A Further discussion

A.1 Task selection

Our primary concern when choosing tasks is mainly the computational cost. With limited computation resources, we couldn't afford to run the extensive $5e6$ steps for DMC and $2e8$ steps for Atari as the Dreamer papers did. Instead, we only had a $1e6$-step budget for DMC and a $1e7$-step budget for Atari. Consequently, in the case of DMC, we focus on choosing tasks from different environments on which Dreamer can roughly reach convergence in under $1e6$ steps. Similarly, for Atari, we focus on tasks that DreamerV2 could reach convergence within our $1e7$ budget. We are certainly enthusiastic about exploring how TEMPO performs on more challenging tasks in the future.

A.2 Network settings

The architectural configurations of our meta weighter, such as the number of layers and hidden dimensions, were informally aligned with the network modules employed in DreamerV2. Specifically, we mirrored the architectural choices found in modules like the reward module of RSSM, the actor, and the critic, all of which adopt a similar MLP (Multi-Layer Perceptron) structure. Given that these modules can handle state representation well, it was a logical choice to build a similar architecture for our meta weighter as well. Perhaps the most important thing we discovered about architecture design is that batch normalization in the meta weighter greatly boosts the final performance.

For the mapping the network outputs to meta weights, we toyed with $weight = \sigma(output) + 0.5$ and $weight = 0.5 \times \sigma(output) + 0.75$ initial experiments and didn't observe a noticeable difference, so we stuck with the latter in the hope of stable training.

A.3 Tuning strategy

One of the key pursuits when we built TEMPO was to design a paradigm that could outperform DreamerV2 with little tuning demand. Since TEMPO operates as an additive bonus algorithm, essentially fine-tuning the training sample weights within DreamerV2, it is quite easy for TEMPO to at least meet DreamerV2's performance, resulting in low tuning demand. Specifically, we tuned TEMPO on DMC Walker Walk and Atari Pong such that meta weighter converges gradually alongside the RSSM. This is to ensure the weighter doesn't reach early convergence on a meaningless state representation. We have no doubt that superior architecture and hyperparameters exist for TEMPO, and we remain committed to further exploration and refinement.

A.4 Performance analysis

Based on our findings, TEMPO demonstrates a notable advantage over DreamerV2 in specific tasks, such as Finger Spin, while in others, like Walker Walk, the two methods exhibit similar performance levels. We hypothesize that the distance between naive state representation from MLE and the task-aware state representation can vary on different tasks. When the distance is large, meaning that the features that can effectively lower reconstruction error don't coincide well with the task-specific features, DreamerV2 can stuck in learning task-irrelevant features like a local minimum, while TEMPO's bi-level learning quickly captures the task-relevant features, and lets the agent performance snowball. In other tasks, where the distance is not significant, TEMPO can have less advantage.

We also observe that TEMPO shows larger variances in some tasks than others. We suspect the reason is that, in these tasks, the meta weighter learns too "eagerly" relative to the RSSM, leading to less meaningful meta weights, potentially disturbing the learning of RSSM. Lowering the learning rate for the weighter may be helpful in reducing the variance in these tasks.

A.5 Future directions

World models, especially MLE-based models, encode a broad spectrum of environmental dynamics. They carry the potential of multi-tasking with a single model, presenting a possible way to multi-tasking agents, even general AI. With the hierarchical environment modeling paradigm that TEMPO offers, we aspire for it to serve as a foundational framework for enhancing task awareness across multiple specific tasks within future world models.

B Derivations

The variational bound, i.e. Evidence Lower Bound (ELBO), for the Recurrent State-space Model (RSSM) with a variational posterior $q(s_{1:T}|o_{1:T}, a_{1:T}) = \prod_{t=1}^{T} q(s_t|h_t, o_t)$, is written as

$$\begin{aligned}
&\log p(o_{1:T}, r_{1:T}|a_{1:T}) \\
&= \log \int \prod_{t=1}^{T} p(s_t|h_t) \cdot p(o_t, r_t|s_t, h_t) \, \mathrm{d}s_{1:T} \\
&= \log \int \prod_{t=1}^{T} q(s_t|h_t, o_t) \cdot \frac{p(s_t|h_t)\, p(o_t, r_t|s_t, h_t)}{q(s_t|h_t, o_t)} \, \mathrm{d}s_{1:T} \\
&= \log \mathrm{E}_{q(s_{1:T}|o_{1:T}, a_{1:T})} \left[\prod_{t=1}^{T} \frac{p(s_t|h_t)\, p(o_t, r_t|s_t, h_t)}{q(s_t|h_t, o_t)} \right] \\
&\geq \mathrm{E}_{q(s_{1:T}|o_{1:T}, a_{1:T})} \left[\sum_{t=1}^{T} \log p(o_t, r_t|s_t, h_t) + \log p(s_t|h_t) - \log q(s_t|h_t, o_t) \right] \\
&= \sum_{t=1}^{T} \left(\mathrm{E}_{q(s_t|h_t, o_t)} \big[p(o_t, r_t|s_t, h_t) \big] - \mathrm{E}_{q(s_{t-1}|h_{t-1}, o_{t-1})} \Big[\mathrm{KL}\big[q(s_t|h_t, o_t) \,\|\, p(s_t|h_t) \big] \Big] \right)
\end{aligned} \tag{1}$$

C Hyperparameters

Module	Name	Value
World model	Batch size	16
	Trajectory length	50
	KL balancing	0.8
for DMC	Deterministic state dimension	200
	Stochastic state dimension	32-dim continuous
	Learning rate	$3\text{e}{-4}$
	KL scale	1.0
for Atari	Deterministic state dimension	600
	Stochastic state dimension	32-dim discrete (32 classes/dim)
	Learning rate	$2\text{e}{-4}$
	KL scale	0.1
Agent	Imagination horizon	15
	Discount factor	0.99
	Discount factor for λ-target	0.95
	Target critic update interval	100
for DMC	Actor learning rate	$8\text{e}{-5}$
	Critic learning rate	$8\text{e}{-5}$
for Atari	Actor learning rate	$4\text{e}{-5}$
	Critic learning rate	$1\text{e}{-4}$
Meta weighter	Number of dense layers	5
	Hidden dimension	400
	Activation function	ELU
	Normalization	Batch
	Learning rate	$1\text{e}{-4}$
for DMC	Input dimension	$200 + 2 \times 32 + 1 \times 2$
for Atari	Input dimension	$600 + 2 \times 32 \times 32 + 1 \times 2$
Common	Optimizer	Adam
	Gradient clipping	100
	Adam epsilon	$1\text{e}{-5}$
	Weight decay	$1\text{e}{-6}$

Table 1: Main hyperparameters of TEMPO. DMC stands for DeepMind Control tasks. We use the default setting of Dreamerv2 in World model, Agent, and Common. All experiments were run on a single Nvidia RTX 3090 GPU with Python 3.7 and Tensorflow 2.6. Refer to our sample code for full implementation details.

D Full results on continuous control tasks

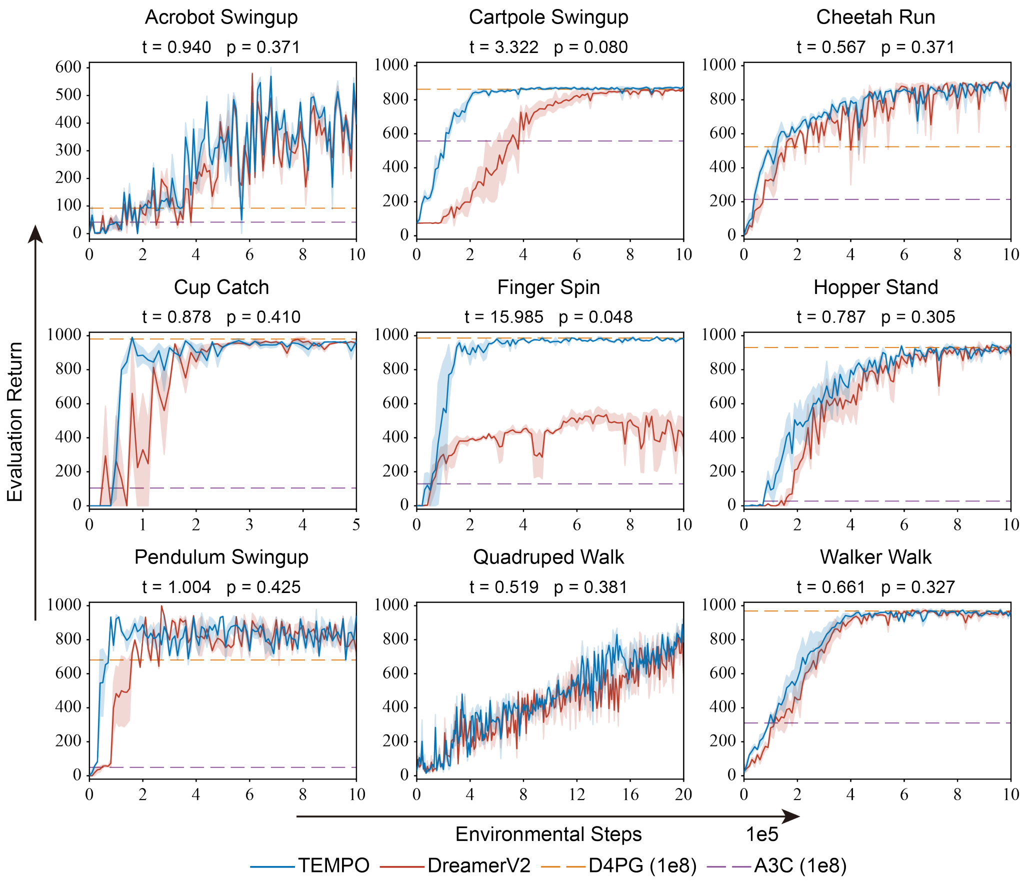

Figure 1: Evaluation of TEMPO on continuous control tasks from DeepMind Control Suite. The lines show mean scores and the shaded areas show the standard deviation across 3 random seeds.

E Full results on discrete control tasks

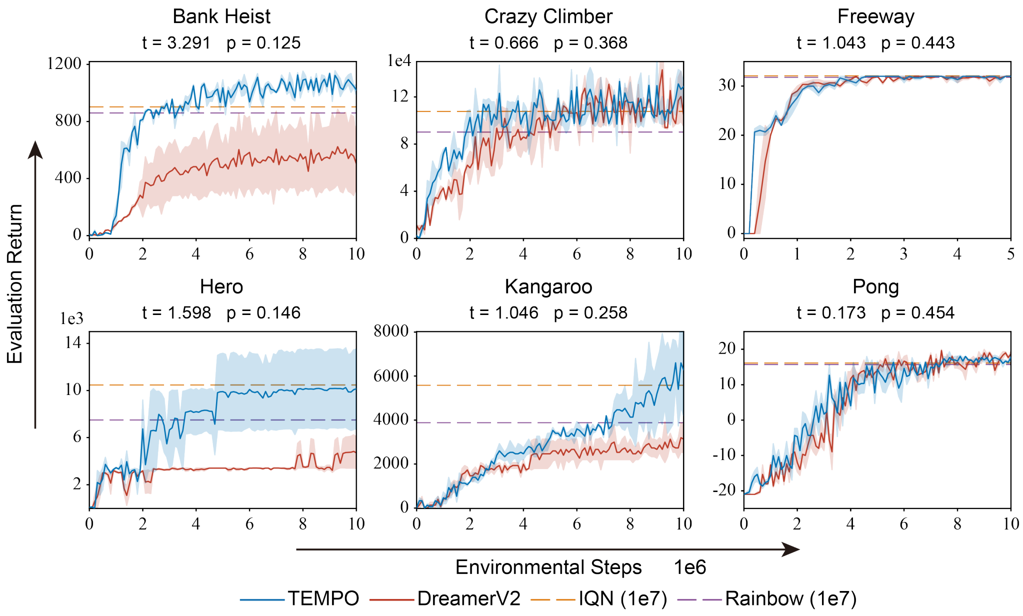

Figure 2: Evaluation of TEMPO on discrete control tasks from Atari video games. The lines show mean scores and the shaded areas show the standard deviation across 3 random seeds.

F Training curves

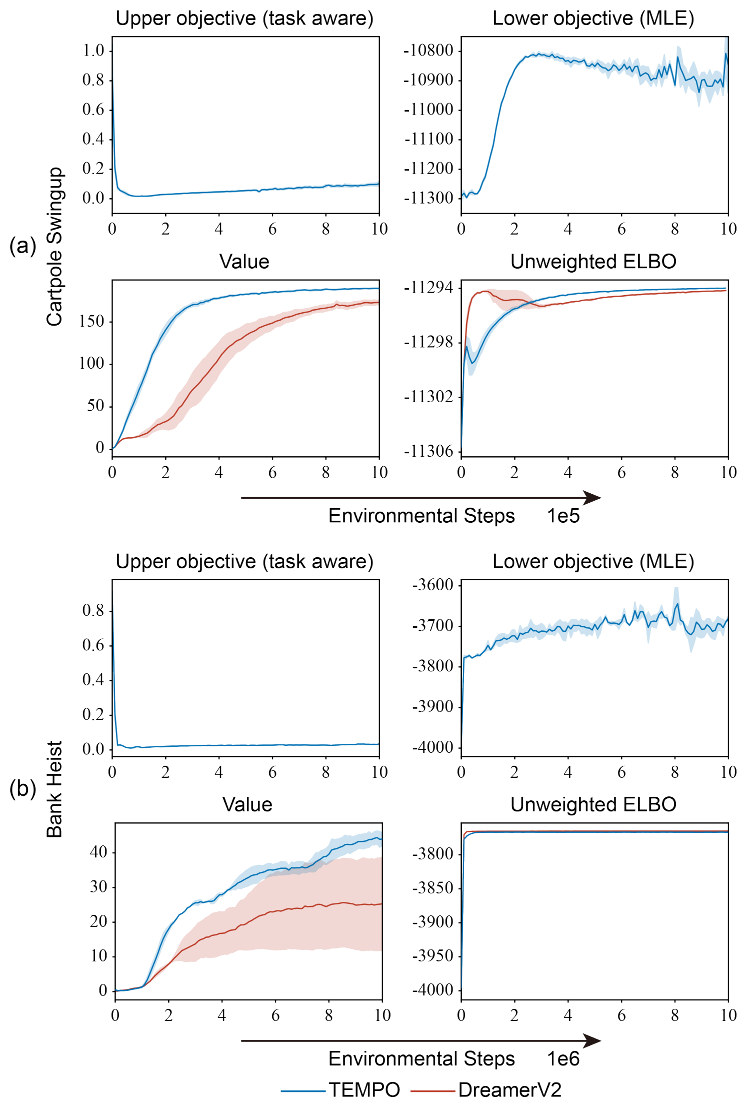

Figure 3: Training curves of TEMPO on (a) Cartpole Swingup from DeepMind Control Suite and (b) Bank Heist from Atari, including curves of value estimation, upper objective, lower objective, and unweighted ELBO. The lines show means and the shaded areas show the standard deviation across 3 random seeds.

G Sample weights

Set	**Type**	**Weights ($w_{1:50}$)**									**Pearson (r)**
1	meta	[1.18	1.20	1.20	1.16	1.20	1.04	0.99	0.95	...]	-0.20
	naive	[1.57	0.27	0.35	1.16	0.52	1.07	0.43	0.78	...]	
2	meta	[1.17	1.06	1.08	1.13	0.97	0.93	1.03	1.16	...]	-0.09
	naive	[0.29	1.17	0.68	0.43	0.35	1.67	1.70	0.70	...]	
3	meta	[1.07	1.09	0.88	0.80	1.10	0.91	0.94	0.93	...]	0.29
	naive	[2.69	4.74	1.42	1.14	2.39	0.49	0.44	0.39	...]	
4	meta	[1.09	1.13	1.01	1.17	1.14	1.09	0.92	0.96	...]	0.02
	naive	[0.27	0.28	0.79	0.27	0.34	0.28	0.57	0.32	...]	

Table 2: 4 sets of meta weights and corresponding naive weights after $5e5$ steps of training in DMC Cartpole Swingup. The Pearson correlation coefficients show that there is no apparent linear relation between the two types of weights.