Multi-step Decision Aware Dreamer

Maxwell Bridgewater **Mishaal Kandapath** **Sophia Lee** **Devan Srinivasan**

Abstract

To plan in complex environments under the Reinforcement Learning paradigm, RL agents are equipped with world models to simulate the future and learn efficiently. These World Models are most effective when projected into a compressed latent space where only the most important information in the environment must be learned for planning. Our report focuses on enhancing one prominent latent World Model method, DreamerV2. Recognizing a few potential inefficiencies in Dreamer, we experiment with integrating decision-aware methods into the base DreamerV2 architecture. Additionally, we train the latent space to be informative of multiple steps into the future. We test our full architecture against the baseline DreamerV2 in the Crafter environment, and follow up with several ablation experiments. The results suggest that decision-aware agents are less capable of pursuing long-term goals. The agents trained to optimize the multi-step predictiveness of the latent World Model, however, performed significantly better than the base DreamerV2 (MStep ~ 8.4, DreamerV2 ~ 6).

1 Introduction

In order to intelligently act, agents must be able to navigate complex environments with sparse feedback, just as people do in real life. Particularly, one challenge is learning to act towards rewards across long-horizons. In these cases, the agent needs to learn to anticipate the dynamics of the world itself rather than simply react to current information. In Reinforcement Learning, this process is simulated by learning environment dynamics, called a World Model, and the approach is termed model-based reinforcement learning (MBRL). [1] However, the exact environment is typically high dimensional, where predicting the full observations becomes an intensive operation. To bypass this, an agent can learn and plan across latent representations of the environment for a compressed world model. In this approach, agents that are able to create more informative compressed representations will be more successful in generalizing across complex and long-horizon tasks.

The effectiveness of this latent planning approach was shown in the Dreamer architectures [6] [7] [8]. DreamerV2 achieved human-level performance on ATARI tasks, the first model-based agent to outperform the top model-free agents, and do so with a light computational budget. The Dreamer architectures are critically able to optimize actions by simulating future events across longer horizons due to the manageable latent space size. This is not possible without a world model, suggesting that this approach is the way forward for control tasks. However, DreamerV2 relies only on single-step image reconstruction loss and no signals from value prediction, both of which contain valuable information for planning. As such, we selected DreamerV2 as an architecture to improve off of, particularly focusing on methods to improve the quality of the world model's latent representations.

Decision-aware methods theoretically assert that, provided use of latent representations, world models can be trained on scalar losses to converge to the optimal solution [11]. Although decision-aware methods perform inconsistently in practice, these may be compatible with Dreamer's latent world

[1] Unless specified otherwise, this report uses the term "model" to only refer to the world model for consistency.

Preprint. Under review.

model and allow DreamerV2 to incorporate decision-aware loss for performance enhancement.

Long-term planning in rich and diverse environments remains a challenging problem for the RL paradigm. In this report, we modify the base DreamerV2 architecture and study its performance in long-term planning goals. In addition to decision-aware loss, we introduce a novel objective that explicitly conditions latent representations to hold information about the future. We reason that such a world model should better anticipate the states in the future rather than just focusing on learning the present. We test the comprehensiveness of our experiments on a complex, long-term planning game called Crafter [5].

Contribution 1- Decision-Aware Dreamer: Train world models that are informed by the value

- We encode value information into the world model's latent states during training by exploiting the connected computation graphs between the World Model and Actor-Critic Module.
- In considering the literature on the decision-aware reinforcement learning framework, we use a theoretically supported decision-aware MBRL method, iterVAML, for the same [4].

Contribution 2- Multistep Prediction: Representations that embody information about the future are more apt for future planning

- We integrate multi step information into the hidden states by training individual hidden states to be reconstructable several observations into the future.

2 Prior Work

As opposed to model-free reinforcement learning methods, MBRL seeks to leverage domain knowledge in planning. However, despite their perceived ability for directed exploration [10], generalizing to novel tasks [2], model-based agents have had difficulty comparing to SOTA model-free agents. Within progress towards the same, we draw direct motivation from the following:

The primary background for the work in this report is DreamerV2 [7], a single-GPU agent that achieves human level performance on the ALE benchmarks [1] that learns the dynamics model entirely from scratch. The main recurrent component of DreamerV2 World Model is as follows:

$$\begin{aligned} \textbf{Recurrent Model: } & h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}) \\ \textbf{Representation Model: } & z_t \sim q_\phi(z_t | h_t, x_t) \\ \textbf{Transition Prediction Model: } & \hat{z}_t \sim p_\phi(\hat{z} | h_t) \end{aligned} \tag{1}$$

where a refers to an action, h is the hidden state of the RNN involved, x is an image observation, z is a state representation (referred to as *posterior*) representing the distribution of current state informed using previous states, previous actions, as well as the current observation, whereas $\hat{z}$ refers to a state representation (referred to as *prior*) representing the distribution of the current state informed by previous states and actions. The prior is used in planning in latent spaces since it does not require any ground truth image observations and thus can be easily unrolled, whereas the posterior is used to train the priors during the training phase (see Section 3 for details) with information from ground truth observations. DreamerV2 showed that model-based learning with latent representations achieve strong results that outperform carefully engineered model-free agents on challenging RL tasks. As an extra and much appreciated perk, DreamerV2 did all this under modest computation requirements: a single NVIDIA V100 32GB GPU.

Another influential line of work in MBRL is that of decision-aware model learning (DAML) [11], which holds that good models are those that estimate value correctly. The paradigm does so by passing value gradients to inform world model representations and can ideally reduce/extinguish the role of reconstruction (and/or similar objectives) in training latent representations. It's loss functions involve differences between value estimates (for more details, see Section 3 for the iterVAML loss function). Two most prominent approaches in this space are iterVAML [4] and MuZero [9]. MuZero value loss has been successfully deployed in a variety of works, but according to [11]

although it can achieve low error in stochastic environments, MuZero can never recover the optimal value function w.r.t its loss function even with the correct model. On the other hand, provided that iterVAML operates on latent state representations under some more theoretical assumptions, iterVAML converges to the optimal value function under its loss.

As such, the incorporation of decision aware objectives in MBRL shows much promise in learning better models for planning across challenging RL tasks.

3 Method

3.1 Data

We found that the game Crafter was well suited for our goals [5]. Crafter is a smaller version of Minecraft, that requires agents to succeed in a variety of long-term objectives and generalize over randomized terrains. Environment observations are 64x64x3 images, which are manageable for our limited computational resources (Fig 1).

Crafter has 22 achievements (see Fig 2) with varying levels of difficulty. Rewards are sparse, forcing the agent to discover new achievements: (+1) for unlocking an achievement for the first time in an episode, (-0.1) when a health point is lost, (+0.1) when a health point is gained (health points $\in [0, 9]$). Furthermore, evaluation on Crafter is limited to 1M steps and thus prioritizes sample efficiency and not computational capacity. Success in Crafter is indicative of strong generalization, deep exploration, good latent representations, and long-term planning [5].

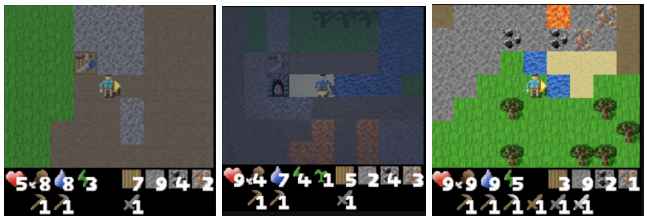

Figure 1: Crafter Observations

3.2 Base Architecture

We build upon the DreamerV2 architecture, as implemented in [7][2][3].
DreamerV2 consists of a World Model that has a Recurrent State Space Model (RSSM) [7] (see Section 2 for details). Notably, the posterior is trained by feeding in the posterior to an image decoder trained towards outputting the original observation. Furthermore, the World Model consists of MLPs to predict reward at a state and discount factor based off the latents. In total, the DreamerV2 loss function amounts to:

$$L(\phi) = E_{q_\phi(z_{1:T}|a_{1:T},x_{1:T})}(\sum_t -\ln p_\phi(x_t|h_t,z_t) - \ln p_\phi(r_t|h_t,z_t) - \ln p_\phi(\gamma_t|h_t,z_t) + \beta KL[q_\phi(z_t|h_t,x_t)||p_\phi(z_t,h_t))) \quad (2)$$

The loss aims to maximize the log-likehood of dataset observations (images, rewards, and discount factors).Additionally, equation 2 contains the ELBO of a hidden Markov model conditioned on an action sequence, which is the KL-divergence term that drives the prior and posterior close (higher weighting w.r.t prior).

Finally, DreamerV2 uses the actor critic framework for planning along the critic loss:

$$L(\epsilon) = E_{p_\phi,p_\psi}(\sum_{t=1}^{H-1}(0.5 * (v_\epsilon(sg(\hat{z}_t)) - sg(V_t^\lambda))^2)) \quad (3)$$

[2]https://github.com/danijar/dreamerv2

[3]See https://github.com/mishaalkandapath/valuedream for our modifications

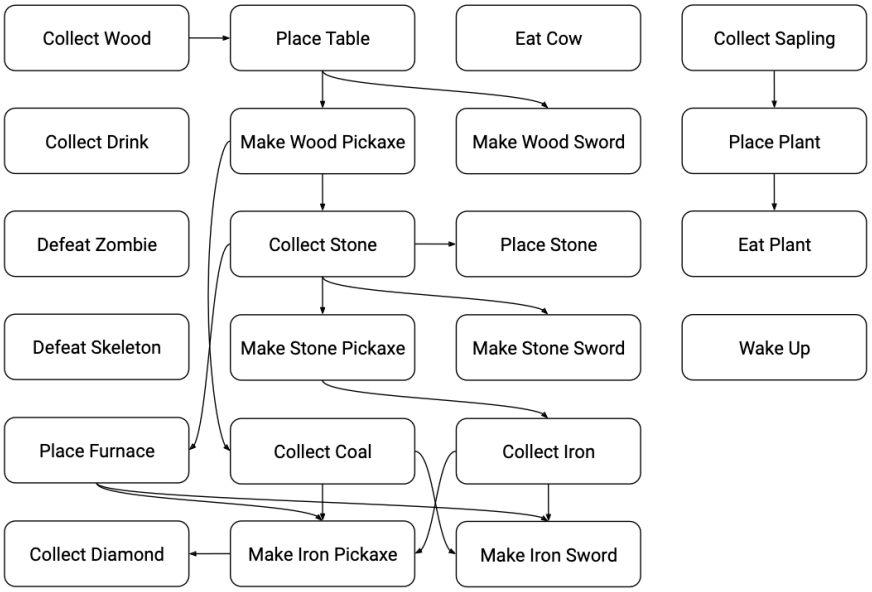

Figure 2: 22 achievements in Crafter, with arrows indicating dependencies of achievements on others. Bottom achievements are the most complex. Adapted from [5]

and actor loss:

$$L(\psi) = E_{p_\phi, p_\psi}(\sum_{t=1}^{H-1} -\ln p_\psi(\hat{a_t}|\hat{z}_t) sg(V_t^\lambda - v_\epsilon(\hat{z}_t)) - \eta H(a_t|\hat{z}_t)) \tag{4}$$

where

$$V_t^\lambda = \hat{r_t} + \hat{\gamma_t} \begin{cases} (1-\lambda) v_\epsilon(\hat{z_{t+1}} + \lambda V_{t+1}^\lambda) & t \leq H \\ v_\epsilon(\hat{z_H}) & t = H \end{cases} \tag{5}$$

where H is the planning in latent space horizon (imagining/dreaming), $\hat{a}$ is the predicted action under the policy, and the actor loss' second term is an entropy regularizer for exploration (see Fig 3).

3.3 Our Modifications

3.3.1 Value-Aware Latent States

Since the actor-critic module never has direct access to observations, performance is highly dependent on the quality of the latent representations learned by the world model. Yet, the world model only receives feedback to improve accuracy in representing the pixel-level environment, without any direct value feedback. For our first modification, we build directly on this observation.

As seen in Figure 3, the actor-critic module is trained off of the latent representations of the world model meaning the computation graphs between the two modules are inherently connected. We experiment by explicitly connecting these computation graphs in training and gradient propagation, allowing the value loss to inform the world model parameters. Thus, the critic loss function as:

$$L(\epsilon, \phi)^4 = E_{p_\phi, p_\psi}(\Sigma_{t=1}^{H-1}(0.5 * (v_\epsilon((\hat{z}_t)) - sg(V_t^\lambda))^2)) \tag{6}$$

We do not expect this approach to work too well since the critic loss only includes information from the priors (see Equation 3) which is poor in the beginning and is subject to adjustments by the posterior. However, if it does, it would indicate that value-signals are rich signals for long-term planning.

[4]Note the omission of the sg on $\hat{z}$

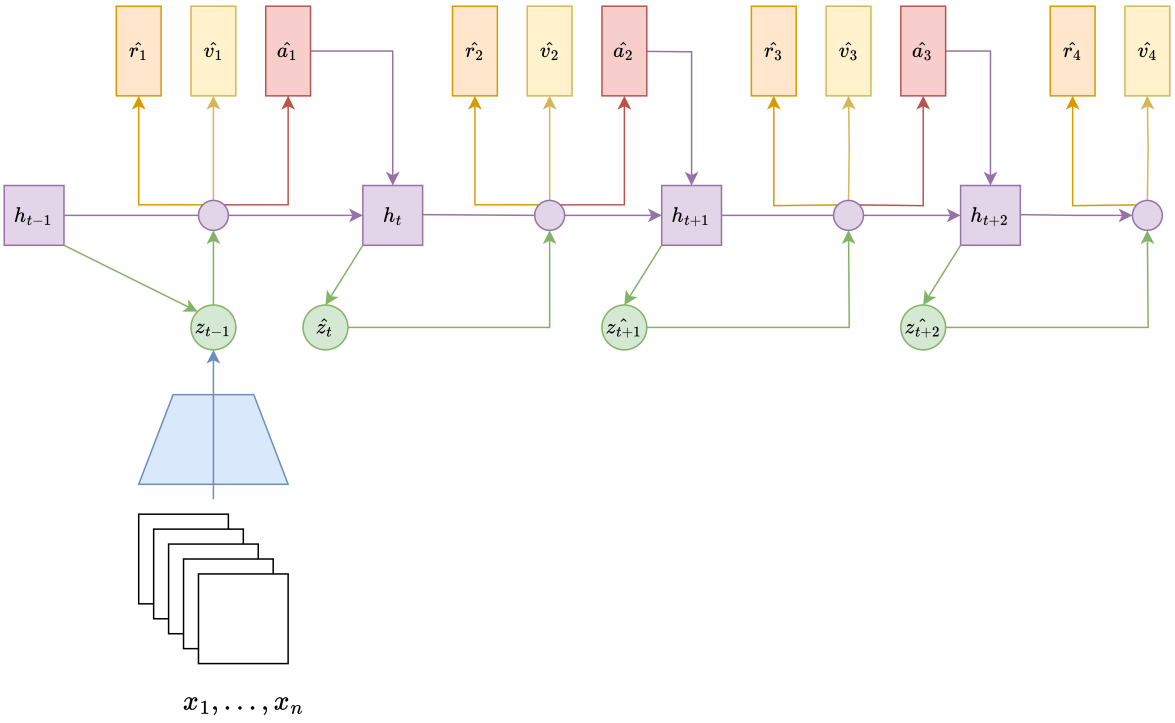

Figure 3: Planning in latent space

3.3.2 Decision-Aware World Model: Latent iterVAML

We next integrate decision-aware model learning. In place of DreamerV2's critic loss (3), we use iterVAML: loss function prominent in the DAML framework [4].

$$L_{\text{iterVAML}}(\epsilon) = \frac{1}{N \cdot n} \sum_{i=1}^{N} \sum_{j=1}^{n} |E_{p_\phi}[v_\epsilon(\hat{z}_t)] - sg(v_\epsilon(z_t))|^2 \tag{7}$$

where N is the batch size and n is the sequence length. This equation enforces the expected value from unrolled latent states to be close to the value of a latent "ground-truth". Aligning with the claim of Voelker et al. (2023), we apply iterVAML in the latent space. $\hat{z}$ is sampled from the prior and z from the posterior.

Although iterVAML has struggled to be empirically validated, this and other decision-aware methods are sensitive to the contexts in which they are used [11]. Specifically, decision-aware approaches have been proposed to perform well in environments with sparse rewards and challenging objectives, consistent with the challenges posed by the Crafter environment.

3.3.3 Multi-step representations

DreamerV2 uses the prior representation to plan in the latent space. The posterior is more expressive than the prior given its direct access to the observation, and it is thus that it is used to train the prior. The current World Model loss function 2 maximizes the posteriors to be predictive on merely the single observation they were generated from, without considering successive states. We postulate that the posterior lacks key information on the future to train optimal priors for latent space planning.

This is well studied in the neuroscience literature. In neuroscience, a cognitive map is a structure that organizes knowledge to enable flexible behaviour, which must represent state along its relevant dimensions [12]. Existing neuroscience models for planning representations in the hippocampus like CANNs and VCOs, use RNNs which consider current velocity and past state to

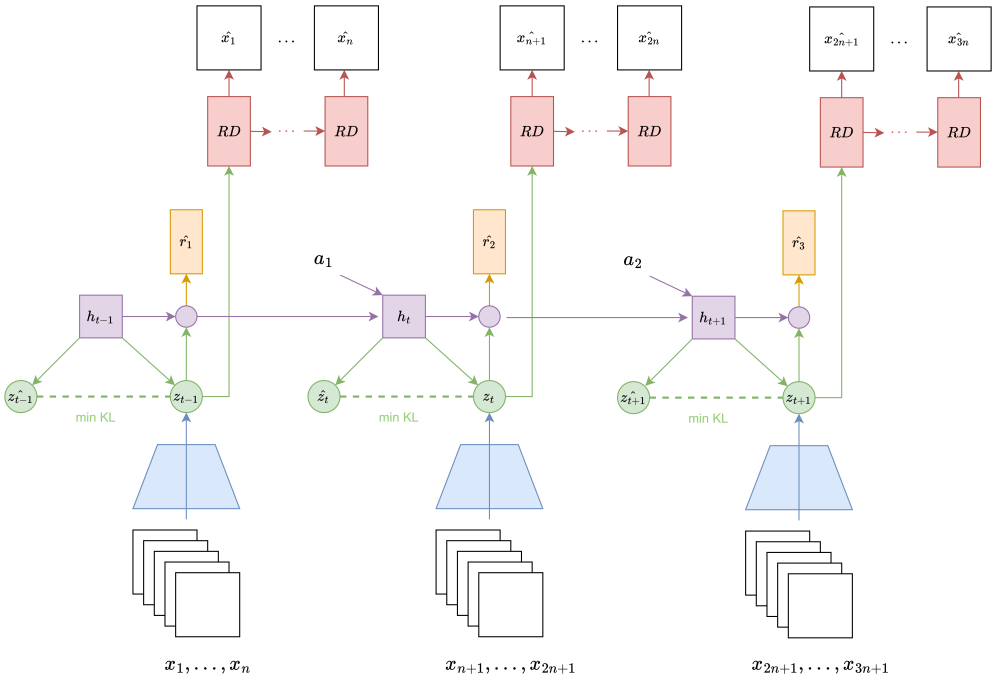

Figure 4: Multi-step World Model. RD: recursive decoder

update all existing state representations, resulting in dynamic representations suited for planning. For example, heading North from South versus heading South from the West, then East, and then North, should finish in similar places. Such multi-step information needs to be integrated in order for you to know that North Bakery is nearby in both cases. As per [12], these models can be easily extended to sensory observations by explicitly ensuring that state representations are predictive of multiple observations. Incorporating such information enhances robustness to environmental perturbations.

We modify the decoder in DreamerV2 to be recurrent, leading to the addition of the following modules:

$$\begin{aligned} j_t &= g_{\phi'}(j_{t-1}, a_{t-1}) \\ \hat{x_t} &\sim p_{\phi'}(\hat{x_t}|j_t) \end{aligned} \tag{8}$$

where ϕ' is in addition to ϕ and Equation 8 is used to unroll representations to get subsequent images. The capacity of this RNN is much smaller than the RNN used for latent planning to preserve computational budgets.
The following loss objective is used to replace the first term in 2:

$$\sum_{t'=t}^{t+G-1} -\ln p_{\phi'}(\hat{x_t}|j_{t'}) \tag{9}$$

where $j_{t'=t}$ is a non-linear transformation on z_t and is the start state for the recurrent decoder. See Figure 4 for an illustration of the same.

We expect that applying the multi-step loss to the posterior distribution leads to increased stability and efficient use of information. We expect that this enhanced reconstruction loss will encourage latent representations of observations in neighboring timesteps to be similar, smoothing the latent space. Since the current base architecture has little regularization, multistep reconstruction loss may improve agent performance. This may also potentially lead to stabilizing effects on iterVAML loss due to higher quality latents earlier in training. [11].

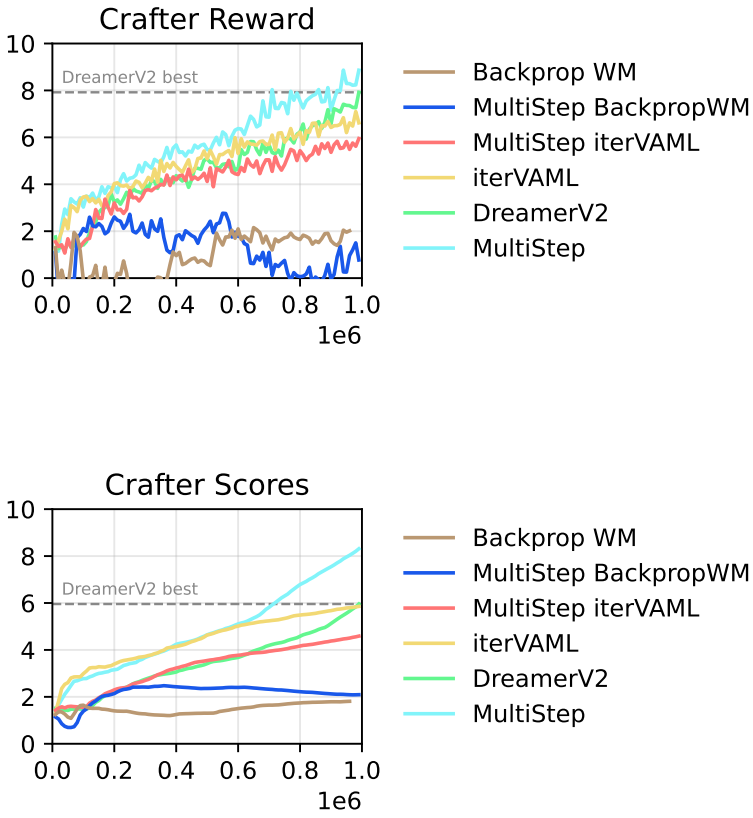

Figure 5: Reward and score over environment steps per agent

4 Experimental Results

We run our modified DreamerV2 architecture on the Crafter environment and compare it against the baseline: DreamerV2 [7]. To gain more insight on to the final results, we run ablation studies testing combinations of our additional components. We also test MultiStep Dreamer, our best performing modified model, and DreamerV2 on an unseen Crafter environment to evaluate explicit generalization.

4.0.1 Training

Each training instance is run for 1e6 steps with the same hyperparameters as done in the original DreamerV2 paper, with the most important configurations being that we pre-filled our replay buffer with 10^4 environment steps, with a batch size of 16, hidden state dimensions of 1024, GRU [3] cells for all our RNNs (as done in [7]), fully connected layers of width 400, and a batch size of 16 (see repository config.yaml file for more parameters). For most of our experiments we use a 24GB NVIDIA GPU (L4 or RTXA5000). For running multi-step dreamer with iterVAML, we used a 48GB NVIDIA A40 due to increased number of Jacobians in the GPU. All experiments were conducted on single GPU machines on a maximum of 10 CPU cores with 90GB of VRAM. [5]

[5]Due to compute limitations, our reported DreamerV2 scores are results from single runs of the DreamerV2 architecture, which is different from the rigorously tested official scores for DreamerV2 on Crafter.

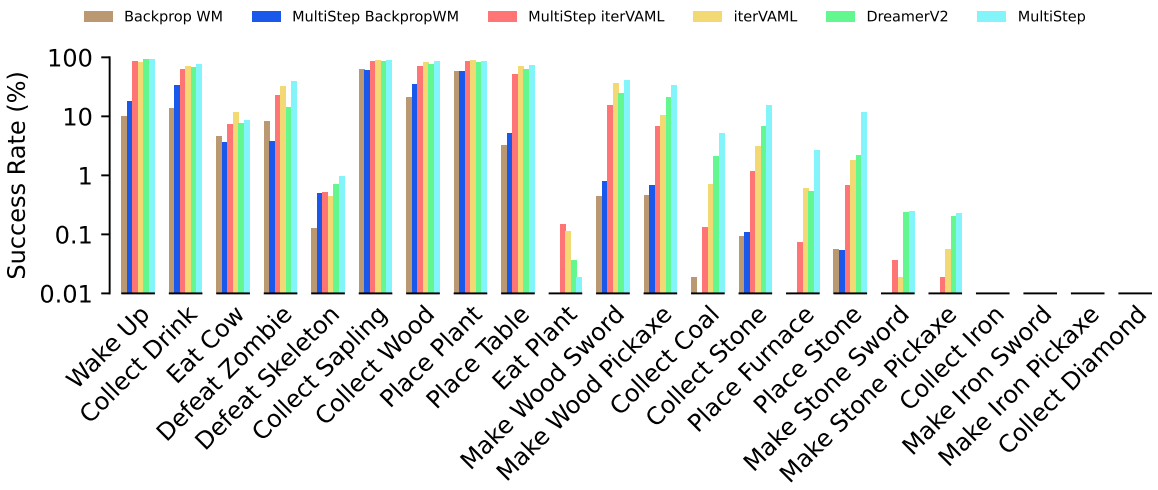

Figure 6: Success rate per achievements, achievements ordered by tree depth (increasing from left to right)

4.1 Evaluation Metrics

To evaluate, we used the given Crafter scoring metric [5]. The primary evaluation metric in Crafter is not the reward but the *score* calculated as:

$$S = \exp(\frac{1}{N}(\sum_{i=1}^{N} \ln(1 + s_i))) - 1 \tag{10}$$

where s_i is the agent's success rate of achieving achievement i across all episodes [5]. We evaluate the performance of DreamerV2 augmented with multi-step latents and decision-aware losses and its individual counterparts against DreamerV2. As per the DreamerV2 evaluation guidelines, we only consider the first $1M$ steps of the agent in all cases when calculating scores.

4.2 Results

Multi-Step Decision Aware Dreamer performs worse than expected with reward ~ 6 and a score ~ 4.6, significantly below most other methods. A breakdown of the achievements (Fig 6) reveals that while it does compare on goals with fewer dependencies and it performs significantly worse as dependencies increase. The score over time (see Fig 5) thus grows approximately linearly with increased training rather than exponential.

Given this performance, we conduct ablation studies on the individual modifications:

Multi-Step Dreamer performs the best of all, with a final score ~ 8 as compared to DreamerV2's ~ 6 and a modest increase in reward. This agent has significantly higher counts for achievements of increased difficulty (see Fig 2 pickaxe, furnace, and stone related achievements), demonstrating its superior grasp of long-term planning. This indicates that our multi-step modification worked as expected, successfully capturing future states within each latent and stabilizing the latent space.

Decision-Aware Dreamer's (iterVAML) performance offers insights into why Multi-Step Decision Aware Dreamer performed worse than expected. We see the scores curve (Fig 5) follows the same linear pattern and similar distributions in the achievement breakdown (Fig 6) as Multi-step Decision Aware Dreamer. Consistent across both Decision-Aware agents, this suggests that the iterVAML loss itself resulted in agents who were unable to prepare for more long-term goals while the achievements that required less anticipation and long-term planning were consistently reached. Like this, we see that the value function decision-aware models converge to is different from that

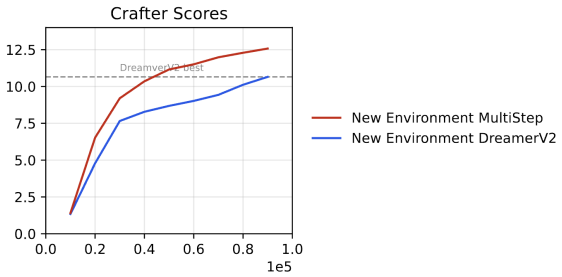

Figure 7: Scores over time for Crafter with a new environment seed

which multi-step dreamer is converging to, suggesting that iterVAML leads to training the critic to only look at short-term goals and converges pre-maturely (see figures 11, 8, 9 and 10 in Section A.2

World Model Value Backprop approaches did not perform well since the mere gradient connection between the critic loss and world model does not provide much additional information for the model to learn from. The target value function in DreamerV2 (see Section 3.2 for details), is bootstrapped from its own reward predictions from the priors. As opposed to iterVAML, where value targets integrate information from the posterior. As such, minimizing value-prediction errors based on priors offered less useful signals than doing so with posterior information as expected.

Robustness to new Crafter environments: Crafter creates environments randomly (random positions of lakes, trees, etc). Here we examine how robust the models are to changing positions of objects in the environment. To do so, we set the same random seed for both agents, different from their original seeds, and test their performance on this new environment setting for 100K steps. Given performance of the models in the first experiment, we opt to only evaluate base Dreamer and Multi-step Dreamer for this experiment due to compute limitations. In Fig 7, we see that multi-step Dreamer validates our expectation of predictive multi-step latent states being robust to environmental perturbations.

5 Discussion

We present Decision Aware DreamerV2 with Multi-Step Latents to study the impact of decision aware losses and increased expressivity in latents in improving the planning ability and efficiency of DreamerV2. From performance difference between our BackpropWM agents and iterVAML agents, we see that the information in the posteriors heavily drive the performance of Dreamer agents. Value-based gradients do not seem to be capable of capturing or making up for that information, so it is important that they be used in tandem with the best latent representation available. These findings are empirically in line with previous theoretical and empirical work well summarized in [11].
However, on deploying iterVAML into the Dreamer agents, they seem to impede performance rather than increase the same, which brings into question their utility for long-term planning environments like Crafter. As covered in Section 4.2, we see that iterVAML fails to think ahead and deploy long-term reasoning to achieve achievements further down the line in Figure 2, even in the presence of more expressive latents as in our multi-step augmentations. Due to the KL-balancing inherent in Dreamer agents (see Section 3.2), it could be that there is a tussle between the new information that the prior has gained from its decision-aware signal and what the posterior is trying to model. An experiment with various KL-Balancing coefficients would reveal the validity of this hypothesis, but unfortunately we were short on compute, money, and time for the same. In either case, further investigation must be done on the adaptability of decision-aware losses to long-term planning environments with proxy rewards (like Crafter).
Finally, we see that our multi-step augmentation satisfied its predictions for latents with increased expressivity with better long-term planning, and increased robbustness to novel settings. This success is motivation for further work into increasing the expressivity of latents by incorporating long-horizon information within the same. In [12], long-term memory is said to play a key role in human abilities to learn and generalize efficiently. Memory is said to deployed and updated actively during planning in novel environments. The memory present in RNNs are more akin to working memory and the increased performance of our multi-step latents suggest that some limited form of memory (in so far

as storing the next 5 observations go) may be emerging from the multi-step objective. Thus, the incorporation of explicit long-term memory nodes for RL problems is a promising avenue to explore for improvements in planning.

In conclusion, our work demonstrates that multi-step objectives for latent representations in planning for RL is an encouraging direction for better Reinforcement Learning. It also demonstrates that decision-aware losses, specifically iterVAML, has trouble with long-term reasoning in environments even with latents trained on multi-step objectives.

References

[1] M. G. Bellemare et al. "The Arcade Learning Environment: An Evaluation Platform for General Agents". In: *Journal of Artificial Intelligence Research* 47 (June 2013), pp. 253–279. ISSN: 1076-9757. DOI: 10.1613/jair.3912. URL: http://dx.doi.org/10.1613/jair.3912.

[2] Arunkumar Byravan et al. "Imagined Value Gradients: Model-Based Policy Optimization with Tranferable Latent Dynamics Models". In: *Proceedings of the Conference on Robot Learning*. Ed. by Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura. Vol. 100. Proceedings of Machine Learning Research. PMLR, 30 Oct–01 Nov 2020, pp. 566–589. URL: https://proceedings.mlr.press/v100/byravan20a.html.

[3] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL].

[4] Amir-Massoud Farahmand, Andre Barreto, and Daniel Nikovski. "Value-Aware Loss Function for Model-based Reinforcement Learning". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, 20–22 Apr 2017, pp. 1486–1494. URL: https://proceedings.mlr.press/v54/farahmand17a.html.

[5] Danijar Hafner. "Benchmarking the spectrum of agent capabilities". In: *arXiv preprint arXiv:2109.06780* (2021).

[6] Danijar Hafner et al. *Dream to Control: Learning Behaviors by Latent Imagination*. 2020. arXiv: 1912.01603 [cs.LG].

[7] Danijar Hafner et al. "Mastering Atari with Discrete World Models". In: *arXiv preprint arXiv:2010.02193* (2020).

[8] Danijar Hafner et al. *Mastering Diverse Domains through World Models*. 2024. arXiv: 2301.04104 [cs.AI].

[9] Julian Schrittwieser et al. "Mastering Atari, Go, chess and shogi by planning with a learned model". In: *Nature* 588.7839 (Dec. 2020), pp. 604–609. ISSN: 1476-4687. DOI: 10.1038/s41586-020-03051-4. URL: http://dx.doi.org/10.1038/s41586-020-03051-4.

[10] Ramanan Sekar et al. *Planning to Explore via Self-Supervised World Models*. 2020. arXiv: 2005.05960 [cs.LG].

[11] Claas A Voelcker et al. *λ-models: Effective Decision-Aware Reinforcement Learning with Latent Models*. 2024. arXiv: 2306.17366 [cs.LG].

[12] James C. R. Whittington et al. "How to build a cognitive map". In: *Nature Neuroscience* 25.10 (2022), pp. 1257–1272. DOI: 10.1038/s41593-022-01153-y. URL: https://doi.org/10.1038/s41593-022-01153-y.

A Appendix

A.1 Work Distribution

All authors contributed equally to the writing of the report.
Maxwell Bridgewater created the illustrations, made world model value backprop implementations and set up the experiments for the same.
Mishaal Kandapath implemented the multi-step modifications, set up the experiments for the multi-step modifications, and set up the experiments for the generalization task.
Sophia Lee implemented the iterVAML modifications, set up iterVAML-related experiments, and summarized plots for the various experiments.
Devan Srinivasan made world model value backprop implementations, set up the experiments for the same, and organized the Github for easy reproducibility and interpretibility.

A.2 Supplementary Figures

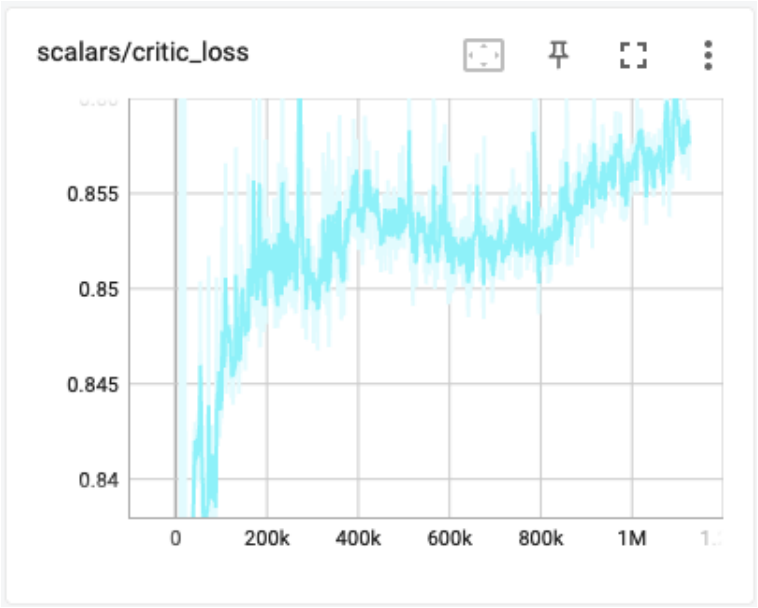

Figure 8: Critic Loss over time for Multi-step Dreamer

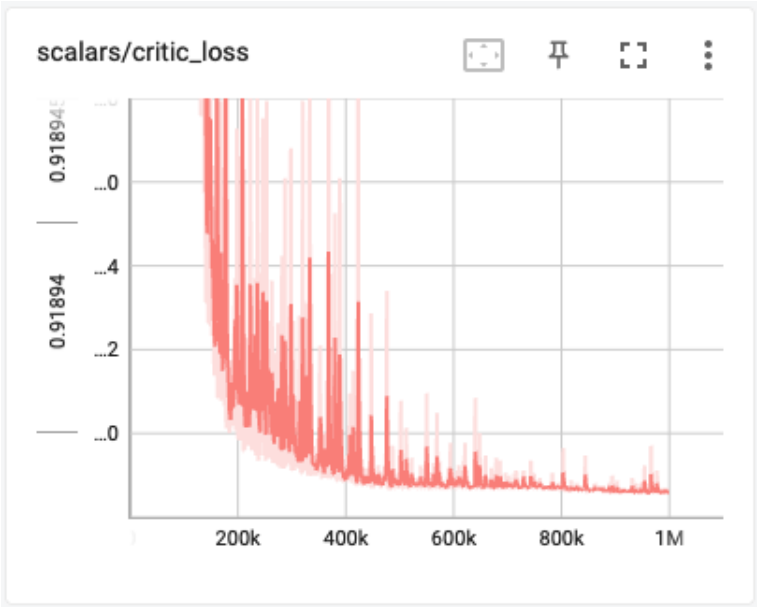

Figure 9: Critic Loss over time for Multi-step iterVAML Dreamer

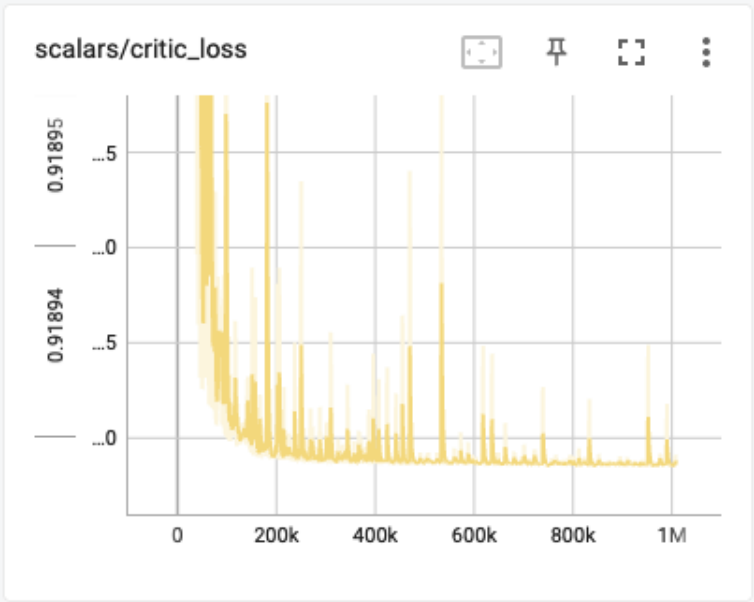

Figure 10: Critic Loss over time for iterVAML Dreamer

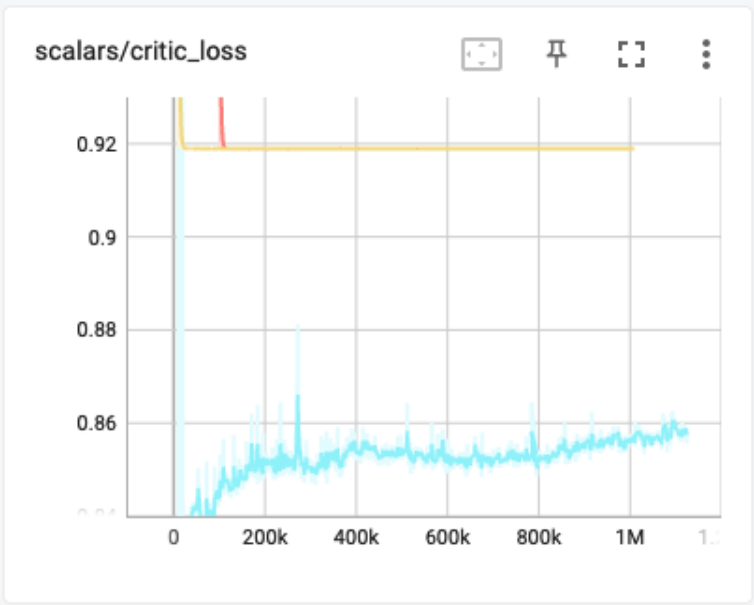

Figure 11: Critic Loss over time for Multi-step Dreamer Multistep iterVAML Dreamer, and iterVAML Dreamer

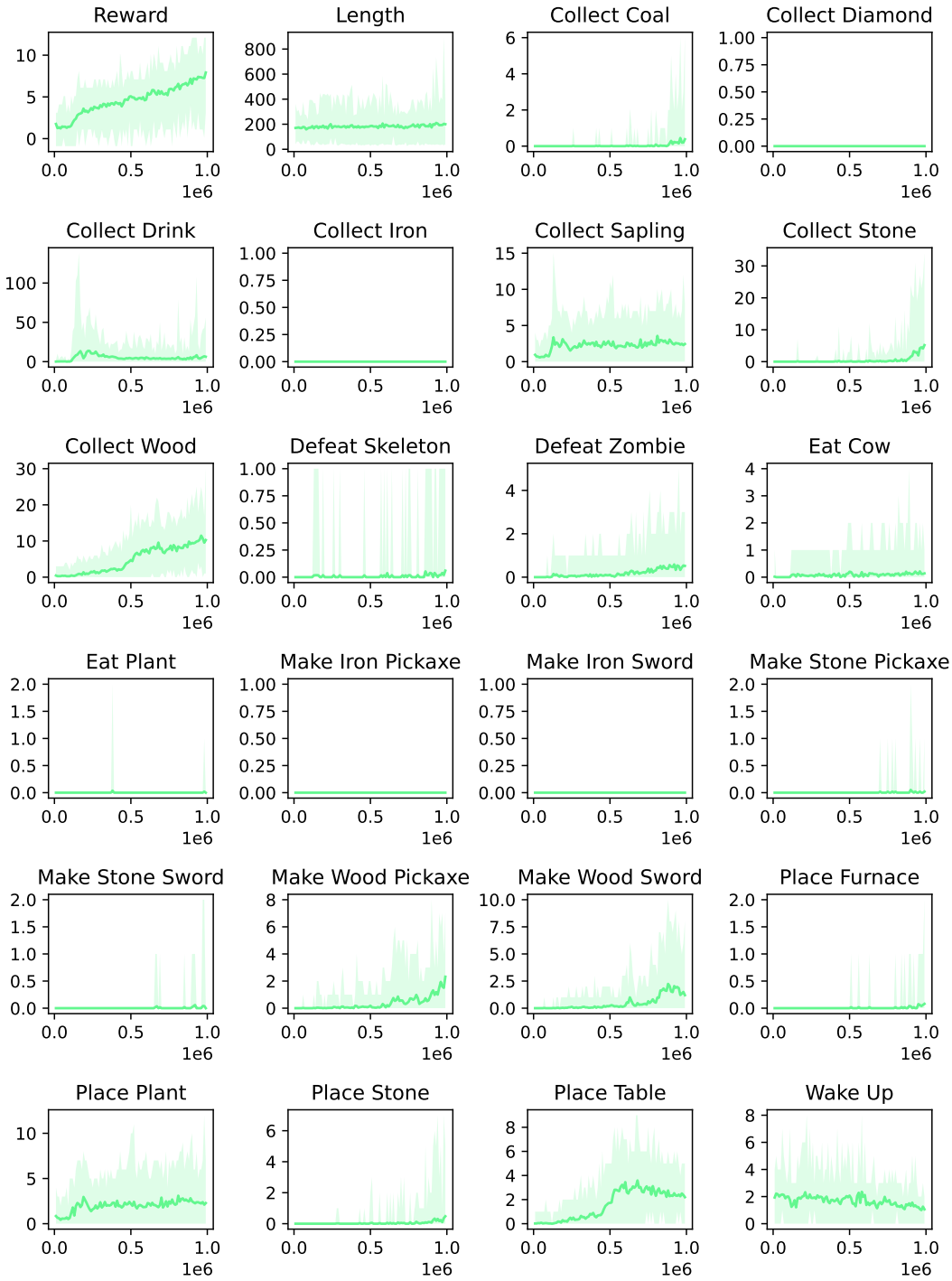

Figure 12: Achievement Statistic over time for Base Dreamer

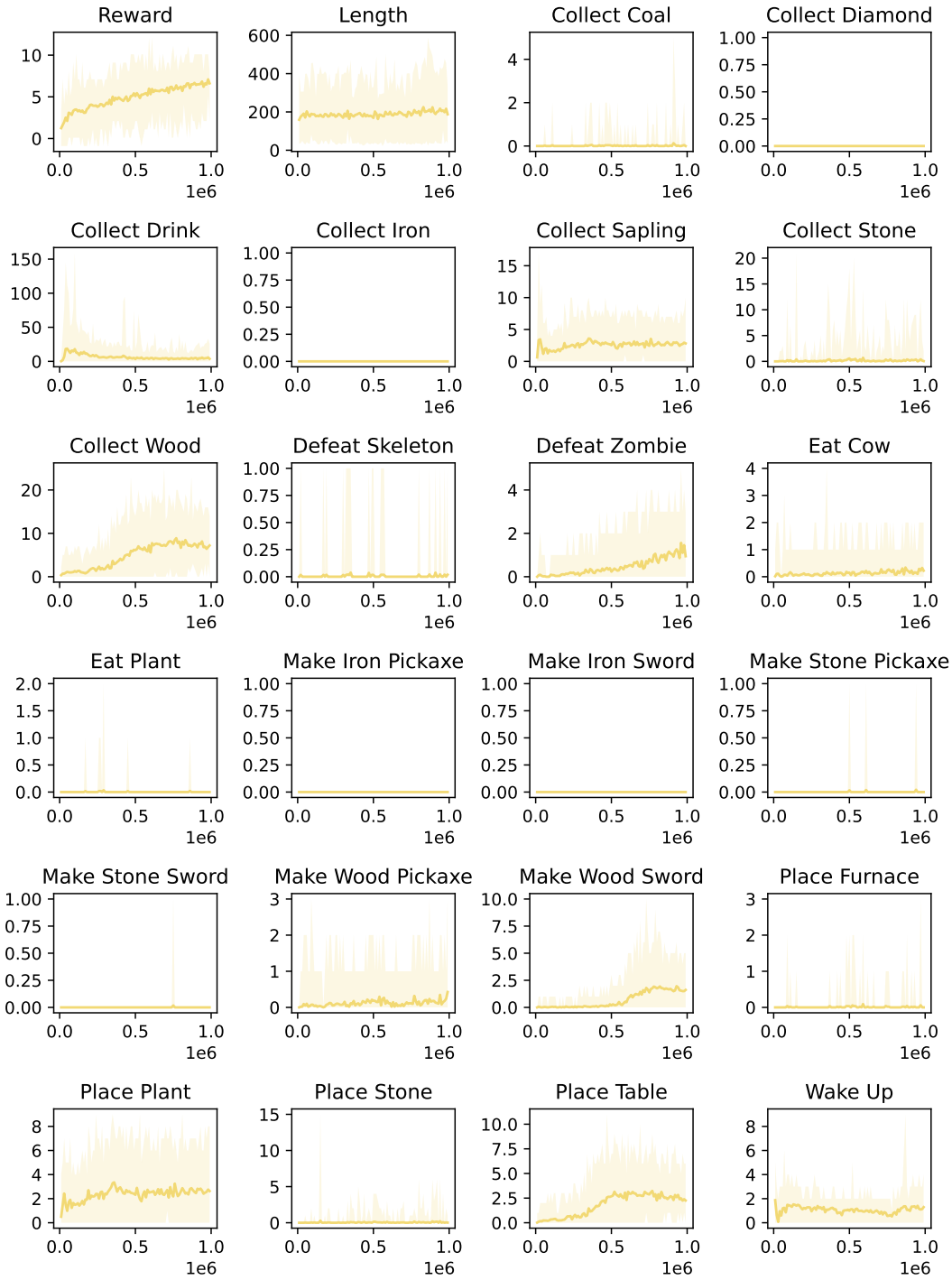

Figure 13: Achievement statistics over time for iterVAML Dreamer

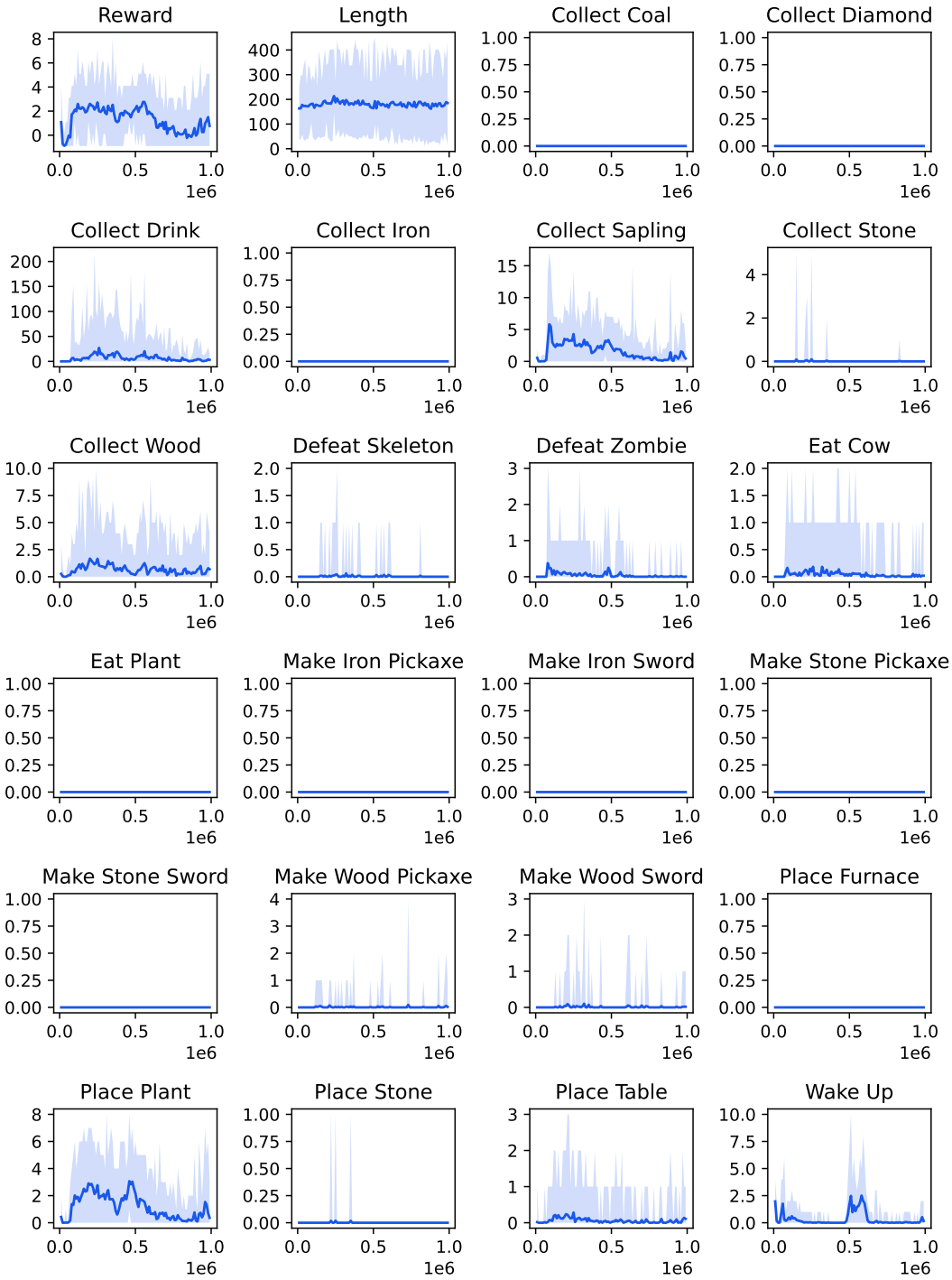

Figure 14: Achievement statistics over time for MStep WM Value Backprop

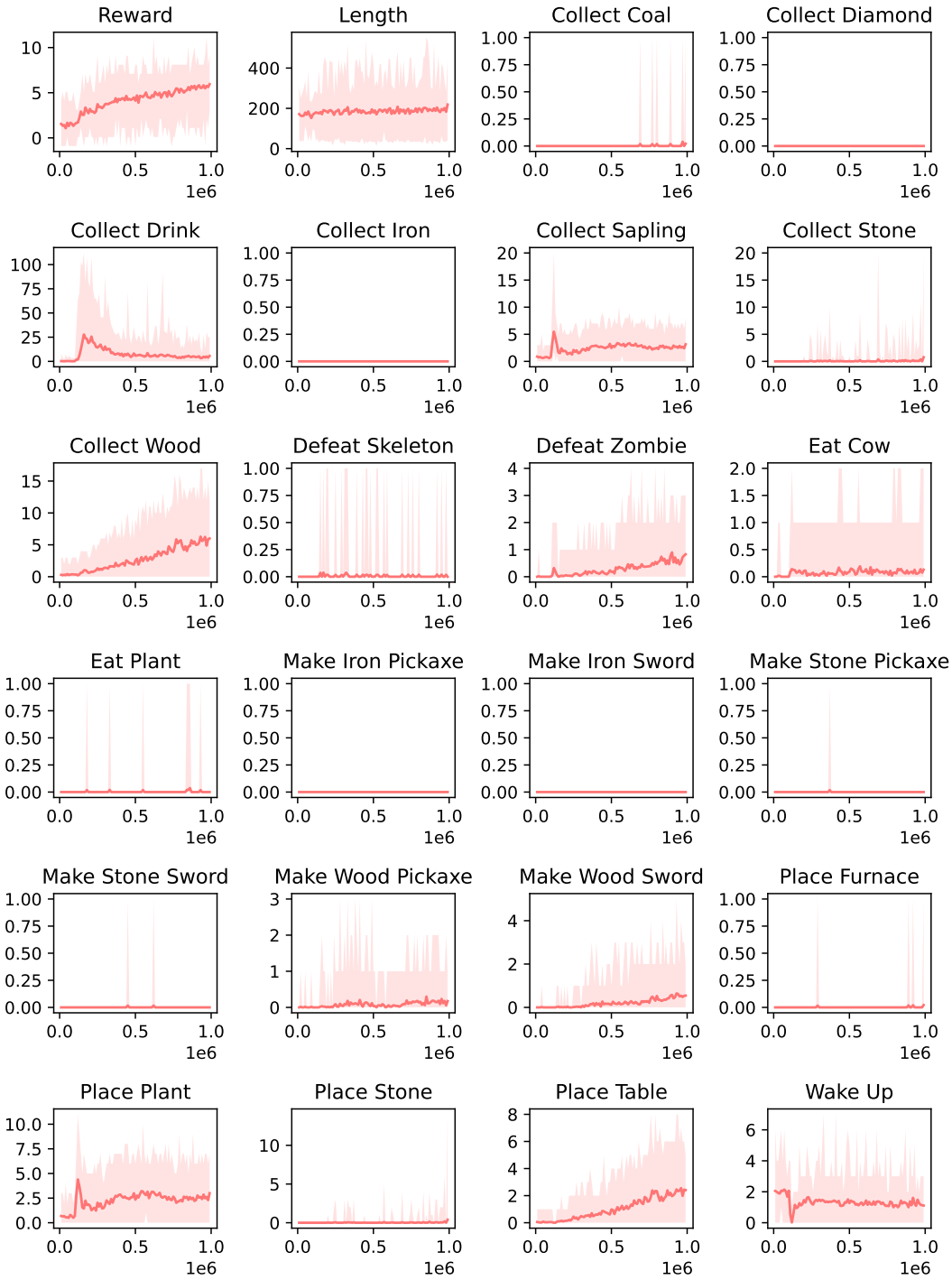

Figure 15: Achievement statistics over time for MStep IterVAML

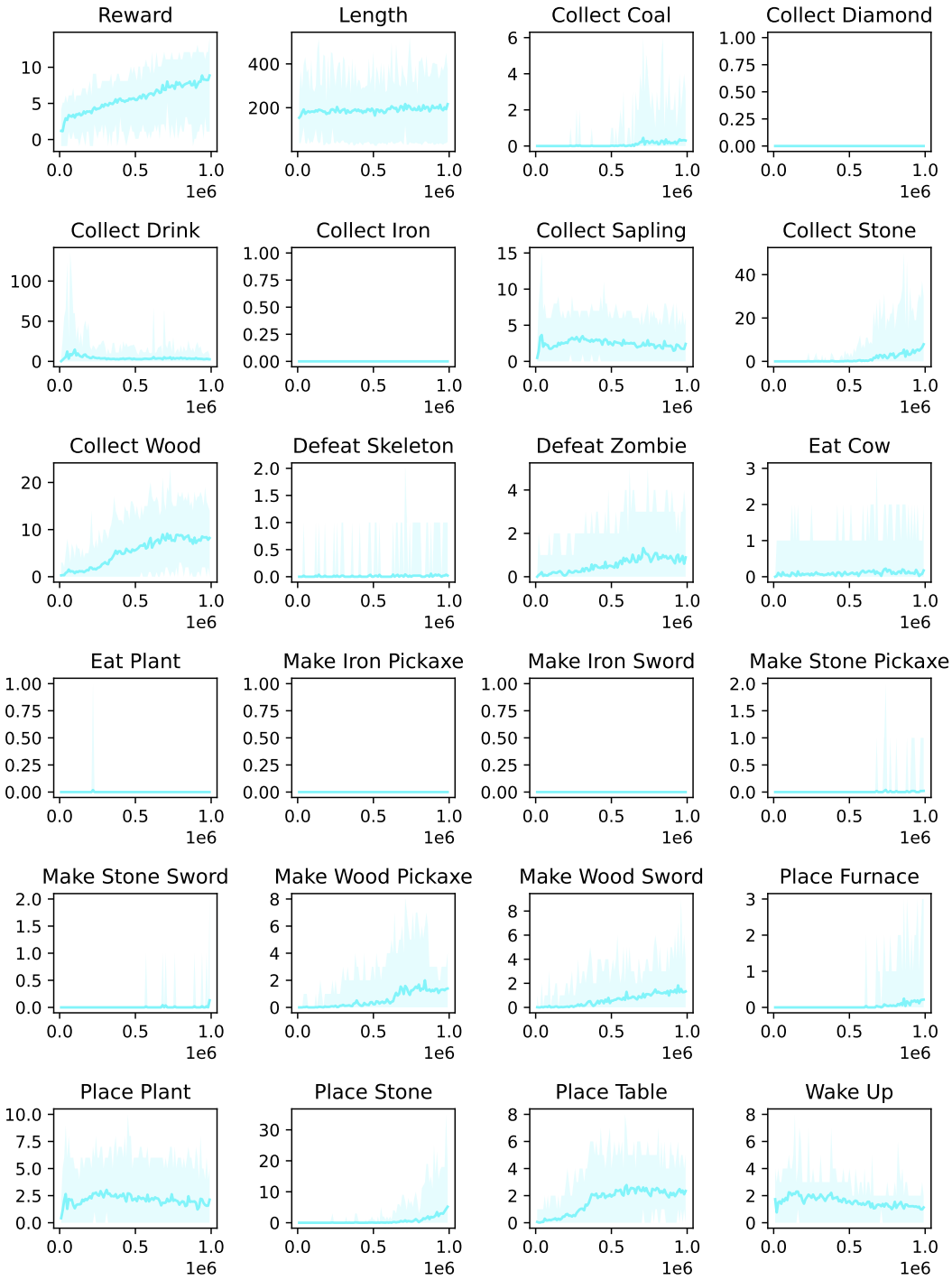

Figure 16: Achievement statistics over time for multi-step Dreamer