BLAST: Latent Dynamics Models from Bootstrapping

Keiran Paster*
Department of Computer Science
University of Toronto, Vector Institute
keirp@cs.toronto.edu

Lev McKinney*
Department of Computer Science
University of Toronto, Vector Institute
lev.mckinney@mail.utoronto.ca

Sheila A. McIlraith & Jimmy Ba
Department of Computer Science
University of Toronto, Vector Institute
{sheila, jimmy}@cs.toronto.edu

Abstract

State-of-the-art world models such as DreamerV2 [15] have significantly improved the capabilities of model-based reinforcement learning. However, these approaches typically rely on a reconstruction loss to shape their latent representations, which is known to fail in environments with high fidelity visual observations [8]. Previous work has found that when learning latent dynamics models without a reconstruction loss by using only the signal provided by the reward, the performance can also drop dramatically [1]. We present a simple set of modification to DreamerV2 to remove its reliance on reconstruction inspired by the recent self-supervised learning method Bootstrap Your Own Latent [10]. The combination of adding a stop-gradient to the posterior, using a powerful auto-regressive model for the prior, and using a slowly updating target encoder, which we call BLAST, allows the world model to learn from signals present in both the reward and observations, improving efficiency on our tested environment as well as being significantly more robust to visual distractors.

1 Introduction

Many environments have simple dynamics and yet require millions or billions of potentially expensive agent interactions to be solved using deep reinforcement learning (DRL) algorithms. One promising way to improve the sample efficiency of DRL is by learning a model of the environment. Then learning a policy within this model. Model-based RL (MBRL) is particularly efficient when learning a world model is relatively easy compared to learning the policy. Luckily, the state of the art in generative modeling is constantly improving, and as new techniques are discovered, these can bring significant improvements to our ability to learn world models for MBRL, making it a viable approach for learning a growing set of complex tasks.

With recent progress in generative modeling, MBRL can solve control tasks from pixels and master several Atari games. However, MBRL still struggles to solve problems with highly complex observations. This is largely because most MBRL techniques today rely on reconstruction to learn their models, where the world model is trained to predict future frames at a pixel level. This makes MBRL susceptible to changes in how an environment is rendered. While an environment may have

*Equal contribution

Deep Reinforcement Learning Workshop at the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

fundamentally simple dynamics, it can be rendered with high fidelity textures, in 3D, or with a natural video playing in the background. All of these changes have been shown to cripple the performance of MBRL since a large amount of modeling capacity is taken up by modeling features that are irrelevant to the control task.

DreamerV2 [15], a state-of-the-art MBRL agent that learns a discrete world model, uses reconstruction to form its representations and thus suffers from this problem. The authors describe a modification that doesn't use a reconstruction loss, where it simply learns a latent representation that can predict rewards and the distribution of the next latent state. While learning a latent dynamics model this way should, in theory [9], be sufficient, empirical studies have shown that across multiple tasks, the signal from predicting future observations is critical and without it, performance suffers dramatically [1, 15].

In our work, we show a simple way to regain this performance without using reconstruction or even contrastive [20, 22] losses. Additionally, our method of leveraging image signals for representation learning without reconstruction is significantly more robust to distractors than DreamerV2 and can perform well even when a video is displayed in the background of the environment. Our method, which we call BLAST, consists of the following changes on top of DreamerV2:

- While DreamerV2 experiments with weighing the gradients that go to the prior and posterior in the KL loss, we show that **by completely stopping the gradient from going to the posterior**, the KL loss encourages representations that are informative of the predictable features of the observations.
- Since our method does not update the posterior to be predictable by the prior, we choose to use an **auto-regressive prior** [32] to better fit the latent distribution.
- Inspired by similarities to recent advancements in self-supervised learning, we use techniques from Bootstrap Your Own Latent (BYOL) [10] such as a **slowly updating target network for prior targets to stabilize training** and using **batch normalization [16] in the encoder**.

In order to evaluate our method, we run experiments on a highly-customizable grid world environment as well as continuous control experiments on the DeepMind Control Suite (DMC) [30]. In our grid world experiments, we evaluate the performance of our method on several rendering modifications, including environments with a small agent sprite as well as video backgrounds. We show empirically that DreamerV2 performs poorly on these environments while BLAST can learn a strong world model. On DMC, we find that BLAST can match the performance of several other MBRL agents without relying on reconstruction or contrastive losses. We conduct an extensive ablation study on the proposed changes and find that while all of the changes improve performance, batch normalization makes a substantial difference in several environments. Overall, BLAST represents a simple set of changes to DreamerV2 that enables practical reconstruction-free MBRL.

2 BLAST: Bootstrapped LAtents for Simulating Trajectories

2.1 Problem Formulation

In reinforcement learning, an agent acts in an unknown environment and optimizes its actions to increase its reward. In this work, we assume the agent is interacting with a POMDP, or partially observable Markov decision process. A POMDP is defined by a tuple $(S, A, T, R, \gamma, \Omega, O)$. S is a set of states; A is a set of actions; the transition probabilities $T : S \times A \times S \to [0, 1]$ define the probability of the environment transitioning from state s to state s' given that the agent acts with action a; the reward function $R : S \times A \to \mathbb{R}$ deterministically maps a state-action transition to a real number; $0 \leq \gamma \leq 1$ is the discount factor, which controls how much an agent should value rewards sooner rather than later. Since a POMDP is partially observable, it also includes Ω, a set of observations, and $O : S \times O \to [0, 1]$, a set of conditional probabilities defining how likely a specific observation is for an underlying state. The agent only receives observations; the true state of the system is unknown and can only be inferred using the observation and action history, the set of which we will refer to as belief states B. An agent acts in the POMDP with a policy $\pi : B \times A \to [0, 1]$, which determines the probability that it takes action a while in belief state b.

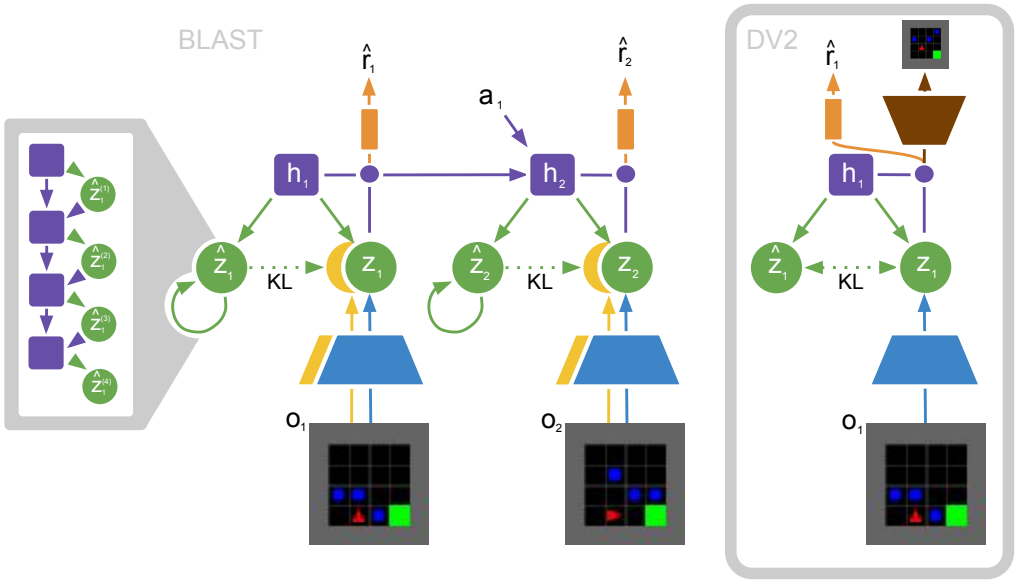

Figure 1: In BLAST, we modify DreamerV2 by removing its reconstruction loss, adding a stop gradient to prevent the posterior from move towards the prior *at all*, introduce a target encoder and make the prior autoregresive.

The expected return of a policy is denoted:

$$J_{\mathrm{RL}}(\pi) = \mathbb{E}_{\tau \sim P(\tau|\pi)} \left[\sum_t \gamma^t R(s_t, a_t) \right] \tag{2.1}$$

A fundamental challenge in RL is that we do not have access to the ground truth environment dynamics. Instead, an agent interacts with the POMDP by taking actions and receiving rewards. Since this interaction can be costly in environments that are expensive to simulate or in real life, MBRL is seen as a promising approach to improving sample efficiency, since it may be faster to learn to model the dynamics of the POMDP than to learn a policy through interaction.

2.2 Modeling Dynamics in a Latent Space

As in prior works [12, 14, 13, 15], we wish to learn a *latent dynamics model*, where a model is trained to predict the future conditioned on actions in a compact latent space rather than in observation space. By predicting the future in this latent space, a latent dynamics model avoids the memory and compute intensive task of having to predict each frame when simulating the environment, allowing thousands of imagined trajectories to be simulated in parallel on a GPU. A policy can also be learned directly on this latent state, potentially accelerating policy learning since the policy does not have to directly deal with high dimensional visual inputs.

Most prior works use reconstruction to form the latent representation. While the observations contain important signals that can accelerate learning, precisely reconstructing image observations has many disadvantages. Losses in pixel space (e.g. mean squared error) typically do not place the most weight on reconstructing the most "important" elements in the frame. World models learned using reconstruction can fail to learn when features important to acting are represented by only a small percentage of pixels (e.g. the ball in Pong). In high fidelity or real world scenes where most pixels are taken up by uncontrollable elements like the clouds or a billboard, reconstruction is not a feasible approach.

2.3 Dreamer without Reconstruction

In this work, we build off of DreamerV2, a state-of-the-art MBRL algorithm. In Dreamer [13, 15], a latent dynamics model called a Recurrent State-Space Model (RSSM) [13] is learned. This model consists of:

$$\begin{aligned}
&\text{Recurrent model:} && h_t = f_\theta(h_{t-1}, z_{t-1}, a_{t-1}) \\
&\text{Representation model:} && z_t \sim q_\theta(z_t | h_t, x_t) \\
&\text{Transition predictor:} && \hat{z}_t \sim p_\theta(\hat{z}_t | h_t) \\
&\text{Image predictor:} && \hat{x}_t \sim p_\theta(\hat{x}_t | h_t, z_t) \\
&\text{Reward predictor:} && \hat{r}_t \sim p_\theta(\hat{r}_t | h_t, z_t) \\
&\text{Discount predictor:} && \hat{\gamma}_t \sim p_\theta(\hat{\gamma}_t | h_t, z_t).
\end{aligned} \tag{2.2}$$

The world model is trained by minimizing the following loss, which corresponds to an ELBO of a hidden Markov model conditioned on the action sequence, making the RSSM a sequential VAE [15, 18]:

$$\mathcal{L}(\theta) \doteq \mathrm{E}_{q_\theta(z_{1:T} | a_{1:T}, x_{1:T})} \Bigg[\sum_{t=1}^{T} \underbrace{-\beta_{\text{img}} \ln p_\theta(x_t | h_t, z_t)}_{\text{image log loss}} \underbrace{-\beta_r \ln p_\theta(r_t | h_t, z_t)}_{\text{reward log loss}} \underbrace{-\beta_\gamma \ln p_\theta(\gamma_t | h_t, z_t)}_{\text{discount log loss}} \tag{2.3}$$

$$\underbrace{+\beta_{\text{KL}} \mathrm{KL}(q_\theta(z_t | h_t, x_t) \| p_\theta(z_t | h_t))}_{\text{KL loss}} \Bigg]$$

In the original work [13, 14, 15], various representation learning ablations were done, including the removal of the image log loss. However, without image gradients to help form representations, Dreamer fails to learn an accurate world model.

While this naive reward-only version of Dreamer might not work well in practice, there have been several works that learn representations in Dreamer without reconstruction, primarily through contrastive learning. The original Dreamer paper [14] used contrastive learning to learn a representation that has maximal mutual information with the encoded observation. Temporal Predictive Coding (TPC) [20] proposes to augment this loss with a contrastive approach that operates between timesteps, encouraging representations of predictable elements. While Dreamer with contrastive representations did not achieve performance on par with reconstruction-based representations, TPC is competitive with Dreamer with reconstruction on several tasks. Theoretically, TPC learns representations that are sufficient for control, but it is unclear whether the loss is still a lower bound on the log probability of the data. In practice, TPC requires carefully balancing four separate losses to avoid collapsed representations in practice.

2.3.1 Learning Bootstrapped Representations

In order to learn a world model with neither reconstruction nor contrastive losses, we note that there are two ways to optimize the KL loss in DreamerV2. We can change the latent representations from the past to be more informative of future representations (forward) or we can change our future representation to be more predictable by the past (reverse). DreamerV2 [15] introduces an additional hyperparameter called the KL balance, where a parameter α is used to scale the gradients going into the prior and posterior in the KL loss. In experiments on Atari, DreamerV2 uses a KL balance of $\alpha = 0.8$, which updates both the posterior and prior distributions. In this section, we argue that by completely stopping the gradients that update the future representation ($\alpha = 1$), we are able to learn representations not only from the reward signal, but from bootstrapping off of the information present in the embeddings of future observations.

By only allowing gradients to flow into the prior and not the posterior, we recover a representation learning algorithm similar to BYOL [10], a self-supervised learning algorithm that has achieved strong performance on benchmarks without the use of the negative samples that are necessary for contrastive learning. Intuitively, even the representation produced by our encoder at initialization has some information about the observation it is encoding. By predicting this future representation, the past representations are encouraged to contain information that can help with the prediction task, which in turn increases the amount of information that is present in the representation.

In order to stabilize training, we use techniques from BYOL. Primarily, we employ a slowly updating target encoder with parameters ξ to produce the posterior, resulting in the following loss:

$$\mathcal{L}(\theta) \doteq \mathrm{E}_{q_\theta(z_{1:T}|a_{1:T},x_{1:T})}\Big[\sum_{t=1}^{T} \underbrace{-\beta_r \ln p_\theta(r_t|h_t,z_t)}_{\text{reward log loss}} \underbrace{-\beta_\gamma \ln p_\theta(\gamma_t|h_t,z_t)}_{\text{discount log loss}} \underbrace{+\beta_{\text{KL}}\text{KL}(\text{stop\_grad}(q_\xi(z_t|h_t,x_t))\|p_\theta(z_t|h_t))}_{\text{KL loss}}\Big] \tag{2.4}$$

The target encoder's parameters ξ are updated as an exponential moving average of θ:

$$\xi \leftarrow \tau\xi + (1-\tau)\theta$$

2.3.2 Auto-Regressive Prediction of Latent Representations

While BYOL uses continuous representations and predicts the mean, we would like to use our prediction model to make accurate predictions about future latent states, even in stochastic environments where they may be multi-modal. Like in DreamerV2, [15], we use k discrete categorical variables for each latent representation. Unlike DreamerV2, an independent prior will likely not be able to accurately fit the distribution of future latents since we do not update the encoder to be predictable when we use a KL balance of $\alpha = 1$.

Therefore, for the prior $p_\theta(z_t|h_t)$ we use a powerful auto-regressive model so that we can accurately capture the joint distribution of these variables. Since the prior predicts the distribution of future observations encoded by the target representation encoder, during training we set the inputs to the prior model to come from both the online encoder and the target encoder.

2.3.3 Batch Normalization

It has been noted that while batch normalization [16] is not strictly necessary to make BYOL work [26]. However, others have proposed that batch normalization plays a similar role in BYOL as negative samples in contrastive learning. This stems from the fact that both batch normalization and negative samples effectively push representations within a batch apart from each other [7]. In our experiments, we found that batch normalization can dramatically improve stability and performance.

3 Experiments

We run experiments to evaluate the following hypotheses:

- (H1) The combination of proposed changes (BLAST) enable the learning of a strong world model without the use of reconstruction.
- (H2) BLAST is more robust to changes in environment rendering compared to DreamerV2.
- (H3) The most effective combination of the proposed changes is the combination of all five, which we call BLAST.

3.1 Environment

We primarily run experiments on a modified grid world environment as well as the DeepMind Control Suite (DMC) [30]. We choose to use the dynamic obstacles environment in gym-minigrid [4]. In this environment, three obstacles randomly move to adjacent tiles each time-step and the agent must navigate around them to reach a goal without colliding with any obstacles. This environment was primarily chosen since it is easy to modify while retaining characteristics that make it a good test-bed for testing world models, such as environment stochasticity and sparse reward. We also chose to evaluate on DMC to allow comparison to prior work which tested on this benchmark.

3.1.1 Environment Rendering Modifications

We modified the base environment `MiniGrid-Dynamic-Obstacles-6x6-v0` in several ways to evaluate robustness to different rendering styles. While prior works primarily evaluate robustness to videos or images placed in the backgrounds of visual environments, we want to evaluate robustness to other scenarios that could potentially harm reconstruction-based agents in order to evaluate (H2).

Examples observations from these environments are shown in Figure 2.

- `color_direction`: A different color is used to represent each of the possible directions that the agent could be facing.
  Scenario: *uncertainty in reconstruction can be confused with another possible observation.*
- `smaller_agent`: The agent is rendered as a triangle using one quarter of the pixels as in the original environment.
  Scenario: *important elements have a low weight in the reconstruction loss.*
- `random_frames`: The background is set to a random frame of a fixed video at each time-step.
  Scenario: *environment observations have temporally uncorrelated and uncontrollable elements in the background.*
- `video`: The background is set to a random frame of a fixed video upon resetting the environment. The video is then displayed in order in subsequent steps.
  Scenario: *environment observations have temporally correlated and uncontrollable elements in the background.*

For all experiments, we used a fixed horizon of 100 time-steps. The return is the number of times the agent reaches the green square in the episode minus the number of times it collides with an obstacle. For video background we use a video drawn from the the kinetics400 [17] driving class across all experiments.

3.2 Experimental Setup

We ran all experiments using code forked from the official DreamerV2 [15] implementation. The code was modified to allow the use of batch normalization, a separate target encoder with weights set to an exponentially moving average of online weights, and an autoregressive prior within the RSSM. This allows us to configure DreamerV2 with any subset of our proposed changes and fairly compare the different configurations. We opted to fix hyperparameters for individual environments rather than individually tune each configuration of the agent since we found that generally, hyperparameters that worked well on the base DreamerV2 agent also worked well on BLAST. More details about hyperparameter and how to reproduce our experiments can be found in appendix A.1. For our plots, we plot the mean over 5 seeds for grid world environments and 6 seeds[2] for DMC tasks. Error bars represent standard deviation.

3.3 Summary of Modifications

- **(-recon)** We remove the reconstruction loss from DreamerV2 by setting its weight to zero.
- **(+SG)** We add a stop gradient to the posterior in the KL loss by setting a KL balance of $\alpha = 1$.
- **(+EMA)** We use an exponential moving average for the encoder when we compute the posterior in the KL loss.
- **(+BN)** We add batch normalization to the encoder.
- **(+AR)** We use an autoregressive prior in the RSSM to better fit the discrete latent distribution.

3.4 Grid World Experiments

Figure 3 shows how performance differs on the grid world environments when the proposed modifications are added to DreamerV2 to create BLAST. DreamerV2 works well on the unmodified

[2]We used data directly from the paper [20] for TPC results. Due to some crashed runs, we were unable to get 6 seeds for all configurations under our time constraint. Please see Appendix A.6 for more details.

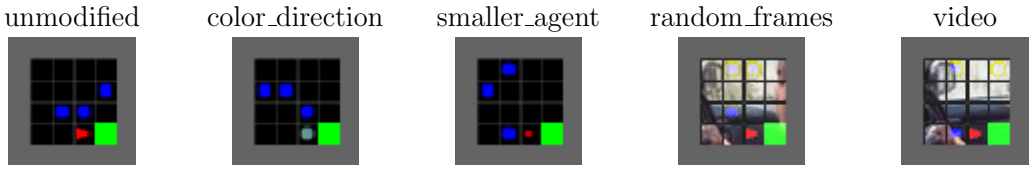

Figure 2: Observations from our modified grid world environments.

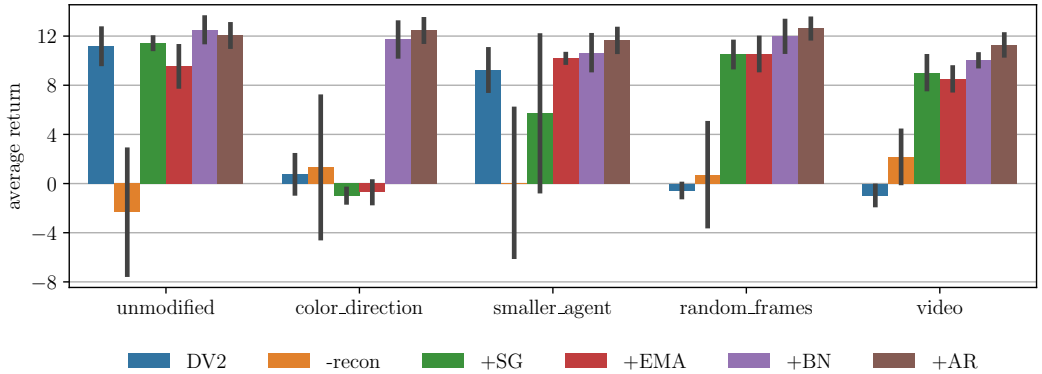

Figure 3: We performed a series of experiments to understand how our proposed changes affect performance across our various grid world modifications. Each subsequent change includes all of the modifications to the left. For example, the rightmost +AR represents the combination of all five changes. DreamerV2 struggles on the modified environments while BLAST (+AR) is robust to all rendering types and performs well. In general, BLAST (+AR) and BLAST without AR (+BN) are the only two configurations that consistently work well. Error bars represent standard deviation.

environment but struggles to learn in the presence of most of the rendering modifications. BLAST is able to achieve strong performance on all of the rendering modifications, confirming our hypothesis that BLAST would be more robust to adversarial rendering (H2). We found that only BLAST (+AR) and BLAST without an autoregressive prior (+BN) consistently performed well. We note that there are some environments such as `color_direction` where the use of batch normalization is vital to achieve strong performance.

We also display training curves in Figure 7 in the appendix, which shows that BLAST can be significantly more sample efficient than DreamerV2 with the same hyperparameters. We believe this is primarily due to not needing to train a decoder.

3.5 DeepMind Control Suite

Our experiments on the modified grid world environments confirm that BLAST is significantly more robust to rendering changes than DreamerV2. In order to compare with prior state of the art model based RL algorithms, we ran continuous control experiments on several standard DMC environments. Figure 4 shows that BLAST can achieve performance that is essentially on par with vanilla DreamerV2 and Dreamer. Surprisingly, we found that batch normalization had an even larger effect on these environments, with several tasks achieving practically zero return until batch normalization was added. We also found that the autoregressive prior did not have a significant effect on performance in DMC environments, likely due to the deterministic nature of the environment. Figure 4 also shows that BLAST achieves comparable performance to TPC, both in terms of asymptotic performance as well as sample complexity. Despite using neither reconstruction nor contrastive losses, BLAST achieves performance that is on par with existing state-of-the-art MBRL approaches on DMC tasks.

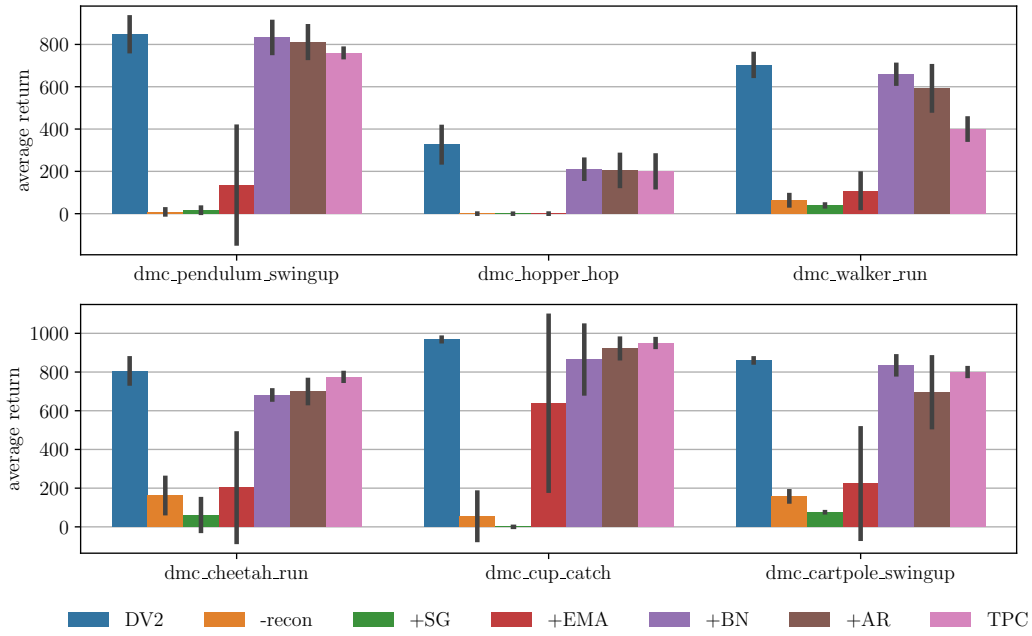

Figure 4: We performed a series of experiments to understand how our proposed changes affect performance across various continuous control tasks in DMC. Each subsequent change includes all of the modifications to the left. For example, the rightmost +AR represents the combination of all five changes. BLAST achieves comparable performance to DreamerV2 and TPC without the use of negative samples or reconstruction. Batch normalization has a large effect on most of the tested tasks. Error bars represent standard deviation.

4 Related Work

Self-supervised learning In recent years, contrastive methods have largely overtaken their unsupervised counterparts as the state-of-the-art technique for visual representations learning [3, 23]. These methods learn their representation using pairs of views on the same underlying data. These pairs come in two flavors: positive pairs where the views are of the same underlying datum and negative pairs where they are not. Contrastive methods produce a latent space where positive pairs converge and negative pairs spread apart. With the correct setup, this technique can be shown to optimize a lower bound on the mutual information between the input images and the latent space [23].

Recent work in the self-supervised learning literature has shown that the negative samples that characterize contrastive learning are unnecessary to learn good representations. Like contrastive learning approaches, these works employ matching latent representations. They differentiate by disposing of negative pairs and thus remove the need to store a bank of negative samples and function well with lower batch sizes [2, 10].

Latent dynamics models Traditional MBRL methods work best on low dimensional state representations [13, 5]. One of the most successful technique for scaling MBRL has been the use of latent dynamics models. In a latent dynamics model, high-dimensional observations are encoded into a compact latent state. Then these latent states are used to predict future latent states, rewards and discount factors. This process allows for training to proceed entirely within the latent dynamics model [12, 14, 13, 15]. These models generally achieve this accuracy by using variational methods that are primarily or entirely driven by a reconstruction loss. In theory, this reconstruction loss ensures that all the information contained in the high-dimensional observations finds its way into the latent states and in theory helps to ensure the world model's accuracy.

Robust world models Though effective for simple environments, latent dynamics models that rely on reconstruction can fail to learn in more realistic settings [8, 22, 20, 15]. They can be distracted by

irrelevant information in their inputs, like video backgrounds, and have issues learning to model small or fast-moving objects [8, 22, 20, 33]. Several papers attempt to alleviate these issues. Paster et al. [25] learns a latent world model using inverse dynamics models to model only controllable features in an environment, but the method is primarily used on goal-directed tasks. Fu et al. [8] attempts to solve the problem of distracting background imaginary by learning a partitioned state space using an adversarial objective. Their objective tries to ensure that the reward is only recoverable from half of the state space. The method can handle distracting backgrounds. However, it does not handle cases where the environment combines relevant and irrelevant information in a way that a simple pixel mask cannot partition.

Perhaps most similar to our work is that of Nguyen et al. [20], Okada and Taniguchi [22] and Ma et al. [19]. These works attempt to learn world models purely using contrastive losses. Nguyen et al. [20] finds improved performance over DreamerV1 on the DeepMind Control Suite [30]. In addition to this, despite not explicitly discouraging irrelevant information from entering the latent representations, the authors find substantially improved performance on video background environments. Similarly, Okada and Taniguchi [22] finds that their model sees improved results over DreamerV1 in environments that require the agent to perceive fine details. While these works learn representations that can avoid being distracted and resolve environments with fine detail, they still require negative samples. BLAST does not.

Learning from just rewards Another successful method of learning world models stems from the Value Prediction Networks and MuZero [29, 21, 27]. These methods learn their environment model by predicting just rewards, values and actions derived by planning. While these MBRL methods have proven spectacularly successful [27], they cannot handle stochastic or partially observable environments and are computationally expensive to train since they do not make use of the training signal provided by observations [24, 28]. Recent work has attempted to address these issues in MuZero by augmenting the method with pre-trained embedding that tries to represent the environment's stochasticity [24]. However, this method again relies on reconstruction to produce these embeddings in the form of a Vector Quantised-Variational AutoEncoder [31]. Finally, there are several works that attempt to address the problem of representation learning with a focus on predicting future latent representations and bisimulation metrics [6]. While Zhang et al. [34], Gelada et al. [9], Guo et al. [11] learn dynamics models, they are only used for representation learning and never for planning.

5 Discussion

We present BLAST, a modification of DreamerV2 which adds a stop-gradient to the posterior, a powerful auto-regressive prior, and a slowly updating target network to learn representations without using reconstruction. BLAST performs far better than DreamerV2 on a range of differently rendered grid world environments, including ones with a video embedded in the background where DreamerV2 fails entirely to learn. We empirically justify our modifications and show evidence that adding the stop-gradient encourages the latent representations to contain more information about the observation.

While BLAST shows strong performance in our grid world environments as well as on continuous control tasks, we acknowledge several limitations and opportunities for future work. The primary limitation of this work is the need for an expressive prior such as an autoregressive model in stochastic environments. Since autoregressive models are considerably slower than the independent prior used in DreamerV2, our model takes longer to run simulations. We also believe that more work should be done to differentiate self-predictive approaches to representation learning such as BLAST to contrastive approaches in order to provide a stronger recommendation to practitioners looking to use reconstruction-free world models.

Acknowledgements

We gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chairs Program, and Microsoft Research. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute for Artificial Intelligence (www.vectorinstitute.ai/partners).

References

[1] M. Babaeizadeh, M. T. Saffar, D. Hafner, H. Kannan, C. Finn, S. Levine, and D. Erhan. Models, Pixels, and Rewards: Evaluating Design Trade-offs in Visual Model-Based Reinforcement Learning. *CoRR*, abs/2012.04603, 2020. URL https://arxiv.org/abs/2012.04603.

[2] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, NeurIPS 2020, december 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/70feb62b69f16e0238f741fab228fec2-Abstract.html.

[3] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th international conference on machine learning, ICML 2020, 13-18 july 2020, virtual event*, pages 1597–1607, 2020. URL http://proceedings.mlr.press/v119/chen20j.html.

[4] M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic Gridworld Environment for OpenAI Gym, 2018. URL https://github.com/maximecb/gym-minigrid. Publication Title: GitHub repository.

[5] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4759–4770, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3de568f8597b94bda53149c7d7f5958c-Abstract.html.

[6] N. Ferns, P. Panangaden, and D. Precup. Bisimulation metrics for continuous markov decision processes. *SIAM J. Comput.*, 40(6):1662–1714, 2011. doi: 10.1137/10080484X. URL https://doi.org/10.1137/10080484X.

[7] A. Fetterman and J. Albrecht. Understanding self-supervised and contrastive learning with "Bootstrap Your Own Latent" (BYOL). URL https://generallyintelligent.ai/blog/2020-08-24-understanding-self-supervised-contrastive-learning/.

[8] X. Fu, G. Yang, P. Agrawal, and T. Jaakkola. Learning task informed abstractions. In M. Meila and T. Zhang, editors, *Proceedings of the 38th international conference on machine learning*, volume 139 of *Proceedings of machine learning research*, pages 3480–3491. PMLR, July 2021.

[9] C. Gelada, S. Kumar, J. Buckman, O. Nachum, and M. G. Bellemare. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2170–2179. PMLR, May 2019. URL https://proceedings.mlr.press/v97/gelada19a.html. ISSN: 2640-3498.

[10] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Pires, Z. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, NeurIPS 2020, december 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html.

[11] Z. D. Guo, B. A. Pires, B. Piot, J.-B. Grill, F. Altché, R. Munos, and M. G. Azar. Bootstrap Latent-Predictive Representations for Multitask Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3875–3886. PMLR, Nov. 2020. URL https://proceedings.mlr.press/v119/guo20g.html. ISSN: 2640-3498.

[12] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in neural information processing systems 31: Annual conference on neural information processing systems 2018, NeurIPS 2018, december 3-8, 2018, montréal, canada*, pages 2455–2467, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

[13] D. Hafner, T. P. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 june 2019, long beach, california, USA*, volume 97 of *Proceedings of machine learning research*, pages 2555–2565. PMLR, 2019. URL http://proceedings.mlr.press/v97/hafner19a.html.

[14] D. Hafner, T. P. Lillicrap, J. Ba, and M. Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

[15] D. Hafner, T. P. Lillicrap, M. Norouzi, and J. Ba. Mastering Atari with Discrete World Models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

[16] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456. PMLR, June 2015. URL https://proceedings.mlr.press/v37/ioffe15.html. ISSN: 1938-7228.

[17] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. URL http://arxiv.org/abs/1705.06950.

[18] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

[19] X. Ma, S. Chen, D. Hsu, and W. S. Lee. Contrastive variational model-based reinforcement learning for complex observations. *CoRR*, abs/2008.02430, 2020. URL https://arxiv.org/abs/2008.02430. arXiv: 2008.02430 tex.bibsource: dblp computer science bibliography, https://dblp.org tex.biburl: https://dblp.org/rec/journals/corr/abs-2008-02430.bib tex.timestamp: Mon, 24 Aug 2020 13:03:01 +0200.

[20] T. D. Nguyen, R. Shu, T. Pham, H. Bui, and S. Ermon. Temporal Predictive Coding For Model-Based Planning In Latent Space. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 8130–8139, 2021. URL http://proceedings.mlr.press/v139/nguyen21h.html.

[21] J. Oh, S. Singh, and H. Lee. Value prediction network. In *Advances in neural information processing systems 30: Annual conference on neural information processing systems 2017, december 4-9, 2017, long beach, CA, USA*, pages 6118–6128, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/ffbd6cbb019a1413183c8d08f2929307-Abstract.html.

[22] M. Okada and T. Taniguchi. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction. *CoRR*, abs/2007.14535, 2020. URL https://arxiv.org/abs/2007.14535.

[23] A. v. d. Oord, Y. Li, and O. Vinyals. Representation Learning with Contrastive Predictive Coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

[24] S. Ozair, Y. Li, A. Razavi, I. Antonoglou, A. van den Oord, and O. Vinyals. Vector quantized models for planning. In *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 july 2021, virtual event*, pages 8302–8313, 2021. URL http://proceedings.mlr.press/v139/ozair21a.html.

[25] K. Paster, S. A. McIlraith, and J. Ba. Planning from Pixels using Inverse Dynamics Models. Sept. 2020. URL https://openreview.net/forum?id=V6BjBgku7Ro.

[26] P. H. Richemond, J.-B. Grill, F. Altché, C. Tallec, F. Strub, A. Brock, S. Smith, S. De, R. Pascanu, B. Piot, and M. Valko. BYOL works even without batch statistics. *arXiv:2010.10241 [cs, stat]*, Oct. 2020. URL http://arxiv.org/abs/2010.10241. arXiv: 2010.10241.

[27] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. ISSN 0028-0836. doi: 10.1038/s41586-020-03051-4.

[28] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to Explore via Self-Supervised World Models. In *ICML*, 2020.

[29] D. Silver, H. Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, and T. Degris. The Predictron: End-To-End Learning and Planning.

In *Proceedings of the 34th International Conference on Machine Learning*, pages 3191–3199. PMLR, July 2017. URL https://proceedings.mlr.press/v70/silver17a.html. ISSN: 2640-3498.

[30] Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, and N. Heess. dm_control: Software and Tasks for Continuous Control, 2020. _eprint: 2006.12983.

[31] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *Advances in neural information processing systems 30: Annual conference on neural information processing systems 2017, december 4-9, 2017, long beach, CA, USA*, pages 6306–6315, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[33] A. Zhang, Y. Wu, and J. Pineau. Natural environment benchmarks for reinforcement learning. *CoRR*, abs/1811.06032, 2018. URL http://arxiv.org/abs/1811.06032.

[34] A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine. Learning Invariant Representations for Reinforcement Learning without Reconstruction. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

A Appendix

A.1 Experimental Details

For all of our experiments we built on top of the open source implementation of DreamerV2 provided by the authors at https://github.com/danijar/dreamerv2. Hyperparameters for the grid world experiments were optimized for maximum performance for DreamerV2 across the tested environments. The hyperparameters with the greatest effect on DreamerV2's performance were the dimension of the latent state `rssm.stoch` and the scale of the reward loss `loss_scales.reward`. We found that BLAST generally worked well for these same hyperparameters and therefore we fixed the hyperparameters across all of our configurations. For DMC experiments, we used the default hyperparameters for DreamerV2 on DMC and similarly kept them fixed across all of the tasks and configurations.

A.2 Hyperparameters

Gym-MiniGrid	
`prefill`	10,000
`train_every`	1
`steps`	5e4
`pred_discount`	False
`discount` γ	0.99
`rssm.hidden`	600
`rssm.deter`	600
`rssm.stoch`	16
`rssm.discrete`	64
`actor_grad`	reinforce
`loss_scales.kl`	0.1
`loss_scales.reward`	100.0
`reward_clamp`	tanh
`kl.balance`	0.8
DMC	
`prefill`	1000
`train_every`	5
`steps`	1e6
`pred_discount`	False
`discount` γ	0.99
`rssm.hidden`	200
`rssm.deter`	200
`rssm.stoch`	8
`rssm.discrete`	64
`actor_grad`	dynamics
`loss_scales.kl`	1.0
`loss_scales.reward`	1.0
`reward_clamp`	none
`kl.balance`	0.8
BLAST	
`kl.balance`	1.0
`encoder.norm`	bn
`rssm.obs_out_norm`	bn
`target_ema`	0.99
`target_input_p`	0.5
`rssm.ar_steps`	same as `rssm.stoch`
`use_target_encoder`	true

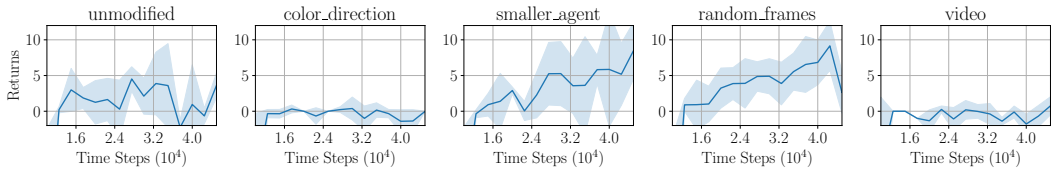

Figure 5: Without using reconstruction or reward gradients to form its latent representations, BLAST can learn to model the environment using the prediction of future latent alone.

A.3 Learning Representations with Only Latent Prediction

Due to its similarity to self-supervised learning algorithms like BYOL [10], we hypothesize that the signal from predicting future latents may be sufficient for learning representations in BLAST. To test this, we stopped the gradients from the reward and discount prediction heads of BLAST so that the world model and observation representation would only be updated with the KL loss. As demonstrated in Figure 5, BLAST, with no reward or reconstruction gradients, still manages to learn a surprisingly strong policy on the unmodified and random frames environment, empirically showing that **the latent prediction loss alone can be enough to learn a world model**.

A.4 Training Curves

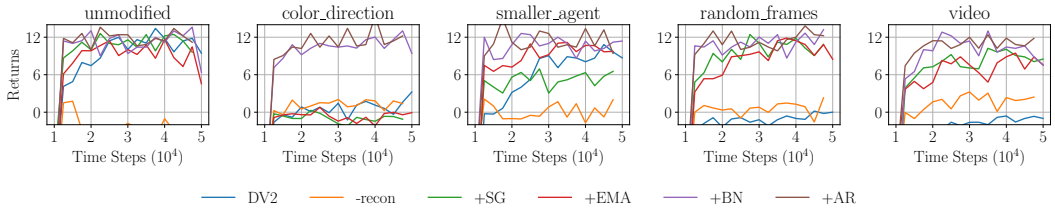

Figure 6: Training curves on our modified grid world environments.

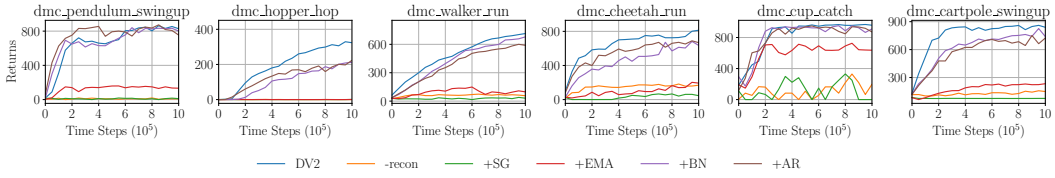

Figure 7: Training curves for DMC tasks.

A.5 Visual Reconstruction with Different Representations

In order to visually inspect the quality of the representations learned by DreamerV2 without reconstruction, we trained a reconstruction network on top of the learned representations. Figure 8 shows that even on an environment modified with a video playing in the background, BLAST improves the quality of representations and allows for a clearer reconstruction.

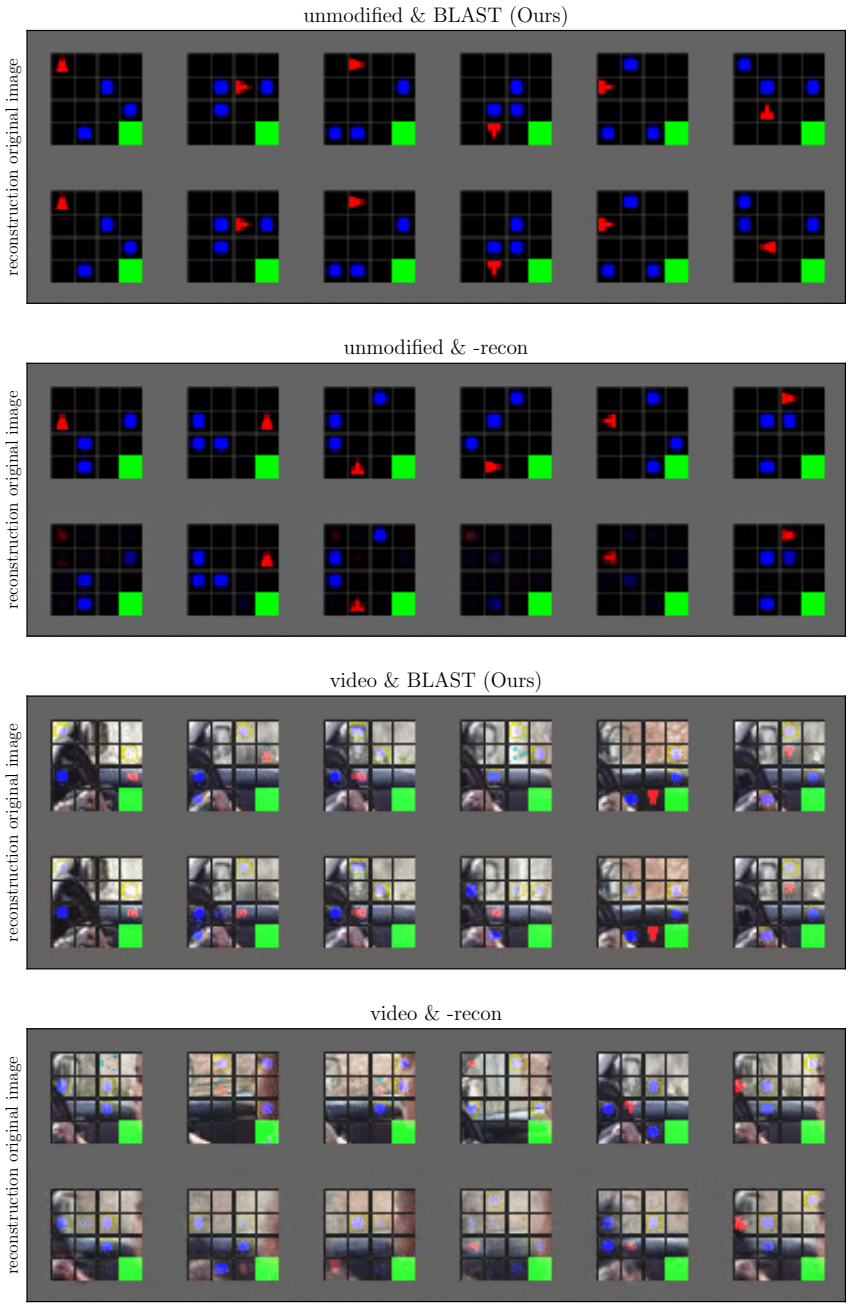

Figure 8: Examples of reconstructions of the tenth frame of a simulated roll out by DreamerV2 without reconstruction compared with BLAST. The decoder in all four experiments was only used for visualizations and all representations in the world model are learned only from reward, discount, and KL losses. BLAST produces noticably clearer reconstructions, showing that BLAST additions are helping to learn more accurate representations of the entire environment.

A.6 Number of Seeds for DMC Experiments

dmc_cartpole_swingup	
DV2	6 seeds
-recon	6 seeds
+SG	6 seeds
+EMA	5 seeds
+BN	4 seeds
+AR	5 seeds
dmc_cheetah_run	
DV2	6 seeds
-recon	6 seeds
+SG	3 seeds
+EMA	4 seeds
+BN	5 seeds
+AR	6 seeds
dmc_cup_catch	
DV2	6 seeds
-recon	6 seeds
+SG	2 seeds
+EMA	6 seeds
+BN	6 seeds
+AR	6 seeds
dmc_hopper_hop	
DV2	6 seeds
-recon	6 seeds
+SG	6 seeds
+EMA	6 seeds
+BN	6 seeds
+AR	4 seeds
dmc_pendulum_swingup	
DV2	6 seeds
-recon	6 seeds
+SG	6 seeds
+EMA	6 seeds
+BN	6 seeds
+AR	4 seeds
dmc_walker_run	
DV2	6 seeds
-recon	6 seeds
+SG	4 seeds
+EMA	2 seeds
+BN	5 seeds
+AR	5 seeds