MUDREAMER: LEARNING PREDICTIVE WORLD MODELS WITHOUT RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

The DreamerV3 agent recently demonstrated state-of-the-art performance in diverse domains, learning powerful world models in latent space using a pixel reconstruction loss. However, while the reconstruction loss is essential to Dreamer's performance, it also necessitates modeling unnecessary information. Consequently, Dreamer sometimes fails to perceive crucial elements which are necessary for task-solving, significantly limiting its potential. In this paper, we present MuDreamer, a reinforcement learning agent that builds upon the DreamerV3 algorithm by learning a predictive world model without the need for reconstructing input signals. Rather than relying on pixel reconstruction, hidden representations are instead learned by predicting the environment value function and previously selected actions. Similar to predictive self-supervised methods for images, we find that the use of batch normalization is crucial to prevent learning collapse. We also study the effect of KL balancing between model posterior and prior losses on convergence speed and learning stability. We evaluate MuDreamer on the widely used DeepMind Visual Control Suite and achieves performance comparable to DreamerV3. MuDreamer also demonstrates promising results on the Atari100k benchmark. Research code will be made available publicly.

1 INTRODUCTION

Deep Reinforcement learning has achieved great success in recent years, solving complex tasks in diverse domains. Researchers made significant progress applying advances in deep learning for learning feature representations (Mnih et al., 2013). The use of deep neural networks as function approximations made it possible to train powerful agents directly from high-dimensional observations like images, achieving human to superhuman performance in challenging and visually complex domains like Atari (Mnih et al., 2015; Hessel et al., 2018), visual control (Lillicrap et al., 2016; Barth-Maron et al., 2018), the game of Go (Silver et al., 2018; Schrittwieser et al., 2020), StarCraft II (Vinyals et al., 2019) and even Minecraft (Baker et al., 2022; Hafner et al., 2023). However, these approaches generally require a large amount of environment interactions (Tassa et al., 2018) or behavior cloning pretraining (Silver et al., 2016) to achieves strong performances.

To address this issue, concurrent works have chosen to focus on model-based approaches (Silver et al., 2017; Watter et al., 2015), aiming to enhance the agent performance while reducing the number of necessary interactions with the environment. Reinforcement learning algorithms are typically categorized into two main groups: model-free algorithms, which directly learn value and/or policy functions through interaction with the environment, and model-based algorithms, which learn a model of the world by interacting with the environment. World models (Sutton, 1991; Ha & Schmidhuber, 2018) summarize an agent's experience into a predictive model that can be used in place of the real environment to learn complex behaviors. Having access to a model of the environment enables the agent to simulate multiple plausible trajectories in parallel, which not only enhances generalization but also improves sample efficiency.

Recent works have shown that model-based agents can effectively be trained from images, leading to enhanced performance and sample efficiency compared to model-free approaches (Hafner et al., 2019; 2020; Kaiser et al., 2020; Schrittwieser et al., 2020; Hafner et al., 2021; Ye et al., 2021; Micheli et al., 2023). The DreamerV3 agent (Hafner et al., 2023) recently demonstrated state-of-the-art performance across diverse domains, learning powerful world models in latent space using a

pixel reconstruction loss. The agent solves long-horizon tasks from image inputs with both continuous and discrete action spaces. However, while the reconstruction loss is essential for Dreamer's performance, it also necessitates modeling unnecessary information (Okada & Taniguchi, 2021; Ma et al., 2021; Nguyen et al., 2021; Paster et al., 2021; Deng et al., 2022; Bharadhwaj et al., 2022). Consequently, Dreamer sometimes fails to perceives crucial elements which are necessary for task-solving, significantly limiting its potential.

In this paper, we present MuDreamer, a reinforcement learning agent that builds upon the DreamerV3 (Hafner et al., 2023) algorithm by learning a predictive world model without the necessity of reconstructing input signals. Taking inspiration from the MuZero (Schrittwieser et al., 2020) agent, MuDreamer learns a world model in latent space by predicting the environment rewards, continuation flags and value function, focusing on information relevant to the task. We also propose to incorporate an action prediction branch to predict the sequence of selected actions from the observed data. This additional task proves particularly beneficial for learning hidden representations in scenarios where environment rewards are extremely sparse. Similar to predictive self-supervised methods used for image data, we find that the use of batch normalization is crucial to prevent learning collapse in which the model produces constant or non-informative hidden states. Following Paster et al. (2021), we solve this issue by introducing batch normalization inside the model representation network. We also study the effect of KL balancing between model posterior and prior losses on convergence speed and learning stability. We evaluate MuDreamer on the widely used DeepMind Visual Control Suite (Tassa et al., 2018). Our approach achieves performance comparable to DreamerV3 while being faster to train. Furthermore, we evaluate MuDreamer on the Atari100k benchmark (Kaiser et al., 2020) and present promising results.

2 Related Works

2.1 Model-based Reinforcement Learning

In recent years, there has been a growing interest in using neural networks as world models to simulate environments and train reinforcement learning agents from hypothetical trajectories. Initial research primarily focused on proprioceptive tasks (Gal et al., 2016; Silver et al., 2017; Henaff et al., 2017; Wang et al., 2019; Wang & Ba, 2019), involving simple, low-dimensional environments. However, more recent efforts have shifted towards learning world models for environments with high-dimensional observations like images (Kaiser et al., 2020; Hafner et al., 2019; 2020; Schrittwieser et al., 2020; Ye et al., 2021; Micheli et al., 2023). For example, SimPLe (Kaiser et al., 2020) successfully demonstrated planning in Atari games by training a world model in pixel space, then utilizing it to train a Proximal Policy Optimization (PPO) agent (Schulman et al., 2017). This approach involves a convolutional autoencoder for generating discrete latent states and an LSTM-based recurrent network for autoregressively generating latent states. PlaNet (Hafner et al., 2019) proposed to learn a Recurrent State-Space Model (RSSM) in latent space using a pixel reconstruction loss, planning using model-predictive control. Dreamer (Hafner et al., 2020; 2021; 2023) extended PlaNet by incorporating actor and critic networks trained from simulated trajectories and imaginary rewards. MuZero (Schrittwieser et al., 2020) took a different approach, focusing on learning a model of the environment by predicting quantities crucial for planning, such as reward, action-selection policy, and value function. This approach allowed MuZero to excel in reinforcement learning tasks without relying on the reconstruction of input observations. EfficientZero improved upon MuZero's sample efficiency by incorporating a representation learning objective (Chen & He, 2021) to achieve better performance with limited data. Lastly, IRIS (Micheli et al., 2023) proposed a discrete autoencoder (Van Den Oord et al., 2017) model for imagining trajectories, predicting discrete latent tokens in an autoregressive manner, using a transformer model.

2.2 Self-Supervised Representation Learning for Images

Self-supervised learning (SSL) of image representations has attracted significant research attention in recent years for its ability to learn hidden representations from large scale unlabelled datasets. Reconstruction-based approaches proposed to learn hidden representations by reconstructing a corrupted version of the input image (He et al., 2022; Xie et al., 2022; Feichtenhofer et al., 2022). Many works focused on pixel space reconstruction while other proposed to predict hand-designed features like histograms of oriented gradient (HOG) (Wei et al., 2022). Contrastive approaches proposed to

learn hidden representations using joint embedding architectures where output features of a sample and its distorted version are bought close to each other, while negative samples and their distortions are pushed away (Hjelm et al., 2019; Oord et al., 2018; He et al., 2020; Chen et al., 2020). These methods are commonly applied to Siamese architectures, where two identical networks are trained together, sharing parameters. In contrast, SwAV (Caron et al., 2020) proposed a different approach by ensuring consistency between cluster assignments produced for different augmentations of the same image, rather than directly comparing features. Predictive approaches proposed to predict the hidden representation of a similar view of the input signal without relying on negative samples (Grill et al., 2020; Caron et al., 2021; Ermolov et al., 2021; Chen & He, 2021; Zbontar et al., 2021; Baevski et al., 2022; Assran et al., 2023). These methods prevent learning collapse using various architectural tricks such as knowledge distillation (Hinton et al., 2015), normalizing output representations, or the application of additional constraints to output representations like VICReg (Bardes et al., 2021).

2.3 Reconstruction-Free Dreamer

Following advances in the area of self-supervised representation learning for image data, several works proposed to apply reconstruction-free representation learning techniques to Dreamer. The original Dreamer paper (Hafner et al., 2020) initially experimented with contrastive learning (Oord et al., 2018) to learn representations having maximal mutual information with the encoded observation but found that it did not match the performance of reconstruction-based representations. Subsequently, several works proposed Dreamer variants using contrastive learning (Okada & Taniguchi, 2021; Ma et al., 2021; Nguyen et al., 2021; Okada & Taniguchi, 2022; Bharadhwaj et al., 2022), successfully competing with Dreamer on several tasks of the Visual Control Suite. Dreaming (Okada & Taniguchi, 2021) proposed to use a multi-step InfoNCE loss to sequentially predict future time steps representations of augmented views. Temporal Predictive Coding (TPC) (Nguyen et al., 2021) followed a similar approach using contrastive learning to maximize the mutual information between past and the future latent states. Similar to SwAV, DreamerPro (Deng et al., 2022) proposed to encourage uniform cluster assignment across batches of samples, implicitly pushing apart embeddings of different observations. Concurrently, BLAST (Paster et al., 2021) proposed to learn hidden representation using a slow-moving teacher network to generate target embeddings (Grill et al., 2020). BLAST also demonstrated that batch normalization was critical to the agent performance. In this paper, we present a reconstruction-free variant of DreamerV3 achieving comparable performance without using negatives samples, augmented views of images, or an additional teacher network.

3 Background

3.1 Dreamer

Our method is built on the DreamerV3 (Hafner et al., 2023) algorithm which we refer to as Dreamer throughout the paper. Dreamer (Hafner et al., 2020) is an actor-critic model-based reinforcement learning algorithm learning a powerful predictive world model from past experience in latent space using a replay buffer. The world model is learned from self-supervised learning by predicting the environment reward, episode continuation and next latent state given previously selected actions. The algorithm also uses a pixel reconstruction loss using an autoencoder architecture such that all information about the observations must pass through the model hidden state. The actor and critic neural networks learn behaviors purely from abstract sequences predicted by the world model. The model generate simulated trajectories from replayed experience states using the actor network to sample actions. The value network is trained to predict the sum of future reward while the actor network is trained to maximize the expected sum of future reward from the value network.

DreamerV2 (Hafner et al., 2021) applied Dreamer to Atari games, utilizing categorical latent states with straight-through gradients (Bengio et al., 2013) in the world model to improve performance, instead of Gaussian latents with reparameterized gradients (Kingma & Welling, 2013). It also introduced KL balancing, separately scaling the prior cross entropy and the posterior entropy in the KL loss to encourage learning an accurate temporal prior.

DreamerV3 (Hafner et al., 2023) mastered diverse domains using the same hyper-parameters with a set of architectural changes to stabilize learning across tasks. The agent uses Symlog predictions

for the reward and value function to address the scale variance across domains. The networks also employ layer normalization (Ba et al., 2016) to improve robustness and performance while scaling up to larger model sizes. It regularizes the policy by normalizing the returns and value function using an Exponential Moving Average (EMA) of the returns percentiles. Using these modifications, the agent solves Atari games and DeepMind Control tasks while collecting diamonds in Minecraft.

3.2 MuZero

MuZero (Schrittwieser et al., 2020) is a model-based algorithm combining Monte-Carlo Tree Search (MCTS) (Coulom, 2006) with a world model to achieve superhuman performance in precision planning tasks such as Chess, Shogi and Go. The model is learned by being unrolled recurrently for K steps and predicting environment quantities relevant to planning. All parameters of the model are trained jointly to accurately match the TD value (Sutton, 1988) and reward, for every hypothetical step k. The MCTS algorithm uses the learned model to simulate environment trajectories and output an action visit distribution over the root node. This potentially better policy compared to the neural network one is used to train the policy network. MuZero excels in discrete action domains but struggles with high-dimensional continuous action spaces, where a discretization of possible actions is required to apply MCTS (Ye et al., 2021). Our proposed method, MuDreamer, draws inspiration from MuZero to learn a world model by predicting the expected sum of future rewards. MuDreamer solves tasks in both continuous and discrete action spaces, without the need for a reconstruction loss.

4 MuDreamer

We present MuDreamer, a reconstruction-free version of the Dreamer algorithm, which learns a world model in latent space by predicting not only rewards and continuation flags but also the environment value function and previously selected actions. Figure 1 illustrates the learning process of the MuDreamer world model. Similar to Dreamer, MuDreamer comprises three neural networks: a world model, a critic network, and a policy network. These three networks are trained concurrently using an experience replay buffer that collects past experiences. This section provides an overview of the world model and the modifications applied to the Dreamer agent. We also detail the learning process of the critic and actor networks.

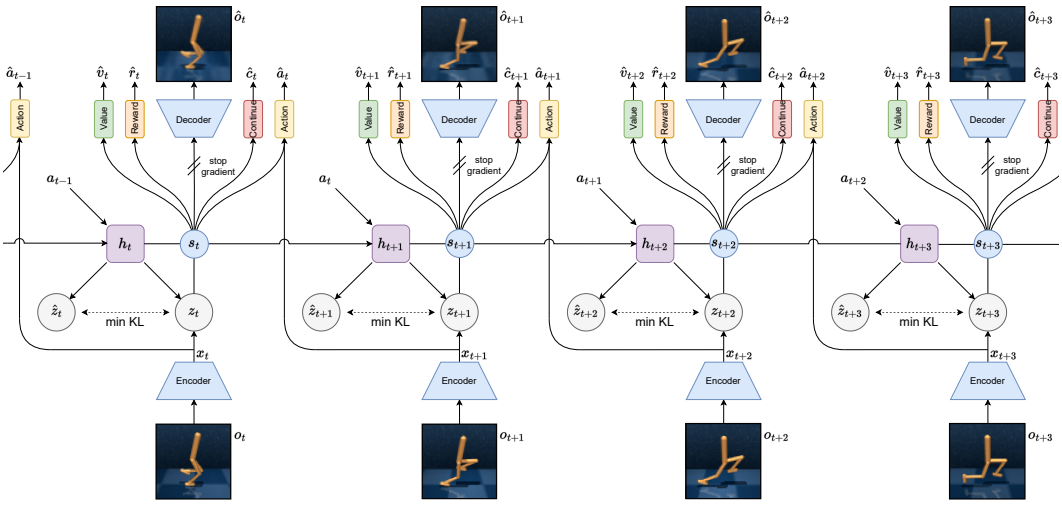

Figure 1: MuDreamer world model training. A sequence of image observations $o_{1:T}$ is sampled from the replay buffer. The sequence is mapped to hidden representations $x_{1:T}$ using a CNN encoder. At each step, the RSSM computes a posterior state z_t representing the current observation o_t and a prior state $\hat{z}_t$ that predict the posterior without having access to the current observation. Unlike Dreamer, the decoder gradient is not back-propagated to the rest of the model. The hidden representations are learned solely using value, reward, episode continuation and action prediction heads.

4.1 WORLD MODEL LEARNING

Following DreamerV3, we learn a world model in latent space, using a Convolutional Neural Networks (CNN) (LeCun et al., 1989) encoder to map high-dimensional visual observations o_t to hidden representations x_t. The world model is implemented as a Recurrent State-Space Model (RSSM) (Hafner et al., 2019) composed of three sub networks: A sequential network using a GRU (Cho et al., 2014) to predict the next hidden state h_t given past action a_{t-1}. A representation network predicting the current stochastic state z_t using both h_t and encoded features x_t. And a dynamics network predicting the stochastic state z_t given the current recurrent state h_t. The concatenation of h_t and z_t forms the model hidden state $s_t = \{h_t, z_t\}$ from which we predict environment rewards r_t, episode continuation $c_t \in \{0, 1\}$ and value function v_t. We also learn an action predictor network using encoded features x_t and preceding model hidden state s_{t-1} to predict the action which led to the observed environment change. The trainable world model components are the following:

$$
\begin{aligned}
&& \text{Encoder:} && x_t = enc_\phi(o_t) \\
\text{RSSM} &\begin{cases} \\ \\ \\ \end{cases} & \text{Sequential Network:} && h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}) \\
& & \text{Representation Network:} && z_t \sim q_\phi(z_t \mid h_t, x_t) \\
& & \text{Dynamics Predictor:} && \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) \\
&& \text{Reward Predictor:} && \hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t) \\
&& \text{Continue Predictor:} && \hat{c}_t \sim p_\phi(\hat{c}_t \mid h_t, z_t) \\
&& \text{Value Predictor:} && \hat{v}_t \sim p_\phi(\hat{v}_t \mid h_t, z_t) \\
&& \text{Action Predictor:} && \hat{a}_{t-1} \sim p_\phi(\hat{a}_{t-1} \mid x_t, h_{t-1}, z_{t-1}) \\
&& \text{Decoder:} && \hat{o}_t \sim p_\phi(\hat{o}_t \mid sg(h_t), sg(z_t))
\end{aligned}
\tag{1}
$$

Given a sequence batch of inputs $x_{1:T}$, actions $a_{1:T}$, rewards $r_{1:T}$, and continuation flags $c_{1:T}$, the world model parameters (ϕ) are optimized end-to-end to minimize a prediction loss L_{pred}, dynamics loss L_{dyn}, and representation loss L_{rep} with corresponding loss weights $\beta_{pred} = 1.0$, $\beta_{dyn} = 0.95$ and $\beta_{rep} = 0.05$. The loss function for learning the world model is:

$$
L_{model}(\phi) = \mathrm{E}_{q_\phi}\left[\sum_{t=1}^{T}(\beta_{pred}L_{pred}(\phi) + \beta_{dyn}L_{dyn}(\phi) + \beta_{rep}L_{rep}(\phi))\right] \tag{2}
$$

The prediction loss trains the reward, continue, value and action predictors to learn hidden representations. We optionally learn a decoder network to reconstruct the sequence of observations while using the stop gradient operator $sg(.)$ to prevent the gradients from being back-propagated to other network parameters. The dynamics loss trains the dynamics predictor to predict the next representation by minimizing the KL divergence between the predictor $p_\phi(z_t \mid h_t)$ and the next stochastic representation $q_\phi(z_t \mid h_t, x_t)$. While the representation loss trains the representations to become more predictable if the dynamics cannot predict their distribution:

$$
\begin{aligned}
L_{pred}(\phi) = &\underbrace{-\ln p_\phi(r_t \mid z_t, h_t)}_{\text{reward log loss}} \underbrace{-\ln p_\phi(c_t \mid z_t, h_t)}_{\text{continue log loss}} \underbrace{-\ y_t \ln p_\phi(\cdot \mid z_t, h_t)}_{\text{discrete returns regression}} \\
&\underbrace{-\ln p_\phi(a_{t-1} \mid x_t, z_{t-1}, h_{t-1})}_{\text{action log loss}} \underbrace{-\ln p_\phi(o_t \mid sg(z_t), sg(h_t))}_{\text{image log loss}} \\
L_{dyn}(\phi) = &\ \text{max}(1, \text{KL}[\ sg(q_\phi(z_t \mid h_t, x_t)) \ ||\ \ \ p_\phi(z_t \mid h_t)\ ]) \\
L_{rep}(\phi) = &\ \text{max}(1, \text{KL}[\ \ \ q_\phi(z_t \mid h_t, x_t)) \ ||\ sg(p_\phi(z_t \mid h_t)\ ])
\end{aligned}
\tag{3}
$$

Value Predictor We use a value network to predict the environment value function from the model hidden state. The value function aims to represent the expected λ-returns (Sutton, 1988) with λ set 0.95, using a slow moving target value predictor $v'_{\phi'}$ with EMA momentum $\tau = 0.01$:

$$
R_t^\lambda = r_t + \gamma c_t \begin{cases} (1-\lambda)v'_{\phi'}(s_{t+1}) + \lambda R_{t+1}^\lambda & \text{if t} < \text{T} \\ v'_{\phi'}(s_T) & \text{if t} = \text{T} \end{cases} \tag{4}
$$

Following previous works (Schrittwieser et al., 2020; Hafner et al., 2023), we set the discount factor γ to 0.997 and use a discrete regression of λ-returns to let the critic maintain and refine a distribution over potential returns. The λ-return targets are first transformed using the Symlog function and discretized using a *twohot* encoding. Given twohot-encoded targets $y_t = sg(twohot(symlog(R_t^\lambda)))$ the value predictor minimizes the categorical cross entropy loss with the predicted value logits.

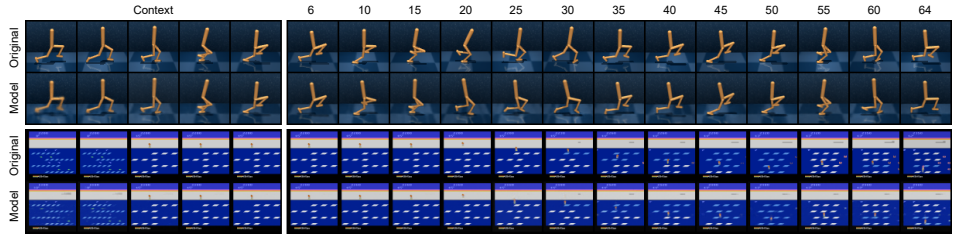

Figure 2: Reconstruction of MuDreamer model predictions over 64 time steps. We takes 5 context frames and generate trajectories of 59 steps into the future using the model sequential and dynamics networks. Actions are predicted using the policy network given generated latent states. MuDreamer generates accurate long-term predictions similar to Dreamer without requiring reconstruction loss gradients during training to compress the observation information into the model hidden state.

Action Predictor The action predictor network learns to identify the previously selected actions for each time step of the sampled trajectory. It shares the same architecture as the actor network but takes the current encoded features x_t and preceding model hidden state s_{t-1} as inputs. Since the current model hidden state s_t already depends of the action target a_t.

Batch Normalization In order to prevent learning collapse to constant or non-informative model states, like observed in predictive self-supervised learning for image data (Grill et al., 2020; Chen et al., 2020), we apply batch normalization (Ioffe & Szegedy, 2015) to MuDreamer. Following BLAST (Paster et al., 2021), we replace the hidden normalization layer inside the representation network by a batch normalization layer. We find that it is sufficient to prevent collapse on all tasks.

4.2 Behavior Learning

Following DreamerV3 (Hafner et al., 2023), MuDreamer policy and value functions are learned by imagining trajectories using the world model (Figure 2). Actor-critic learning takes place entirely in latent space, allowing the agent to use a large batch size of imagined trajectories. To do so, the model hidden states of the sampled sequence are flattened along batch and time dimensions. The world model imagines $H = 15$ steps into the future using the sequential and dynamics networks, selecting actions from the actor. The actor and critic use parameter vectors (θ) and (ψ), respectively:

$$\begin{aligned} &\text{Actor:} && \hat{a}_t \sim \pi_\theta(\hat{a}_t|\hat{s}_t) \\ &\text{Critic:} && v_\psi(\hat{s}_t) \approx \mathrm{E}_{p_\phi \pi_\theta}[\hat{R}^\lambda_t] \end{aligned} \tag{5}$$

Critic learning Similarly to the value predictor branch, the critic is trained by predicting the discretized λ-returns, but using rewards predictions imagined by the world model:

$$\hat{R}^\lambda_t = \hat{r}_t + \gamma \hat{c}_t \begin{cases} (1-\lambda) v_\psi(\hat{s}_{t+1}) + \lambda \hat{R}^\lambda_{t+1} & \text{if t} < \text{H} \\ v_\psi(\hat{s}_H) & \text{if t} = \text{H} \end{cases} \tag{6}$$

The critic also do not use a target network but relies on its own predictions for estimating rewards beyond the prediction horizon. This requires stabilizing the critic by adding a regularizing term of the predicted values toward the outputs of its own EMA network:

$$L_{critic}(\psi) = \sum_{t=1}^{H} \Big(\underbrace{- \hat{y}_t \ln p_\psi(\cdot \mid \hat{s}_t)}_{\text{discrete returns regression}} \underbrace{- v'_{\psi'}(\hat{s}_t) \ln p_\psi(\cdot \mid \hat{s}_t)}_{\text{critic EMA regularizer}} \Big) \tag{7}$$

Actor learning The actor network learns to select actions that maximize advantages $A^\lambda_t = R^\lambda_t - v_\psi(\hat{s}_t)$ while regularizing the policy entropy to ensure exploration both in imagination and during data collection. In order to use a single regularization scale for all domains, DreamerV3 stabilizes the scale of returns using moving statistics. The actor loss computes policy gradients using stochastic back-propagation thought the model sequential and dynamics networks for continuous actions ($\rho = 0$) and using reinforce (Williams, 1992) for discrete actions ($\rho = 1$):

$$L_{actor}(\theta) = \mathrm{E}_{\pi_\theta, p_\phi} \Big[\sum_{t=1}^{H} \Big(\underbrace{- \rho \ln \pi_\theta(\hat{a}_t \mid \hat{s}_t) sg(A^\lambda_t)}_{\text{reinforce}} \underbrace{- (1-\rho) A^\lambda_t}_{\text{dynamics backprop}} \underbrace{- \eta \mathrm{H}[\pi_\theta(\hat{a}_t \mid \hat{s}_t)]}_{\text{entropy regularizer}} \Big) \Big] \tag{8}$$

5 EXPERIMENTS

In this section, we aim to evaluate the performance of our MuDreamer algorithm compared to its reconstruction-based version and other published model-based and model-free methods. We evaluate MuDreamer on the Visual Control Suite from DeepMind (Table 1) as well as the atari100k benchmark (Table 2). We also proceed to a detailed ablation study on the Visual Control Suite, studying the effect of batch normalization, action-value predictors and KL balancing on performance and learning stability (Table 3). Our PyTorch (Paszke et al., 2019) implementation of MuDreamer takes 4 hours to reach 1M steps on the 'walker run' visual control task using a single NVIDIA RTX 3090 GPU and 16 CPU cores, while DreamerV3 takes 4 hours and 20 minutes. The total amount of trainable parameters for MuDreamer and DreamerV3 are 15.3M and 17.9M, respectively.

5.1 RESULTS

Visual Control Suite The DeepMind Control Suite was introduced by Tassa et al. (2018) as a set of continuous control tasks with a standardised structure and interpretable rewards. The suite is intended to serve as performance benchmarks for reinforcement learning agents in continuous action space. The tasks can be solved from low-dimensional inputs and/or pixel observations. In this work, we evaluate our method on the Visual Control Suite benchmark which contains 20 tasks where the agent receives only high-dimensional images as inputs and a budget of 1M environment steps. Similarly to DreamerV3 (Hafner et al., 2023), we use 4 environment instances during training with a training ratio of 512 replayed steps per policy step.

Table 1 compares MuDreamer with DreamerV3 and other recent methods on the Visual Control Suite using 1M environment steps. MuDreamer achieves performance comparable to DreamerV3 without reconstructing the input signal, outperforming SAC (Haarnoja et al., 2018), CURL (Laskin et al., 2020) and DrQ-v2 (Yarats et al., 2021). As shown in Figure 2, although the reconstruction gradients are not propagated to the whole network, the decoder easily reconstruct the input image, meaning that the model hidden state contains all necessary information about the environment. MuDreamer shows faster convergence on the Quadruped task. However, we found it to be slightly slower to converge on Cheetah and Walker where the agent body occupy a large portion of the image input, diminishing the impact of the background on the reconstruction loss. Performance curves showing a comparison of MuDreamer and DreamerV3 over the 20 tasks using 3 different seeds can be found in Appendix B.

Task	SAC‡	CURL‡	DrQ-v2‡	DreamerV3	DreamerV3†	MuDreamer
Acrobot Swingup	5.1	5.1	128.4	210.0	**227.7**	191.8
Cartpole Balance	963.1	979.0	991.5	**996.4**	995.8	994.4
Cartpole Balance Sparse	950.8	981.0	996.2	**1000.0**	**1000.0**	**1000.0**
Cartpole Swingup	692.1	762.7	**858.9**	819.1	843.7	852.7
Cartpole Swingup Sparse	154.6	236.2	706.9	792.9	**802.8**	554.8
Cheetah Run	27.2	474.3	691.0	728.7	**828.6**	675.5
Cup Catch	163.9	**965.5**	931.8	957.1	961.3	962.1
Finger Spin	312.2	**877.1**	846.7	818.5	460.9	507.5
Finger Turn Easy	176.7	338.0	448.4	787.7	677.2	**888.2**
Finger Turn Hard	70.5	215.6	220.0	**810.8**	369.0	637.7
Hopper Hop	3.1	152.5	189.9	**369.6**	239.4	247.4
Hopper Stand	5.2	786.8	893.0	900.6	**923.7**	896.7
Pendulum Swingup	560.1	376.4	839.7	806.3	834.1	**843.2**
Quadruped Run	50.5	141.5	407.0	352.3	513.7	**723.4**
Quadruped Walk	49.7	123.7	660.3	352.6	723.0	**909.8**
Reacher Easy	86.5	609.3	910.2	898.9	**950.3**	823.3
Reacher Hard	9.1	400.2	**572.9**	499.2	445.7	142.3
Walker Run	26.9	376.2	517.1	**757.8**	585.1	542.7
Walker Stand	159.3	463.5	974.1	**976.7**	973.7	960.7
Walker Walk	38.9	828.8	762.9	**955.8**	950.7	939.9
Mean	225.3	504.7	677.4	**739.6**	715.3	714.7
Median	78.5	431.8	734.9	808.5	816.5	**829.6**

Table 1: Visual Control Suite scores (1M environment steps). † denotes our implementation of DreamerV3. ‡ results were taken from Hafner et al. (2023). We use 3 different seeds per experiment.

Atari100k The Atari100k benchmark was proposed in Kaiser et al. (2020) to evaluate reinforcement learning agents on Atari games in low data regime. The benchmark includes 26 Atari games with a budget of 400K environment steps, amounting to 100K steps using action repeat. This represents 2

hours of real-time play. The current state-of-the-art is held by EfficientZero (Ye et al., 2021), which uses look-ahead search to select the best action, with a human mean score of 190% and median of 116%. Concerning the Atari100k benchmark, we only use a single environment instance during training with a training ratio of 1024 replayed steps per policy step.

Table 2 compares MuDreamer with DreamerV3 and other methods on the Atari100k benchmark. We compare the returns across games using human-normalized metrics. MuDreamer fails to solve Boxing, which severely penalizes the final human-normalized score. It nevertheless demonstrates promising results, improving DreamerV3 performance on Battle Zone and Frostbite. To obtain a more meaningful comparison with respect to DreamerV3, we compare DreamerV3-normalized metrics, excluding Freeway and Chopper Command since DreamerV3 does not achieve better than random score in these games. Our MuDreamer agent achieves a DreamerV3-normalized mean score of 95%, reaching better performance than model-free algorithms like SimPLe (Kaiser et al., 2020) and CURL (Laskin et al., 2020). Performance curves showing a comparison of MuDreamer and DreamerV3 over the 26 games using 3 different seeds can be found in Appendix C.

Task	Random	Human	SimPLe	CURL‡	EfficientZero	IRIS	DreamerV3	DreamerV3†	MuDreamer
Alien	228	7128	617	711	**1140**	420	959	908	713
Amidar	6	1720	74	114	102	**143**	139	125	114
Assault	222	742	527	501	1407	**1524**	706	635	594
Asterix	210	8503	1128	567	**16844**	854	932	997	813
Bank Heist	14	753	34	65	362	53	649	**950**	561
Battle Zone	2360	37188	4031	8998	17938	13074	12250	13933	**22633**
Boxing	0	12	8	1	44	70	**78**	68	-8
Breakout	2	30	16	3	**406**	84	31	19	13
Chopper Com.	811	7388	979	784	**1794**	1565	420	1213	780
Crazy Climber	10780	35829	62584	9154	80125	59324	**97190**	77360	80017
Demon Attack	152	1971	208	646	**13298**	2034	303	264	237
Freeway	0	30	17	28	22	**31**	0	7	0
Frostbite	65	4335	237	1226	314	259	909	938	**2306**
Gopher	258	2412	597	401	3518	2236	**3730**	3097	1347
Hero	1027	30826	2657	4988	8530	7037	11161	**13451**	4521
James Bond	29	303	100	331	459	**463**	445	282	227
Kangaroo	52	3035	51	740	962	838	**4098**	2967	3993
Krull	1598	2666	2205	3049	6047	6616	**7782**	5900	6408
Kung Fu Master	258	22736	14862	8156	**31112**	21760	21420	24933	20117
Ms Pacman	307	6952	1480	1064	1387	999	1327	**2279**	1460
Pong	–21	15	13	-18	**21**	15	18	18	11
Private Eye	25	69571	35	82	100	100	882	**6488**	3607
Qbert	164	13455	1289	727	**15458**	746	3405	1905	3128
Road Runner	12	7845	5641	5006	**18512**	9615	15565	9960	9003
Seaquest	68	42055	683	315	**1020**	661	618	491	389
Up N Down	533	11693	3350	2646	**16096**	3546	7600	5199	4766
Human Mean	0%	100%	33%	26%	**190%**	105%	112%	93%	62%
Human Median	0%	100%	13%	9%	**116%**	29%	49%	44%	36%
DreamerV3 Mean	0%	981%	43%	50%	**623%**	130%	100%	120%	95%
DreamerV3 Median	0%	137%	37%	35%	**107%**	74%	100%	86%	79%

Table 2: Atari100k scores (400K environment steps). † denotes our implementation of DreamerV3. ‡ results were taken from Micheli et al. (2023). We use 3 different seeds per experiment.

5.2 Ablation Study

In order to understand the necessary components of MuDreamer, we conduct ablation studies applying one modification at a time. We study the impact of using value and action prediction branches, removing one or both branches during training. We study the effect of using batch normalization in the representation network to stabilize learning and avoid collapse without the reconstruction loss. We also study the effect of KL balancing hyper-parameters on learning speed and stability. We perform all these ablation studies on the Visual Control Suite, observing the effect of each modification for diverse tasks. The mean and median score results of these studies are summarized in Table 3 and Figure 3. Please refer to the appendix for the score curves of individual tasks.

Action-Value predictors We study the necessity of using action and value prediction branches to learn hidden representations and successfully solve the tasks without reconstruction loss. We observed that removing the action or value prediction heads led to a degradation of MuDreamer performance and decoder reconstruction quality on most of the Visual Control tasks. The action

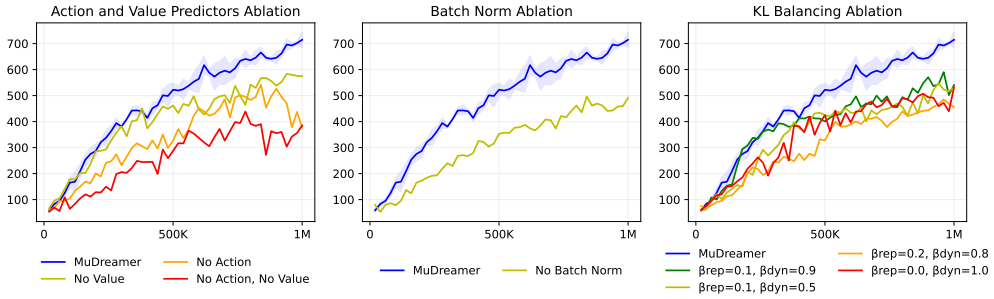

Figure 3: Ablations mean scores on the Visual Control Suite using 1M environment steps.

Agent	Mean	Median	Tasks Score $\geq$ MuDreamer
MuDreamer	**714.7**	**829.6**	–
No Value Predictor	574.5	600.4	5 / 20
No Action Predictor	377.1	340.0	0 / 20
No Action and Value Predictors	385.0	270.0	2 / 20
No Batch Normalization	491.9	519.6	3 / 20
$\beta_{rep} = 0.0$, $\beta_{dyn} = 1.0$	531.6	475.8	3 / 20
$\beta_{rep} = 0.1$, $\beta_{dyn} = 0.9$	540.5	522.2	1 / 20
$\beta_{rep} = 0.1$, $\beta_{dyn} = 0.5$	531.5	618.8	3 / 20
$\beta_{rep} = 0.2$, $\beta_{dyn} = 0.8$	455.8	506.3	0 / 20

Table 3: Ablations to MuDreamer evaluated on the Visual DeepMind Control Suite using 1M environment steps. Each ablation applies only one modification to the MuDreamer agent.

and value prediction losses require the model to learn environment dynamics which leads to a more accurate model and better performance.

batch normalization We study the effect of using batch normalization inside the representation network to stabilize representation learning. We find that batch normalization prevents collapse in which the model produces constant or non-informative hidden states. Without batch normalization, we observe that MuDreamer fails to learn representations for some of the tasks. Batch normalization solves this problem by stabilizing dynamics and representation losses.

KL balancing We find that applying the default KL balancing parameters of DreamerV3 ($\beta_{dyn} = 0.5$, $\beta_{rep} = 0.1$) slows down convergence for some of the tasks, restraining the model from learning representations. Similar to BLAST and predictive SSL approaches (Grill et al., 2020; Chen et al., 2020), we experiment using the stop gradient operation with $\beta_{rep} = 0.0$. We find that this solves the issue but generates learning instabilities with spikes for the dynamics and prediction losses. We also observed a degradation of the agent performance after a certain amount of steps. Using a slight regularization of the representations toward the prior with $\beta_{rep} = 0.05$ solved both of these issues.

6 Conclusion and Future Work

We presented MuDreamer, a variant of DreamerV3 solving tasks from image inputs with both continuous and discrete action spaces, all without the need to reconstruct input signals. MuDreamer learns a world model by predicting environment rewards, value function, and continuation flags, focusing on information relevant to the task. We also proposed to incorporate an action prediction branch to predict the sequence of selected actions. Our approach is faster to train than DreamerV3, as it does not require training an additional decoder network for generating video sequences. MuDreamer achieves performance comparable to DreamerV3 on the DeepMind Visual Control Suite. Furthermore, it demonstrates promising results on the Atari100k benchmark using limited data.

Looking ahead, our future research will explore the impact of multi-step predictions on performance by unrolling the world model recurrently for several steps. We also plan to apply MuDreamer to control tasks with random background and more complex environments like Minecraft.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*, pp. 1298–1312. PMLR, 2022.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributional policy gradients. In *International Conference on Learning Representations*, 2018.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Homanga Bharadhwaj, Mohammad Babaeizadeh, Dumitru Erhan, and Sergey Levine. Information prioritization through empowerment in visual model-based rl. In *International Conference on Learning Representations*, 2022.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15750–15758, 2021.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Fei Deng, Ingook Jang, and Sungjin Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 4956–4975. PMLR, 2022.

Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. In *International Conference on Machine Learning*, pp. 3015–3024. PMLR, 2021.

Christoph Feichtenhofer, Yanghao Li, Kaiming He, et al. Masked autoencoders as spatiotemporal learners. *Advances in neural information processing systems*, 35:35946–35958, 2022.

Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-efficient machine learning workshop, ICML*, volume 4, pp. 25, 2016.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Mikael Henaff, William F Whitney, and Yann LeCun. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177*, 2017.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Xiao Ma, Siwei Chen, David Hsu, and Wee Sun Lee. Contrastive variational reinforcement learning for complex observations. In *Conference on Robot Learning*, pp. 959–972. PMLR, 2021.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample efficient world models. In *International Conference on Learning Representations*, 2023.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Tung D Nguyen, Rui Shu, Tuan Pham, Hung Bui, and Stefano Ermon. Temporal predictive coding for model-based planning in latent space. In *International Conference on Machine Learning*, pp. 8130–8139. PMLR, 2021.

Masashi Okada and Tadahiro Taniguchi. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction. In *2021 ieee international conference on robotics and automation (icra)*, pp. 4209–4215. IEEE, 2021.

Masashi Okada and Tadahiro Taniguchi. Dreamingv2: Reinforcement learning with discrete world models without reconstruction. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 985–991. IEEE, 2022.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Keiran Paster, Lev E McKinney, Sheila A McIlraith, and Jimmy Ba. Blast: Latent dynamics models from bootstrapping. In *Deep RL Workshop NeurIPS 2021*, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

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, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Hado Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and planning. In *International Conference on Machine Learning*, pp. 3191–3199. PMLR, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*, 2019.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *Advances in neural information processing systems*, 28, 2015.

Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14668–14678, 2022.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9653–9663, 2022.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pp. 12310–12320. PMLR, 2021.

A World Model Predictions

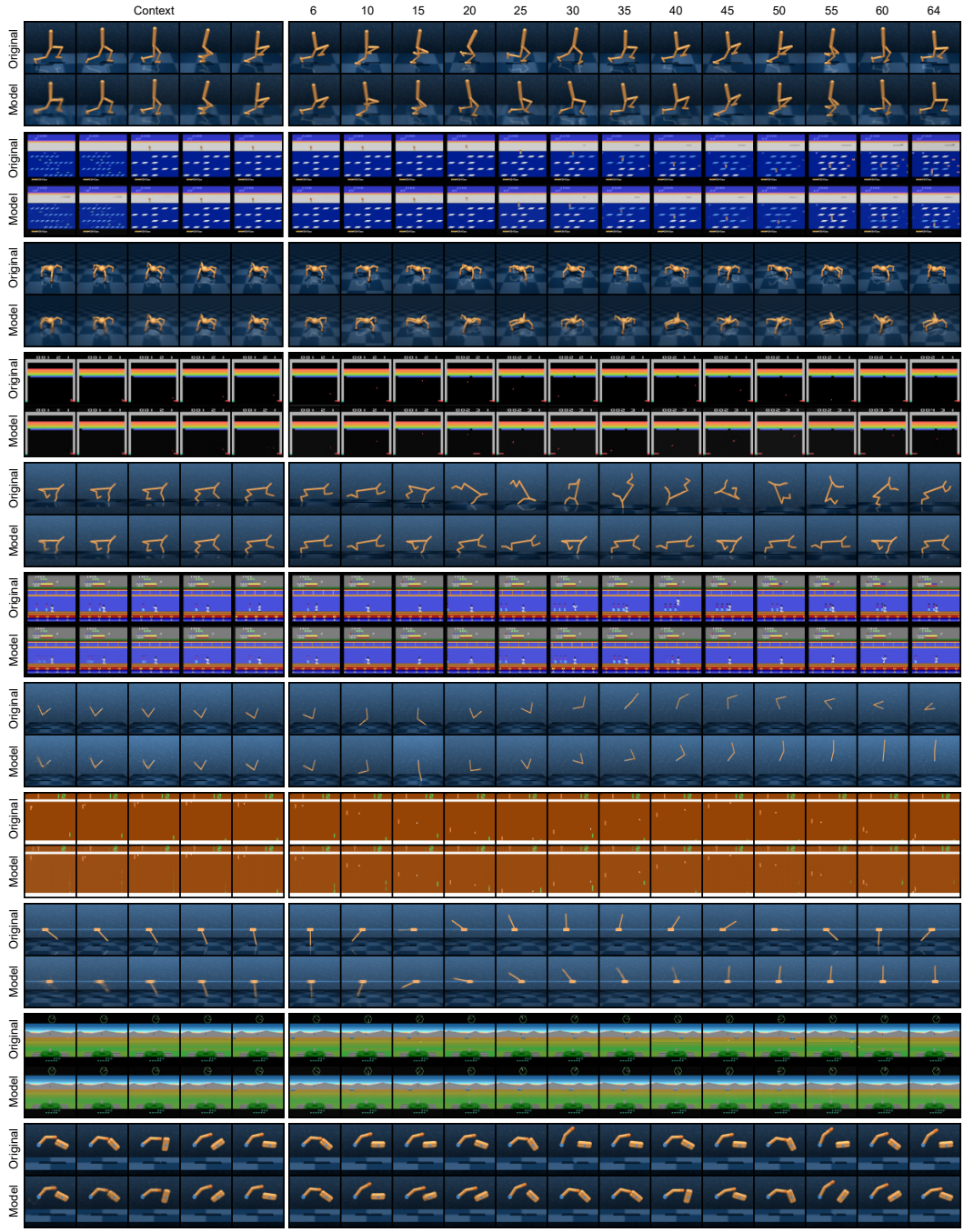

Figure 4: Trajectories imagined by the world model over 64 time steps using 5 context frames. DreamerV3 generates accurate long-term predictions for various tasks without requiring reconstruction loss gradients during training to compress the observation information into the model hidden state. Although the reconstruction gradients are not propagated to the whole network, the decoder successfully reconstruct the input image, meaning that the model hidden state contains all necessary information about the environment.

B VISUAL CONTROL COMPARISON

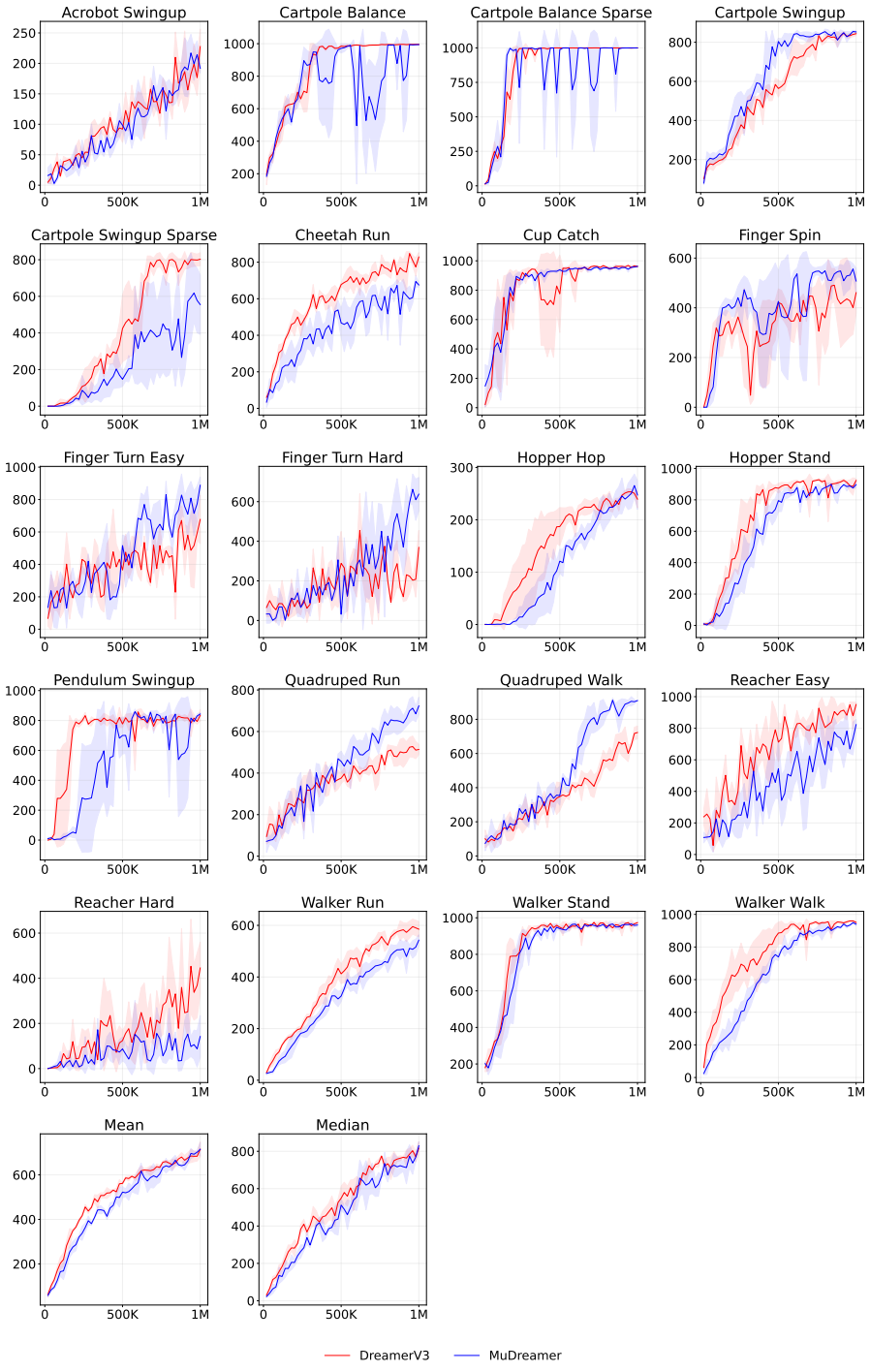

Figure 5: Comparison of MuDreamer and DreamerV3 scores on the Deep Mind Control Suite (1M environment steps). The curves show mean and standard deviation across 3 different seeds. While MuDreamer demonstrates faster convergence on Quadruped tasks, it is slightly slower to converge on Walker and Cheetah. Overall, MuDreamer achieves comparable performance to DreamerV3 in aggregated score metrics after 1M environment steps.

C ATARI100K COMPARISON

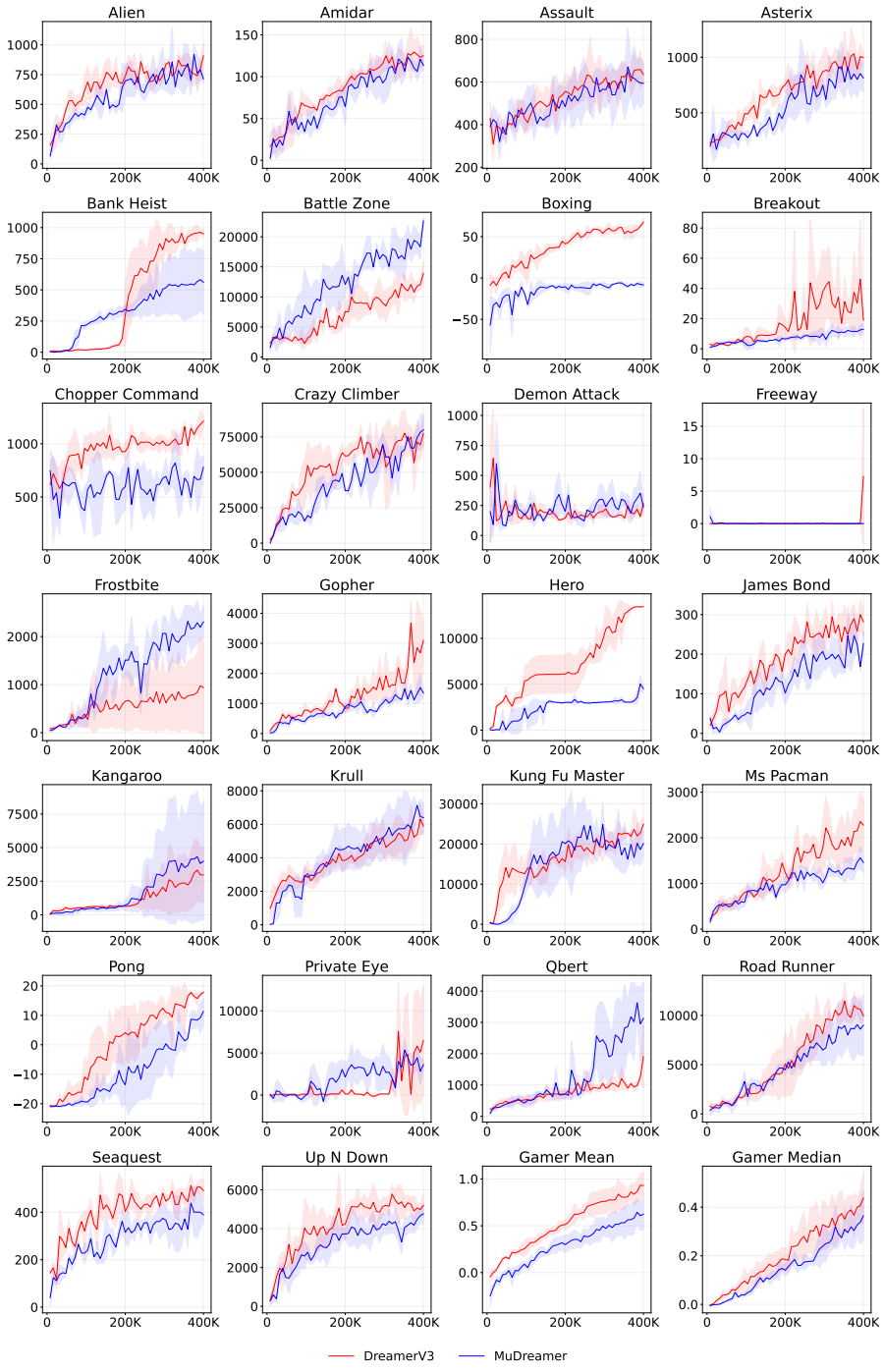

Figure 6: Comparison of MuDreamer and DreamerV3 on the Atari100k benchmark (400K environment steps). The curves show mean and standard deviation across 3 different seeds. MuDreamer fails to solve Boxing, which severely penalizes the final human-normalized score. It nevertheless demonstrates promising results, improving DreamerV3 performance on Battle Zone and Frostbite.

D BATCH NORMALIZATION ABLATION

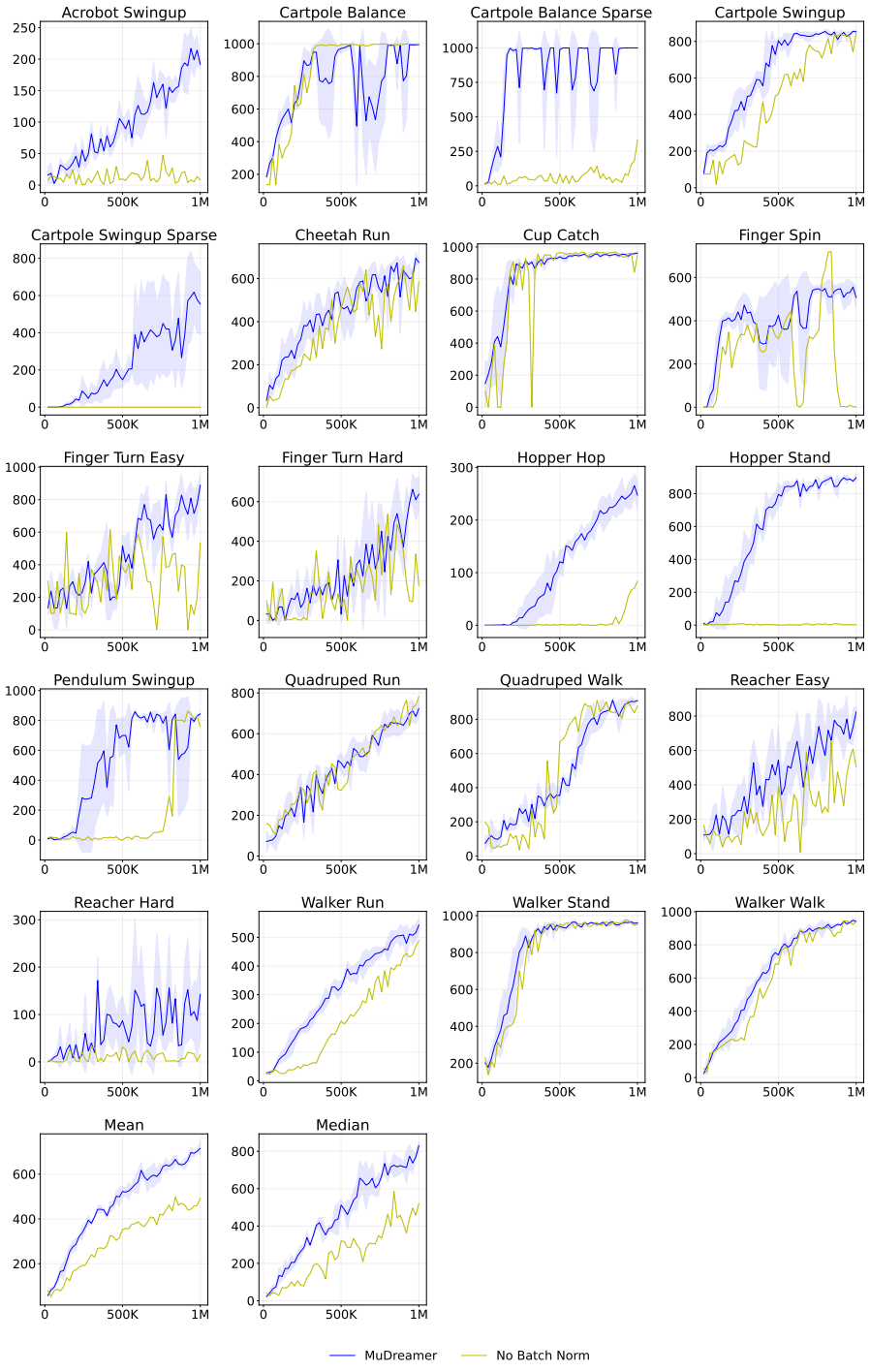

Figure 7: Comparison of MuDreamer, using layer normalization instead of batch normalization in the representation network. Removing the batch normalization layer leads to optimization difficulties for some of the tasks. The model hidden states collapse to constant or non-informative representations. This makes it impossible for the decoder to reconstruct the input observations.

E ACTION AND VALUE PREDICTORS ABLATION

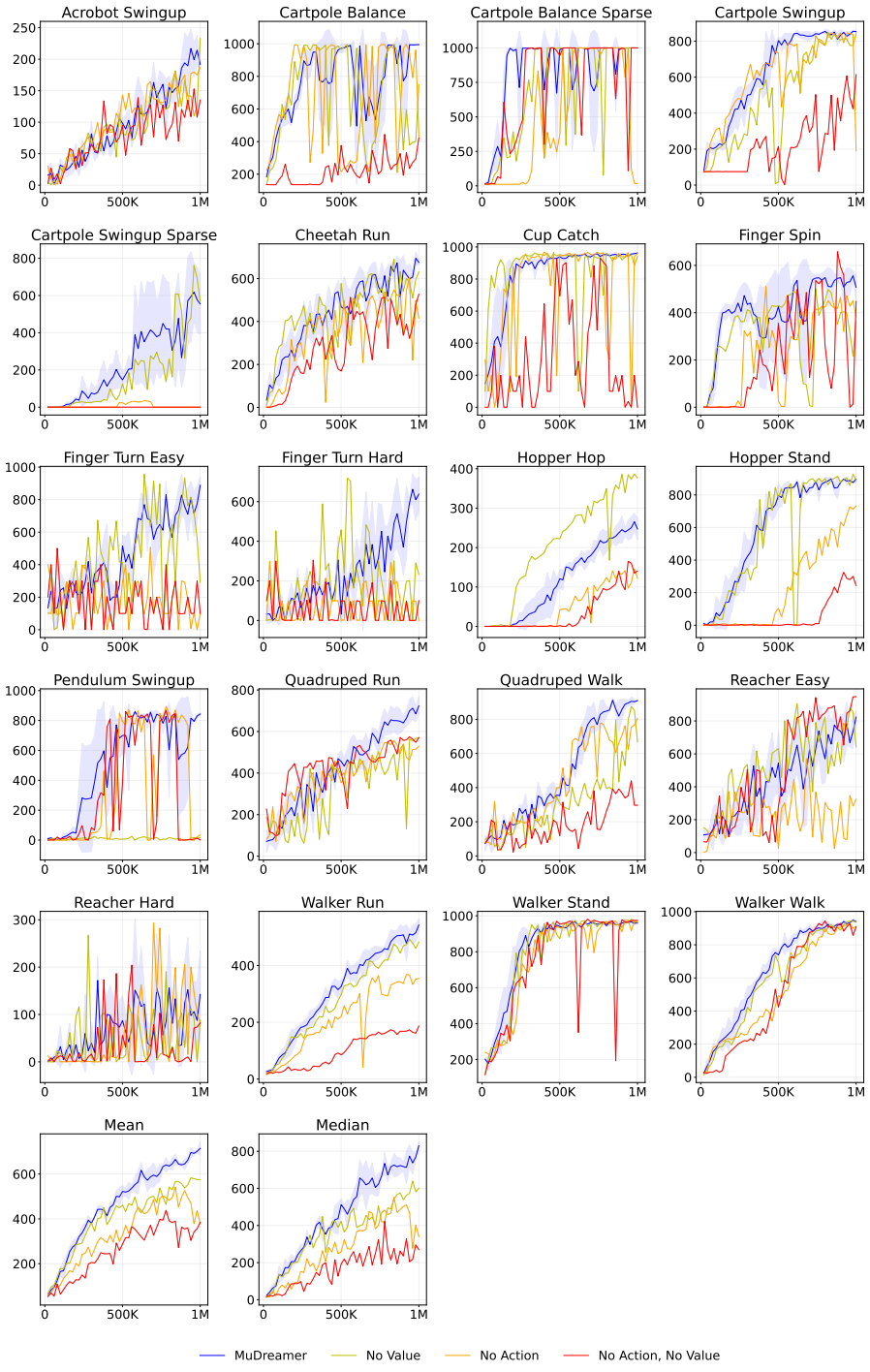

Figure 8: Comparison of MuDreamer, removing the action and/or value prediction heads. Removing the action and/or value heads during training deteriorates MuDreamer performance. We find that action prediction significantly helps tasks with sparse rewards like Hopper Hop or Cartpole Swingup Sparse. The value prediction head improves stability, leading to more stable learning. Both heads help the model to learn representations, leading to faster convergence and lower reconstruction loss.

F KL BALANCING ABLATION

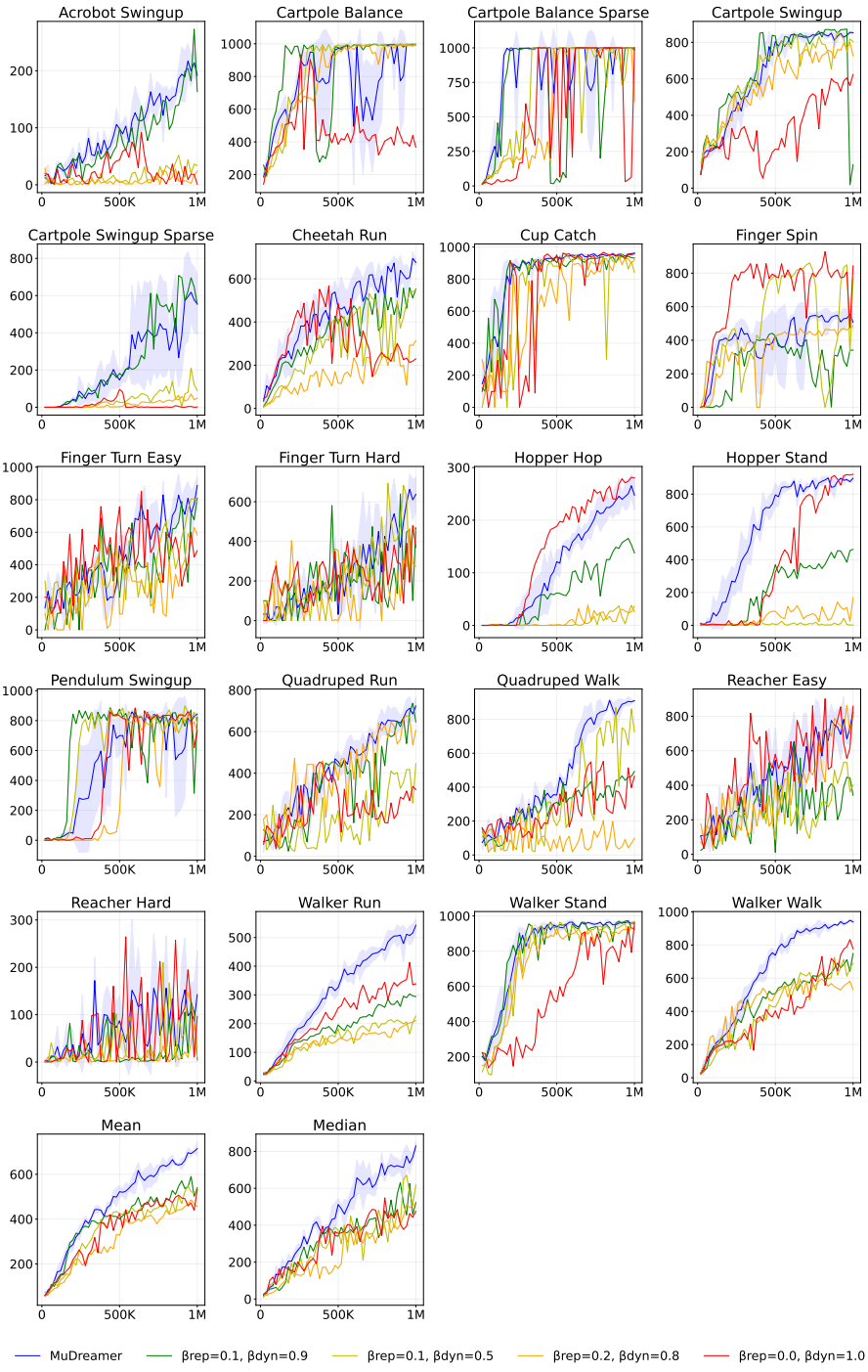

Figure 9: Comparison of MuDreamer, studying the effect of β_{dyn} and β_{rep}. We find that applying the default KL balancing parameters of DreamerV3 slows down convergence for some of the tasks, restraining the world model from learning representations. Setting β_{rep} to zero improves learning speed but results in instabilities and a degradation of performance after a certain amount of steps. Using a slight regularization of the representations toward the prior with $\beta_{rep} = 0.05$ improved convergence speed while maintaining stability.

G MuDreamer Hyper Parameters

Name	Symbol	Value
General		
Replay Buffer Capacity (FIFO)	—	10^6
Batch Size	B	16
Batch Length	T	64
Activation	—	LayerNorm + SiLU
Model Size	—	DreamerV3 Small
Input Image Resolution	—	64×64 RGB
World Model		
Number of Latents	—	32
Classes per Latent	—	32
Prediction Loss Scale	β_{pred}	1.0
Dynamics Loss Scale	β_{dyn}	0.95
Representation Loss Scale	β_{rep}	0.05
Learning Rate	—	10^{-4}
Adam Epsilon	ϵ	10^{-8}
Gradient Clipping	—	1000
Slow Value Momentum	τ	0.01
Model Discount	γ	0.997
Model Return Lambda	λ	0.95
Activation Representation Network	—	BatchNorm + SiLU
Actor Critic		
Imagination Horizon	H	15
Discount	γ	0.997
Return Lambda	λ	0.95
Critic EMA Decay	—	0.98
Critic EMA regularizer	—	1.0
Return Normalization Percentiles	—	5^{th} and 95^{th}
Return Normalization Decay	—	0.99
Actor Entropy Scale	η	$3 \cdot 10^{-4}$
Learning Rate	—	$3 \cdot 10^{-5}$
Adam Epsilon	ϵ	10^{-5}
Gradient Clipping	—	100

Table 4: MuDreamer Hyper parameters. We apply the same hyper-parameters for the DeepMind Visual Control Suite and Atari100k benchmark. Discount and Return Lambda values are kept the same for World Model and Actor Critic training.

H VISUAL CONTROL SCORES (5M ENVIRONMENT STEPS)

Task	PlaNet‡	DreamerV1	DreamerV3†	MuDreamer
Acrobot Swingup	3.2	365.3	411.2	**422.7**
Cartpole Balance	452.6	979.6	**999.2**	999.0
Cartpole Balance Sparse	164.7	941.9	**1000.0**	**1000.0**
Cartpole Swingup	312.6	833.7	**866.7**	851.2
Cartpole Swingup Sparse	0.6	812.2	841.2	**847.8**
Cheetah Run	496.1	894.6	**916.1**	891.6
Cup Catch	456.0	962.5	**967.5**	961.4
Finger Spin	495.3	498.9	388.9	**785.4**
Finger Turn Easy	451.2	825.9	823.3	**894.9**
Finger Turn Hard	312.6	891.4	896.6	**936.1**
Hopper Hop	242.0	369.0	**640.2**	328.4
Hopper Stand	6.0	923.7	**943.0**	909.8
Pendulum Swingup	3.3	833.0	793.6	**849.0**
Quadruped Run	280.5	888.4	**918.2**	845.3
Quadruped Walk	238.9	931.6	940.7	**943.5**
Reacher Easy	468.5	935.1	**972.9**	965.2
Reacher Hard	187.0	817.1	**974.9**	644.0
Walker Run	626.3	824.7	**827.0**	803.4
Walker Stand	759.2	978.0	**986.0**	977.0
Walker Walk	944.7	961.7	**973.3**	963.5
Mean	333.0	823.4	**854.0**	841.0
Median	312.6	889.9	917.2	893.3

Table 5: Visual Control Suite scores (5M environment steps). † denotes our implementation of DreamerV3. ‡ results were taken from Hafner et al. (2020). We train MuDreamer for 5M environment steps on the Visual Control Suite using a single seed. MuDreamer achieves results comparable to state-of-the-art, competing with DreamerV1 with mean score of 841.0 and median score of 893.3.