MASTERING ATARI WITH DISCRETE WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Intelligent agents need to generalize from past experience to achieve goals in complex environments. World models facilitate such generalization and allow learning behaviors from imagined outcomes to increase sample-efficiency. While learning world models from image inputs has recently become feasible for some tasks, modeling Atari games accurately enough to derive successful behaviors has remained an open challenge for many years. We introduce DreamerV2, a reinforcement learning agent that learns behaviors purely from predictions in the compact latent space of a powerful world model. The world model uses discrete representations and is trained separately from the policy. DreamerV2 constitutes the first agent that achieves human-level performance on the Atari benchmark of 55 tasks by learning behaviors inside a separately trained world model. With the same computational budget and wall-clock time, DreamerV2 reaches 200M frames and exceeds the final performance of the top single-GPU agents IQN and Rainbow.

1 INTRODUCTION

To successfully operate in unknown environments, reinforcement learning agents need to learn about their environments over time. World models are an explicit way to represent an agent's knowledge about its environment. Compared to model-free reinforcement learning that learn through trial and error, world models facilitate generalization and can predict the outcomes of potential actions to enable planning (Sutton, 1991). Capturing general aspects of the environment, world models have been shown to be effective for transfer to novel tasks (Byravan et al., 2019), directed exploration (Sekar et al., 2020), and generalization from offline datasets (Yu et al., 2020). When the inputs are high-dimensional images, latent dynamics models predict ahead in a abstract latent space (Watter et al., 2015; Ha and Schmidhuber, 2018; Hafner et al., 2018; Zhang et al., 2019). Predicting compact representations instead of images can reduce the effect of accumulating errors and their small memory footprint enables thousands of parallel predictions on a single GPU (Hafner et al., 2018; 2019). Using this approach, the recent Dreamer agent (Hafner et al., 2019) solved a wide range of continuous control tasks from image inputs.

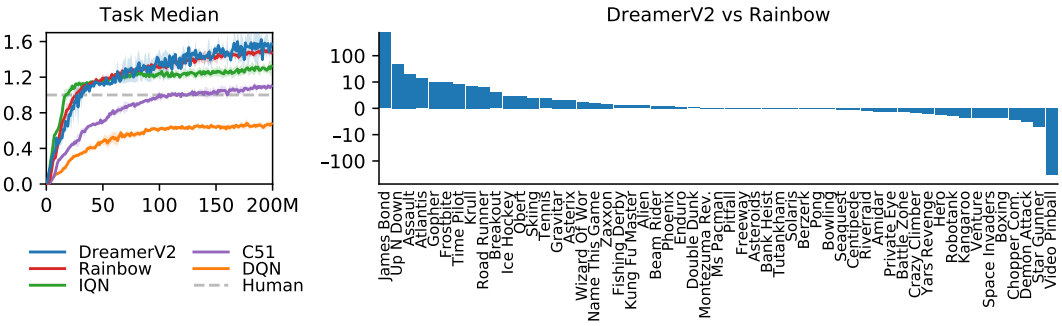

Figure 1: Performance on the Atari benchmark of 55 games with sticky actions over 200M steps. DreamerV2 is the first agent that learns within a separately trained world model to achieve human-level performance on the Atari suite, demonstrating the high accuracy of it learned world model. DreamerV2 further outperforms the state-of-the-art single-GPU agents Rainbow (Hessel et al., 2018) and IQN (Dabney et al., 2018), achieving comparable or higher performance on almost all games. The baseline scores are takes from the Dopamine framework (Castro et al., 2018).

Despite their intriguing properties, world models have so far not been accurate enough to compete with the state-of-the-art model-free algorithms on the most competitive benchmarks. The well-established Atari benchmark (Bellemare et al., 2013) historically required model-free algorithms to achieve human-level performance, such as DQN (Mnih et al., 2015), A3C (Mnih et al., 2016), or Rainbow (Hessel et al., 2018). Several attempts at learning accurate world models of Atari games have been made, without achieving competitive performance (Oh et al., 2015; Chiappa et al., 2017; Kaiser et al., 2019). On the other hand, the recently proposed MuZero agent (Schrittwieser et al., 2019) shows that planning can achieve impressive performance given extensive engineering effort and a vast computational budget. However, its implementation is not available to the public and it would require over 2 months of computation to train even one agent on a GPU, rendering it impractical for most research groups.

In this paper, we introduce DreamerV2, the first reinforcement learning agent that achieves human-level performance on the Atari benchmark by learning behaviors purely within a separately trained world model, as shown in Figure 1. Learning successful behaviors purely within the world model demonstrates that the world model learns to accurately represent the environment. To achieve this, we apply small modifications to the Dreamer agent (Hafner et al., 2019), such as using discrete latents and balancing terms within the KL loss. Using a single GPU and a single environment instance, DreamerV2 outperforms top single-GPU Atari agents Rainbow (Hessel et al., 2018) and IQN (Dabney et al., 2018), which rest upon years of model-free reinforcement learning research (Van Hasselt et al., 2015; Schaul et al., 2015; Wang et al., 2016; Bellemare et al., 2017; Fortunato et al., 2017). Moreover, aspects of these algorithms are complimentary to our world model and could be integrated into the Dreamer framework in the future. To rigorously compare the algorithms, we report scores normalized by both a human gamer (Mnih et al., 2015) and the human world record (Toromanoff et al., 2019) and make a suggestion for reporting scores going forward.

2 DreamerV2

We present DreamerV2, an evolution of the Dreamer agent (Hafner et al., 2019). We refer to the original Dreamer agent as DreamerV1 throughout this paper. This section describes the complete DreamerV2 algorithm, consisting of the three typical components of a model-based agent (Sutton, 1991). We learn the world model from a dataset of past experience, learn an actor and critic from imagined sequences of compact model states, and execute the actor in the environment to grow the experience dataset. In Appendix A, we include a list of changes that we applied to the open sourced DreamerV1 implementation[1] and which of them we found to increase empirical performance.

2.1 World Model Learning

World models summarize an agent's experience into a predictive model that can be used in place of the environment to learn behaviors. When inputs are high-dimensional images, it is beneficial to learn compact state representations of the inputs to predict ahead in this learned latent space (Watter et al., 2015; Karl et al., 2016; Ha and Schmidhuber, 2018). These models are called latent dynamics models. Predicting ahead in latent space not only facilitates long-term predictions, it also allows to efficiently predict thousands of compact state sequences in parallel in a single batch, without having to generate images. DreamerV2 builds upon the world model that was introduced by PlaNet (Hafner et al., 2018) and used in DreamerV1, by replacing its Gaussian latents with categorical variables.

Experience dataset The world model is trained from the agent's growing dataset of past experience that contains sequences of of images $x_{1:T}$, actions $a_{1:T}$, rewards $r_{1:T}$, and discount factors $\gamma_{1:T}$. The discount factors equal a fixed hyper parameter $\gamma = 0.995$ for time steps within an episode and are set to zero for terminal time steps. For training, we use batches of $B = 50$ sequences of fixed length $L = 50$ that are sampled randomly within the stored episodes. To observe enough episode ends during training, we sample the start index of each training sequence uniformly within the episode and then clip it to not exceed the episode length minus the training sequence length.

[1] https://github.com/google-research/dreamer

Model components The world model consists of five components. Given a batch of sequences from the dataset, the representation model summarizes the history of images $x_{1:t}$ and actions $a_{1:t-1}$ into a compact Markovian model state z_t. From this model state, we predict the current image x_t, current reward r_t, discount factor γ_t, and next model state z_{t+1}. The model components are:

$$\begin{aligned}
&\text{Representation model:} && z_t \sim q_\phi(z_t \mid z_{t-1}, a_{t-1}, x_t) \\
&\text{Transition predictor:} && \hat{z}_t \sim p_\phi(\hat{z}_t \mid z_{t-1}, a_{t-1}) \\
&\text{Image predictor:} && \hat{x}_t \sim p_\phi(\hat{x}_t \mid z_t) \\
&\text{Reward predictor:} && \hat{r}_t \sim p_\phi(\hat{r}_t \mid z_t) \\
&\text{Discount predictor:} && \hat{\gamma}_t \sim p_\phi(\hat{\gamma}_t \mid z_t).
\end{aligned} \tag{1}$$

All components are implemented as neural networks and ϕ describes their combined parameter vector. The transition predictor guesses the next model state only from the current model state and the action but without using the next image, so that we can later learn behaviors by predicting sequences of model states without having to observe or generate images. The discount predictor lets us estimate the probability of an episode end when learning behaviors from model predictions.

Neural networks The representation model is implemented as a Convolutional Neural Network (CNN; LeCun et al., 1989) followed by a Recurrent State Space Model (RSSM; Hafner et al., 2018). The RSSM uses a Gated Recurrent Unit (GRU; Cho et al., 2014) that outputs a stochastic variable at each time step and feeds it back for the next time step. The model state z_t is the concatenation of deterministic GRU state and a sample of the stochastic variable. The transition predictor is an RSSM that shares most of its parameters with the representation model as in PlaNet. The image predictor is a transposed CNN and the reward and discount predictors are Multi-Layer Perceptrons (MLPs). We down-scale the 84×84 input images to 64×64 pixels so that we can apply the convolutional architecture of DreamerV1. We use the ELU activation function for all components of the model (Clevert et al., 2015). The world model uses a total of 20M trainable parameters.

Distributions The image predictor outputs the mean of a diagonal Gaussian likelihood with unit variance, the reward predictor outputs a univariate Gaussian with unit variance, and the discount predictor outputs a Bernoulli likelihood. In prior work, the latent variable in the model state was a diagonal Gaussian that used reparamterization gradients during backpropagation (Kingma and Welling, 2013; Rezende et al., 2014). In DreamerV2, we instead use a vector of several categorical variables and leverage straight-through gradients during backpropagation (Bengio et al., 2013), which are also easy to implement using automatic differentiation.

Loss function All components of the world model are optimized jointly. The distributions produced by the image predictor, reward predictor, discount predictor, and transition predictor are trained to maximize the log-likelihood of their corresponding targets. The representation model is trained to produce model states that facilitates these prediction tasks, through the expectation below. Moreover, it is regularized to produce model states with high entropy, such that the model becomes robust to many different model states during training. The loss function for learning the world model is:

$$\begin{aligned}
\mathcal{L}(\phi) \doteq \mathrm{E}_{q_\phi(z_{1:T} \mid a_{1:T}, x_{1:T})} \Big[\tfrac{1}{T} \textstyle\sum_{t=1}^{T} \big(& \underbrace{-\eta_i \ln p_\phi(r_t \mid z_t)}_{\text{image log loss}} \underbrace{-\eta_r \ln p_\phi(x_t \mid z_t)}_{\text{reward log loss}} \\
& \underbrace{-\eta_\gamma \ln p_\phi(\gamma_t \mid z_t)}_{\text{discount log loss}} \underbrace{-\eta_t \ln p_\phi(z_t \mid z_{t-1}, a_{t-1})}_{\text{transition log loss}} \underbrace{+\eta_q \ln q_\phi(z_t \mid z_{t-1}, a_{t-1}, x_t)}_{\text{entropy regularizer}} \big) \Big].
\end{aligned} \tag{2}$$

We jointly minimize the loss function with respect to the vector ϕ that contains all parameters of the world model using the Adam optimizer (Kingma and Ba, 2014). We use the loss scales $\eta_i = 1/(64 \cdot 64 \cdot 3)$ for the image, $\eta_r = 1$ for the reward, $\eta_\gamma = 1$ for the discount, $\eta_t = 0.08$ for the transition, and $\eta_q = 0.02$ for the entropy regularizer. Scaling the transition loss up compared to the entropy regularizer allows us to encourage learning an accurate transition function.

Probabilistic interpretation The world model loss function in Equation 2 is the ELBO or variational free energy of an hidden Markov model that is conditioned on the action sequence. The world model can thus be interpreted as a sequential VAE, where the representation model is the approximate posterior and the transition predictor is the temporal prior. In the ELBO objective, the transition loss and entropy regularizer together are known as the KL regularizer. Scaling the image loss relative to the KL regularizer is known as beta-VAE (Higgins et al., 2016). Additionally, we scale the two terms within the KL term, the prior cross entropy and the posterior entropy, relative to each other. We

refer to this technique as KL balancing. KL balancing encourages learning an accurate prior over increasing posterior entropy, so that the prior better approximates the aggregate posterior.

2.2 Behavior Learning

DreamerV2 learns long-horizon behaviors purely within its world model using an actor and a critic. The actor chooses actions for predicting imagined sequences of compact model states. The critic accumulates the future predicted rewards to take into account rewards beyond the planning horizon. Both the actor and critic operate on top of the learned model states and thus benefit from the representations learned by the world model. The world model is fixed during behavior learning, so the actor and value gradients do not affect its representations. Not predicting images during behavior learning lets us efficiently simulate 2500 latent trajectories in parallel on a single GPU.

Imagination MDP To learn behaviors within the latent space of world model, we define the imagination MPD as follows. The distribution of initial states $\hat{z}_0$ in the imagination MDP is the distribution of compact model states encountered during world model training. From there, the transition predictor $p_\phi(\hat{z}_t \mid \hat{z}_{t-1}, \hat{a}_{t-1})$ outputs sequences $\hat{z}_{1:H}$ of compact model states up to the imagination horizon $H = 15$. The mean of the reward predictor $p_\phi(\hat{r}_t \mid \hat{z}_t)$ is used as reward sequence $\hat{r}_{1:H}$. The discount predictor $p_\phi(\hat{\gamma}_t \mid \hat{z}_t)$ outputs the discount sequence $\hat{\gamma}_{1:H}$ that is used to down-weight rewards. Moreover, we weight the loss terms of the actor and critic by the cumulative predicted discount factors to softly account for the possibility of episode ends.

Model components To learn long-horizon behaviors in the imagination MDP, we leverage a stochastic actor that chooses actions and a deterministic critic. The actor and critic are trained coorperatively, where the actor aims to output actions that lead to states that maximize the critic output, while the critic aims to accurately estimate the sum of future rewards achieved by the actor from each imagined state. The actor and critic use the parameter vectors ψ and ξ, respectively:

$$\begin{aligned} &\text{Actor:} && \hat{a}_t \sim p_\psi(\hat{a}_t \mid \hat{z}_t) \\ &\text{Critic:} && v_\xi(\hat{z}_t) \approx \mathrm{E}_{p_\phi, p_\psi}\Big[\textstyle\sum_{\tau \geq t} \hat{\gamma}^{\tau - t} \hat{r}_\tau\Big]. \end{aligned} \tag{3}$$

In contrast to the actual environment, the latent state sequence is Markovian, so that there is no need for the actor and critic to condition on more than the current model state. The actor and critic are both MLPs with ELU activations (Clevert et al., 2015) and use 1M trainable parameters each. The actor outputs a categorical distribution over actions and the critic has a deterministic output. The two components are trained from the same imagined trajectories but optimize separate loss functions.

Critic loss function The critic aims to predict the discounted sum of future rewards that the actor achieves in a given model state, known as the state value. For this, we leverage temporal-difference learning, where the critic is trained toward a value target that is constructed from intermediate rewards and critic outputs for later states. A common choice is the 1-step target that sums the current reward and the critic output for the following state. However, the imagination MDP lets us generate on-policy trajectories of multiple steps, suggesting the use of n-step targets that incorporate reward information into the critic more quickly. We follow DreamerV1 in using the more general λ-target (Sutton and Barto, 2018; Schulman et al., 2015) that is defined recursively as follows:

$$V_t^\lambda \doteq \hat{r}_t + \hat{\gamma}_t \begin{cases} (1-\lambda) v_\xi(\hat{z}_{t+1}) + \lambda V_{t+1}^\lambda & \text{if} \quad t < H, \\ v_\xi(\hat{z}_H) & \text{if} \quad t = H. \end{cases} \tag{4}$$

Intuitively, the λ-target is a weighted average of n-step returns for different horizons, where longer horizons are weighted exponentially less. We set $\lambda = 0.95$ in practice, to focus more on long horizon targets than on short horizon targets. Given a trajectory of model states, rewards, and discount factors, we train the critic to regress the λ-return using a squared loss:

$$\mathcal{L}(\xi) \doteq \mathrm{E}_{p_\phi, p_\psi}\Big[\tfrac{1}{H-1} \textstyle\sum_{t=1}^{H-1} \tfrac{1}{2}\big(v_\xi(\hat{z}_t) - \mathrm{sg}(V_t^\lambda)\big)^2\Big] \tag{5}$$

We optimize the critic loss with respect to the critic parameters ξ using the Adam optimizer. There is no loss term for the last time step because the target equals the critic at that step. We stop the gradients around the targets, denoted by the $\mathrm{sg}(\cdot)$ function, as typical in the literature. We stabilize value learning using a target network (Mnih et al., 2015), namely, we compute the targets using a copy of the critic that is updated every 100 gradient steps.

Actor loss function The actor aims to output actions that maximize the prediction of long-term future rewards made by the critic. To incorporate intermediate rewards more directly, we train the actor to maximize the same λ-return that was computed for training the critic. There are different gradient estimators for maximizing the targets with respect the the actor parameters. DreamerV2 combines unbiased but high-variance Reinforce gradients with biased but low-variance straight-through gradients. Moreover, we regularize the entropy of the actor to encourage exploration where feasible while allowing the actor to choose precise actions when necessary.

Learning by Reinforce (Williams, 1992) maximizes the actor's probability of its own sampled actions weighted by the values of those actions. The variance of this estimator can be reduced by subtracting the state value as baseline, which does not depend on the current action. Intuitively, subtracting the baseline centers the weights and leads to faster learning. The benefit of Reinforce is that it produced unbiased gradients and the downside is that it can have high variance, even with baseline.

DreamerV1 relied entirely on reparameterization gradients (Kingma and Welling, 2013; Rezende et al., 2014) to train the actor directly by backpropagating value gradients through the sequence of sampled model states and actions. DreamerV2 uses both discrete latents and discrete actions. To backpropagate through the sampled actions and state sequences, we leverage straight-through gradients (Bengio et al., 2013). This results in a biased gradient estimate with low variance. The combined actor loss function is:

$$\mathcal{L}(\psi) \doteq \mathrm{E}_{p_\phi, p_\psi}\Big[\tfrac{1}{H-1}\textstyle\sum_{t=1}^{H-1}\big(\underbrace{-\eta_s \ln p_\psi(\hat{a}_t \mid \hat{z}_t)\,\mathrm{sg}(V_t^\lambda - v_\xi(\hat{z}_t))}_{\text{reinforce}} \underbrace{-\eta_d V_t^\lambda}_{\text{dynamics backprop}} \underbrace{-\eta_e \mathrm{H}[a_t|\hat{z}_t]}_{\text{entropy regularizer}}\big)\Big]. \tag{6}$$

We optimize the actor loss with respect to the actor parameters ψ using the Adam optimizer. We use the loss scales $\eta_s = 0.9$ for reinforce, $\eta_d = 0.1$ for dynamics backpropagation, and linearly anneal the entropy regularizer $\eta_e = 3 \cdot 10^{-3} \to 3 \cdot 10^{-4}$ over the first 40M environment frames. We hypothesize that combining the two gradient estimators is beneficial because the low-variance straight-through gradients can accelerate early learning, while the unbiased Reinforce gradients can help find a better final solution.

3 Experiments

We evaluate DreamerV2 on the well-established Atari benchmark with sticky actions, comparing to four strong model-free algorithms. DreamerV2 outperforms the four model-free algorithms in all scenarios. For an extensive comparison, we report four scores according to four aggregation protocols and give a recommendation for meaningfully aggregating scores across games going forward. We also ablate the importance of discrete representations in the world model. Our implementation of DreamerV2 reaches 200M environment steps in about 10 days, while using only a single Nvidia V100 GPU and a single environment instance. To help others build upon this work, we will release the implementation and training curves with the final version of the paper.

Experimental setup We select the 55 games that prior works in the literature from different research labs tend to agree on (Mnih et al., 2016; Brockman et al., 2016; Hessel et al., 2018; Castro et al., 2018; Badia et al., 2020) and recommend this set of games for evaluation going forward. We follow the evaluation protocol of Machado et al. (2018) with 200M environment steps, action repeat of 4, a time limit of 108,000 steps per episode that correspond to 30 minutes of game play, and sticky actions. The experiments use a single-task setup where a separate agent is trained for each game. Moreover, each agent uses only a single environment instance. We compare the algorithms based on both human gamer and human world record normalization (Toromanoff et al., 2019).

Model-free baselines We compare the learning curves and final scores of DreamerV2 to four model-free algorithms, IQN (Dabney et al., 2018), Rainbow (Hessel et al., 2018), C51 (Bellemare et al., 2017), and DQN (Mnih et al., 2015). We use the scores of these agents provided by the Dopamine framework (Castro et al., 2018) that use sticky actions. These may differ from the reported results in the papers that introduce these algorithms in the deterministic Atari setup. The training time of Rainbow was reported at 10 days on a single GPU and using one environment instance.

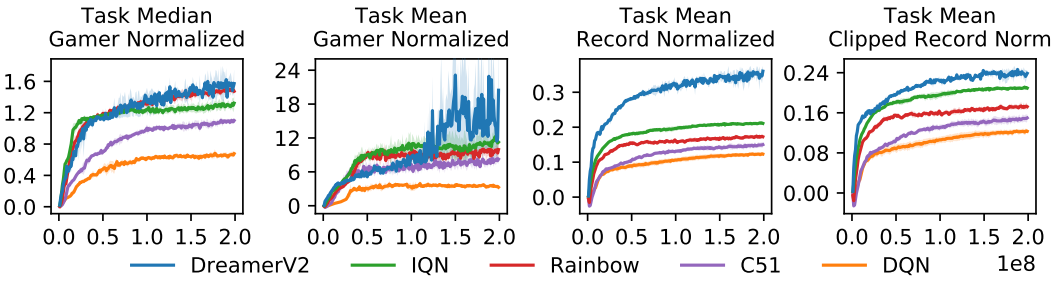

Figure 2: Atari performance over 200M steps. The standards in the literature to aggregate over tasks are shown in the left two plots. They normalize scores by a professional gamer and compute the median or mean over tasks (Mnih et al., 2015; 2016). In Section 3, we point out limitations of this methodology. As a robust measure of performance, we recommend the metric in the right-most plot. We normalize scores by the human world record (Toromanoff et al., 2019) and then clip them, such that exceeding the record does not further increase the score, before computing the mean.

Atari performance The performance curves of DreamerV2 and four standard model-free algorithms are visualized in Figure 2. The final scores at 200M environment steps are shown in Table 1 and the scores on individual games are included in Table D.1. There are different approaches for aggregating the scores across the 55 games and we show that this choice can have a substantial impact on the relative performance between algorithms. To extensively compare DreamerV2 to the model-free algorithms, we consider the following four aggregation approaches:

- **Task Median, Gamer Normalized** Atari scores are commonly normalized based on a random policy and a professional gamer, and the median over tasks is reported (Mnih et al., 2015; 2016). However, if almost half of the scores would be zero, the median would not be affected. Thus, we argue that median scores are not reflective of the robustness of an algorithm.
- **Task Mean, Gamer Normalized** Compared to the task median, the task mean considers all tasks. However, the gamer performed poorly on a small number of games, such as Crazy Climber, James Bond, and Video Pinball. This makes it easy for algorithms to achieve a high normalized score on these few games, which then dominate the task mean.
- **Task Mean, Record Normalized** Instead of normalizing based on the professional gamer, Toromanoff et al. (2019) suggest to normalize based on the registered human world record of each game. This partially addresses the outlier problem but the mean is still dominated by a small number of games, where the algorithms achieve super-human performance.
- **Task Mean, Clipped Record Norm** To overcome these limitations, we recommend normalizing by the human world record and then clipping the scores to not exceed a value of 1, so that performance above the record does not further increase the score. The result is a robust measure of algorithm performance on the Atari suite that considers performance across all games.

From Figure 2 and Table 1, we see that the different aggregation approaches let us examine agent performance from different angles. Interestingly, Rainbow clearly outperforms IQN in the first

Agent	**Task Median** Gamer Normalized	**Task Mean** Gamer Normalized	**Task Mean** Record Normalized	**Task Mean** Clipped Record Norm
DreamerV2	**1.57**	**20.46**	**0.36**	**0.24**
IQN	1.32	11.27	0.21	0.21
Rainbow	1.47	9.95	0.17	0.17
C51	1.10	8.25	0.15	0.15
DQN	0.68	3.28	0.12	0.12

Table 1: Atari performance at 200M steps. The scores of the 55 games are aggregated using the four different protocols described in Section 3. To overcome limitations of the previous metrics, we recommend the task mean of clipped record normalized scores as a robust measure of algorithm performance, shown in the right-most column. DreamerV2 outperforms previous single-GPU agents across all metrics. The baseline scores are taken from Dopamine Baselines (Castro et al., 2018).

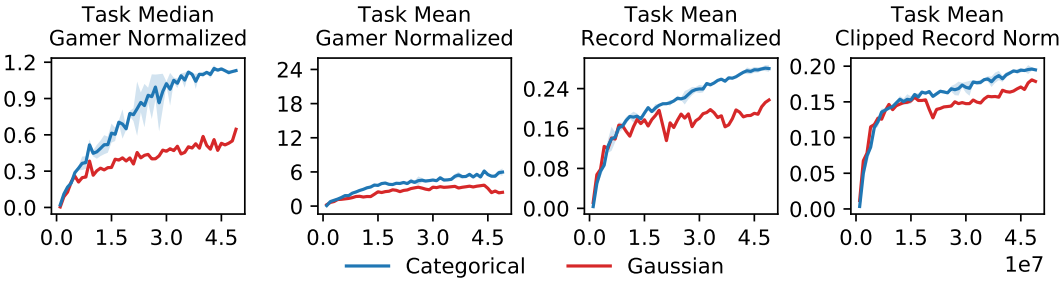

Figure 3: Comparison of DreamerV2 with categorical latents to an ablation with Gaussian latents, as used by DreamerV1. Both world models optimize the same objective based on the ELBO optimized via straight-through gradients for categorical variables and reparameterization gradients for Gaussian latents. We find that categorical latents outperform Gaussian latents in all four metrics.

aggregation method but IQN clearly outperforms Rainbow in the remaining setups. DreamerV2 outperforms the model-free agents in all four metrics, with the largest margin in record normalized mean performance. Despite this, we recommend clipped record normalized as the most meaningful aggregation method, as it considers all tasks to a similar degree without being dominated by a small number of outlier scores.

Discrete latents In a separate experiment, we compare the performance of equipping the world model with categorical latents, as in DreamerV2, to Gaussian latents, as in DreamerV1. The training curves for a smaller number of environment steps are visualized in Figure 3. Across all aggregation approaches, categorical variables outperform Gaussian latents, with the largest difference in gamer normalized task median. Discrete latent variables could be beneficial is that they better capture the discrete objects present in many Atari games or because they are easier to optimize.

Individual games The scores on individual Atari games at 200M environment steps are included in Table D.1, alongside the model-free algorithms and the baselines of random play, human gamer, and human world record. We filled in reasonable values for the 2 out of 55 games that have no registered world record. Figure C.1 compares the score differences between DreamerV2 and each model-free algorithm for the individual games. DreamerV2 achieves comparable or higher performance on all games except for Video Pinball. We hypothesize that the reconstruction loss of the world model does not encourage learning a meaningful latent representation because the most important object in the game, the ball, occupies only a single pixel. One the other hand, DreamerV2 achieves the strongest improvements over the model-free agents on the games James Bond, Up N Down, and Assault.

4 Related Work

Model-free Atari The majority of agents applied to the Atari benchmark have been trained using model-free algorithms. DQN (Mnih et al., 2015) showed that deep neural network policies can be trained using Q-learning by incorporating experience replay and target networks. Several works have extended DQN to incorporate bias correction as in DDQN (Van Hasselt et al., 2015), prioritized experience replay (Schaul et al., 2015), architectural improvements (Wang et al., 2016), and distributional value learning (Bellemare et al., 2017; Dabney et al., 2017; 2018). Besides value learning, agents based on policy gradients have targeted the Atari benchmark, such as ACER (Schulman et al., 2017a), PPO (Schulman et al., 2017a), ACKTR (Wu et al., 2017), and Reactor (Gruslys et al., 2017). Another line of work has focused on improving performance by distributing data collection, often while increasing the budget of environment steps beyond 200M (Mnih et al., 2016; Schulman et al., 2017b; Horgan et al., 2018; Kapturowski et al., 2018; Badia et al., 2020).

World models Several model-based agents focus on proprioceptive inputs (Watter et al., 2015; Gal et al., 2016; Higuera et al., 2018; Henaff et al., 2018; Chua et al., 2018; Wang et al., 2019; Wang and Ba, 2019), model images without using them for planning (Oh et al., 2015; Krishnan et al., 2015; Karl et al., 2016; Chiappa et al., 2017; Babaeizadeh et al., 2017; Gemici et al., 2017; Denton and Fergus, 2018; Buesing et al., 2018; Doerr et al., 2018; Gregor and Besse, 2018), or combine the benefits of model-based and model-free approaches (Kalweit and Boedecker, 2017; Nagabandi et al., 2017; Weber et al., 2017; Kurutach et al., 2018; Buckman et al., 2018; Ha and Schmidhuber, 2018; Wayne et al., 2018; Igl et al., 2018; Srinivas et al., 2018; Lee et al., 2019). Most world model agents

Algorithm	Reward Modeling	Image Modeling	Latent Transitions	Single GPU	Trainable Parameters	Atari Frames	Accelerator Days
DreamerV2	✓	✓	✓	✓	22M	200M	10
SimPLe	✓	✓	✗	✓	74M	4M	40
MuZero	✓	✗	✓	✗	40M	20B	80
MuZero Reanalyze	✓	✗	✓	✗	40M	200M	80

Table 2: Conceptual comparison of recent RL algorithms that leverage planning with a learned model. DreamerV2 and SimPLe learn complete models of the environment by leveraging the learning signal provided by the image inputs, while MuZero learns its model through value gradients that are specific to an individual task. The Monte-Carlo tree search used by MuZero is effective but adds complexity and is challenging to parallelize. This component is orthogonal to the world model proposed here.

with image inputs have thus far been limited to relatively simple control tasks (Watter et al., 2015; Ebert et al., 2017; Ha and Schmidhuber, 2018; Hafner et al., 2018; Zhang et al., 2019; Hafner et al., 2019). We explain the two model-based approaches that were applied to Atari in detail below.

SimPLe The SimPLe agent (Kaiser et al., 2019) learns a video prediction model in pixel-space and uses its predictions to train a PPO agent (Schulman et al., 2017a), as shown in Table 2. The model directly predicts each frame from the previous four frames and receives an additional discrete latent variable as input. The authors evaluate SimPLe on a subset of Atari games for 400k and 2M environment steps, after which they report diminishing returns. Some recent model-free methods have followed the comparison at 400k steps (Srinivas et al., 2020; Kostrikov et al., 2020). However, the highest performance achieved in this data-efficient regime is a gamer normalized median score of 0.28 (Kostrikov et al., 2020) that is far from human-level performance. Instead, we focus on the well-established and competitive evaluation after 200M frames, where many successful model-free algorithms are available for comparison.

MuZero The MuZero agent (Schrittwieser et al., 2019) learns a sequence model of rewards and values (Oh et al., 2017) to solve reinforcement learning tasks via Monte-Carlo Tree Search (MCTS; Coulom, 2006; Silver et al., 2017). The sequence model is trained purely by predicting task-specific information and does not incorporate explicit representation learning using the images, as shown in Table 2. MuZero shows that with significant engineering effort and a vast computational budget, planning can achieve impressive performance on several board games and deterministic Atari games. However, MuZero is not publicly available, and it would require over 2 months to train an Atari agent on one GPU. By comparison, DreamerV2 is a simple algorithm that achieves human-level performance on Atari on a single GPU in 10 days, making it reproducible for many researchers. Moreover, the advanced planning components of MuZero are complimentary and could be applied to the accurate world models learned by DreamerV2. DreamerV2 leverages the additional learning signal provided by the input images, analogous to recent successes by semi-supervised image classification (Chen et al., 2020; He et al., 2020; Grill et al., 2020).

5 Discussion

We present DreamerV2, a model-based agent that achieves human-level performance on the Atari 200M benchmark by learning behaviors purely from the latent-space predictions of a separately trained world model. Using a single GPU and a single environment instance, DreamerV2 outperforms top model-free single-GPU agents Rainbow and IQN using the same computational budget and training time. To develop DreamerV2, we apply several small modifications to the Dreamer agent (Hafner et al., 2019). We confirm experimentally that learning a categorical latent space improves the performance of agent. We believe that DreamerV2 serves as valuable proof of concept that shows that model-based RL can outperform top model-free algorithms on the most competitive RL benchmarks, despite the years of research and engineering effort that modern model-free agents rest upon. Beyond achieving strong performance on individual tasks, world models open avenues for efficient transfer and multi-task learning, sample-efficient learning on physical robots, and global exploration based on uncertainty estimates.

REFERENCES

M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017.

A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, and C. Blundell. Agent57: Outperforming the atari human benchmark. *arXiv preprint arXiv:2003.13350*, 2020.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*, 2017.

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, pages 8224–8234, 2018.

L. Buesing, T. Weber, S. Racaniere, S. Eslami, D. Rezende, D. P. Reichert, F. Viola, F. Besse, K. Gregor, D. Hassabis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

A. Byravan, J. T. Springenberg, A. Abdolmaleki, R. Hafner, M. Neunert, T. Lampe, N. Siegel, N. Heess, and M. Riedmiller. Imagined value gradients: Model-based policy optimization with transferable latent dynamics models. *arXiv preprint arXiv:1910.04142*, 2019.

P. S. Castro, S. Moitra, C. Gelada, S. Kumar, and M. G. Bellemare. Dopamine: A research framework for deep reinforcement learning. *arXiv preprint arXiv:1812.06110*, 2018.

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

S. Chiappa, S. Racaniere, D. Wierstra, and S. Mohamed. Recurrent environment simulators. *arXiv preprint arXiv:1704.02254*, 2017.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

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*, pages 4754–4765, 2018.

D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

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

W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. *arXiv preprint arXiv:1710.10044*, 2017.

W. Dabney, G. Ostrovski, D. Silver, and R. Munos. Implicit quantile networks for distributional reinforcement learning. *arXiv preprint arXiv:1806.06923*, 2018.

E. Denton and R. Fergus. Stochastic video generation with a learned prior. *arXiv preprint arXiv:1802.07687*, 2018.

A. Doerr, C. Daniel, M. Schiegg, D. Nguyen-Tuong, S. Schaal, M. Toussaint, and S. Trimpe. Probabilistic recurrent state-space models. *arXiv preprint arXiv:1801.10395*, 2018.

F. Ebert, C. Finn, A. X. Lee, and S. Levine. Self-supervised visual planning with temporal skip connections. *arXiv preprint arXiv:1710.05268*, 2017.

M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Y. Gal, R. McAllister, and C. E. Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, 2016.

M. Gemici, C.-C. Hung, A. Santoro, G. Wayne, S. Mohamed, D. J. Rezende, D. Amos, and T. Lillicrap. Generative temporal models with memory. *arXiv preprint arXiv:1702.04649*, 2017.

K. Gregor and F. Besse. Temporal difference variational auto-encoder. *arXiv preprint arXiv:1806.03107*, 2018.

J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

A. Gruslys, W. Dabney, M. G. Azar, B. Piot, M. Bellemare, and R. Munos. The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning. *arXiv preprint arXiv:1704.04651*, 2017.

D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

M. Henaff, W. F. Whitney, and Y. LeCun. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177*, 2018.

M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

J. C. G. Higuera, D. Meger, and G. Dudek. Synthesizing neural network controllers with probabilistic model based reinforcement learning. *arXiv preprint arXiv:1803.02291*, 2018.

D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson. Deep variational reinforcement learning for pomdps. *arXiv preprint arXiv:1806.02426*, 2018.

L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

G. Kalweit and J. Boedecker. Uncertainty-driven imagination for continuous deep reinforcement learning. In *Conference on Robot Learning*, pages 195–206, 2017.

S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

R. G. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, and M. Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv preprint arXiv:1708.02596*, 2017.

J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2863–2871, 2015.

J. Oh, S. Singh, and H. Lee. Value prediction network. In *Advances in Neural Information Processing Systems*, pages 6118–6128, 2017.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017a.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.

R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. *arXiv preprint arXiv:2005.05960*, 2020.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676): 354, 2017.

A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn. Universal planning networks. *arXiv preprint arXiv:1804.00645*, 2018.

A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

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

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

M. Toromanoff, E. Wirbel, and F. Moutarde. Is deep reinforcement learning really superhuman on atari? leveling the playing field. *arXiv preprint arXiv:1908.04683*, 2019.

H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*, 2015.

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

T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba. Benchmarking model-based reinforcement learning. *CoRR*, abs/1907.02057, 2019.

Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003, 2016.

M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

G. Wayne, C.-C. Hung, D. Amos, M. Mirza, A. Ahuja, A. Grabska-Barwinska, J. Rae, P. Mirowski, J. Z. Leibo, A. Santoro, et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

T. Weber, S. Racanière, D. P. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017.

T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine. Solar: deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, 2019.

A SUMMARY OF MODIFICATIONS

To develop DreamerV2, we used the Dreamer agent (Hafner et al., 2019) as a starting point. This subsection describes the changes that we applied to the agent to achieve high performance on the Atari benchmark, as well as the changes that were tried but not found to increase performance and thus were not not included in DreamerV2.

Summary of changes that were tried and were found to help:

- **Categorical latents** Using categorical latent states using straight-through gradients in the world model instead of Gaussian latents with reparameterized gradients.
- **Mixed actor gradients** Combining Reinforce and dynamics backpropagation gradients for learning the actor instead of dynamics backpropagation only.
- **Policy entropy** Regularizing the policy entropy for exploration both in imagination and during data collection, instead of using external action noise during data collection.
- **KL balancing** Separately scaling the prior cross entropy and the posterior entropy in the KL loss to encourage learning an accurate temporal prior, instead of using free nats.
- **Model size** Increasing the number of units or feature maps per layer of all model components, resulting in a change from 13M parameters to 22M parameters.
- **Layer norm** Using layer normalization in the GRU that is used as part of the RSSM latent transition model, instead of no normalization.

Summary of changes that were tried but were not shown to help:

- **Binary latents** Using a larger number of binary latents for the world model instead of using categorical latents, which could have encouraged a more disentangled representation.
- **Long-term entropy** Including the policy entropy into temporal-difference loss of the value function, so that the actor seeks out states with high action entropy beyond the planning horizon.
- **Scheduling** Scheduling the learning rate, KL scale, free bits, fraction of Reinforce gradient for the actor. Only scheduling the entropy regularizer for the policy was clearly beneficial.
- **Reinforce only** Using only Reinforce gradients for the actor worked for many games but fails to start learning on some games, possibly because of the high variance of reinforce gradients.

Due to the large computational requirements, a comprehensive ablation study on this list of changes is unfortunately infeasible for us. This would require 55 tasks times 5 seeds for 10 days per change to run, resulting in over 60,000 GPU hours per change. However, we aim to include ablations for the most critical changes in the final version of this paper.

B HYPER PARAMETERS

Name	**Symbol**	**Value**
World Model		
Dataset size (FIFO)	—	$2 \cdot 10^6$
Batch size	B	50
Sequence length	L	50
Discrete latent dimensions	—	32
Discrete latent classes	—	32
RSSM number of units	—	600
Image loss scale	η_i	$1/(64 \cdot 64 \cdot 1)$
Reward loss scale	η_r	1
Discount loss scale	η_γ	1
Transition loss scale	η_t	0.08
Entropy loss scale	η_q	0.02
World model learning rate	—	$2 \cdot 10^{-4}$
Reward transformation	—	$\tanh$
Behavior		
Imagination horizon	H	15
Discount	γ	0.995
λ-target parameter	λ	0.95
Reinforce loss scale	η_s	0.9
Dynamics backprop loss scale	η_d	0.1
Actor entropy loss scale	η_e	$3 \cdot 10^{-3} \xrightarrow{40\text{M}} 3 \cdot 10^{-4}$
Actor learning rate	—	$4 \cdot 10^{-5}$
Critic learning rate	—	$1 \cdot 10^{-4}$
Slow critic update interval	—	100
Common		
Environment steps per update	—	5
MPL number of layers	—	4
MPL number of units	—	400
Gradient clipping	—	100
Adam epsilon	ϵ	10^{-5}
Decoupled weight decay	—	10^{-6}

Table B.1: Hyper parameters of the DreamerV2 agent on Atari. When tuning the agent on a different task, we recommend searching over the actor entropy scale, the discount factor, and the transition and entropy loss scales, while keeping the ratio of the two constant. The number of environment steps per update should be reduced to achieve higher data-efficiency.

C ATARI AGENT COMPARISON

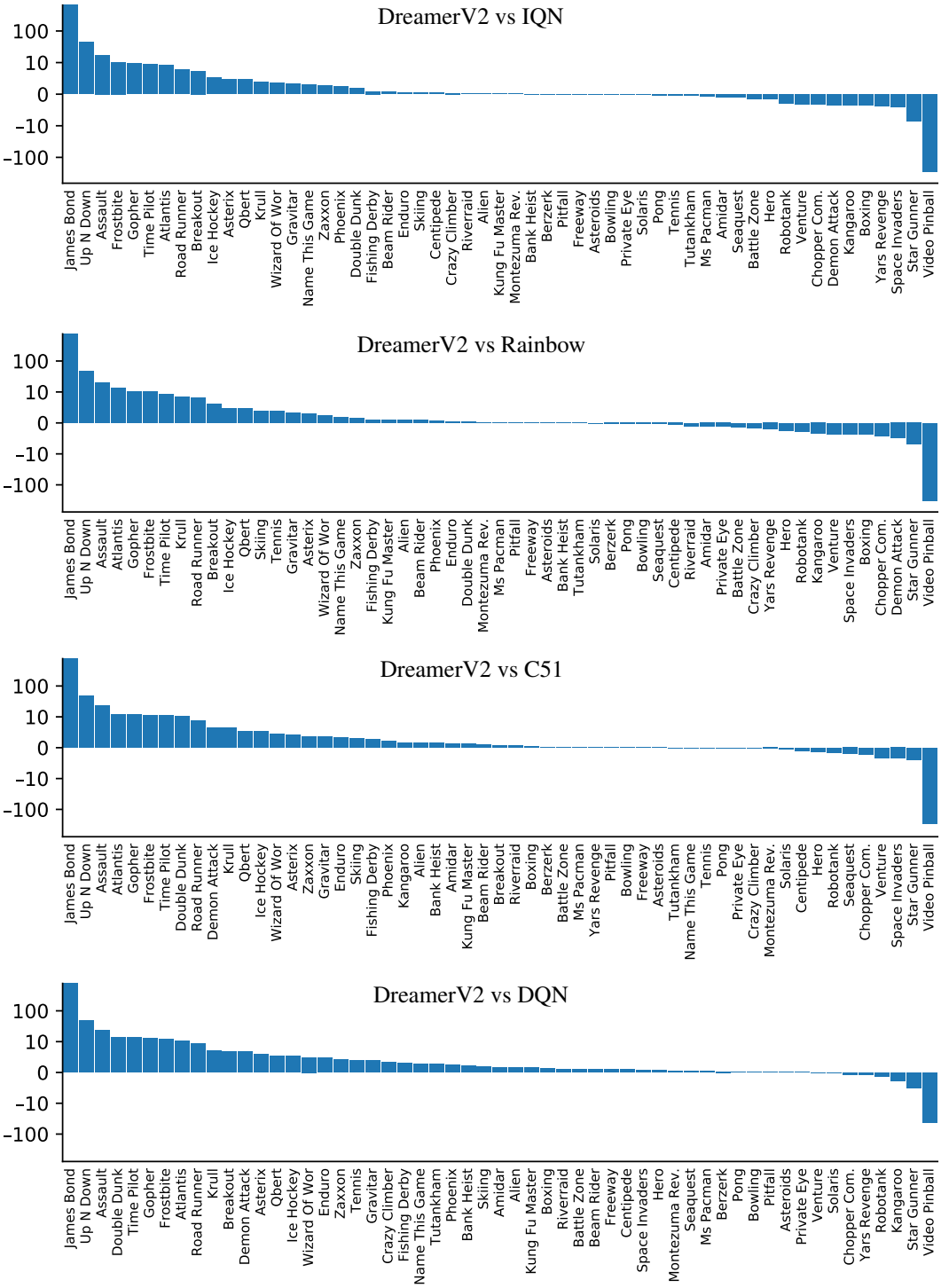

Figure C.1: Atari agent comparison. The bars show difference in gamer normalized scores at 200M steps. DreamerV2 outperforms the four model-free algorithms IQN, Rainbow, C51, and DQN while learning behaviors purely by planning within a separately learned world model. DreamerV2 achieves higher or similar performance on all tasks besides Video Pinball, where we hypothesize that the reconstruction loss does not focus on the ball that makes up only one pixel on the screen.

D ATARI INDIVIDUAL SCORES

	Baselines			Algorithms		
Task	**Random**	**Gamer**	**Record**	**Rainbow**	**IQN**	**DreamerV2**
Alien	228	7128	251916	3457	4961	**5332**
Amidar	6	1720	104159	**2529**	2393	2007
Assault	222	742	8647	3229	4885	**13432**
Asterix	210	8503	1000000	18367	10374	**26925**
Asteroids	719	47389	10506650	1484	**1585**	897
Atlantis	12850	29028	10604840	802548	890214	**1005430**
Bank Heist	14	753	82058	**1075**	**1052**	**1063**
Battle Zone	2360	37188	801000	**40061**	**40953**	27483
Beam Rider	364	16926	999999	6290	7130	**10193**
Berzerk	124	2630	1057940	**833**	648	660
Bowling	23	161	300	**43**	39	32
Boxing	0	12	100	**99**	**98**	82
Breakout	2	30	864	120	79	**208**
Centipede	2091	12017	1301709	**6510**	3728	4898
Chopper Command	811	7388	999999	**12338**	9282	1630
Crazy Climber	10780	35829	219900	**145389**	132738	134512
Demon Attack	152	1971	1556345	**17071**	15350	13208
Double Dunk	-19	-16	21	**22**	**21**	**22**
Enduro	0	860	9500	2200	**2203**	**2316**
Fishing Derby	-92	-39	71	42	45	**57**
Freeway	0	30	38	**34**	**34**	**34**
Frostbite	65	4335	454830	8208	7812	**47384**
Gopher	258	2412	355040	10641	12108	**30535**
Gravitar	173	3351	162850	1272	1347	**4855**
Hero	1027	30826	1000000	**46675**	36058	22744
Ice Hockey	-11	1	36	0	-5	**24**
James Bond	7	29	45550	1097	3166	**17614**
Kangaroo	52	3035	1424600	**12748**	**12602**	9020
Krull	1598	2666	104100	4066	8844	**10347**
Kung Fu Master	258	22736	1000000	26475	**31653**	**32625**
Montezuma Revenge	0	4753	1219200	500	500	**650**
Ms Pacman	307	6952	290090	3861	**5218**	3952
Name This Game	2292	8049	25220	9026	6639	**12378**
Phoenix	761	7243	4014440	8545	5102	**9888**
Pitfall	-229	6464	114000	-20	-13	**0**
Pong	-21	15	21	**20**	**20**	18
Private Eye	25	69571	101800	**21334**	4181	150
Qbert	164	13455	2400000	17383	16730	**42653**
Riverraid	1338	17118	1000000	**20756**	15183	16166
Road Runner	12	7845	2038100	54662	58966	**97720**
Robotank	2	12	76	**66**	**66**	56
Seaquest	68	42055	999999	9903	**17039**	5735
Skiing	-17098	-4337	-3272	-28708	-11162	-9530
Solaris	1236	12327	111420	1583	**1684**	1036
Space Invaders	148	1669	621535	4131	**4530**	2112
Star Gunner	664	10250	77400	57909	**80003**	20280
Tennis	-24	-8	21	0	**23**	21
Time Pilot	3568	5229	65300	12051	11666	**24752**
Tutankham	11	168	5384	**239**	**251**	235
Up N Down	533	11693	82840	34888	59944	**551051**
Venture	0	1188	38900	**1529**	1313	0
Video Pinball	16257	17668	89218328	**466895**	415833	4386
Wizard Of Wor	564	4756	395300	7879	5671	**10832**
Yars Revenge	3093	54577	15000105	45542	**84144**	13028
Zaxxon	32	9173	83700	14603	11023	**18759**

Table D.1: Atari individual scores. We select the 55 games that are common among most papers in the literature. We compare the algorithms DreamerV2, IQN, and Rainbow to the baselines of random actions, DeepMind's human gamer, and the human world record. Algorithm scores are highlighted in bold when they lie within 5% of the best algorithm.