Supplementary Materials to Temporal Predictive Coding For Model-Based Planning In Latent Space

A. Hyper Parameters

A.1. Standard setting

Dreamer, CVRL and TPC share the following hyperparameters:

Model components

- Latent state dimension: 30
- Recurrent state dimension: 200
- Activation function: ELU
- The action model outputs a tanh mean scaled by a factor of 5 and a softplus standard deviation for the Normal distribution that is then transformed using tanh (Haarnoja et al., 2018)

Learning updates

- Batch size: 50 for Dreamer and CVRL, 250 for TPC
- Trajectories length: 50
- Optimizer: Adam (Kingma & Ba, 2014) with learning rates 6×10^{-4} for world model, 8×10^{-5} for value and action.
- Gradient update rate: 100 gradient updates every 1000 environment steps.
- Gradient clipping norm: 100
- Imagination horizon: 15
- $\gamma = 0.99$ and $\lambda = 0.95$ for value estimation

Environment interaction

- The dataset is initialized with $S = 5$ episodes collected using random actions.
- We iterate between 100 training steps and collecting 1 episode by executing the predicted mode action with $\mathcal{N}(0, 0.3)$ exploration noise.
- Action repeat: 2
- Environment steps: 2×10^6

Additionally,

- Dreamer and CVRL clip the KL below 3 nats
- CVRL uses a bi-linear model for the critic function in the contrastive loss: $f_\theta(s_t, o_t) = \exp(z_t^T W_\theta s_t)$, where z_t is an embedding vector for observation o_t and W_θ is a learnable weight matrix parameterized by θ.

- TPC has a fixed set of coefficient in the overall objective for all control tasks: $\lambda_1 = 1, \lambda_2 = 0.1, \lambda_3 = 1, \lambda_4 = 1$. We use a fixed Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.2^2)$ to add to the future latent code when computing temporal CPC, as suggested in (Shu et al., 2020), and also use 0.2 as the fixed variance in static CPC.

In TPC, we also use a target network for the value model and update this network every 100 gradient steps. Note that we also tried to use target value network for Dreamer, but it does not improve the results, as suggested by their original paper (Hafner et al., 2020).

Hyperparameters for DBC We use the same set of hyperparameters as reported in the paper (Zhang et al., 2020)

- Replay buffer capacity: 1000000
- Batch size: 128
- Discount γ: 0.99
- Optimizer: Adam
- Critic learning rate: 10^{-5}
- Critic target update frequency: 2
- Critic Q-function soft-update rate τ_Q : 0.005
- Critic encoder soft-update rate τ_ϕ : 0.005
- Actor learning rate: 10^{-5}
- Actor update frequency: 2
- Actor log stddev bounds: $[-5, 2]$
- Encoder learning rate: 10^{-5}
- Decoder learning rate: 10^{-5}
- Decoder weight decay: 10^{-7}
- Temperature learning rate: 10^{-4}
- Temperature Adam's β_1 : 0.9
- Init temperature: 0.1

Hyperparameters search for TPC TPC has four hyperparameters that can be tuned: λ_1, λ_2, λ_3 and λ_4, which are coefficients for the TPC objective, consistency objective, SPC objective and reward prediction objective, respectively. Since λ_1, λ_3 and λ_4 do not conflict with each other, we fixed them to 1 and only tuned λ_2 in our experiments. We performed grid search for λ_2 in range $\{0.05, 0.1, 0.2\}$[2] *on the Cartpole Swingup task and then used the same set of hyperparameters for all the remaining tasks.*

A.2. Natural background setting

To further encourage the model to focus on task-relevant information from observations, we additionally tune the weight λ_4 of the reward loss in the training objective for both Dreamer and TPC. In each control task they share the same reward coefficient, which is specified in the table below. CVRL and DBC have the same hyperparameters as in the standard setting.

[2]Larger values of λ_2 lead to representation collapse.

Table 1. Reward coefficients for different tasks in the natural backgrounds setting

Task	**Reward coefficient**
Cartpole Swingup, Cup Catch	1000
Cheetah Run, Walker Run, Pendulum Swingup, Hopper Hop	100

B. Proof of Lemma 1

Our goal is to show that, under the conditions in Lemma 1,

$$\eta(\pi^*) \geq \eta(\pi^*_{\text{aux}}) \tag{15}$$

for any choice of auxiliary encoder E'.

We start by denoting $s_t = E^*(o_t)$ and $s'_t = E'(o_t)$. Note that the performance of π_{aux} can be written as

$$\begin{aligned}\eta(\pi_{\text{aux}}) &= \mathbb{E}_{(\pi_{\text{aux}},p)} r(o_{1:T}) & (16)\\ &= \sum_{\tau_{\text{aux}}} r(o_{1:T}) \prod_t p(o_t \mid o_{<t}, a_{<t}) p(s_t, s'_t \mid o_t) \pi_{\text{aux}}(a_t \mid s_{\leq t}, s'_{\leq t}, a_{<t}), & (17)\end{aligned}$$

where τ_{aux} denotes the full trajectory of $(o, s, s', a)_{1:T}$ and $r(o_t)$ evaluates the reward at o_t (for simplicity, we shall assume $p(r_t \mid s_t)$ is deterministic. Since $D_{\text{KL}}(p(r_t \mid o_t) \parallel R^*(r_t \mid E^*(o_t))) = 0$, we can rewrite as

$$\eta(\pi_{\text{aux}}) = \sum_{\tau_{\text{aux}}} R^*(s_{1:T}) \prod_t p(o_t \mid o_{<t}, a_{<t}) p(s_t, s'_t \mid o_t) \pi_{\text{aux}}(a_t \mid s_{\leq t}, s'_{\leq t}, a_{<t}), \tag{18}$$

where, with a slight abuse of notation, we note that $R^*(E^*(o_t)) = r(o_t)$. We now further rewrite $\pi_{\text{aux}}(a_t \mid s_{\leq t}, s'_{\leq t}, a_{<t})$ as

$$p(a_t \mid s_{\leq t}, s_{\leq t}, a_{<t}, \pi_{\text{aux}}), \tag{19}$$

and subsequently collapse the expression of the performance as

$$\begin{aligned}\eta(\pi_{\text{aux}}) &= \sum_{(o,s,s',a)_{1:T}} R^*(s_{1:T}) p(o_{1:T}, s_{1:T}, s'_{1:T}, a_{1:T} \mid \pi_{\text{aux}}) & (20)\\ &= \sum_{(s,s',a)_{1:T}} R^*(s_{1:T}) p(s_{1:T}, s'_{1:T}, a_{1:T} \mid \pi_{\text{aux}}), & (21)\end{aligned}$$

where the last step arises from marginalization of $o_{1:T}$. Note by chain rule that $p(s_{1:T}, s'_{1:T}, a_{1:T} \mid \pi_{\text{aux}})$ becomes

$$\prod_t p(s_t \mid s_{<t}, s'_{<t}, a_{<t}, \pi_{\text{aux}}) p(s'_t \mid s_{\leq t}, s'_{<t}, a_{<t}, \pi_{\text{aux}}) p(a_t \mid s_{\leq t}, s'_{\leq t}, a_{<t}, \pi_{\text{aux}}). \tag{22}$$

By analyzing the Markov blankets in $p(s_{1:T}, s'_{1:T}, a_{1:T} \mid \pi_{\text{aux}})$, we can simplify the above expression to

$$\prod_t p(s_t \mid s_{<t}, s'_{<t}, a_{<t}) p(s'_t \mid s_{\leq t}, s'_{<t}, a_{<t}) p(a_t \mid s_{\leq t}, s'_{\leq t}, a_{<t}, \pi_{\text{aux}}). \tag{23}$$

Note that we omit the dependency on π_{aux} in the first two terms since, given only the history of past actions and observations, the next observation does not depend on our choice of policy but only on the environment dynamics.

Since E^* is optimal under the MI objective, we note that

$$I(S_{<t}, S'_{<t}, A_{<t} \,; S_t, S'_t) = I(S_{<t}, A_{<t} \,; S_t). \tag{24}$$

Eq. (24) implies that $s'_{<t}$ is independent of s_t given $(s_{<t}, a_{<t})$, and that $(s_{<t}, s'_{<t}, a_{<t})$ is independent of s'_t given s_t. This allow us to further simplify Eq. (23) to

$$\prod_t p(s_t \mid s_{<t}, a_{<t}) p(s'_t \mid s_t) \pi_{\text{aux}}(a_t \mid s_{\le t}, s'_{\le t}, a_{<t}). \tag{25}$$

Thus, the performance expression equates to

$$\eta(\pi_{\text{aux}}) = \sum_{\tau_{\text{aux}}} R^*(s_{1:T}) \prod_t p(s_t \mid s_{<t}, a_{<t}) p(s'_t \mid s_t) \pi_{\text{aux}}(a_t \mid s_{\le t}, s'_{\le t}, a_{<t}). \tag{26}$$

Note by way of similar reasoning (up to and including Eq. (23)) that

$$\eta(\pi) = \sum_{\tau} R^*(s_{1:T}) \prod_t p(s_t \mid s_{<t}, a_{<t}) \pi(a_t \mid s_{\le t}, a_{<t}). \tag{27}$$

By comparing Eq. (26) and Eq. (27), we see that $s'_{1:T}$ effectively serves as a source of noise that makes π_{aux} behave like a stochastic policy depending on the seed choice for $s'_{1:T}$. To take advantage of this, we introduce a reparameterization of s' as ϵ such that

$$\eta(\pi_{\text{aux}}) = \sum_{\tau_{\text{aux}}} R^*(s_{1:T}) \prod_t p(s_t \mid s_{<t}, a_{<t}) p(\epsilon_t) \pi_{\text{aux}}(a_t \mid s_{\le t}, \epsilon_{\le t}, a_{<t}) \tag{28}$$

$$= \mathbb{E}_{p(\epsilon_{1:T})} \sum_{(s,a)_{1:T}} R^*(s_{1:T}) \prod_t p(s_t \mid s_{<t}, a_{<t}) \pi_{\text{aux}}(a_t \mid s_{\le t}, \epsilon_{\le t}, a_{<t}) \tag{29}$$

$$\le \max_{\epsilon_{1:T}} \sum_{(s,a)_{1:T}} R^*(s_{1:T}) \prod_t p(s_t \mid s_{<t}, a_{<t}) \pi_{\text{aux}}(a_t \mid s_{\le t}, \epsilon_{\le t}, a_{<t}) \tag{30}$$

$$\le \max_{\pi} \eta(\pi), \tag{31}$$

where the last inequality comes from defining a policy

$$\pi' := \pi_{\text{aux}}(a_t \mid o_{\le t}, \epsilon^*_{\le t}, a_{<t}) \tag{32}$$

and noting that the performance of π' must be bounded by the performance of π^*. $\square$

C. Additional Results

C.1. Comparision with CVRL after 2×10^6 environment steps

In Figure 6, we compare the performance of TPC and CVRL after 2×10^6 environment steps in both the standard setting and the natural background setting. TPC learns faster and achieves much higher rewards compared to CVRL in the standard setting. In the natural control setting, TPC outperforms in 4 out of 6 tasks and is competitive in Walker Run. Both methods do not work on Hopper Hop.

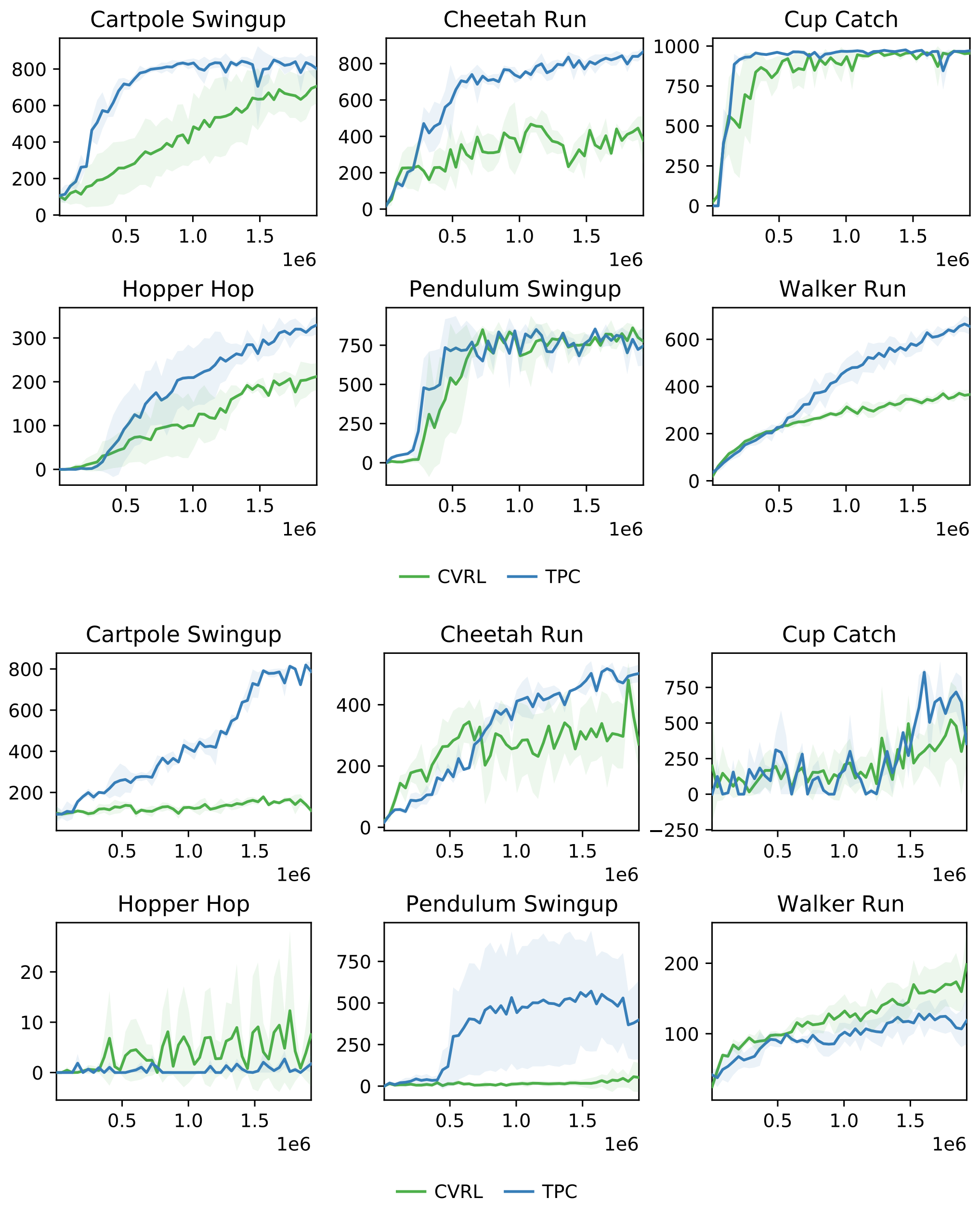

Figure 6. TPC vs CVRL after 2×10^6 in the standard setting (top) and background setting (bottom)

C.2. Comparison with CURL in the natural background setting

As shown in Figure 7, TPC outperforms CURL significantly on 3 of 6 tasks, while CURL performs better on Walker Run. On Hopper Hop and Cup Catch, both methods fail to make progress after 1 million environment steps.

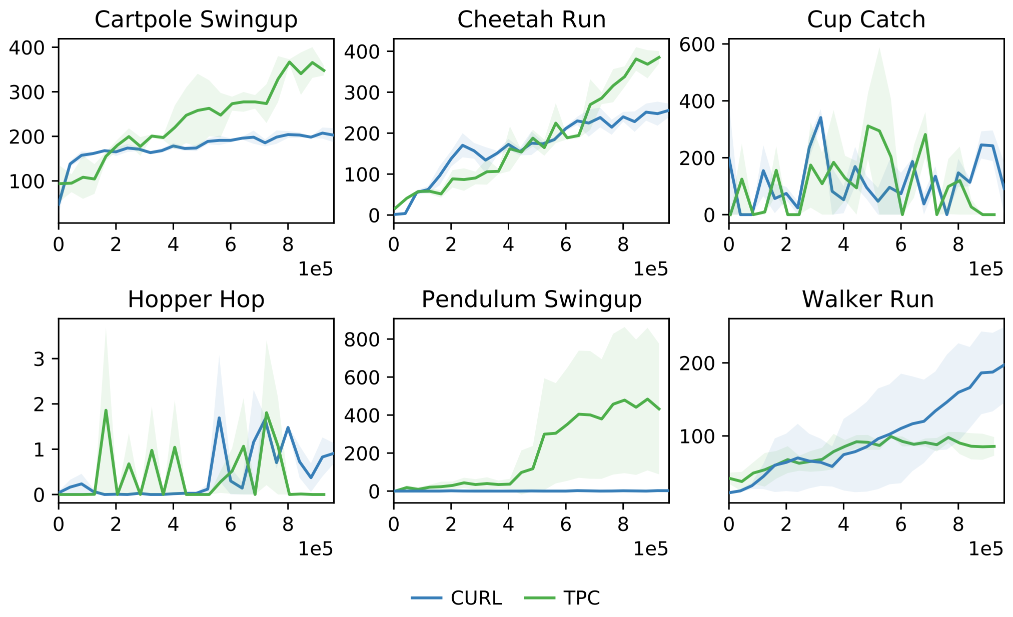

Figure 7. TPC versus CURL in the natural background setting. Each task is run with 3 seeds.

C.3. Importance of dynamics smoothing

We run TPC in the standard setting without dynamics smoothing to investigate the empirical importance of this component. As shown in Figure **??**, TPC 's performance degrades significantly without dynamics smoothing. Without smoothing, the dynamics model cannot handle noisy rollouts during test-time planning, leading to poor performance. Dynamics smoothing prevents this by enabling test-time robustness against cascading error.

C.4. Learning a separate reward model

Joint learning of reward is crucial for all models in the natural background setting (see Appendix A.2). Since background information is also temporally-predictive, increasing the weight of reward loss encourages the model to focus more on the components that are important for reward learning. However, in the random background setting, since all temporally-predictive information is task-relevant, TPC *uniquely* can learn the reward separately, as shown in Figure **??**.

C.5. Reconstructions in the natural background setting

We conduct experiments to investigate what information the encoder in different models learns to encode during training in the natural background setting. To do that, we train auxiliary decoders that try to reconstruct the original observations from the representations learned by Dreamer and TPC. As shown in Figure 8, Dreamer (2$^{\text{nd}}$ row) tries to encode both the agent and the background. In contrast, TPC (3$^{\text{rd}}$ row) prioritizes encoding the agent, which is task-relevant, over the background.

C.6. The simplistic motion background setting

As discussed in Section 3.2, TPC can capture certain task-irrelevant information. However, TPC can choose to not encode the background *whenever* encoding only the agent is sufficient to maximize the mutual information. In the experiments, we found that forcing the model to predict well the reward helps the encoder focus more on the agent, which can be done by increasing the weight of reward loss. Dreamer, in contrast, must encode as much information about the observation as

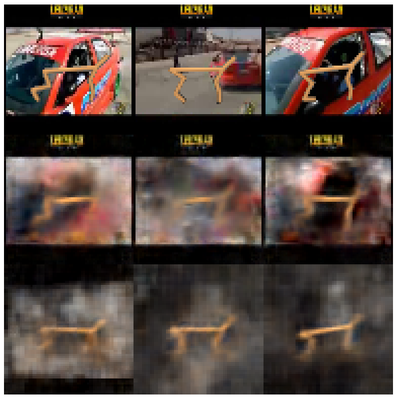

Figure 8. Observation reconstruction of TPC versus Dreamer in the natural background setting

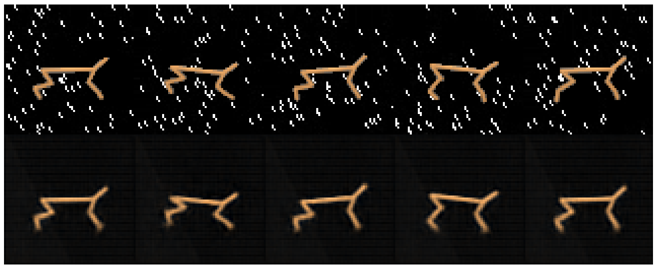

Figure 9. The top row shows a sample sequence of data, and the bottom row shows the reconstruction of TPC.

possible to achieve a good reconstruction loss. To elaborate on this, we conducted an experiment where we replaced the natural background with a simplistic motion, easily predictable background, which is depicted in Figure 9. Figure **??** shows that TPC works well in this setting, and outperforms Dreamer significantly.