in imitation learning, the Generative Adversarial Imitation Learning (GAIL) Ho & Ermon (2016); Torabi et al. (2018) utilize a GAN-like framework to learn the expert policy. In GAIL, a discriminator is used to judge whether the trajectory is generated by an expert. Similarly, we use a trajectory evaluator to discriminate between real sequence and overshoot sequence predicted by the model as well as possible. In every iteration, we sample a number of state-action sequences generated by real interactions from the dataset as real sequences $Seq_{real} = \text{Concat}\{s_t, h_t, a_t, s_{t+1}, h_{t+1}\}$. Then we use the latent model to predict the overshoot trajectories from the same start points as the fake sequences $Seq_{fake} = \text{Concat}\left\{s_t, h_t, a_t, \hat{s}_{t+1}, \hat{h}_{t+1}\right\}$ to train the evaluator. The evaluator aims to output the probability that the input is the real trajectory, that is equal to maximize the following objective function

$$J_{eva} = \text{EVA}(Seq_{real}) - \text{EVA}(Seq_{fake}) \tag{4}$$

3.2 POLICY OPTIMIZATION WITH MEMORY AUGMENTED TRAJECTORIES

The policy $\pi(s; \varphi)$ is optimized by imagined trajectories $\{s_\tau, a_\tau, r_\tau\}_{\tau=t}^{t+H}$ from original data and augmented data predicted by the transition model. We estimate every state's value in the imagined trajectory to trade off the difficulty of long-horizon back-propagation and efficiency of gradient utilization. That means a single trajectory $\{s_\tau, a_\tau, r_\tau\}_{\tau=t}^{t+H}$ generate many sub-trajectories with different rollout steps. As shown in Figure 4, every sub-trajectory is used to update the actor and critic with different reliable weight $w(s_\tau)$ given by the trajectory evaluator. The reliable weight $w(s_\tau)$ is

given by the trajectory evaluator and is normalized in batch.

$$W(s_\tau) = \mathrm{EVA}(s_\tau, a_\tau, s_{\tau+1}), \quad s_\tau^i \sim (D, D^{aug})$$
$$w_{s_\tau^i} = \frac{W(s_\tau^i)}{\sum_{i=0}^{N}\sum_{\tau=t}^{t+H} W(s_\tau^i)} \tag{5}$$

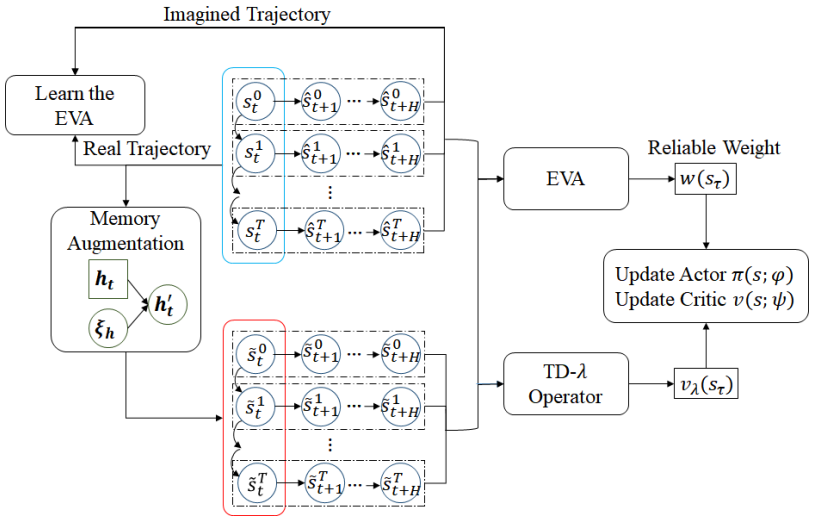


Figure 4: The CLION framework

In an attempt to trade off bias and variance, we calculate the $V(s_\tau)$ by TD-λ method (Srinivas et al. (2018)) similar to Dreamer, which is an exponentially-weighted average of the estimates for different k to make the estimation robust to different predict horizon.

$$\mathrm{V}_\lambda(s_\tau) \doteq (1-\lambda)\sum_{n=1}^{H-1}\lambda^{n-1}\mathrm{V}_\mathrm{N}^n(s_\tau) + \lambda^{H-1}\mathrm{V}_\mathrm{N}^H(s_\tau) \tag{6}$$

$$\mathrm{V}_\mathrm{N}^k(s_\tau) \doteq \mathrm{E}_{q_\theta, q_\varphi}\left(\sum_{n=\tau}^{h-1}\gamma^{n-\tau}r_n + \gamma^{h-\tau}v_\psi(s_h)\right) \quad \text{with} \quad h = \min(\tau+k, t+H) \tag{7}$$

where the V_N^K estimates rewards beyond k steps with the learned value model.

The critic is updated to regress the targets around which we stop the gradient as typical (Srinivas et al. (2018)). The reliable weight is used in the critic updating, which is called weighted regression of target value. The objective for the critic $v_\psi(s_\tau)$, in turn, is to regress the target value

$$\min_\psi \mathrm{E}\left(\sum_{\tau=t}^{t+H}\frac{w(s_\tau)}{2}\|v_\psi(s_\tau) - \mathrm{V}_\lambda(s_\tau)\|^2\right) \tag{8}$$

Different from DDPG (Lillicrap et al. (2015)) and SAC (Haarnoja et al. (2018)) which only maximize immediate Q-values, we leverage gradients through transitions and by update the policy by backpropagation through the value model.

The object of the actor with policy $\pi(s;\varphi)$ is to maximize

$$\mathrm{J}_{actor} = \sum_{i=0}^{N}\left(\sum_{\tau=t}^{t+H} w(s_\tau^i)\mathrm{V}_\lambda(s_\tau^i)\right) \tag{9}$$

Since all steps are implemented as neural networks, we analytically compute $\nabla_\phi \mathrm{J}_{actor}$ by stochastic backpropagation. We use reparameterization for continuous actions and latent states and straight-through gradients for discrete actions. Note that prevous works like DrQ that just uses image-level augmented data as a regularizer for Q-learning, CLION directly learns the policy by back-propagating through both augmented trajectories and original trajectories.

3.3 THEORETICAL ANALYSES OF CLION FRAMEWORK

Based on Janner's work (Janner et al. (2019)), we hope to give the upper bound of the value function estimation error after introducing the latent data augmentation mechanism and provide theoretical inspiration for further study of model-based reinforcement learning with augmentation. The gap between true returns and model returns can be expressed in terms of two error quantities of the model: generalization error ϵ_m due to sampling, and distribution shift ϵ_π due to the updated policy encountering states not seen during model training. As the model is trained with supervised learning, sample error can be quantified by standard PAC generalization bounds, which bound the difference in expected loss and empirical loss by a constant with high probability. We denote this model generalization error as $\epsilon_m = \max_t E_{s\sim\pi_{D,t}}\left[D_{TV}\left(p\left(s', r \mid s, a\right) \| p_\theta\left(s', r \mid s, a\right)\right)\right]$ and the policy distribution shift between iterations $\max_s D_{TV}\left(\pi \| \pi_D\right) \leq \epsilon_\pi$ by the maximum total-variation distance. If we can instead approximate the model error on the distribution of the current policy π, which we denote as ϵ'_m, and approximate the ϵ'_m with a linear function of the policy divergence yields: $\hat{\epsilon}_{m'}\left(\epsilon_\pi\right) \approx \epsilon_m + \epsilon_\pi \frac{\mathrm{d}\epsilon_{m'}}{\mathrm{d}\epsilon_\pi}$, the upper bound of returns estimation error of k-branched model rollout is illustrated in Proposition 3.1.

Proposition 3.1. (Janner et al. (2019)) Under the k-branched rollout method, using model error under the updated policy $\epsilon_{m'} \geq \max_t E_{s\sim\pi_{D,t}}\left[D_{TV}\left(p\left(s' \mid s, a\right) || \hat{p}\left(s' \mid s, a\right)\right)\right]$, we have

$$\eta^{\text{branch}}\left[\pi\right] - \eta[\pi] \leq 2 r_{\max}\left[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k \epsilon_\pi}{(1-\gamma)} + \frac{k}{1-\gamma}\left(\epsilon_{m'}\right)\right] \tag{10}$$

where the $\eta[\pi]$ denotes the returns of the policy in the true MDP, $\eta^{branch}[\pi]$ denotes the returns of the policy by k steps rollout based on our model . Such a statement guarantees that, as long as we optimize such an upper bound, we can guarantee improvement on the true MDP.

Next, we analyze the CLION framework based on Proposition 3.1. The $\hat{p}\left(s' \mid s, a\right)$ in the CLION framework is equal to $w(s) p_\theta\left(s' \mid s, a\right)$. In the CLION framework the reliable weight $w(s)$ aims to approximate the important sampling factor between the data from real environment and the imagined data, and it is the most significant difference between CLION and Dreamer. We assumed that $w(s)$ is bounded by

$$\max_t \left\{\left| w(s) - \frac{p\left(s' \mid s, a\right)}{p_\theta\left(s' \mid s, a\right)}\right|\right\} \leq \epsilon_w \tag{11}$$

Thus, the total variance distance between the $p_\theta\left(s' \mid s, a\right)$ and $p\left(s' \mid s, a\right)$ can be derived as equation 17 (see Appendix A.1).

$$D_{TV}\left(p\left(s' \mid s, a\right) || \hat{p}\left(s' \mid s, a\right)\right) = \int_{(s,a,s')} \left| p\left(s' \mid s, a\right) - w(s) p_\theta\left(s' \mid s, a\right)\right| \leq \epsilon_w \tag{12}$$

Since the ϵ_m could be bounded by ϵ_w, the returns estimation error upper bound of CLION could be given by Theorem 3.1.

Theorem 3.1 . Under the CLION framework, if the model error under the updated policy is bounded by $\epsilon_{m'} \geq \max_t E_{s\sim\pi_{D,t}}\left[D_{TV}\left(p\left(s' \mid s, a\right) || \hat{p}\left(s' \mid s, a\right)\right)\right]$ and the reweighting coefficient is bounded by $\max_t\left\{\left|w(s) - p\left(s' \mid s, a\right) / p\left(s' \mid s, a\right)\right|\right\} \leq \epsilon_w$, the returns estimation error upper bound is

$$\eta^{\text{branch}}\left[\pi\right] - \eta[\pi] \leq 2 r_{\max}\left[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k \epsilon_\pi}{(1-\gamma)} + \frac{k}{1-\gamma}\left(\epsilon_w + \epsilon_\pi \frac{\mathrm{d}\epsilon_{w'}}{\mathrm{d}\epsilon_\pi}\right)\right] \tag{13}$$

where $\epsilon'_w = \epsilon_w + \epsilon_\pi \frac{\mathrm{d}\epsilon_{w'}}{\mathrm{d}\epsilon_\pi}$. More specifically, for sufficiently low ϵ'_w we have

$$k^* = \operatorname*{argmin}_k \left[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k \epsilon_\pi}{(1-\gamma)} + \frac{k}{1-\gamma}\left(\epsilon'_w\right)\right] > 0 \tag{14}$$

This insight suggests that when ϵ'_w is sufficiently low, the nonzero-length model rollout helps decrease the upper bound of returns estimation error. ϵ_w indicates how well does the evaluator could classify the real transition data and the generated transition data by model. The lower the ϵ_w the smaller the returns estimation error $\eta^{branch}[\pi] - \eta[\pi]$.

4 EXPERIMENTS

4.1 EFFECTIVENESS TESTING

The CLION framework is implemented as Algorithm 1 (see Appendix A.2.1). We use a single Nvidia K80 GPU for each training run. We use the same hyperparameters of Dreamer across all continuous control tasks. We compare with both the SOTA of model-based methods and model-free methods. Considering that the Dreamer algorithm provides a very mature model-based RL training framework, to eliminate the interference of other engineering factors, we mainly choose Dreamer as the model-based baseline to test the effectiveness of the CLION framework. As for model-free methods, Drq (Kostrikov et al. (2020)) uses data regularized Q-learning method to learn the policy and shows a significant advantage in data efficiency compared to CURL (Laskin et al. (2020b)). We choose the Drq (Kostrikov et al. (2020)) as the model-free baseline.

In an attempt to test the agent's ability to handle environmental uncertainty, the experiments are implemented in DMControl tasks (Tassa et al. (2018)) with observation uncertainty by adding a rotation $(-5°, +5°)$ to the input from the Mujoco environment, and test the performance of CLION together with the Dreamer and the Drq. The effectiveness of the CLION framework to tackle the input uncertainty is demonstrated as Figure 5 and Table 1.

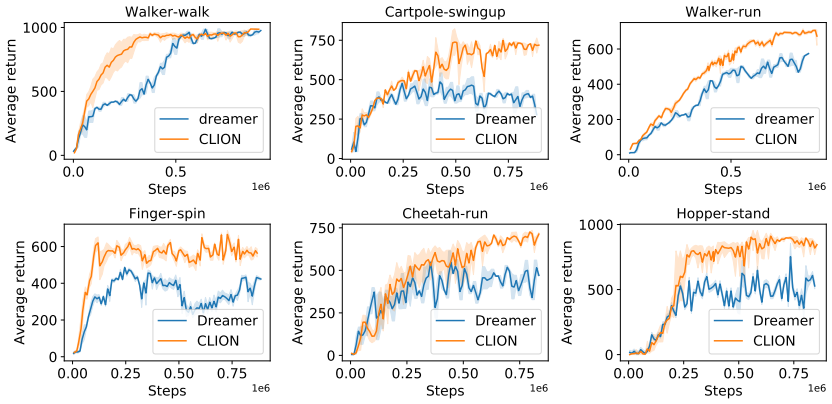


Figure 5: CLION exceeds at visual control tasks that testing with environmental uncertainty.

We also test the robustness of Drq. Since Drq's input is a stack of three sequential images (3 continuous images), we implement 2 different experiments: 1) random rotate the 3 images by the same angle and 2) randomly rotate the 3 images by different angles. All the rotate angles are sampled from (-5,5) degree. The results show that under the first type of uncertainty Drq almost failed, and under the second type of uncertainty,as illustrated in , Drq's performance decreased significantly as shown in Figure 6.

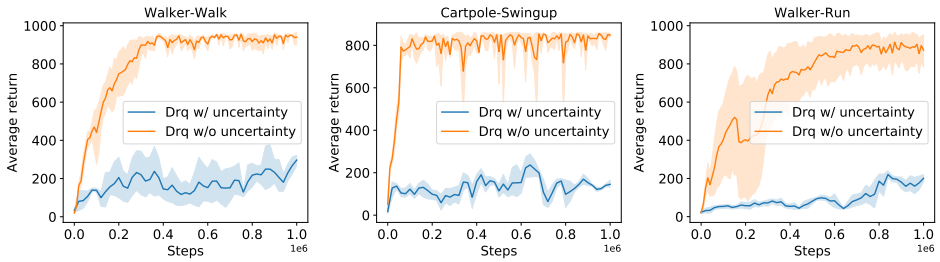


Figure 6: Drq's performance decreased due to the observation uncertainty

In all tasks for CLION and Dreamer, the only observations are third-person camera images of size $(64, 64, 3)$ pixels. All the hyperparameters are shown in Table 3 (see Appendix A.2.2).

Table 1: Performance comparison with image uncertainty

	CLION		Dreamer	
	105K	505K	105K	505K
Walker-Walk	**527.1** $\pm$ 166	**914.9** $\pm$ 61	372.5 $\pm$ 19	876.3 $\pm$ 48
Walker-Run	**214.6** $\pm$ 72	**529.3** $\pm$ 107	116.1 $\pm$ 11	464.2 $\pm$ 65
Hopper-Stand	**92.3** $\pm$ 166	**752.7** $\pm$ 150	85.6 $\pm$ 52	389.7 $\pm$ 162
Cartpole-Swingup	**384.5** $\pm$ 58	**736.0** $\pm$ 35	303.3 $\pm$ 43	423.2 $\pm$ 66
Cheetah-Run	165.3 $\pm$ 24	**631.5** $\pm$ 42	**245.9** $\pm$ 132	526.3 $\pm$ 241
Finger-Spin	**522.5** $\pm$ 133	**685.7** $\pm$ 61	283.5 $\pm$ 138	374.9 $\pm$ 178

Table 2: Performance comparison without external image uncertainty

500k step scores	Drq	CLION	Dreamer
Walker Walk	921 $\pm$ 45	**959** $\pm$ 22	897 $\pm$ 49
Walker Run	459 $\pm$ 139	**536** $\pm$ 25	482 $\pm$ 68
Walker stand	942 $\pm$ 22	**979** $\pm$ 28	966 $\pm$ 35
Pendulum Swingup	548 $\pm$ 287	**724** $\pm$ 104	682 $\pm$ 203
Cheetah Run	**660** $\pm$ 96	631 $\pm$ 103	570 $\pm$ 253
100k step scores			
Walker Walk	612 $\pm$ 164	**630** $\pm$ 113	277 $\pm$ 12
Walker Run	185 $\pm$ 148	**211** $\pm$ 162	161 $\pm$ 112
Walker stand	**843** $\pm$ 251	728 $\pm$ 46	494 $\pm$ 128
Pendulum Swingup	86 $\pm$ 83	**182** $\pm$ 128	43 $\pm$ 34
Cheetah Run	**344** $\pm$ 67	328 $\pm$ 48	235 $\pm$ 137

In order to show the data efficiency of CLION, compared to Dreamer and Drq (Kostrikov et al. (2020), we test the performance of CLION in DMControl environments without image uncertainty with 10 random seeds. As illustrated in Table 2, CLION still outperforms Drq and Dreamer in these tasks.

Overall, the experimental results suggest that CLION outperforms Dreamer and Drq in terms of data-efficiency, robustness to uncertainty, and final performance, which proves that our proposed memory augmentation method can facilitate the agent to learn better behavior with limited experiences.

4.2 Ablation Experiments

We implemented three groups of ablation experiments which are tested in 3 DMContol tasks with observation uncertainty: a) two constraints for RSSM with latent trajectory augmentation only. As

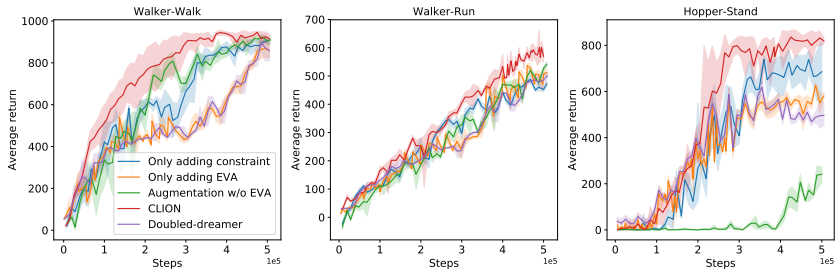


Figure 7: Contribution of each component in CLION to the performance improvement

illustrated in Figure 7, the CLION performs better than Dreamer but are insufficient compared with CLION's advantages. For instance, at the 100k of Walker-walk, its performance is 26.6% better than Dreamer, but CLION is 41.6% better than Dreamer; b) trajectory evaluator only. The experimental results show that its performance is basically similar to Dreamer; c) memory augmentation with the constraints for RSSM but without the trajectory evaluator. Although it performs better than the Dreamer in Walker-Walk, it has lower performance in Hopper-Stand due to lack of reliable guarantee for the augmented data. We also compared CLION with the method that utilizes two times

batch size in Dreamer (Doubled-Dreamer), as shown in Figure 7, the ClION also outperforms the Doubled-Dreamer. Overall, the latent trajectory augmentation with constraints, and the trajectory evaluator are critical components for CLION, and they complement each other.

5 RELATED WORKS

Model-based policy optimization: Prior works can be divided into the following categories, including Dyna-like algorithms (Sutton et al. (2012)), value expansion, adaptive programming (Bertsekas et al. (1995)), and sampling-based planning. Dyna-like algorithms alternate between model learning from environmental interaction, data generation, and policy improvement by model-free methods, e.g., ME-TRPO (Kurutach et al. (2018)). Value expansion algorithm utilizes the dynamic model to improve the estimation accuracy of the cumulative return (Feinberg et al. (2018)). The adaptive dynamic programming is built upon the use of the analytic gradient of state value backpropagation through the dynamic transition. Typical algorithms in this category include Dreamer (Hafner et al. (2019a)), PILCO (Deisenroth & Rasmussen (2011)), and SVG (Heess et al. (2015)). The idea of sampling-based planning is to choose the best action by a large number of samples, and regard it as the objective for policy network. Representative algorithms include the cross-entropy method (Bekker (2012)) in continuous space, which is used in PlaNet (Hafner et al. (2018)) and PETS (Chua et al. (2018)), and MCTS (Silver et al. (2017)) in discrete space. To our best knowledge, the Dreamer algorithm (Hafner et al. (2019a)) is the most successful model-based algorithm for image tasks.

Latent world model: In MBRL for high dimension input tasks, the latent world model offers a flexible way to represent key information of the observations with lower dimension and more accurate forward prediction. World Models (Ha and Schmidhuber, 2018) learn latent dynamics in a two-stage process: representation learning and latent dynamic learning. However, it lacks the coordination of the two processes. PlaNet (Hafner et al. (2019b)) proposes a recurrent stochastic state model (RSSM), which learns them jointly and learn the optimal policy by latent online planning. Dreamer utilizes the RSSM to learn behavior by long-term imagination. Dreaming (Okada & Taniguchi (2020)) develops a non-reconstruction based latent model to solve target vanishing problems. To our best knowledge, the latent recurrent stochastic state model (RSSM) used in Planet and Dreamer is the most popular method to represent the image state dynamics (Hafner et al. (2019b)).

Data augmentation for reinforcement learning: Data augmentation has been investigated in the context of RL to improve generalization and data efficiency. There are many prior works that suggest the potential conclusion that data augmentation is helpful for data-efficient reinforcement learning from image. CURL (Laskin et al. (2020b)) utilizes data augmentations for learning representations in the RL setting which minimize the contrastive loss between an image and its augmented version using the MoCo (He et al. (2020)) mechanism. RAD (Laskin et al. (2020a)) attempts to directly train the RL objective on multiple augmented views of the observations without any auxiliary loss, thereby ensuring consistencies on the augmented views implicitly. Both CURL and RAD aim to get the representation of the input image to avoid overfitting. Cocurrent but independent to them, Drq (Kostrikov et al. (2020)) aims to solve the distribution mismatch problem in off-policy RL, which utilized random cropping and regularized Q-functions in conjunction.

6 CONCLUSION

We present a robust memory augmentation method with constrained latent imagination (CLION), which aims to improve the policy robustness by effectively enlarge the model's latent space with the enhancement of agent's memory diversity. For this, we propose a constrained RSSM framework to embed the experiential knowledge into the latent world model with memory robustness. CLION optimizes a parametric policy with the augmented trajectories by propagating analytic gradients of multi-step values back through learned latent dynamics. CLION exceeds previous methods in data-efficiency, robustness to uncertainty, and final performance on a variety of challenging continuous control tasks with image inputs. The combination with advanced contrastive representation learning proposed by RAD, CURL, and dreaming for more extensive applications will be investigated in future.

REFERENCES

James Bekker. *Applying the cross-entropy method in multi-objective optimisation of dynamic stochastic systems*. PhD thesis, Stellenbosch: Stellenbosch University, 2012.

Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 4759–4770, 2018.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

Brigitta Gundersen. Forming artificial memories during sleep. *Nature neuroscience*, 18(4):483–483, 2015.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565, 2019b.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pp. 4565–4573, 2016.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pp. 12519–12530, 2019.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020a.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2003.06417*, 2020b.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Masashi Okada and Tadahiro Taniguchi. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction. *arXiv preprint arXiv:2007.14535*, 2020.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks. *arXiv preprint arXiv:1804.00645*, 2018.

Richard S Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael P Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. *arXiv preprint arXiv:1206.3285*, 2012.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

A Appendix

A.1 Proof for the returns estimation error upper bound

Lemma A.1.1 (Returns bound, branched rollout (Janner et al. (2019))). Assume we run a branched rollout of length Before the branch ("pre" branch),w e assume that the dynamics distributions are bounded as $\max_t E_{s\sim p_1^t(s)} D_{KL}\left(p_1^{\text{pre}}\left(s', a \mid s\right) \| p_2^{\text{pre}}\left(s', a \mid s\right)\right) \leq \epsilon_m^{\text{pre}}$ and after the branch as $\max_t E_{s\sim p_1^t(s)} D_{KL}\left(p_1^{\text{post}}\left(s', a \mid s\right) \| p_2^{\text{post}}\left(s', a \mid s\right)\right) \leq \epsilon_m^{\text{post}}$. Likewise, the policy divergence is bounded pre- and post- branch by $\epsilon_\pi^{\text{pre}}$ and $\epsilon_\pi^{\text{post}}$, repsectively. Then the K-step returns are bounded as:

$$|\eta_1 - \eta_2| \leq 2r_{\max}\left[\frac{\gamma^{k+1}}{(1-\gamma)^2}\left(\epsilon_m^{\text{pre}} + \epsilon_\pi^{\text{pre}}\right) + \frac{k}{1-\gamma}\left(\epsilon_m^{\text{post}} + \epsilon_\pi^{\text{post}}\right) + \frac{\gamma^k}{1-\gamma}\epsilon_\pi^{\text{pre}} + \frac{1}{1-\gamma}\epsilon_\pi^{\text{post}}\right] \tag{15}$$

Next we prove the Theorem 3.1 metioned in Section 4.3 by using Lemma A.1.1. Assume that the reliable weight $w(s)$ is bounded by

$$\max_t \left\{\left| w(s) - \frac{p\left(s' \mid s, a\right)}{p_\theta\left(s' \mid s, a\right)}\right|\right\} \leq \epsilon_w \tag{16}$$

Thus, the total variance distance between the $p_\theta\left(s' \mid s, a\right)$ and $p\left(s' \mid s, a\right)$ can be derived as

$$D_{TV}\left(p\left(s' \mid s, a\right) || \hat{p}\left(s' \mid s, a\right)\right) = \int_{(s,a,s')} \left|p\left(s' \mid s, a\right) - w(s) p_\theta\left(s' \mid s, a\right)\right| \tag{17}$$

when $p\left(s^{\prime} \mid s, a\right)-w(s) p_{\theta}\left(s^{\prime} \mid s, a\right)>0$, we have

$$\begin{aligned}
&\int_{(s, a, s^{\prime})} p\left(s^{\prime} \mid s, a\right)-w(s) p_{\theta}\left(s^{\prime} \mid s, a\right) \\
=&\int_{(s, a, s^{\prime})} p\left(s^{\prime} \mid s, a\right)-w(s) p_{\theta}\left(s^{\prime} \mid s, a\right) \\
\leq&\int_{(s, a, s^{\prime})} p\left(s^{\prime} \mid s, a\right)-\left(\frac{p\left(s^{\prime} \mid s, a\right)}{p_{\theta}\left(s^{\prime} \mid s, a\right)}-\epsilon_{w}\right) p_{\theta}\left(s^{\prime} \mid s, a\right) \\
=&\int_{(s, a, s^{\prime})} \epsilon_{w} p_{\theta}\left(s^{\prime} \mid s, a\right)=\epsilon_{w}
\end{aligned} \tag{18}$$

when $p\left(s^{\prime} \mid s, a\right)-w(s) p_{\theta}\left(s^{\prime} \mid s, a\right) \leq 0$, we have

$$\begin{aligned}
&\int_{(s, a, s^{\prime})} p\left(s^{\prime} \mid s, a\right)-w(s) p_{\theta}\left(s^{\prime} \mid s, a\right) \\
=&\int_{(s, a, s^{\prime})} w(s) p_{\theta}\left(s^{\prime} \mid s, a\right)-p\left(s^{\prime} \mid s, a\right) \\
\leq&\int_{(s, a, s^{\prime})}\left(\epsilon_{w}+\frac{p\left(s^{\prime} \mid s, a\right)}{p\left(s^{\prime} \mid s, a\right)}\right) p\left(s^{\prime} \mid s, a\right)-p\left(s^{\prime} \mid s, a\right)=\epsilon_{w}
\end{aligned} \tag{19}$$

Therefore the ϵ_m could be bounded by ϵ_w. Since the ϵ_m could be bounded by ϵ_w.

The choice of reference quantity is a branched rollout which executes the old policy π_D under the true dynamics until the branch point, then executes the new policy π under the true dynamics for k steps. We denote the returns under this scheme as $\eta^{\pi_D, \pi}$. We can split the returns as follows:

$$\eta[\pi]-\eta^{\text{branch}}=\underbrace{\eta[\pi]-\eta^{\pi D, \pi}}_{L_1}+\underbrace{\eta^{\pi_D, \pi}-\eta^{\text{branch}}}_{L_2} \tag{20}$$

We can bound both terms L_1 and L_2 using Lemma A.1.1. L_1 accounts for the error from executing the old policy instead of the current policy. This term only suffers from error before the branch begins, and we can use Lemma A.1.1 $\epsilon_{\pi}^{\text{pre}} \leq \epsilon_{\pi}$ and all other errors set to 0. This implies:

$$\left|\eta[\pi]-\eta^{\pi_D, \pi}\right| \leq 2 r_{\max }\left[\frac{\gamma^{k+1}}{(1-\gamma)^{2}} \epsilon_{\pi}+\frac{\gamma^{k}}{1-\gamma} \epsilon_{\pi}\right] \tag{21}$$

L_2 incorporates model error under the new policy incurred after the branch. Again we use Lemma A.1.1, setting $\epsilon_{m}^{\text{post}} \leq \epsilon_{m}$ and all other errors set to 0. This implies:

$$\left|\eta[\pi]-\eta^{\pi_D, \pi}\right| \leq 2 r_{\max }\left[\frac{k}{1-\gamma} \epsilon_{m^{\prime}}\right] \leq 2 r_{\max }\left[\frac{k}{1-\gamma} \epsilon_{w^{\prime}}\right] \tag{22}$$

Adding L_1 and L_2 together, the returns estimation error upper bound can be derived as

$$\eta[\pi] \geq \eta^{\text{branch}}[\pi]-2 r_{\max }\left[\frac{\gamma^{k+1} \epsilon_{\pi}}{(1-\gamma)^{2}}+\frac{\gamma^{k} \epsilon_{\pi}}{(1-\gamma)}+\frac{k}{1-\gamma}\left(\epsilon_{w}+\epsilon_{\pi} \frac{\mathrm{d} \epsilon_{w^{\prime}}}{\mathrm{d} \epsilon_{\pi}}\right)\right] \tag{23}$$

where $\epsilon_{w}^{\prime}=\epsilon_{w}+\epsilon_{\pi} \frac{\mathrm{d} \epsilon_{w^{\prime}}}{\mathrm{d} \epsilon_{\pi}}$.

A.2 Algorithm flowchart and HYPER PARAMETERS

A.2.1 Algorithm flowchart

The CLION framework is implemented as Algorithm 1, which mainly contains 4 key processes: (1) learning the constrained recurrent stochastic state model, (2) memory augmentation and latent imagination, (3) behavior learning, and (4) environment interaction.

Algorithm 1: CLION (Robust memory augmentation by constrained latent imagination)

Initialize dataset D with S random seed episodes.
Initialize neural network parameters θ, φ and ψ randomly.
while *not convergence* **do**
  for *update step c = 1..C* **do**
    // Learning the constrained RSSM
    Draw B data sequences $\{(a_t, o_t, r_t)\}_{t=k}^{k+L} \sim \mathcal{D}$;
    Compute model state $s_t \sim p_\theta\left(s_t \mid s_{t-1}, a_{t-1}, o_t\right)$;
    Update θ by $\min_\theta \{J_M + \beta_1 ||e'_t - e'_t||_2 + \beta_2 ||\tilde{o}_t - o_t||_2\}$
    // Memory augmentation and latent imagination
    Sample start point of the augment trajectories
     $h_\tau^{Aug} \sim N(h_\tau, \sigma_h^2), \quad s_\tau^{Aug} \sim N(m_\tau, \sigma_\tau^2)$
    Imagine trajectories on both original and augmented data $\{(s_\tau, a_\tau)\}_{\tau=t}^{t+H}$ from each s_t
    Compute the reweighting score $W(s_\tau) = \mathrm{EVA}(s_\tau, a_\tau, s_{\tau+1})$ for every imagined trajectory and normalize the score in the mini batch
    Predict rewards $\mathrm{E}\left(q_\theta\left(r_\tau \mid s_\tau\right)\right)$ and values $v_\psi\left(s_\tau\right)$
    Compute value estimates $\mathrm{V}_\lambda\left(s_\tau\right)$
    // Behavior learning
    Update $\phi \leftarrow \phi + \alpha \nabla_\phi \sum_{\tau=t}^{t+H} w(s_\tau) V_\lambda\left(s_\tau\right)$
    Update $\psi \leftarrow \psi - \alpha \nabla_\psi \sum_{\tau=t}^{t+H} \frac{w(s_\tau)}{2} \left\| v_\psi\left(s_\tau\right) - \mathrm{V}_\lambda\left(s_\tau\right) \right\|^2$
  / / Environment interaction
  $o_1 \leftarrow$ env.reset ()
  for *time step* $t = 1..T$ **do**
    Compute $s_t \sim p_\theta\left(s_t \mid s_{t-1}, a_{t-1}, o_t\right)$ from history.
    Compute $a_t \sim q_\phi\left(a_t \mid s_t\right)$ with the action model.
    Add exploration noise to action.
    $r_t, o_{t+1} \leftarrow$ env.step $\left(a_t\right)$
  Add experience to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \left\{(o_t, a_t, r_t)_{t=1}^{T}\right\}$.

Table 3: Hyper parameters table

Name	Symbol	Value
Collect interval	C	100
Interact interval	T	1000
Batch size	B	50
Sequence length	L	15
Imagination horizon	H	50
Learning rate for model	α_0	$6\mathrm{e}-4$
Learning rate for critic	α_1	$8\mathrm{e}-5$
Learning rate for actor	α_2	$8\mathrm{e}-5$
Penalty coefficient	β_1	0.01
Penalty coefficient	β_2	0.5

A.2.2 HYPER PARAMETERS

The main parameters of the CLION algorithm are listed in Table 3. Here, we discuss some tricks used in the implement. We scale down gradient norms that exceed 100 and clip the KL-divergence in equation 1 below 3 free nats as in Dreamer and PlaNet. We compute the v_λ targets with $\gamma = 0.99$ and $\lambda = 0.95$. The dataset is initialized with $S = 5$ episodes collected using random actions. Similar to Dreamer, we fix the action repeat to 2 for all environments.