A APPENDIX

A.1 DREAMERV2

Hafner et al. (2020) makes several additional changes to the framework that are found to improve performance on the Atari environment. First, instead of using a continuous stochastic hidden state, a discrete state is used. Second, straight-through gradients (Bengio et al., 2013) are used to differentiate through the discrete states and actions. Due to the bias induced by the straight-through estimator, REINFORCE gradient or a mixed gradient of REINFORCE and the dynamics backpropagation is used. Lastly, they use KL balancing, separately scaling the prior cross entropy and the posterior entropy in the KL loss.

A.2 TRANSDREAMER LOSS FUNCTION

We optimize the following objective, which is the negative ELBO of the action conditioned model with additional terms for predicting the reward and discount,

$$\mathcal{L}_{\text{TSSM}}(\phi) = \sum_{t=1}^{T} \Bigg(\mathbb{E}_{\prod_{\tau=1}^{t} q_\phi(z_\tau|x_\tau)} [-\eta_x \ln p_\phi(x_t|h_t, z_t) - \eta_r \ln p_\phi(r_t|h_t, z_t) - \eta_\gamma \ln p_\phi(\gamma_t|h_t, z_t)] \\ + \mathbb{E}_{\prod_{\tau=1}^{t-1} q_\phi(z_\tau|x_\tau)} \left[D_{\text{KL}} \left[q_\phi(z_t|x_t) \parallel p_\phi(z_t|z_{1:t-1}, a_{1:t-1}) \right] \right] \Bigg).$$

Here, η_x, η_r, and η_γ are hyperparameters used to scale the loss terms. The derivation of the ELBO can be found in the below.

A.2.1 ELBO

The generative model is $p(o_t, z_{1:T}|a_{1:T}) = \prod_t p(o_t|h_t, z_t)p(z_t|z_{1:t-1}, a_{1:t-1})$ where $o_t = (x_t, r_t, \gamma_t)$ and $h_t = f_{\text{transformer}}(z_{1:t-1}, a_{1:t-1})$. By approximating the posterior by $q(z_t|x_t)$, a variational posterior is $q(z_{1:T}|o_{1:T}, a_{1:T}) = \prod_t q(z_t|x_t)$. By the importance weighting and Jensen's inequality, we can write as follows:

$$\begin{aligned}
\ln p(o_{1:T}|a_{1:T}) &= \ln \mathbb{E}_{p(z_{1:T}|o_{1:T}, a_{1:T})} \left[\prod_{t=1}^{T} p(o_t|h_t, z_t) \right] \\
&= \ln \mathbb{E}_{q(z_{1:T}|o_{1:T}, a_{1:T})} \left[\prod_{t=1}^{T} p(o_t|h_t, z_t)p(z_t|z_{1:t-1}, a_{1:t-1})/q(z_t|x_t) \right] \\
&\geq \mathbb{E}_{\prod_{t=1}^{T} q(z_t|x_t)} \left[\sum_{t=1}^{T} \ln p(o_t|h_t, z_t) + \ln p(z_t|z_{1:t-1}, a_{1:t-1}) - \ln q(z_t|x_t) \right] \\
&= \sum_{t=1}^{T} \Bigg(\mathbb{E}_{\prod_{\tau=1}^{t-1} q(z_\tau|x_\tau)} [\ln p(o_t|h_t, z_t)] \\
&\quad - \mathbb{E}_{\prod_{\tau=1}^{t-1} q(z_\tau|x_\tau)} \left[D_{\text{KL}} \left[q(z_t|x_t) \parallel p(z_t|z_{1:t-1}, a_{1:t-1}) \right] \right] \Bigg) \qquad (1) \\
&= \sum_{t=1}^{T} \Bigg(\mathbb{E}_{\prod_{\tau=1}^{t-1} q(z_\tau|x_\tau)} [\ln p(x_t|h_t, z_t) + \ln p(r_t|h_t, z_t) + \ln p(\gamma_t|h_t, z_t)] \\
&\quad - \mathbb{E}_{\prod_{\tau=1}^{t-1} q(z_\tau|x_\tau)} \left[D_{\text{KL}} \left[q(z_t|x_t) \parallel p(z_t|z_{1:t-1}, a_{1:t-1}) \right] \right] \Bigg) \qquad (2)
\end{aligned}$$

where $p(o_t|h_t, z_t) = p(x_t|h_t, z_t)p(r_t|h_t, z_t)p(\gamma_t|h_t, z_t)$.

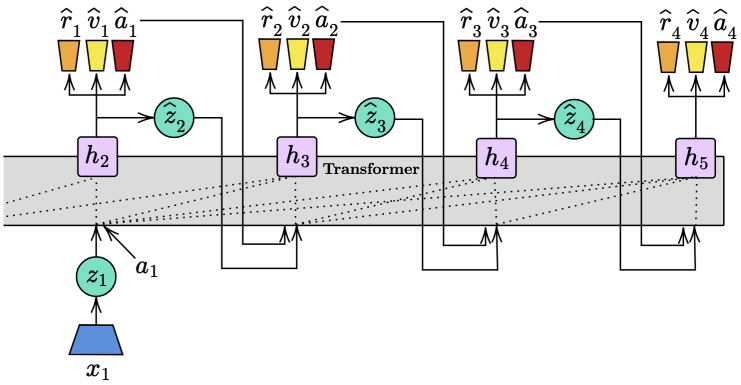

Figure 6: Transfomer-Based Trajectory Rollout for Actor Critic Learning.

A.3 DMC AND ATARI

As written in Sec. 5.5, we used almost identical configurations with Dreamer and DreamerV2 by referring to the configuration file in https://github.com/danijar/dreamerv2 (e.g., action repeat and training World Model and policy every 5 steps for DMC). The one configuration we did modify is the number of imagined trajectories, which is not configurable in Dreamer, but is necessary in TransDreamer because imagining from every state in the batch requires too much computational resources. We control this through a hyperparamter that limits the number of imagined trajectories per training sample. For DMC and Atari, we use 3 imagined trajectories per sample.

Several other hyperparmeters are specific to the TSSM. These include: whether or not to use gating or identity map reording as is done in GTrXL (Parisotto et al., 2020), the number of layers and heads to use for the transformer, the size of the hidden state for the MLP in the transformer, and whether or not to use relational positional embedding (Dai et al., 2019). For DMC and Atari, we generally use 2-layer Transformer (Vaswani et al., 2017) without dropout, gating, or identity map reordering. One exception is for Atari Pong where we did find identity map reordering performed better. We use 10 heads in the Multihead Attention and the dimensions of the hidden state for the MLP and Attention are 200 for DMC and 600 for Atari. These are the same as the dimensions used in the deterministic state in DreamerV2.

A.4 HIDDEN ORDER DISCOVERY

For 2D and 3D Hidden Order Discovery tasks, we measure the model in two aspects, the ability to deal with complex memory-based reasoning and the ability to extract long-term knowledge. Thus we design tasks either with an increased number of objects or the distances between any two objects or both. Specifically, on 2D tasks, we increase the number of balls from 4 to 6, while not changing the distance. The distance here is measured as the number of cells between any two balls. We sum the absolute difference along the x-axis and y-axis as the distance between any two balls. To control the long-term dependency, a threshold of 2 is applied to the distance of balls, i.e. the minimum distance between any two balls should not be less than 2 cells. For 3D tasks, we tested not only the reasoning complexity but also the ability to capture long-term dependency. For reasoning complexity, we compare 5-Ball Dense with 4-Ball Dense. For long term dependency, we compare 4-Ball sparse against 4-Ball Dense. The sparse setting has a larger distance between any two balls. We use the Euclidean distance as a measure of distance. Any two balls have a distance at least 4 units in the sparse setting, while in the dense setting, it is 2 units. 1 unit equals 1 ball size (diameter). Thus, in sparse setting, the distance between any two balls is at least 3 ball-size. For each task, the maximum steps for an episode is set as 100.

We implemented a 6-layer transformer with identity map reordering as TSSM for both 2D and 3D Hidden Order Discovery tasks. During imagination, only one state was randomly sampled as the starting state for imagination. We imagined till the trajectory's max step was reached. Empirically we found concatenating the intermediate output of attention layers together as h_t accelerates the converge speed, so we applied this during experiments. Other hyperparameter configurations are kept the same as DreamerV2 crafter configuration, see https://github.com/danijar/crafter/issues/1

for details. For DreamerV2, we use the same configuration as DreamerV2 for crafter, except that we imagined 30 steps for agent learning.

As explained in the paper, to help train the world model better, we used a prioritized replay buffer for Dreamer and TransDreamer with $\alpha = 0.5$. The sampling probability for each trajectory is set at the return of this trajectory divided by the overall return of the whole data buffer collected till that time, thus a trajectory with higher rewards will have a higher chance to be sampled. The rest 50% of the batch are sampled uniformly from the whole data buffer.

Table 3 shows the ratio of successfully completing at least one round of the hidden order on all the 2D and 3D tasks. We deploy the well-trained agent for each task to collect 1000 trajectories and count the ratio in the whole trajectories. As we can see, TransDreamer performs better than Dreamer by this metric. Note that the episode length is limited to 100, and succeeding in this task in 100 steps is difficult. For example, in the 4-Ball case, the chance of guessing the right order is 0.04 (1 over 4!), and the agent needs to start collecting from the first ball when it collects an incorrect ball. Therefore the agent needs to start over again and again during exploration. When increasing the number of balls from 4 to 5, the chance of randomly guessing the order decreases by a factor of 5. We can observe this relation approximately on TransDreamer's performance, $23\% \rightarrow 5\%$, while Dreamer nearly fails.

Table 3: Success Rate for Complete Order Visitation

Task	2D Object Room			3D Object Room		
	4-Ball	5-Ball	6-Ball	4-Ball Dense	4-Ball Sparse	5-Ball Dense
TransDreamer	**23%**	**5%**	**1%**	**18%**	**11%**	**4%**
DreamerV2	7%	0%	0%	10%	1%	0 %

A.4.1 Full Quantitative Results

To compute the foreground MSE, we use Unity (Juliani et al., 2020) to render a foreground image, Figure 7, and from the foreground image, we infer a binary foreground mask to filter out the background from the predicted image. The full MSE result is reported in Table 4. As can be seen, more than half of the overall MSE gap between TransDreamer and Dreamer happens in the foreground. For example, in the 4-Ball Dense, 60 context setting, the overall MSE gap between TransDreamer and Dreamer is 119.8, while 70.7 of the error occurs in the foreground.

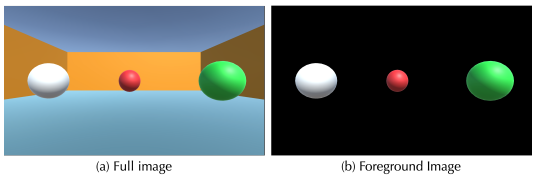

Figure 7: Image from Unity Foreground Camera

Table 5 shows the reward prediction accuracy on both zero-reward and nonzero-reward timesteps for the 3D tasks. As mentioned in the paper, to measure prediction accuracy for +3 reward case, we classify it by labeling 3 ± 0.3 as positive. For 0 reward case, we classify it by labeling ± 0.01 as positive. We can see that TransDreamer outperforms Dreamer by a large gap on nonzero-reward in the 4-Ball Dense and the 5-Ball Dense. Both models perform well generally on zero-reward. In 4-Ball Sparse setting, The gap is smaller, we hypothesis that it is because in the sparse setting, the foreground balls are seen less frequently.

Table 4: Image Generation MSE

Task	Model	60 contexts / 40 targets		70 contexts / 30 targets		80 contexts / 20 targets	
		Overall	Foreground	Overall	Foreground	Overall	Foreground
4-Ball Dense	TransDreamer	**458.0**	**211.2**	**281.9**	**133.1**	**146.0**	**69.8**
	DreamerV2	577.8	281.9	380.0	194.2	206.2	110.8
4-Ball Sparse	TransDreamer	**448.8**	**195.5**	**261.4**	**115.2**	**128.1**	**56.8**
	DreamerV2	462.6	215.8	279.7	138.6	145.1	72.4
5-Ball Dense	TransDreamer	**516.0**	**245.2**	**329.9**	**163.1**	**167.4**	**85.0**
	DreamerV2	605.1	300.9	413.8	217.0	231.6	124.9

Table 5: Reward Prediction Accuracy

Task	Model	60 contexts / 40 targets		70 contexts / 30 targets		80 contexts / 20 targets	
		Zero	Non-zero	Zero	Non-zero	Zero	Non-zero
4-Ball Dense	Transdreamer	**94.9**	**46.9**	**94.7**	**53.2**	**95.4**	**73.2**
	DreamerV2	93.7	28.2	93.6	34.6	94.2	50.5
4-Ball Sparse	Transdreamer	**96.4**	**32.4**	96.0	**36.5**	**96.6**	**48.6**
	DreamerV2	95.6	32.0	**96.2**	33.3	95.5	42.3
5-Ball Dense	Transdreamer	**92.5**	**17.7**	**93.2**	**18.1**	**93.3**	**32.35**
	DreamerV2	91.1	9.8	92.3	6.2	92.4	15.3

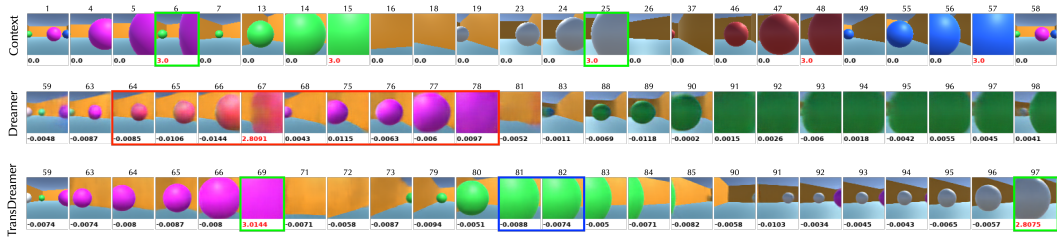

Figure 8: Imagined trajectories comparison between DreamerV2 and TransDreamer given same context

A.5 World Model Imagination with Same Context

Different from Figure 4, in Figure 8, we illustrated imagined trajectories from TransDreamer and Dreamer given the same contexts. This 5-Ball Dense sample is collected from Dreamer's agent learning process, so for TransDreamer, it is an out of distribution sample. This is the same context given to Dreamer in Figure 4. Despite being an out of distribution sample, TransDreamer can still correctly imagine the balls and predicts rewards for the purple and white balls (Green box) correctly. However, it does make a mistake predicting the reward for the green ball (blue box). Nevertheless, even in this setting, the quality of imagination in TransDreamer is better than Dreamer.