Learning State Representations via Temporal Cycle-Consistency Constraint in MBRL

Changmin Yu[1][*], **Dong Li**[2], **Hangyu Mao**[2], **Jianye Hao**[2], **Neil Burgess**[1]
[1]UCL, London, United Kingdom
[2]Huawei Noah's Ark Lab, Beijing, China
{changmin.yu.19; n.burgess}@ucl.ac.uk;
{lidong106; maohangyu; haojianye}@huawei.com

Abstract

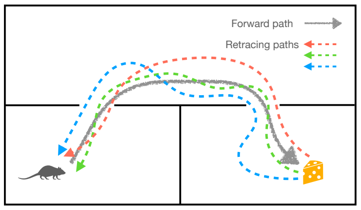
Representation learning is a popular approach for reinforcement learning (RL) tasks with partially observable Markov decision processes. Existing works on learning representations utilise the dynamics models in model-based RL to perform training through model predictive reconstruction in a temporally forward fashion. However, temporally backward state predictions also yield useful supervision signals as they convey information about the future states given the action choices. We argue that combining them with forward passes will facilitate stronger representation learning and improve the sample efficiency of RL. Here we propose a general framework for learning state representations for RL tasks, utilising both forward and backward passes by imposing temporal cycle-consistency constraints, which can be integrated with any model-based RL algorithms leveraging a latent dynamics model. We show improved empirical performance in terms of sample-efficiency and convergence score over several baselines on continuous control benchmarks.

1 Introduction

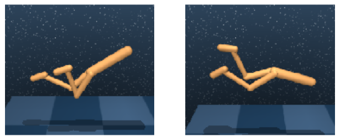
Reinforcement learning (RL) in Partially Observable Markov Decision Processes (POMDPs) is usually a non-trivial task due to the curse of dimensionality and curse of history (Kaelbling et al., 1998). Representation learning is a commonly adopted approach for extracting abstract features that compactly characterise the high-dimensional raw inputs, such that RL can be performed more efficiently. Both model-free and mode-based RL methods have been studied extensively for solving the POMDPs. Here we focus on learning representations in model-based RL.

In model-based RL, an environment model is typically approximated and combined with planning methods (Chua et al., 2018) or learning methods (Hafner et al., 2020) in the latent space to facilitate behaviour learning. In their seminal work, Ha & Schmidhuber (2018) propose to learn world models given trained encoders. The world model can be used either for learning behaviours or for real-time planning. More recently, a number of works propose joint training of the embedding function and the latent dynamics model (Srinivas et al., 2018; Hafner et al., 2019; Schrittwieser et al., 2020b; Lee et al., 2020a). Such model learning is performed by minimising the deviation of the imaginary predictive states from the real observations.

However, we argue that the aforementioned model learning is inefficient in terms of the sample usage. More meaningful



(a)



(b)

Figure 1: **Demonstration of learning representation via retracing.** (a): Retracing in navigation tasks yields detection of key states (doors); (b): Retracing in continuous control tasks helps with identifying catastrophic states and behaviour (graphical demonstration from the DeepMind Control Suite (Tassa et al., 2018)).

[*]Please send any enquiries to: changmin.yu.19@ucl.ac.uk

supervision can be obtained by employing the cycle-consistency constraint between the imaginary state predictions in forward and backward temporal directions. The key intuition is that if the model can translate the agent from state s_0 to s_T by following a trajectory, and the model also has the capability to bring the agent back from s_T to s_0' along a similar trajectory, where s_0 and s_0' are expected to be similar. Based on this intuition, backward imaginary predictions can be utilised to facilitate model learning in a self-supervised manner. As a motivating example, consider a rat trying to navigate towards a goal location represented by a cheese in a cluttered room shown in Figure 1a. By imposing the temporal cycle-consistency constraint, multiple backwards imaginative paths can be simulated. In order to retrace to the starting position, the rat needs to pass through two "doors" in all paths. This allows the rat to identify key states (the doorways) that are essential for successful navigation towards the goal.

Moreover, backward rollouts can pass information about the future states back to previous states. This approximately resembles the smoothing operation in state-space models (Kalman, 1960; Sahani, 1999), which potentially supports more accurate inference over latent states.

We also expect that learning representation by retracing will improve the agent's ability of avoiding catastrophic actions in certain situations. Consider a toy example of continuous control in Figure 1b, where no action will take the falling humanoid (right) back to its previous state (left), yielding large discrepancy between the true and retraced states. The error will then be propagated back to inform the agent to avoid the behaviours that lead to irrecoverable states.

Joining forward and backward inferences in the latent space is also supported by evidence from the neuroscience community. Hippocampal "replay" has been suggested to play an essential role in decision-making and model-based inferences (Mattar & Daw, 2018). Connections between the latent state inference and forward and reversed hippocampal "replay" have been proposed and substantiated by experimental evidence (Penny et al., 2013). Following this motivation, we thus expect the retrace operation to play an essential role in learning representations for POMDP tasks.

The main contributions are summarised as follows.

- We introduce a self-supervised representation learning algorithm via imposing the cycle-consistency constraint to facilitate model learning. To the best of our knowledge, this is the first work that introduce cycle-consistency to learning a world model for RL.
- We instantiate our method utilising the Recurrent State-Space Model (RSSM (Hafner et al., 2019)) as the world model, resulting in a new model-based RL agent termed as *Cycle-Consistent World Model (CCWM)*.
- We empirically show that *CCWM* outperforms the baseline algorithms, and is able to reach the asymptotic performance of state-based model-free methods on a number of challenging continuous control benchmarks.

2 RELATED WORKS

Representation learning in RL. Finding useful state representations that could aid reinforcement learning has long been studied, from the introduction of manually constructed features such as tile coding and Fourier basis in combination (Sutton & Barto, 2018), to the learned embedding vectors using trained neural networks (Mnih et al., 2013; Schrittwieser et al., 2020a). Even under partial observability, we can train an RL agent in the embedding space using a neural network encoder without the requirements of an explicit model of the POMDP (Kaelbling et al., 1998). However, it has been noted that naive joint training of RL with representation learning can be inefficient due to the sparse and delayed reward structure of the MDP (Shelhamer et al., 2017). Many auxiliary tasks, e.g. weakly-supervised classification and location recognition, have been proposed for resolving such issue (Lee et al., 2020b; Mirowski et al., 2017; Oord et al., 2018). In the current work, we propose an additional cycle-consistency constrained objective for learning the representations.

There exists two commonly adopted sources of supervisions for learning representations in RL tasks, the reconstruction of observations (Gelada et al., 2019; Hafner et al., 2020; Lee et al., 2020a), and self-supervised signals such as contrastive losses (Oord et al., 2018). Recently, Zhang et al. (2020) and Agarwal et al. (2021) proposed to learn the embeddings using bisimulation metrics (Ferns et al., 2011), which yields representations that are agnostic to task-irrelevant information, supporting faster learning and stronger generalisation. Here we adopt a combination of the reconstruction and bisimulation metrics supervision signals for model training.

Learning the dynamics model. In addition to learning state abstractions based solely on the observations, it is also possible to learn the latent representation along with the dynamics model using latent state-space models (LSSM (Murphy, 2012)). Ha & Schmidhuber (2018) introduced the world model, for modelling the dynamics of the latent space based on the pre-trained state embeddings. The world model leads to improved performance in POMDP tasks such as Atari games, as well as generates reasonable predictions in the observable space through rollouts in the latent space. The state representation can also be learnt jointly with the latent dynamics model. A number of works proposed to treat the embedding function as part of the recognition model in a probabilistic graphical model (Buesing et al., 2018; Lee et al., 2020a; Hafner et al., 2020). These methods treat the learning of representations and latent dynamics model as a joint task, which can be trained via doing predictive inference over the latent states given the observations and actions. The learned embeddings can then be used as the inputs to the value function approximator in the downstream RL component for learning the optimal policy. The learned latent dynamics model can be used for generating imaginary trajectories through latent model-based rollouts for RL training, which improves the sample-efficiency of learning. Our proposed approach is similar to this line of work since we also leverage latent stochastic sequential modes for joint learning of the representations and the latent dynamics model, and our method can be efficiently combined with downstream RL algorithm to learn the optimal policy using model-based rollouts following a similar paradigm. However, we note a major difference between our method and the existing approaches: all related works use predictive supervision in a temporally forward fashion, whereas our method combines predictive supervision in both the temporally forward and backward directions by imposing a temporal cycle-consistency constraint.

Cycle-Consistency. Cycle-consistency is a commonly adopted approach in computer vision and natural language processing (Zhou et al., 2016; Zhu et al., 2017; Yang et al., 2017; Dwibedi et al., 2019), where the fundamental idea is the validation of matches between cycling through multiple samples. Here we adopt similar design principles for sequential decision-making tasks: rollouts in a temporally forward direction alone yield under-constrained learning of the dynamics model of the task. We propose to additionally incorporate backwards rollouts to perform model learning in a self-supervised fashion. We do this by imposing the *temporal cycle-consistency* constraint (Dwibedi et al., 2019): we introduce an auxiliary objective based on the bisimulation metric that ensures the forward and backward temporal dynamics yield the same MDP structure.

3 Learning Representation via Retracing

We consider learning in a POMDP, which is represented by the tuple $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, r, T, E, \gamma \rangle$, where $\mathcal{S}, \mathcal{O}, \mathcal{A}$ are the state space, the observation space and the action space, respectively. $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition distribution between states, $E : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the conditional observation distribution, and $\gamma \in (0, 1]$ is the discounting factor. At each time step t, the agent receives an observation $O_t \in \mathcal{O}$, and chooses its action according to its policy $a_t \sim \pi(a|O_t)$ to interact with the environment. The goal is to learn the optimal policy that maximises the expected return over all states defined as $\mathbb{E}_\pi \left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau | \, s\right]$.

In the current work, instead of working directly with $\mathcal{O}$, we wish to learn an embedding function $q_\psi : \mathcal{O} \to Z$, along with a latent dynamics model, $q_\psi : Z \times \mathcal{A} \to Z$, such that model-based RL can be performed efficiently in the learned latent space.

3.1 Learning Representations via Forward Passes

Existing works on learning the state representation of a latent dynamics model rely on proposing a stochastic sequential model as the world model, which is trained by performing latent forward rollouts such that the future environmental attributes such as observations or rewards are accurately predicted (Lee et al., 2020a; Hafner et al., 2020). Specifically, we are interested in model-based RL algorithms with a dynamics model defined in terms of the following components:

$$\begin{aligned} \textit{embedding function} :& q_\psi(O_t), \\ \textit{posterior distribution} :& p(z_{t+1}|z_t, a_t, O_{t+1}), \\ \textit{variational posterior distribution} :& q_\psi(z_{t+1}|z_t, a_t, O_{t+1}), \\ \textit{variational transition distribution} :& q_\psi(z_{t+1}|z_t, a_t), \\ \textit{generative distribution} :& p_\theta(O_{t+1}|z_{t+1}), \end{aligned} \tag{1}$$

where ψ and θ represent the parameters associated with the recognition and generative models, respectively. Note that we use a variational approximation to the posterior distribution since the true posterior $p(z_{t+1}|z_t, a_t, O_{t+1})$ is usually intractable in practice.

At each time step t, the agent receives an observation O_t, which is then embedded into a latent feature vector $z_t = q_\psi(O_t)$. The feature vector is then rolled out in a forward fashion given the action a_t, yielding one-step prediction into the future as $\hat{z}_{t+1} \sim q_\psi(z|z_{\leq t}, a_{\leq t})$ following the variational transition distribution $q_\psi(z'|z, a)$. The parameters of the recognition and generative models shown in Eq. 1 are jointly learned by maximising the variational lower bound (ELBO, (Wainwright & Jordan, 2008)), which takes the general form as following:

$$\begin{aligned} \mathcal{L}_{\text{ELBO}} =& \mathbb{E}_{q_\psi}[\log p_\theta(O_{t+1}|\hat{z}_{t+1}) - \\ & \beta \mathcal{D}_{\text{KL}}[q_\psi(z_{t+1}|z_t, a_t, O_{t+1}) || q_\psi(z_{t+1}|z_t, a_t)]], \end{aligned} \tag{2}$$

where the expectation is taken with respect to the variational posterior distribution, $q_\psi(z_{t+1}|z_t, a_t, O_{t+1})$. Note that it is also possible to generate predictions over multiple steps into the future instead of one for model learning, see, e.g. Lee et al. (2020a).

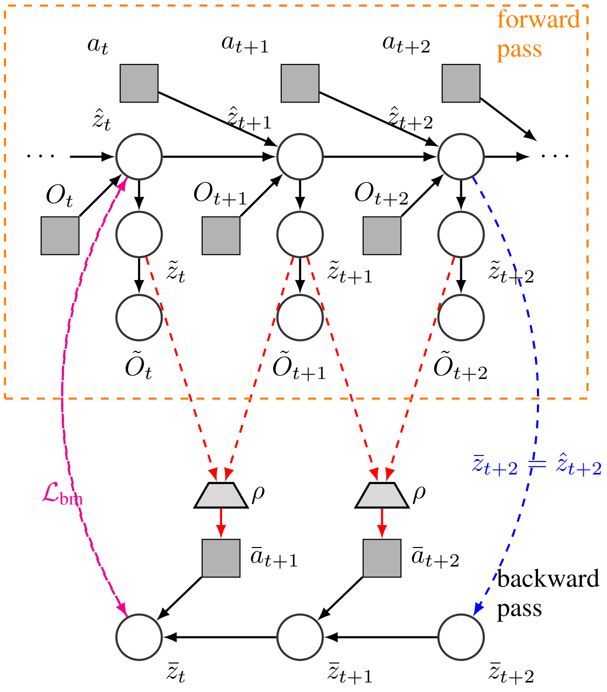


Figure 2: **Probabilistic graphical models of the model learning process in *CCWM*.** The forward pass (bounded by the orange rectangle) and retrace phase for learning the representations and latent dynamics model via latent imaginations are shown. Please refer to Section 3.2 for detailed description.

3.2 Learning Representations via Retracing

The backward rollouts convey information of future states given the actions taken back to the previous states, similar to the smoothing operation in state-space models (Kalman, 1960). This potentially supports more accurate inference of the latent states and stronger representation learning. Moreover, the backward rollouts provide additional supervision for training the dynamics model, which are expected to improve the sample efficiency of learning. Here we introduce a general framework for learning representations for the latent dynamics model which combines both forward and backward passes through latent imagination while constraining the temporal cycle-consistency between the forward and backward passes. We refer to the proposed approach as *Cycle-Consistent World Model* (*CCWM*). The pseudocode for *CCWM* is shown in Algorithm 1.

Consider the same dynamics model described in Section 3.1, we additionally define a reverse action approximator, $\rho : Z \times Z \to \mathcal{A}$, which takes in a tuple of latent states (z_1, z_2) and outputs an action $\bar{a}$ that approximates the action that leads the transition from z_2 to z_1. We assume the parameters of ρ are learned jointly with the model learning in an end-to-end fashion.

Suppose the imagination horizon for predictive inference is k. For $l = 1, \ldots, L - k$, the prior estimates for the k-step predictions and posterior estimates given the ground-truth observations are defined similarly as in the one-step case (corresponding to line 7 of Algorithm 1 and the forward dynamics chain in Figure 2).

$$
\begin{aligned}
\hat{z}_{l+1:l+k} &\sim q_\psi(z_{l+1:l+k}|z_l, a_{l:l+k-1}) = \prod_{j=1}^{k} q_\psi(z_{l+j}|\hat{z}_{l+j-1}, a_{l+j-1}), \\
\tilde{z}_{l+1:l+k} &\sim q_\psi(z_{l+1:l+k}|z_l, a_{l:l+k-1}, O_{l+1:l+k}) = \prod_{j=1}^{k} q_\psi(z_{l+j}|\hat{z}_{l+j-1}, a_{l+j-1}, O_{l+j}),
\end{aligned}
\tag{3}
$$

where the initial condition is taken to be $\hat{z}_l = z_l$.

The retracing steps are similar to the one-step case, which can be computed via backwards imagination using the same latent dynamics model given the predicted prior estimates from the forward passes.

$$
\bar{z}_{l+k-1:l} \sim q_\psi(z_{l+k-1:l}|\hat{z}_{l+k}, \tilde{z}_{l:l+k}, \rho(\cdot,\cdot)) = \prod_{j=1}^{k} q_\psi(z_{l+k-j}|\bar{z}_{l+k-j+1}, \rho(\tilde{z}_{l+k-j}, \tilde{z}_{l+k-j+1})),
\tag{4}
$$

for $l = 1, \dots, L-k$, where the initial retraced state is taken to be $\bar{z}_{l+k} = \hat{z}_{l+k}$. The retracing computation corresponds to Line 8 in Algorithm 1 and is demonstrated graphically in the reversed dynamics chain in Figure 2.

Model learning supervised by cycle-consistency. We learn the representation with the auxiliary task for satisfying the temporal cycle-consistency of the dynamics models, i.e., we wish to impose a constraint on the similarity between the original (forward-predicted) states and the retraced states in addition to the ELBO loss (Eq. 2). We utilise the bisimulation metrics as the loss function for the auxiliary retrace consistency task (Zhang et al., 2020).

$$
\begin{aligned}
\mathcal{L}_{\text{bm}}(z, z') = &[||z - z'||_1 - \mathcal{D}_{\text{KL}}[\hat{R}(\cdot|z)||\hat{R}(\cdot|z')] - \\
&\gamma W_2(\hat{P}(\cdot|z, \hat{\pi}(z)), \hat{P}(\cdot|z', \hat{\pi}(z')))]^2,
\end{aligned}
\tag{5}
$$

where $\hat{R}(r|z)$ represents the approximated reward model; $\hat{P}(z'|z, \hat{\pi}(z))$ denotes the distribution over the predicted state z', parameterised by the latent transition dynamics model, given the current state z and a sampled action (chosen given the current action distribution, $\hat{\pi}(a|z)$); and $W_2(\cdot,\cdot)$ represents the 2-Wasserstein distance between probability distributions, which has an analytical expression for Gaussian distributions (Givens & Shortt, 1984). We choose the bisimulation metric as the loss function based on the expectation that the additional supervision on the reward and transition distribution yields the learned representation to be consistent with respect to the task MDP. Given the imposed temporal cycle-consistency constraint, the task MDPs embedded in the resulting latent space will be identical for both the temporally forward and backward transitions.This is an appropriate inductive bias in many tasks such as navigation or continuous control (Figure 1). Hence, for each $l = 1, \dots, k$, the overall objective for training the dynamics model via retracing given imagination over k-steps take the following form (corresponding to line 9 in Algorithm 1).

Algorithm 1 *Cycle-Consistency World Model (CCWM)*

```
1: Given: policy $\hat{\pi}(a|s)$, encoder $q_\psi(O)$ and decoder $q_\theta(O|z)$ (Eq. 1), reverse action approximator $\rho(z, z')$, imagination horizon $k$, running average loss $\mathcal{L}_{\text{avg}} = 0$, counter $c = 0$.
2: Input: Batch of $N$ sampled trajectories of length $L$, $\{\mathbf{O}_{i,t_i:t_{i+L}}, \mathbf{a}_{i,t_i:t_{i+L}}\}_{i=1}^{N}$.
3: for $n = 1$ to $N$ do
4:     $O = \{\mathbf{O}_{n,t_n+j}\}_{j=1}^{k}$, $a = \{\mathbf{a}_{n,t_n+j}\}_{j=0}^{k-1}$
5:     Compute embedding vectors $z = \{q_\psi(O_j)\}_{j=0}^{k}$ (which we only use the first $L-k$ for learning representation given $k$-step imaginations).
6:     for $j = 1$ to $L-k$ do
7:         Compute prior $\hat{z}_{j+1:j+k}$ and posterior estimates $\tilde{z}_{j+1:j+k}$ using Eq. 3
8:         Compute retraced states $\bar{z}_{j:j+k}$ using Eq. 4
9:         Compute the latent model loss $\mathcal{L}_j$ using Eq. 6.
10:        Update $\mathcal{L}_{\text{avg}} \leftarrow \frac{c}{c+1}\mathcal{L}_{\text{avg}} + \frac{1}{c+1}\mathcal{L}_j$.
11:        Increment counter, $c \leftarrow c + 1$.
12:    end for
13: end for
14: Compute the gradient, $\nabla_\Theta \mathcal{L}_{\text{avg}}$.
15: Update the model parameters, $\Theta \leftarrow \Theta + \alpha\nabla_\Theta \mathcal{L}_{\text{avg}}$.
```

$$
\mathcal{L} = \sum_{i=1}^{k} \mathcal{L}_{\text{ELBO}}(O_{l+i}, \tilde{z}_{t+i}; \theta, \psi) + \lambda \mathcal{L}_{\text{bm}},
\tag{6}
$$

where λ is a scalar controlling the strength we wish to impose the temporal cycle-consistency. The dynamics model parameters $\Theta = \{\theta, \psi\}$ are updated via following the gradients of the variational objective in Eq. 6 with respect to Θ, which can be computed using reparameterisation trick followed by backpropagation (Kingma & Welling, 2014; Rezende et al., 2014). Figure 2 demonstrates the complete probabilistic graphical model for *CCWM*.

3.3 Reinforcement Learning in POMDPs

We aim to eventually utilise the learned representation and latent dynamics model for RL in POMDPs. To implement this, we interlace the model learning and the policy learning as in Hafner et al. (2020). For each iteration, we firstly update the models following the steps detailed in Algorithm 1, then use the current transition and reward models to compute the value estimates for the latent states, which can then be used in conjunction with downstream RL algorithms. Essentially *CCWM* can be combined with any RL algorithms in a model-based setting.

4 Experimental Studies

We evaluate *CCWM* on a number of challenging image-based continuous control tasks from the DeepMind Control Suite (Tassa et al., 2018), as shown in Figure 3a. The experimental studies aim to empirically examine the following aspects of the proposed *CCWM* framework: a) whether learning representation via retracing indeed aids model-based reinforcement learning in continuous control tasks using solely image inputs; b) the interpretability of the proposed retracing operation; c) the effects of different aspects of *CCWM* on training. The architecture and hyperparameters configurations of the neural network implementation and training details can be found in Appendix A.

4.1 Experiment Setup

We instantiate *CCWM* with the *RSSM* (Hafner et al., 2019) as the latent dynamics model, and *A3C* as the downstream RL agent for behaviour learning (Mnih et al., 2016). We refer to the resulting model-based agent as *CCWM-A3C*.

Despite introducing the framework of learning representations via retracing in its most general form of multi-step forward and backward rollouts, we commit to one-step case in our implementation, partially based on the empirical finding that multi-step rollouts have marginal improvements over the one-step rollouts in many model-based planning algorithms (Hamrick et al., 2020). We discuss additional motivations for sticking with the one-step case in the current implementation in Section 5.

To provide a comprehensive evaluation of the performance of the proposed method, we use 8 tasks from the DeepMind Control Suite (Tassa et al., 2018) as shown in Figure. 3a. We compare *CCWM-A3C* with the following baselines.

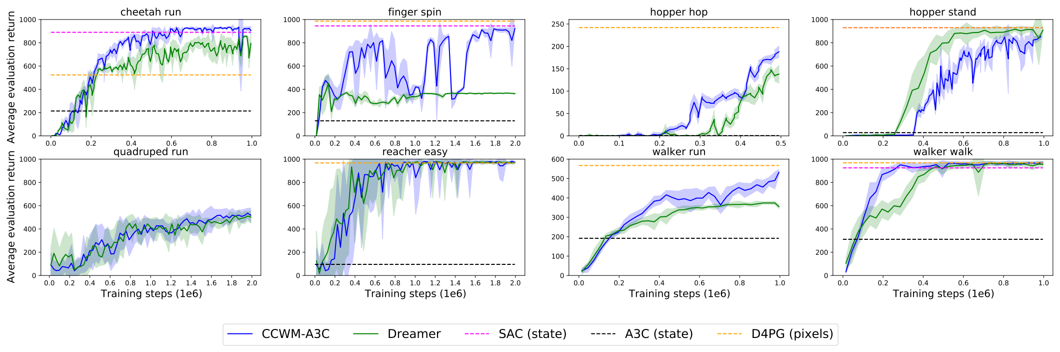
Dreamer (Hafner et al., 2020): We implement Dreamer, which integrates the model-learning and policy learning as approximately independent components for solving long-horizon decision-making problems in POMDPs based on image inputs.

Model-Free Baselines: We additionally compare with the following model-free algorithms: SAC (Haarnoja et al., 2018), D4PG (Barth-Maron et al., 2018), A3C (Mnih et al., 2016). Namely, we implement the SAC algorithm using the state inputs and directly report the asymptotic performance of the D4PG and A3C algorithms after 10^8 training steps given in Tassa et al. (2018). We only report the final performance of the model-free algorithms instead of the training curve due to the difference in training steps.

4.2 Continuous Control Evaluation

The performance of *CCWM-A3C* is shown in Figure 3b, with the comparison with the aforementioned baseline algorithms. The proposed framework in combination with RSSM achieves similar final convergence scores as the state-based model-free agents on 6 out of the 8 presented task environments. Since we chose our implementation built upon Dreamer, the comparison with the baseline Dreamer is essential. We observe that *CCWM-A3C* outperforms Dreamer on 5 of the selected tasks, and is comparable to Dreamer on 2 of the remaining 3 tasks, in terms of both the sample efficiency and final convergence performance. The only task that *CCWM-A3C* is weaker than Dreamer is the hopper stand task. We briefly discuss our intuition and proposed solution for this failure in Section 4.4.

(a)



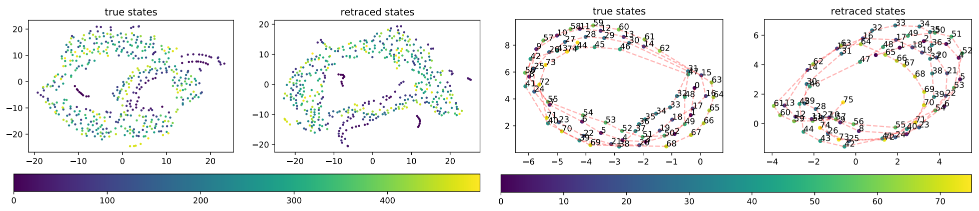
(b)

Figure 3: **Evaluation of *CCWM-A3C* on DeepMind Control Suite.** (a): Demonstration of selected continuous control task environments, from left to right: hopper stand/run, walker run/walk, finger spin, reacher easy, cheetah run, quadruped run. (b): Average evaluation returns (± 1 s.d.) during training (evaluated over 5 randomly chosen evaluation seeds) of *CCWM-A3C* and baseline agents.

The empirical results on the continuous control tasks conform with our hypothesis that utilising backward passes in addition to forward passes provides additional supervision, which improves the sample efficiency in RL task. Moreover, imposing the temporal cycle-consistency constraint yields significant performance increases on the finger spin, hopper hop and walker run tasks, demonstrating that retracing can help to learn stronger representation from pixel inputs for decision making.

4.3 Interpretability of Retracing

One might ask that whether the addition of the retrace stage is truly helpful for learning representations, i.e., whether the latent dynamics model and the reverse action approximator ρ can learn to generate similar predictions during retracing as during forward passes. We examine this by checking whether, after training, retracing yields qualitatively similar predictions over latent states as forward passes. We demonstrate this via showing the low-dimensional embeddings of the retraced and the ground-truth latent states (encoded given the observations) using tSNE visualisation (Van der Maaten & Hinton, 2008) in Figure 4.



(a) (b)

Figure 4: **Visualisation of similarity between the ground-truth states and retraced states using tSNE.** The two-dimensional embeddings for a 500-step trajectory (a) and a 76-step subtrajectory (b) is shown. Color of the scatters indicates the temporal ordering of the states within the trajectories.

In Figure 4a we show the 2-dimensional tSNE embeddings of the true and retraced states over an episode of 500 steps of the trained agent on the cheetah run task. We observe a similar dominant ring structure in both embeddings, which is expected, since during running, the simulated cheetah repeats its states periodically to resume running at a high speed. Such structure is correctly captured by the retracing. During the early stage of the episode (darker dots), the android starts to accelerate from stationarity and the states of the agent are periodic in nature but differs from the states when the agent is running at full speed. The structures for the states in the early phase of the episode in the embedding space for the true and retraced states have high qualitative similarity, again supporting the idea that the agent has successfully learned the correct retracing actions and states upon training.

We take a closer look at the sampled trajectory, by examining a 76-step sub-trajectory from it (Figure 4b, where the floating numbers indicate the timestep). We can see that the embedding of the true and retraced states share the same circular structure, with most states densely distributed on both sides of the ring and more loosely distributed on the paths connecting the two sides. The dynamics in the forward temporal direction causing clock-wise transitions in both embeddings. These suggest that our approach yields interpretable retrace actions.

4.4 Early Truncations

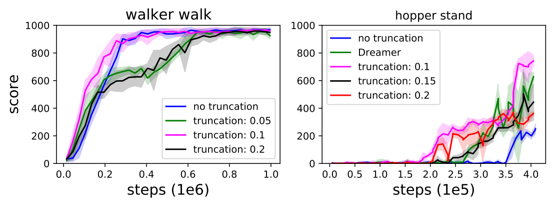


Figure 5: **Early truncation on the performance of *CCWM-A3C*.** *left:* walker walk; *right:* hopper hop.

We observe from Figure 3b that our agent behaves relatively poorly on the hopper stand task. One possible interpretation is that in the selected continuous control tasks, the simulated androids generally fall a lot, especially during the early stage of training. The retracing operation poses the cycle consistency constraint that the retraced states should be similar to the original states, but consider the situation shown in Figure 1b: no valid action can take the agent back to its previous state. This violates the assumption that the same MDP governs the forward and backward temporal dynamics hence might hinder the representation learning. In tasks such as hopper, where irrecoverable failure states occur more often than in other tasks, the effect is more severe.

We propose a potential resolution for this issue, by truncating the sampled episodes such that non-retraceable near-terminal states are left out for learning the representation via retracing.The effects of early truncations on the performance is shown in Figure 5.

We generally observe that introduction of appropriate truncation indeed have positive effects on learning, while either over- or under-truncation impedes the overall learning. We expect that the improvement will be most significant during the early stage of training, when the agent has not learned good behaviours and often stumbles into non-retraceable failure states. In the right panel of Figure 5, we show training curves over the first 4×10^5 steps for *CCWM* with different truncation lengths in the hopper stand task. Comparing to the original *CCWM-A3C* agent (Figure 3b), truncation consistently yields faster acquisition of good policies. With appropriate truncation strength, the resulting agent shows significant improvement in terms of sample efficiency comparing to the baseline Dreamer.

We suggest that truncation can act as a potential resolution for the issues caused by non-retraceable failure. The truncation length is currently set as a fixed hyperparameter and requires careful tuning for each task. Future work could look at the adaptive tuning of the truncation length.

5 Discussion

We proposed *CCWM*, a novel representation learning approach that combines the forward and backward latent rollouts based on the temporal cycle-consistency constraint for model learning. We define a joint objective for model training, which integrates the predictive reconstruction loss based on the ground-truth observation and the self-supervised loss that imposes temporal cycle-consistency between the original and retraced states. *CCWM* is a general framework for model learning in model-based RL, and can be combined with any stochastic sequential model as the latent dynamics model. We empirically show that the model-based instantiation of the proposed approach, *CCWM-A3C*, yields improved performance over state-of-the-art model-based and model-free methods on a number of continuous control benchmarks, in terms of sample-efficiency and final convergence score. We empirically investigated the interpretability and potential improvement schemes of *CCWM*.

We use point estimates from the distribution of the latent variable at the current timestep, instead of the distribution itself, to derive the predictive inference, hence imaginary rollouts can only be performed approximately, and the errors accumulate exponentially with the imagination horizon. Structured VAE deals with probabilistic inferences using deep learning in graphical models (Johnson et al., 2016), but is restricted to conjugate graphical model inferences. Future work could look at neural message passing algorithms for doing predictive inferences in general graphical structures that could support accurate multi-step inferences.

Although hippocampal replay is thought to play a role in learning the model for model-based decision-making, few computational models have been proposed to account for the occurrence of forward and backward replay (Mattar & Daw, 2018). Our model suggests the forward and reversed replay are used interchangeably for learning representations that aid decision-making tasks; the forward replay elicits predictions over future events and the reversed replay supports generalisable behaviours such as identification of bottleneck states. One testable experimental prediction could look at the effect of suppressing reversed hippocampal replay on the ability of within-domain generalisation.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Rishabh Agarwal, Marlos C. Machado, P. S. Castro, and Marc G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. 2021.

Gabriel Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, Dan Horgan, TB Dhruva, Alistair Muldal, N. Heess, and T. Lillicrap. Distributed distributional deterministic policy gradients. *ArXiv*, abs/1804.08617, 2018.

Lars Buesing, T. Weber, Sébastien Racanière, S. Eslami, Danilo Jimenez Rezende, David P. Reichert, F. Viola, F. Besse, K. Gregor, Demis Hassabis, and Daan Wierstra. Learning and querying fast generative models for reinforcement learning. *ArXiv*, abs/1802.03006, 2018.

K. Chua, R. Calandra, Rowan McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

D. Dwibedi, Y. Aytar, J. Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1801–1810, 2019.

N. Ferns, P. Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM J. Comput.*, 40:1662–1714, 2011.

Carles Gelada, S. Kumar, J. Buckman, Ofir Nachum, and Marc G. Bellemare. Deepmdp: Learning continuous latent space models for representation learning. *ArXiv*, abs/1906.02736, 2019.

C. R. Givens and R. Shortt. A class of wasserstein metrics for probability distributions. *Michigan Mathematical Journal*, 31:231–240, 1984.

David R Ha and J. Schmidhuber. World models. *ArXiv*, abs/1803.10122, 2018.

T. Haarnoja, Aurick Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Danijar Hafner, T. Lillicrap, Ian S. Fischer, R. Villegas, David R Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. *ArXiv*, abs/1811.04551, 2019.

Danijar Hafner, T. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *ArXiv*, abs/1912.01603, 2020.

Jessica B Hamrick, Abram L Friesen, Feryal Behbahani, Arthur Guez, Fabio Viola, Sims Witherspoon, Thomas Anthony, Lars Buesing, Petar Veličković, and Théophane Weber. On the role of planning in model-based deep reinforcement learning. *arXiv preprint arXiv:2011.04021*, 2020.

Matthew J Johnson, David Duvenaud, Alexander B Wiltschko, Sandeep R Datta, and Ryan P Adams. Composing graphical models with neural networks for structured representations and fast inference. *arXiv preprint arXiv:1603.06277*, 2016.

L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101:99–134, 1998.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

Alex X. Lee, Anusha Nagabandi, P. Abbeel, and S. Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *ArXiv*, abs/1907.00953, 2020a.

L. Lee, Benjamin Eysenbach, R. Salakhutdinov, Shixiang Gu, and Chelsea Finn. Weakly-supervised reinforcement learning for controllable behavior. *ArXiv*, abs/2004.02860, 2020b.

M. Mattar and N. Daw. Prioritized memory access explains planning and hippocampal replay. *Nature neuroscience*, 21:1609 – 1617, 2018.

P. Mirowski, Razvan Pascanu, F. Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and Raia Hadsell. Learning to navigate in complex environments. *ArXiv*, abs/1611.03673, 2017.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

V. Mnih, Adrià Puigdomènech Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *ArXiv*, abs/1602.01783, 2016.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. 2012.

A. Oord, Y. Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

Will D Penny, Peter Zeidman, and Neil Burgess. Forward and backward inference in spatial cognition. *PLoS Comput Biol*, 9(12):e1003383, 2013.

Danilo Jimenez Rezende, S. Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Maneesh Sahani. *Latent variable models for neural data analysis*. California Institute of Technology Pasadena, CA, 1999.

Julian Schrittwieser, Ioannis Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, Edward Lockhart, Demis Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 7839:604–609, 2020a.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020b.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *ArXiv*, abs/1612.07307, 2017.

A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and Chelsea Finn. Universal planning networks. *ArXiv*, abs/1804.00645, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Y. Tassa, Yotam Doron, Alistair Muldal, T. Erez, Y. Li, D. Casas, D. Budden, Abbas Abdolmaleki, J. Merel, Andrew Lefrancq, T. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *ArXiv*, abs/1801.00690, 2018.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1:1–305, 2008.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. Improving neural machine translation with conditional sequence generative adversarial nets. *arXiv preprint arXiv:1703.04887*, 2017.

Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pp. 2528–2535. IEEE, 2010.

A. Zhang, Rowan McAllister, R. Calandra, Yarin Gal, and S. Levine. Learning invariant representations for reinforcement learning without reconstruction. *ArXiv*, abs/2006.10742, 2020.

Tinghui Zhou, Philipp Krähenbühl, Mathieu Aubry, Qixing Huang, and Alexei A. Efros. Learning dense correspondence via 3d-guided cycle consistency. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 117–126, 2016.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

A Implementation Details

All stochastic variables in *CCWM* are assumed to be Gaussian. We implement the latent transition dynamics model of *CCWM* using the RSSM Hafner et al. (2019), utilising a Gated Recurrent Unit (GRU) as the core component (Chung et al., 2014) in addition to the MLP. The RSSM is used for modelling the predictive inferences, $q_\psi(z_{t+1}|z_t, a_t)$ and $q_\psi(z_{t+1}|z_t, a_t, O_{t+1})$, from Eq. 1. The encoding function is given by a convolutional encoder that deterministically embeds the high-dimensional observation into the latent space. The dimension of the latent space is set to equal 32, there is low sensitivity of the performance with respect to the dimension of the latent embedding. The generative function $q_\psi(O_t|z_t)$ is defined by an MLP followed by a deconvolution network (Zeiler et al., 2010), outputting the means the Gaussian distributions for each pixel location of the decoded observation (note that here we assume all pixel-level Gaussians have variance 1). The reward distribution $\hat{\mathcal{R}}(r|z)$ is modelled by an MLP, outputting a scalar mean Gaussian distribution (again, assuming unit variance). The supervision of the model learning is based on the ELBO and the bisimulation metric as described in Section 3.

In the downstream A3C implementation, we use a Gaussian distribution with means and variances parameterised by an MLP for modelling the value estimates. The policy distribution is also modelled by a Gaussian distribution, with means and variances parameterised by an MLP. We used distributed actors for interacting with the environment for more efficiency collection of sampled episodes as in (Mnih et al., 2016). The value function is optimised by maximising the probability of the "ground-truth" values estimated by the lambda return (Sutton & Barto, 2018). The policy distribution is updated using policy gradient estimated based generalised advantage estimation (Schulman et al., 2015).

We use the Adam optimiser for updating the parameters for the model, value function, and policy networks (Kingma & Ba, 2014).

The specific configurations for the neural network implementations is summarised in Table 1. Note that the activation functions for all non-output layers are assumed to be ReLU activation function unless otherwise stated.

All neural network implementation are carried out using TensorFlow (Abadi et al., 2015) and TensorFlow Distributions (Dillon et al., 2017).

For the actual training, the batch size is chosen to be 64, and all sampled trajectories are taken to be 50 timesteps long. After everything 10^4 training steps, we evaluate the agent over 5 random seeds. We adopt the same scheme for setting the action repeats equal to 2 across all tested environments as in (Hafner et al., 2020). The parameter λ controlling the weights of the retrace auxiliary loss in Eq. 6 is set to 1.0. The discounting factor for the expected value function is set to 0.99.

We implemented the tSNE visualisation shown in Figure 4 by firstly applying principal component analysis initialisation (taking first 50 principal components), followed by standard tSNE embedding.

The python implementation of the *CCWM-A3C* can be found in the supplementary materials.

B Further Discussion on Truncation

In Section 4.4, we introduced *truncation* as a general-purpose method for improving the performance of *CCWM* in continuous control tasks. The complete ablation studies on the effects of the strength of truncation on the remaining tasks in Figure 3a is shown in Figure 6. We observe that the 0.1 truncation generally yields higher performance than the basic *CCWM-A3C* agent.

The basic idea of truncation is to cut off the last proportions of the sampled episodes such that the possible non-retraceable states in the episodes will not be involved in training to update the model parameters. However, the current scheme for setting a fixed truncation proportion throughout the training process seems cumbersome, and hinders learning when either under- or over-truncation occurs. Specifically, in Figure 5, we showed that appropriate truncation significantly improves the performance during the early stage of learning (first 4×10^5 training steps) in the hopper stand task, but is quickly surpassed by Dreamer after the early stage as shown in Figure 6. Despite the fact that the truncation improves the overall performance comparing to the basic version of *CCWM-A3C*,

Component	Attribute	Value
RSSM	Internal state dimension	256
	MLP layer 1 units	256
	MLP layer 2 units	64
Embedding function	Conv Layer 1 number of filters	32
	Conv Layer 1 kernel size	4
	Conv Layer 2 number of filters	64
	Conv Layer 2 kernel size	4
	Conv Layer 3 number of filters	128
	Conv Layer 3 kernel size	4
	Conv Layer 4 number of filters	256
	Conv Layer 4 kernel size	4
	Final output operation	Concatenation
Generative model	MLP layer 1 number of units	1024
	Deconvolution Layer 1 number of filters	128
	Deconvolution Layer 1 kernel size	5
	Deconvolution Layer 2 number of filters	64
	Deconvolution Layer 2 kernel size	5
	Deconvolution Layer 3 number of filters	32
	Deconvolution Layer 3 kernel size	6
	Deconvolution Layer 4 number of filters	3
	Deconvolution Layer 4 kernel size	6
Reward model	MLP layer 1 number of units	512
	MLP Layer 2 number of units	512
	MLP Layer 3 number of units	1
Value model	MLP Layer 1 number of units	512
	MLP Layer 2 number of units	512
	MLP Layer 3 number of units	512
	MLP Layer 4 number of units	1
Policy model	All fully connected layers activation	Elu
	MLP Layer 1 number of units	512
	MLP Layer 2 number of units	512
	MLP Layer 3 number of units	512
	MLP Layer 4 number of units	512
	MLP Layer 5 number of units	2
Adam optimiser	Learning rate for Model Learning	6×10^{-4}
	Learning rate for Value Model	8×10^{-5}
	Learning rate for Policy Model	8×10^{-5}

Table 1: Configurations for the neural network implementations of *CCWM-A3C*.

still, it is not comparable with the baseline Dreamer agent in terms of sample efficiency beyond the early stage. We argue that this is due to the fact that the agent quickly learns the ability to control at a relatively high level, which causes significant decrease in the frequency of falling of the simulated hopper (note that this is shared between the hopper stand and hopper hop tasks). Hence if we stick with a fixed truncation proportion, this will lead to the undesirable consequence that the states containing information that guide the agent to achieve higher performance (in the hopper stand task, this corresponds to the states where the torso is brought to a greater height, leading to higher reward received) will not be used for training the model, hence such information will be embedded into the learnt representation, hence causing slow acquisition of near-optimal behaviours. We leave the study of the adaptive truncation strength for future work.

C Zero-Shot Transfer

Based on the motivation that learning with retracing will improve the generalisability of the agent (Figure 1a), we evaluate the generalisability of our approach on the basis of zero-shot transfer tasks.

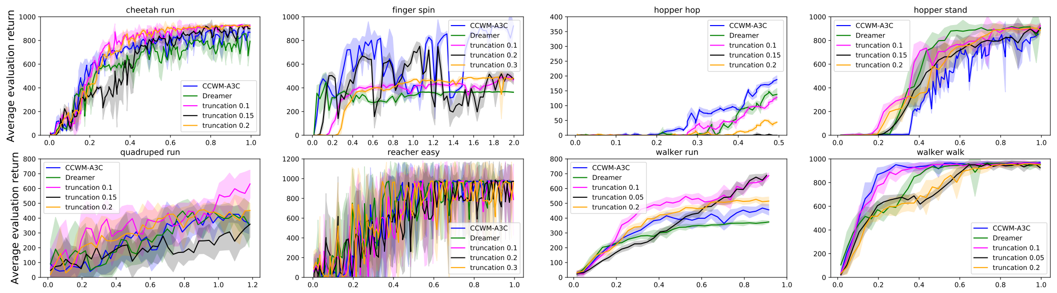


Figure 6: **Full ablation studies on the truncation strength.** We show the effects of truncation strengths on the performance in the dm-control tasks we used in Figure 3a. The training steps (on the horizontal axis) is on the scale of 10^6.

CHANGED COMPONENTS	OUR APPROACH (MEAN ± 1 S.D.)	DREAMER (MEAN ± 1 S.D.)	P-VALUE (3 S.F.)	SIGNIFICANT? ($\alpha = 0.01$)
REWARD FUNCTION	630.40 ± 6.49	629.00 ± 47.01	5.22×10^{-1}	NO
REWARD FUNCTION + MASS	635.37 ± 40.12	597.43 ± 87.68	7.84×10^{-2}	NO
REWARD FUNCTION + FRICTION	643.58 ± 9.29	649.27 ± 3.96	9.76×10^{-1}	NO
REWARD FUNCTION + STIFFNESS	634.80 ± 10.92	646.07 ± 7.78	9.98×10^{-1}	NO
REWARD FUNCTION + MASS + FRICTION	$\mathbf{628.07 \pm 36.95}$	468.82 ± 94.73	$\mathbf{7.27 \times 10^{-6}}$	YES
REWARD FUNCTION + MASS + STIFFNESS + FRICTION	$\mathbf{641.89 \pm 28.67}$	562.58 ± 93.91	$\mathbf{3.97 \times 10^{-3}}$	YES

Table 2: Evaluation of trained *CCWM-A3C* and baseline Dreamer agents on the ability of zero-shot transfer in cheetah run tasks with different configurations.

Specifically, we modify a number of basic configurations of the cheetah run task, such as the mass of the agent, friction coefficients between the joints, reward function of task, etc. The details of the changes can be found in the appendix. Judging from the learning curve of the cheetah run task in Figure 3b, despite the increased sample efficiency of our approach during training, both our approach and the baseline agent, Dreamer, converge at a similar value at 2×10^6 steps. We then evaluate the trained agents directly on the updated cheetah run task without any further training to test their abilities on zero-shot transfer. We show the mean evaluation scores ($\pm$ 1 s.d.) of both agents over 15 random seeds, as well as the one-sided t-test statistics and significance of the difference between the two sets of evaluations in Table 2.

It can be observed from the evaluation results that the overall performance on zero-shot transfer of our approach is comparable with Dreamer on simpler transfer tasks, and significantly outperforms Dreamer on more complicated tasks. Moreover, we see that the introduction of retracing improves the stability of evaluations on zero-shot transfer tasks in general. These confirm our previous hypothesis that retracing poses a positive effect on the generalisability (Figure 1a).