BLAST: Latent Dynamics Models from Bootstrapping

Anonymous Author(s)
Affiliation
Address
email

Abstract

State-of-the-art world models such as DreamerV2 [18] have significantly improved the capabilities of model-based reinforcement learning. However, these approaches typically rely on reconstruction losses to shape their latent representations of the environment, which are known to fail in environments with high fidelity visual observations [11]. When learning latent dynamics models without reconstruction loss using only the signal present in the reward signal, the performance of these methods also drops dramatically [1]. We present a simple modification to DreamerV2 without reconstruction loss inspired by the recent self-supervised learning method Bootstrap Your Own Latent [13]. 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 and 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 to learning a larger set of more 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 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 [18], 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

Submitted to the Deep Reinforcement Learning Workshop at NeurIPS 2021. Do not distribute.

way should, in theory [12], 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, 18].

In our work, we show a simple way to regain this performance without using reconstruction or even contrastive [24, 26] 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 implemented with a transformer** [37] 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) [13] such as a **slowly updating target network for prior targets to stabilize training**.

In order to evaluate our method, we run experiments on a highly-customizable grid world environment. Despite the environment's simplicity, DreamerV2 without reconstruction loss performs poorly. We show empirically that adding the stop-gradient, auto-regressive prior, and target network improve performance to a level higher than the standard DreamerV2 *with reconstruction*. We also evaluate on several modifications to the rendering of our environment, including making the agent smaller and displaying a video in the background. While DreamerV2 cannot learn on any of these modified environments, BLAST converges to an optimal policy using less than twenty thousand time-steps.

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.

The expected return of a policy is denoted:

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

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.

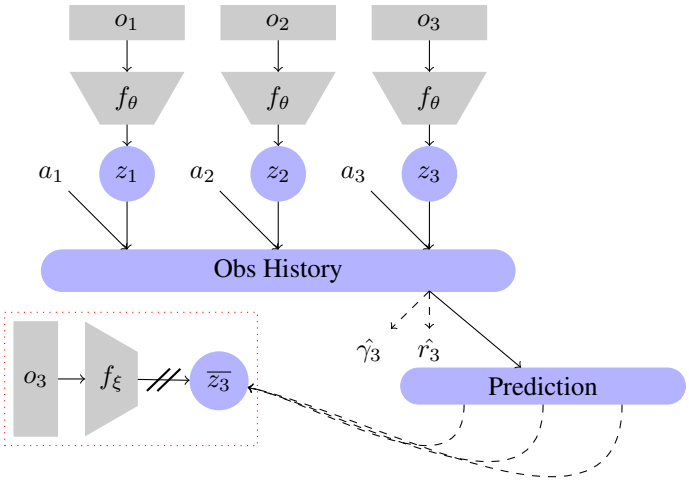

Figure 1: In BLAST, discrete latent representations of observations are learned that are predictive of rewards and future latent states. Unlike prior works, we use a stop-gradient operation so that the future latent representation is not updated to become more predictable by the past representations. We also employ a slow-moving target network to stabilize training.

2.2 Modeling Dynamics in a Latent Space

As in prior works [15, 17, 16, 18], 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 with Reward Representations

In this work, we largly build off of DreamerV2, a state-of-the-art MBRL algorithm. In Dreamer [16, 18], a latent dynamics model called a Recurrent State-Space Model (RSSM) [16] 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 [18, 21]:

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

In the original work [16, 17, 18], 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.

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

While this reward-only version of Dreamer might not work well in practice, there is precedent for learning representations that are predictive of rewards and future latent representations [12, 40]. Under some assumptions, Gelada et al. [12] prove that such objectives produce MDPs with values equivalent to the original MDP.

2.3.1 Learning Bootstrapped Representations

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 or we can change our future representation to be more predictable by the past. DreamerV2 [18] 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 transition predictor (prior) and not the future representation (posterior), we recover a representation learning algorithm similar to BYOL [13]. 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 stably learn such a representation, we use techniques from BYOL. Primarily, we employ a slowly updating target encoder to produce the prediction targets, resulting in the following loss:

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

The target encoder's parameters 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, [18], 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 model the joint distribution of these variables. Since the latent predictor models the distribution of future observations encoded by the target representation encoder, we set a fraction of the inputs to the transition predictor, reward predictor, and discount predictor to come from the target encoder as well.

2.3.3 Implementation

Our representation encoder is implemented using a ResNet [19, 8] followed by a gumbel-softmax [23, 20]. The observation-action history is encoded using a transformer decoder model [37] with causal attention masks rather than a GRU [6]. The hidden representations of the last layer of the transformer are used in place of h_t as input for the reward and discount predictors, implemented with an MLP, and the latent predictor, which is implemented using another transformer decoder. See Figure 1 for a visual description of our algorithm.

In the full version of our model, we also use an encoder $q_\xi(z_t|x_t)$ that does not take the previous hidden state h_t as input, which is equivalent in fully-observable environments. We also condition reward and discount predictions only on the current state h_t and not on a sample of the future state z_t. Unlike DreamerV2, BLAST uses batch normalization in its encoder.

3 Experiments

In our experiments, we aim to answer the following questions:

- How well does DreamerV2 without reconstruction loss perform on our test environment? Does adding a stop-gradient, auto-regressive prior, and target encoder (BLAST) improve performance?
- How robust is BLAST to changes in the rendering of the environment compared to DreamerV2 with reconstruction?
- Which aspects of our approach are most important? Can BLAST learn strong representations with the latent prediction loss alone (without reward or reconstruction gradients)?

3.1 Environment

We choose to evaluate our method on the dynamic obstacles environment in gym-minigrid [5]. While simple, this environment has some properties that make it a good test-bed for testing world models. The environment has three obstacles that randomly move to an adjacent tile at each time-step, making it necessary for the world model to accurately model the stochasticity of the environment in order to learn a strong policy. The environment is also easy to customize, making it easy to evaluate the robustness of an approach to different styles of rendering.

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. Examples observations from these environments are shown in Figure 3.

- `color_direction`: In this environment, we use a different color to represent each of the possible directions that the agent could be facing. In this environment, uncertainty in the reconstruction of a frame could be interpreted as certainty that the agent is facing a wrong direction.

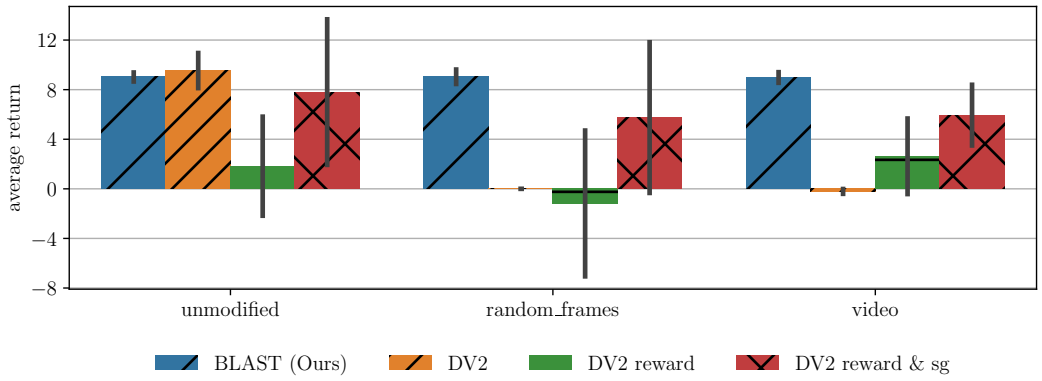

Figure 2: When DreamerV2 does not use reconstruction loss to learn its representation, its performance decreases. Much of this performance can be regained by adding a stop-gradient operation, but the results have a high variance. Adding a target encoder and auto-regressive prior (BLAST) results in stable learning across every environment. Bars show average performance between 40k and 50k time steps.

- `smaller_agent`: In this environment, the agent is rendered as a triangle using one quarter of the pixels as in the original environment.
- `random_frames`: In this environment, the background is set to a random frame of a video at each time-step.
- `video`: In this environment, the background is set to a random frame of a video upon resetting the environment. The video is then displayed in order in subsequent steps.

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.

3.2 Experimental Setup

We run all experiments with a replay buffer prefill of $10,000$ frames and run one step in the real environment for every $2,500$ simulated environment steps. Unless stated otherwise, we tune hyperparameters of BLAST and any baselines separately for maximum performance. More details on the hyperparameters used for this experiment can be found in appendix A.1. We run three seeds and report the standard deviation as well as the mean total return for each experiment.

Since BLAST primarily consists of simple changes on top of DreamerV2 [18], we directly compare to the open-source implementation provided by the authors.

The primary focus of this work is to present a novel way to learn world models and show their benefit over prior approaches in visually distracting tasks. As such, we preserve as much of the approach to policy learning as our main baseline, DreamerV2, as possible. Since the transition encoder predicts the distribution of latents produced by the target encoder, we use the output of the target encoder to produce the representations for the policy and value functions. Since we use a transformer rather than a GRU, we use four stacked latent variables as our policy and value representations.

3.3 From DreamerV2 to BLAST

In order to test how the reward representations, stop-gradient, auto-regressive prior, and target encoder, affect performance, we tested several versions of DreamerV2 as well as BLAST on the base grid world as well as environments with video backgrounds:

- **DreamerV2**: an unmodified baseline version of DreamerV2.
- **DreamerV2 rewad**: a modified version of DreamerV2 where gradients are cut off from the reconstruction head [18]. In this version, representations are learned just from reward and discount prediction.

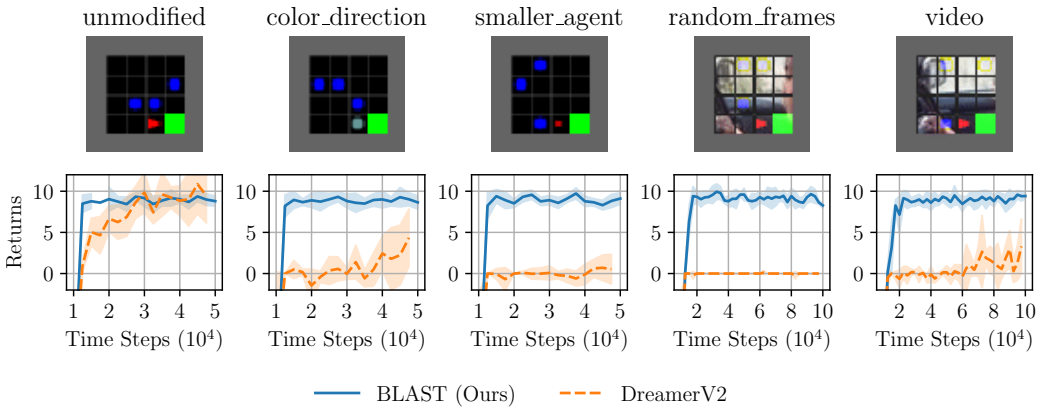

Figure 3: BLAST can still learn an accurate world model and performant policy even in the presence of adversarial rendering such as video backgrounds and rendering the agent with a very small number of pixels.

- **DreamerV2 reward & stop grad**: on top of the previous modification, we test a version of DreamerV2 with a stop-gradient on the posterior in the KL loss, implemented using KL balancing of $\alpha = 1$

- **BLAST (Ours)**: our full set of modifications including a stop-gradient on the posterior, auto-regressive prior, and a slowly updating target network.

As shown in Figure 5, DreamerV2 with reconstruction works well on the base environment but struggles on the environments with video backgrounds. The reward-only version fails to learn a strong policy on any environment, but when the stop-gradient is added, DreamerV2 can learn on every environment, albeit with significant variance and slightly suboptimal performance, likely due to the independence assumption made by the prior. Interestingly, from visualizing a reconstruction network trained on top of the detached representations, Figure 6 in the appendix shows that DreamerV2 with a stop-gradient indeed can reconstruct the observations far better than with KL balance $\alpha = 0.8$, even in video environments.

Finally, BLAST, which uses a stop-gradient, auto-regressive prior, and target network, achieves maximum performance on all environments.

3.4 Robustness to Rendering Changes

In order to test how robust both BLAST and DreamerV2 are to different ways of rendering, we ran both MBRL algorithms on our base environment as well as on environments with the four rendering modifications described in section 3.1.1.

Figure 3 shows that **BLAST is robust to all tested rendering modifications**, while DreamerV2 struggles to learn in all but the unmodified environment. Interestingly, it seems even reducing the number of pixels used to render the agent is enough to greatly reduce the performance of reconstruction-based approaches. The background in `random_frames` is especially simple for our method to deal with, since the background of subsequent frames is completely independent from previous observations. In `video`, BLAST can simply keep track of the time-step of the video. DreamerV2, on the other hand, has to learn to reconstruct each frame of the video. On `random_frames`, DreamerV2 has to learn to sample uniformly among all the frames of the video, making it the most challenging environment for the baseline.

We also note that even in the unmodified environment, BLAST converges to an optimal policy in slightly fewer samples and has significantly less variance than Dreamer. We believe that our method may be faster since it does not have the overhead of learning a decoder.

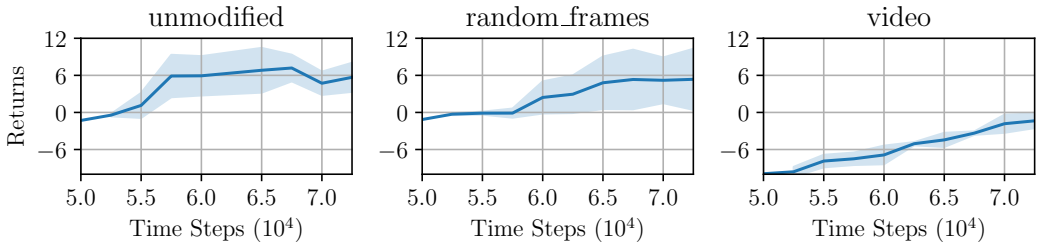

Figure 4: Without using reconstruction or reward gradients to form its latent representations, BLAST can learn to model the environment using the prediction of future latent alone. Note that a prefill of 50000 time steps was used for these experiments.

3.5 Learning Representations with Only Latent Prediction

Due to its similarity to self-supervised learning algorithms like BYOL [13], 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 4, 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 is enough to learn a world model**.

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 [4, 27]. 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 [27].

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 [3, 13].

Latent dynamics models Traditional MBRL methods work best on low dimensional state representations [16, 7]. 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 [15, 17, 16, 18]. 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 [11, 26, 24, 18]. They can be distracted by irrelevant information in their inputs, like video backgrounds, and have issues learning to model small or fast-moving objects [11, 26, 24, 39]. Several papers attempt to alleviate these issues. Paster et al. [29] 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. [11] 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. [24], Okada and Taniguchi [26] and Ma et al. [22]. These works attempt to learn world models purely using contrastive losses. Nguyen et al. [24] finds improved performance over DreamerV1 on the DeepMind Control Suite [35]. 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 [26] 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 [34, 25, 32]. These methods learn their environment model by predicting just rewards, values and actions derived by planning. While these MBRL methods have proven spectacularly successful [32], 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 [28, 33]. 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 [28]. However, this method again relies on reconstruction to produce these embeddings in the form of a Vector Quantised-Variational AutoEncoder [36]. 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 [10]. While Zhang et al. [40], Gelada et al. [12], Guo et al. [14] 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 on our toy environment, we acknowledge that more work needs to be done to show that it scales to larger problems. The primary limitation of this work is the need for an expressive latent predictor such as an auto-regressive model. Since there is no incentive for the latent representation to fit some prior distribution, we cannot assume they are in a computationally convenient form to model as in prior works. Since BLAST uses an auto-regressive predictor, there is at least a linear relationship between the complexity of running simulations in the world model and the number of categorical variables that each state is modeled with, making BLAST slower to run than DreamerV2.

Our current approach of using four categorical variables with 64 events each gives each state approximately 16 million possible values, which may not be enough for more complicated environments. Using a more computationally tractable approach for latent prediction is a promising future direction.

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] L. Biewald. Experiment Tracking with Weights and Biases, 2020. URL https://www.wandb.com/.

[3] 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.

[4] 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.

[5] 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.

[6] K. Cho, B. v. Merrienboer, Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014. doi: 10.3115/v1/d14-1179.

[7] 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.

[8] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1407–1416. PMLR, July 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html. ISSN: 2640-3498.

[9] W. Falcon and The PyTorch Lightning team. PyTorch Lightning, Mar. 2019. URL https://github.com/PyTorchLightning/pytorch-lightning.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. doi: 10.1109/CVPR.2016.90. ISSN: 1063-6919.

[20] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

[21] 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.

[22] 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.

[23] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *5th international conference on learning representations, ICLR 2017, toulon, france, april 24-26, 2017, conference track proceedings*, 2017. URL https://openreview.net/forum?id=S1jE5L5gl.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito,

M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[31] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-Art Natural Language Processing, Oct. 2020. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6. Pages: 38–45 original-date: 2018-10-29T13:56:00Z.

[39] 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.

[40] 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.