
# DreamerV2: Mastering ATARI with Model-based Reinforcement Learning

Benjamin Möckl

Seminar: Menschzentrierte Künstliche Intelligenz

Lecturer: Katharina Weitz

Professor: Prof.Dr. Elisabeth André

December 5, 2023

**Abstract**

The algorithm *DreamerV2* presents a recent breakthrough in the field of model-based reinforcement learning, reaching state-of-the-art performance on the popular ATARI Learning Environment benchmark. This seminar paper gives an overview over model-based reinforcement learning, shows the predecessors of *DreamerV2* and explains the algorithm and neural-network architecture in detail. The results, possible applications and the features of model-based reinforcement learning are discussed.

# Contents

# 1 Introduction

Learning from pixel observations is a challenging problem for reinforcement learning, as it requires learning an accurate interpretation or representation of the high-dimensional inputs and predicting their dynamics under different actions. In contrast, humans are very good at learning from images, as we develop mental models of the games we try to learn. Our brains learn abstract representations of space and time, and we base our decisions on our internal model of the world around us (Ha and Schmidhuber 2018b). Evidence even suggests, that what we perceive is governed by our brain's prediction of the future based on our internal model (Nortmann et al. (2015), Ha and Schmidhuber (2018b)). Model-based reinforcement learning tries to mimic this human-like behaviour by learning a representation model of an environment in unsupervised fashion just by observing it. Model-based learning offers many theoretical benefits over model-free algorithms, but was not put to practice effectively until very recently. The here discussed *DreamerV2* agent from 2021 was able to surpass previous model-free and model-based algorithms and to reach new state-of-the-art performance, empirically showing the effectiveness of said benefits (see results in section 4.3).

# 2 Reinforcement learning

## 2.1 Overview

Reinforcement learning (RL) is a field of study that tries to construct algorithms called *agents*, that are able to learn how to make decisions in an *environment* through trial-and-error interactions. The only feedback an agent receives is in the form of delayed numerical rewards.
The agent interacts continuously with the environment by observing its *state* $s_t$, then choosing an *action* $a_t$ and executing it. The environment changes according to the action, and reports a reward signal $r_{t+1}$ back to the agent, together with the new state $s_{t+1}$. This cycle (see figure 1) continues until the environment reaches a terminal state $s_T$, or a time-limit is reached. This forms a discrete sequence of observations, actions and rewards, denoted with *time* $t \in \mathbb{N}_0$, forming a *trajectory* (1).

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, ... \tag{1}$$

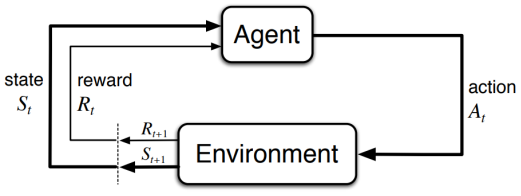



Figure 1: Agent-environment-interaction cycle in an MDP (Sutton and Barto 2018)

A Markov Decision Process (MDP) is a formal description of a reinforcement learning environment, that follows the notion above. It enables precise theoretical statements to be made, to mathematically formulate, analyze and proof reinforcement learning algorithms. Most RL environments and applications can be formulated as a MDP. If the agent doesn't have access to the complete state of the environment it interacts with, but observes only a part of it, the notion of a partially-observable MDP is used.

A MDP is a tuple $\langle S, A, p, r \rangle$, where $S$ denotes the state-space and $A$ the action-space. $p$ (2) is the state transition probability function, describing the dynamics of the MDP and $r$ (3) is the reward function (Sutton and Barto 2018).

$$p : S \times A \times S' \to [0, 1] \tag{2}$$

$$r : S \times A \times S' \to \mathbb{R} \tag{3}$$

$$G_t = \sum_{k=0}^{H} \gamma^t R_{t+k+1};\ 0 \leq \gamma \leq 1 \tag{4}$$

$$G_t = R_{t+1} + \gamma G_{t+1} \tag{5}$$

$$G_H = R_{H+1} \tag{6}$$

An agent seeks to maximise the expected discounted sum of future rewards G (4), where the discount rate $\gamma$ weighs immediate returns higher than ones far in the future. This can also be written as a recursive function (5 & 6).

$$\pi(s_t, a) \to [0, 1] \tag{7}$$

$$V : v_\pi(s) \doteq \mathbb{E}_\pi \left[ G_t \middle| S_t = s \right] \tag{8}$$

$$Q : q_\pi(s, a) \doteq \mathbb{E}_\pi \left[ G_t \middle| S_t = s, A_t = a \right] \tag{9}$$

Learning to solve environments efficiently is usually done with model-free RL. There the agent tries to approximate the value-function $V$ (8), the Q-function $Q$ (9) or the optimal policy function $\pi^*$ (7). Neural networks are the only feasible function approximators for complex environments with numerically big action spaces and/or state spaces, as well as non-linear and probabilistic state transitions, and can be efficiently trained with gradient descent. Using large neural networks in reinforcement learning is called Deep-RL (Sutton and Barto 2018).

## 2.2 Model-based reinforcement learning

Model-based RL tries to learn a model of the environment. A simple approach is learning a different representation for the observations $o_t$ in form of an auto-encoder (Ha and Schmidhuber 2018b), which can be trained efficiently with unsupervised learning techniques. The output of the auto-encoder is called a latent state $s_t$, which is usually a small real-valued vector. This reduces the size of the observations drastically, speeding up learning and reducing the memory footprint, as "learning policies from physical state based features is significantly more sample-efficient than learning from pixels" Laskin et al. (2020)(supported by Lee et al. (2020), Kaiser et al. (2019)).

The benefits of learning a latent space of the original environment are two-fold: It transforms a partially-observable MDP into a fully observe-able MDP, as now the agent can be provided with the complete latent state, resulting in more robust learning and generalization (Ha and Schmidhuber 2018b). Second, due to the compression, training an agent on latent states has a lower computation and memory footprint compared to learning on the original state of an environment (Hafner et al. 2019a).
A more sophisticated approach is learning the dynamics of an environment, which means approximating the state-transition function $p$ (2) and the reward signal $r$ (3). This can also be achieved with unsupervised learning techniques: by training a network to predict the next observation one step into the future from the current one, and calculating a loss from the difference to the real next observation (Hafner et al. 2019b).
To facilitate generalization inside the model and to leverage the efficiency benefits of latent states, the dynamics model is usually prefixed with an auto-encoder and predicts the next latent state $s_{t+1}$ directly from the current output of the encoder $s_t$ (e.g. Hafner et al. (2019b)).
To have an agent act optimally in an environment using a learned model, either a planning algorithm can be used (see Hafner et al. (2019b)), or a model-free RL algorithm can be trained against the learned model (Ha and Schmidhuber (2018b), Hafner et al. (2019a)). As an improved agent policy might discover new unseen states, the model needs to be re-trained to incorporate these new states. Then, re-training the agent might be necessary to optimize its behaviour in the extended world-model. Usually the model and the agent are trained iteratively in this cycle of optimization, that can be seen in figure 2.

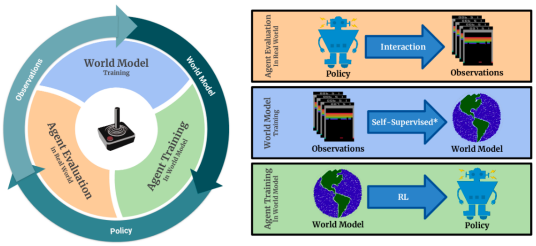



Figure 2: Agent-environment training cycle in model-based RL (Kaiser et al. 2019)

### 2.2.1 Benefits and downsides

Model learning enables learning from the information-rich signal of many observations of the environment, compared to only using the (usually very sparse) reward-signal, as it is done in model-free reinforcement learning. Another benefit is the ability to simulate a slow or inaccessible environment, which enables planning algorithms as an option for the agent, and can be much more efficient in general. A compact latent state, that encodes an environment state more densely, has been hypothesized to have a positive effect on agent learning speed, stability and final score, as it seems to reduce observation noise, reduces the dimension, and increases robustness due to generalizing (Sutton and Barto (2018), Hafner et al. (2020)). Having access to a world-model in form of a

fully differentiable recurrent computation graph (in form of a trained neural network) enables using backpropagation through the world-model for training a policy (Ha and Schmidhuber 2018b). Finally, learned dynamics can be task-independent and have shown to be effective for transfer to novel tasks (Byravan et al. 2020).
These benefits apply only, if the observations provided by the environment aren't already a compact descriptive state, and if no fast simulator exists. Learning a model, despite having access to a fast, perfectly accurate model, can even have downsides by introducing a slight bias, although in practice only increased computation time has been reported so far (see ablation study of *PlaNet* (Hafner et al. 2019b) in figure 5).
Also learning a model might be a much harder task than learning a policy in complex environments with simple tasks. Anecdotal evidence for this is the poor score of *DreamerV2* on *VideoPinball*, see section 4.3.
Despite these many theoretical benefits of model-based RL, no competitive performance compared to state-of-the-art model-free RL algorithms had been achieved until very recently (2018 *PlaNet* Hafner et al. (2019b) for continuous control, 2021 *DreamerV2* Hafner et al. (2020) for *ALE*), highlighting the importance of the here discussed papers.

## 2.3 Benchmarks

The goal of reinforcement learning research is to develop generally competent agents, which comes with the problem on how to evaluate and compare them (Bellemare et al. 2013). Therefore, benchmarking environments were created that can broadly be categorized into two groups:
1) *Continuous control problems* mimic robotic control tasks in the real world, usually have large action-spaces with real-valued inputs and have an underlying physics simulation. They aim to evaluate agents, that can perform well when deployed to robotic applications in the real world.
2) *Game Environments* usually use video games as environments. Video games are very interesting for RL, as they are challenging for both humans and RL agents. They usually have a very large state space, both long-term planning and quick decision-making are needed, and them being popular among human gamers, strong baselines usually exist.
There also exist many purpose-built benchmarks for modelling specific tasks, that are either very challenging for current state-of-the-art RL algorithms, or are modelling possible future applications for RL agents.

Two of the most influential benchmark suites for RL are the following:

### 2.3.1 MuJoCo

Continuous control problems are of great interest in reinforcement learning, due to its close connection to robotics (and controlling robots with RL) and the real world. For decades, simple physics simulations like cart-pole have been used to evaluate agents or controllers. An early description of cart-pole can be found in (Barto et al. 1983) from 1983. *MuJoCo* (Todorov et al. 2012) is a publicly available physics engine, which runs very efficiently. It is used in a variety of continuous control benchmarks, most famously *humanoid* for teaching

a whole human mannequin to walk, as well as robotics simulations and classic control problems like the one described above. Since the release of *OpenAI Gym* (Brockman et al. 2016) in 2016, these environments are conveniently accessible.

#### 2.3.2 ALE

The Arcade Learning Environment *ALE* (Bellemare et al. 2013), is one of the most popular benchmarks for RL algorithms. It combines a rich palette of different ATARI games, which are challenging for reinforcement learning for different reasons: Some require long-term planning, some have very sparse rewards and require sophisticated exploration, and due to their nature of being popular games, there exist strong human records, that in most of the games are yet to be beaten. The inputs to the environment are 18 discrete actions defined by the joystick controller, the outputs are a 2D array of 160 by 210 7-bit pixels, as well as a reward signal. All of the software is publicly available on GitHub and also accessible through *OpenAI Gym* (Brockman et al. 2016).

As a RL benchmark, usually agents are trained and evaluated on a subset of 55 ATARI games, and the results are aggregated and normalized to a "human baseline" (Mnih et al. 2015), mean and median are both common metrics. The authors of *DreamerV2* criticize this metric in their paper, and propose a different one: *Clipped Record Mean*. They propose normalizing the scores not with the "human baseline" scores that were collected by (Mnih et al. 2015), but by the currently standing human world records, which are well documented and provided by (Toromanoff et al. 2019). They also propose clipping the normalized scores at 100%, because in a few very simple games RL algorithms can reach orders of magnitude more points than humanly possible, which would bias the metric. A comparison of the different scoring metrics can be seen in figure 9.

# 3 Related work

## 3.1 Prior work

This section describes three papers, which can be seen as the foundations for the *DreamerV2* algorithm (4), both from a theoretical standpoint as well as architecture wise. *PlaNet* (3.1.2) and *Dreamer* (3.1.3) are even from the same authors, forming the direct predecessors of *DreamerV2*.

#### 3.1.1 WorldModels (2018)

The paper *Recurrent World Models Facilitate Policy Evolution* (Ha and Schmidhuber 2018b) describes the first RL algorithm, that learns a world-model directly from high-dimensional pixel observations. The agent is composed of 3 components (Figure 3):
$V$ is a variational auto-encoder (VAE, Kingma and Welling (2013)), its input is the raw pixel observations $o_t$ of the environment. Its output is an encoded latent state $z_t$.
$M$ is a recurrent neural-network (RNN, urgen Schmidhuber (1990)), which processes $z_t$ together with an action $a_t$ and outputs a Gaussian probability distri-

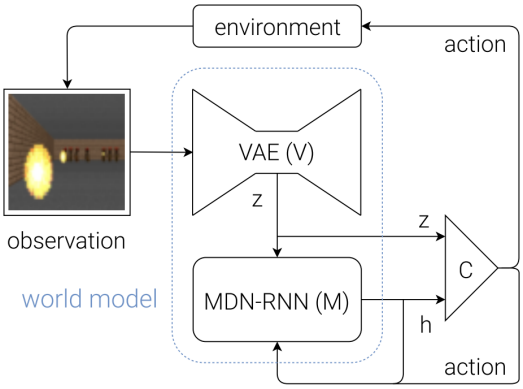



Figure 3: The agent architecture from the paper WorldModels (Ha and Schmidhuber 2018b).

| METHOD | AVG. SCORE |
|---|---|
| DQN (PRIEUR, 2017) | $343 \pm 18$ |
| A3C (CONTINUOUS) (JANG ET AL., 2017) | $591 \pm 45$ |
| A3C (DISCRETE) (KHAN & ELIBOL, 2016) | $652 \pm 10$ |
| CEOBILLIONAIRE (GYM LEADERBOARD) | $838 \pm 11$ |
| V MODEL | $632 \pm 251$ |
| V MODEL WITH HIDDEN LAYER | $788 \pm 141$ |
| **FULL WORLD MODEL** | $\mathbf{906 \pm 21}$ |

Figure 4: Scores of different RL algorithms in the environment *CarRacing-v0* from Ha and Schmidhuber (2018b)

bution of latent states one step into the future $p(z_{t+1})$, which can be sampled to obtain $z_{t+1}$, a prediction of what the next latent state will be.
$C$ is a single layer neural network with no activation, its inputs are the outputs of $V$ and $M$, its output is a vector with values corresponding to different actions in the environment. It is designed specifically to be as simple and small as possible, so the world-model is the main contributor to agent performance.
The training of this algorithm follows 4 steps: First, through a random policy, observations of the environment are collected. Second, $V$ is trained to compress the input images to a latent state by trying to reconstruct the original image from it. Third, $M$ is trained using the observation trajectories from the first step, filtered through $V$. Finally, $C$ is trained via an evolution-based technique. This work was able to solve the *CarRacing-v0* and *VizDoom:TakeCover* benchmarks. The authors did an ablation study, where they compared training the controller with the latent state input only, additionally adding a hidden layer to the controller and with the full described architecture. The results (figure 4) show a clear improvement in performance using the full world-model. There exists an interactive web-demo of the world-model at Ha and Schmidhuber (2018a).

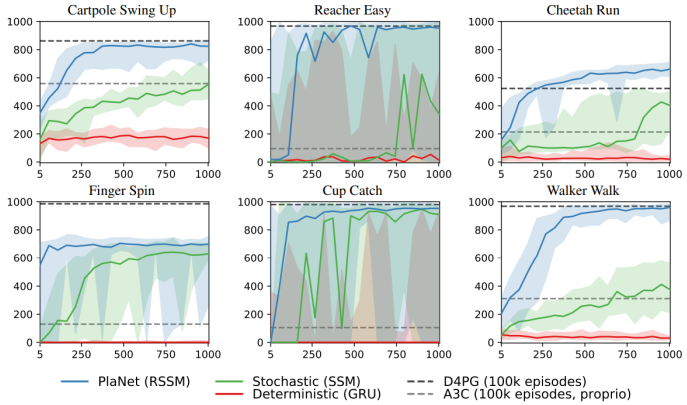



Figure 5: Learning curves for PlaNet ablation studies (Hafner et al. 2019b).

| **Method** | **Modality** | **Episodes** | **Cartpole Swing Up** | **Reacher Easy** | **Cheetah Run** | **Finger Spin** | **Cup Catch** | **Walker Walk** |
|---|---|---|---|---|---|---|---|---|
| A3C | proprioceptive | 100,000 | 558 | 285 | 214 | 129 | 105 | 311 |
| D4PG | pixels | 100,000 | 862 | 967 | 524 | 985 | 980 | 968 |
| PlaNet (ours) | pixels | 1,000 | 821 | 832 | 662 | 700 | 930 | 951 |
| CEM + true simulator | simulator state | 0 | 850 | 964 | 656 | 825 | 993 | 994 |
| Data efficiency gain PlaNet over D4PG (factor) | | | 250 | 40 | 500+ | 300 | 100 | 90 |

Figure 6: Scores of different algorithms on MuJoCo continuous control problems from Hafner et al. (2019b).

### 3.1.2 PlaNet (2018)

The Paper *Learning Latent Dynamics for Planning from Pixels* (Hafner et al. 2019b) trains a world-model to solve continuous control problems from raw pixels by using online planning, and achieves performance close to and sometimes higher than strong model-free algorithms. One of the key contributions of this work is the use of a recurrent state-space model (RSSM) architecture for the world-model. This contains both deterministic and stochastic components for predicting the next model state. In an ablation study (see figure 5) they were able to show that the combination of both statistical and deterministic predictions yields significantly better results than either of these alone. The statistical model outputs gaussian distributions in latent space $p(z_{t+1})$ for the next timestep analoguous to the model $M$ described in 3.1.1, the deterministic model is a RNN, that outputs a prediction for $z_{t+1}$ directly.
The final scores of the described architecture named *PlaNet* can be seen in figure 6. The planning algorithm used as the agent's policy *CEM* (Henaff et al. 2017) was run on the true environments directly for comparison. The results show that the learned world-model performed nearly as well, while also reaching or even surpassing the performance of D4PG (Barth-Maron et al. 2018).
The authors also use the term *data efficiency* when comparing to model-free algorithms. *PlaNet* was able to reach its reported performance in only 1000

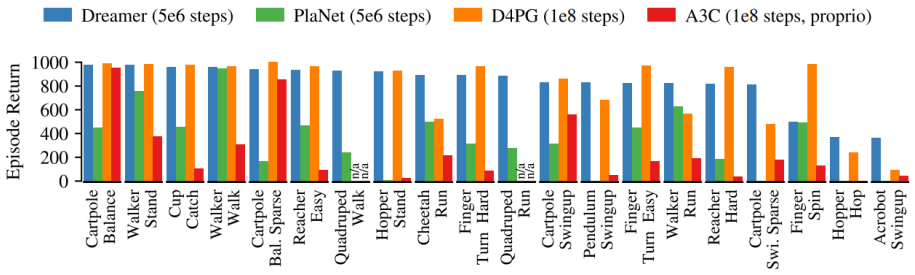



Figure 7: Performance graph of different algorithms on MuJoCo continuous control problems from Hafner et al. (2019a).

episodes on the environment, clearly showcasing some of the theoretical benefits of model-based RL (see 2.2.1). This however is not indicative of faster learning (usually measured in wallclock-time or GPU hours) or general data efficiency (e.g. *PlaNet* requires a replay buffer of past trajectories, which model-free policy-optimization algorithms typically don't use), but is an important metric to take into consideration if the environment is slow to evaluate episodes in, e.g. real world robotic tasks.

### 3.1.3 Dreamer (2019)

The paper *DREAM TO CONTROL [...]* (Hafner et al. 2019a) presents a RL framework, that builds on the already powerful capabilities of world-models, to integrate them with model-free reinforcement learning agents. By training an actor-critic agent on their world-model, they were able to reach new state-of-the-art performance, data-efficiency and computation time on *MuJoCo* (2.3.1) continuous control tasks, see figure 7.
Their agent *'Dreamer'* is composed of the following networks: They use the convolutional encoder and decoder networks from *WorldModels* (3.1.1), the *RSSM* network from *PlaNet* (3.1.2) and a value-network together with a policy-network forming an actor-critic agent. This actor-critic agent was trained on trajectories of 15 steps, that were produced purely by its interaction with the world-model, after feeding it frames from the real environment as starting points. Generating trajectories with the *RSSM* model also enabled the use of multi-step returns for training the value network.
An ablation study by the authors (see Figure 8) showed the importance of learning a value function for agent performance, compared to only training a policy network.
*Dreamer* was evaluated on some ATARI games too, but the performance was not competitive against state-of-the-art model-free algorithms (Hafner et al. 2019a).

## 3.2 Related work

### 3.2.1 SimPle (2020)

*Model-Based RL for Atari* (Kaiser et al. 2019) describes the algorithm *'Simulated Policy Learning'* (SimPLe). It uses a video prediction model as world-model, which is trained to predict future frames accurately. They train a policy

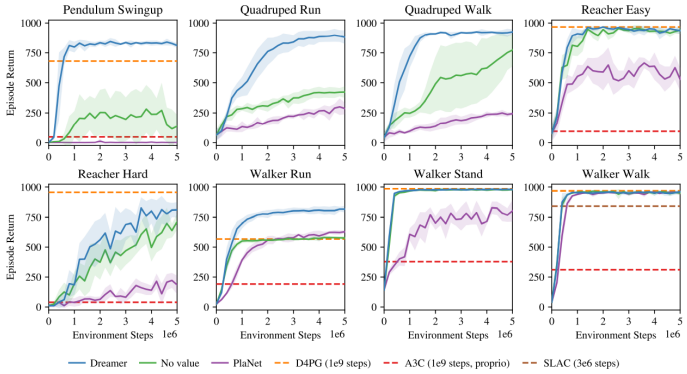



Figure 8: Ablation study of the Dreamer agent: Using an actor-critic agent (blue) is compared to training only a policy network (green) (Hafner et al. 2019a).

optimization algorithm on their world-model, and outperform previous state-of-the-art model-free algorithms on a subset of the *ALE* benchmark. Different from *Dreamer*, they train their policy not on a latent state, but on generated predicted video frames (and a predicted reward signal) using the *PPO* (Schulman et al. 2017) algorithm.

### 3.2.2 CURL (2020)

The paper *Contrastive Unsupervised Representations for RL* (Laskin et al. 2020) describes a method to learn a world-model from raw pixels using contrastive learning and performs off-policy control on top of the extracted features. CURL outperforms prior pixel-based methods (including *Dreamer* and *SimPLe*), both model-based and model-free ones, on continuous control tasks, but performs worse than *SimPLe* on *ALE*. The main contribution of this work is the use of contrastive loss Henaff (2020) in the training of the world-model, which is an unsupervised learning technique based on automatic data augmentation.

## 3.3 Non-model-based RL algorithms for comparison

### 3.3.1 D4PG (2018)

*Distributed Distributional Deterministic Policy Gradients* (D4PG) (Barth-Maron et al. 2018) is a distributed framework for off-policy Policy Gradient learning, incorporating distribution learning (Bellemare et al. 2017), N-step returns (Sutton and Barto 2018) and prioritized experience replay (Schaul et al. 2015). *D4PG* is able to solve difficult obstacle-based locomotion tasks and a wide variety of continuous control tasks (*MuJoCo* benchmarks, see 2.3.1) with new state-of-the-art performance.

### 3.3.2 Rainbow-DQN (2018)

*RainbowDQN* is a Q-learning based algorithm from 2018 (Hessel et al. 2018), that combined 6 different improvements to the original Deep Q-Networks (DQN)

(Mnih et al. 2013) algorithm, greatly increasing stability and final performance, reaching a new state-of-the-art on the *ALE* benchmark (2.3.2).

### 3.3.3 IQN (2018)

*Implicit Quantile Networks* (IQN) (Dabney et al. 2018) iteratively improved on distributional Q-Learning (Bellemare et al. 2017). Its performance gets beaten by *RainbowDQN*, but it was the second strongest algorithm on the *ALE* (2.3.2) benchmark when it came out.

# 4 DreamerV2 (2021)

*MASTERING ATARI WITH DISCRETE WORLD MODELS* (Hafner et al. 2020) is an iterative improvement over *Dreamer* (3.1.3). While the *Dreamer* agent achieved state-of-the-art performance on continuous control tasks, it fell short on the *ALE* benchmark compared to the best model-free algorithms. *DreamerV2* not only closes this performance gap, but surpasses them, see Figure 9.

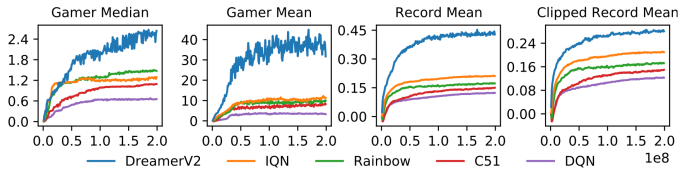



Figure 9: Comparison of learning curves on *ALE* (aggregated of 55 ATARI games) (Hafner et al. 2020).

## 4.1 Architecture

### 4.1.1 World-model

The world-model of *DreamerV2* can be seen in figure 10, the numbers in the following section correspond to the numbers in the graphic.
Network (1) is a convolutional neural network, that compresses image observations $o_t$ from the environment into a more compact state. From there, Network (3) encodes it into the categorical latent state $z_t$, which is the latent state output of the model, on which also the actor-critic agent will be trained. Network (1) and (3) together are called the *Representation model*. The second input of Network (3) is the hidden state $h_t$ of the *Recurrent model* (2), which encodes deterministic memory about past states of the environment, using the latent state of the previous timestep $z_{t-1}$ and the action $a_{t-1}$ that was taken. The *Transition predictor* (4) predicts the categorical latent state $\hat{z}_t$ only from that hidden state. The *Image predictor* (5) is a deconvolutional neural network, that outputs a reconstruction of the observation $\hat{o}_t$ from the latent state and the hidden state. The *Reward predictor* (6) predicts the reward $\hat{r}_t$, that the real environment would provide as $r_t$ after taking action $a_{t-1}$ on state $z_{t-1}$ together with the resulting state $z_t$. The *Discount predictor* (also 6) predicts if

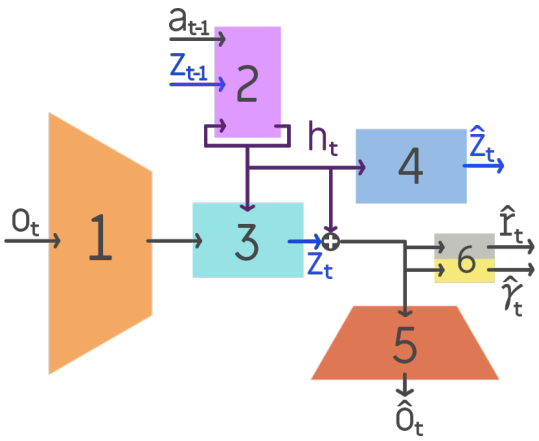



Figure 10: Diagram of the *DreamerV2* world-model architecture, extracted from its description in its paper (Hafner et al. 2020). For more details see (4.1). It is composed of an convolutional encoder network (1), a *RSSM* (Hafner et al. 2019b) (2, 3, 4), reward- and discount predictor networks (6) and a deconvolutional decoder network (5).

an episode ended after taking action $a_{t-1}$ on state $z_{t-1}$, in form of the discount factor $\hat{\gamma}_t$ that will be set to zero on episode ends, and is $\gamma$ otherwise, usually 0.999. Also should be noted, that all outputs except the categorical latent states are not values, but Gaussian distributions over these values (this also includes the predicted image $o_t$).

### 4.1.2 World-model loss function

All of the world-model's components are optimized jointly using the Adam optimizer (Kingma and Ba 2014). The authors use a single loss function, which has 4 terms summed together:
1) The first term *Image loss* measures the difference between an observation $o_t$ from the environment and its reconstruction $\hat{o}_t$.
2) The second term *KL-loss* (Kulback-Leibler divergence) measures the difference between the real categorical latent-state $z_t$ and the model's prediction of it: $\hat{z}_t$. It is also weighted with a factor $\beta$, which was set to 0.1 for *ALE* and to 1.0 for continuous control.
3) The *reward loss* measures the difference between the predicted reward $\hat{r}_t$ and the actually received reward $r_t$ from the environment.
4) The *discount loss* measures, if the discount predictor correctly predicted the end of an episode (where the discount should be zero instead of $\gamma$, usually 0.999).

### 4.1.3 Actor-Critic agent

The actor-critic agent (adapted from the *A3C* algorithm Mnih et al. (2016)) is composed of two neural networks: An *actor*, which computes a categorical distribution over actions from a categorical latent state $z_t$, which can then be

sampled to receive a chosen action $a_t$ to execute in the environment or world-model (modelling equation 7). And a *critic* network, that predicts the value of the current latent state $z$ from it (see equation 8).
To train the actor-critic agent on top of the world-model, first, initial states $\hat{z}_0$ are sampled from latent states that were encountered during world-model training. For these initial states, the actor network predicts an action $a_0$ to take. Then, the *Recurrent model* (2) uses $z_0$, $a_0$ and its randomly initiated hidden state $h_0$ to compute $z_1$, as well as $r_1$ and $\gamma_1$ using the *Discount predictor* and *Reward predictor* (6). This would suffice for 1-step RL learning, but by feeding back $z_1$ into the actor network to gather $a_1$, and feeding both values into the *Recurrent model* (2), and repeating this cycle up to $H$ times, the use of n-step returns is enabled. For training *DreamerV2*, the dream horizon $H$ was set to 15. The trajectories produced by this cycle are called *dreams*, as they are hallucinated by the world-model, giving the algorithm its name.

#### 4.1.4 Agent loss function

The critic is trained using n-step $\lambda$-targets (Sutton and Barto 2018). The actor loss can be configured to use one of two terms: a *Reinforce* gradient loss (Sutton and Barto 2018) with the value from the critic as baseline, or straight-through gradients (Hafner et al. 2019a), which backpropagate directly through the learned dynamics. Additionally, to both variants an entropy regularizing term is added, which is weighted with the factor $\eta = 10^{-3}$ for *ALE* and with $\eta = 10^{-4}$ for continuous control tasks. Empirically tested, Reinforce gradients worked substantially better for *ALE*, and straight-through gradients respectively did much better for continuous control tasks (Hafner et al. 2020).

### 4.2 Modifications from DreamerV1

The main changes of the *DreamerV2* agent compared to its predecessor were the following:
1) The latent state $z$ of the world-model no longer uses Gaussian distributions, but categorical variables. An ablation study by the authors shows a significant improvement in performance, as can be seen in Figure 11 in the first graph.
2) The loss term for the KL divergence between latent state $z$ and by the *Transition model* predicted latent state $\hat{z}$ of the world-model was weighted down with a factor of $\beta = 0.1$ for *ALE* and $\beta = 1.0$ for continuous control. The authors refer to this as *KL Balancing*, and state that this facilitates learning a good latent state representation of the environment $z$ first (through the first loss term *Image loss*, see 4.1.2), and then improving the *Transition model*. The benefits of using *KL Balancing* can be seen in a second ablation study in Figure 11 in the second graph.
3) The number of parameters of the world-model was nearly doubled from 13M to 22M parameters.
4) Dynamics backpropagation was turned off to only use *Reinforce* gradients for training on *ALE*, and inversely for training on continuous control problems, as described in section 4.1.4, due to it significantly increasing performance in both benchmarks.
5) An entropy loss term was added to the actor network, replacing the need to add random noise to actions for exploration.

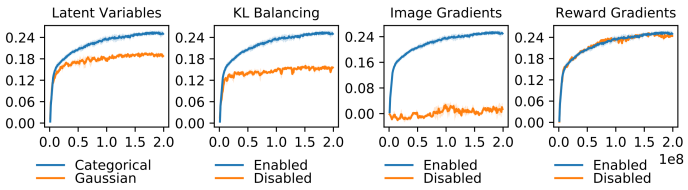



Figure 11: Ablation studies with modified versions of *DreamerV2* on *ALE* (record-clipped normalized means of 55 ATARI games) (Hafner et al. 2020).

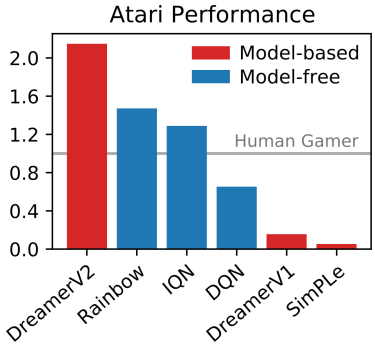
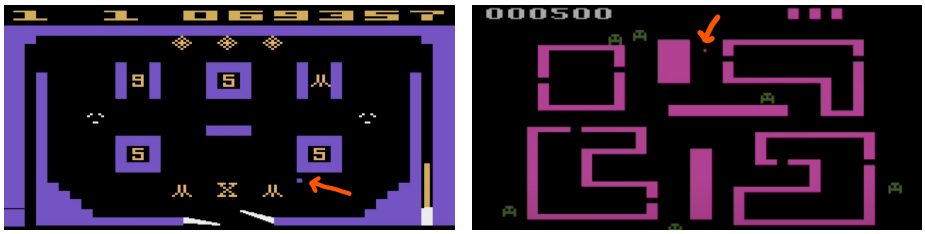



Figure 12: Performance on *ALE* of state-of-the-art model-free and model-based RL algorithms (human gamer normalized median score of 55 ATARI games) (Hafner et al. 2020).

## 4.3 Results

As seen in figures 9 and 12, the *DreamerV2* algorithm is the first model-based RL algorithm that not only catches up to recent model-free algorithms, but significantly outperforms them, reaching new state-of-the-art performance. The use of a discrete categorical latent state can be seen as one of the major discoveries.

Not all of the 55 ATARI games were solved by *DreamerV2*. In *VideoPinball*, it reached only one tenth of the scores of *Rainbow* and *IQN*, and in *Venture* it only achieved 2 points compared to 1529 points for *Rainbow*. Both environments have in common that the pinball or respectively the player is represented by a single pixel, while the surrounding environment is intricate and includes numerous moving elements (refer to the orange arrows in figure 13). The authors hypothesize that the image reconstruction loss of the world-model didn't encourage learning a meaningful latent representation due to the most important object in the game occupying only a single pixel.
For continuous control, only results for one experiment were published, due to the focus of the paper on the *ALE* benchmark. *DreamerV2* was able to solve the very challenging *humanoid-v4* environment from pixel inputs, which requires a human mannequin to first stand up from a lying position, and then walking as far as possible. A learning curve can be seen in figure 14. The authors state that "to the best of [their] knowledge, this constitutes the first published result of solving the humanoid environment from only pixel inputs" (Hafner et al. 2020).



(a) *Video Pinball* (b) *Venture*

Figure 13: Screenshots from two games of the *ALE* benchmark.

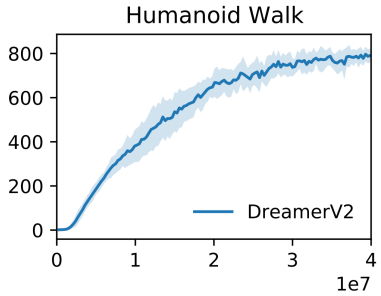


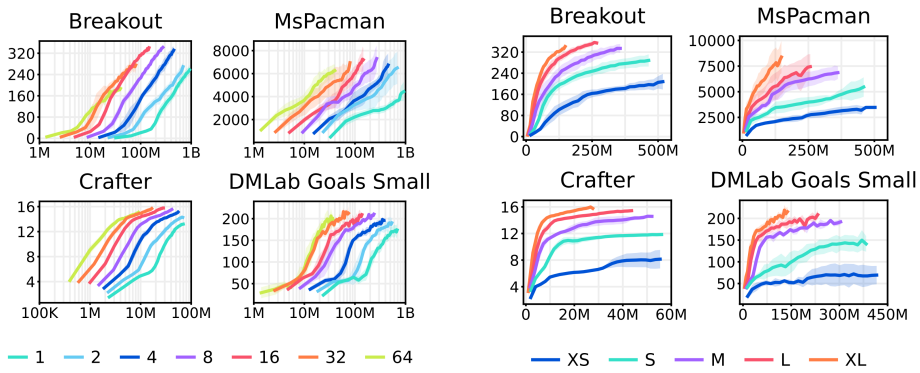

Figure 14: Performance on the *humanoid-v4* environment averaged over 5 runs (Hafner et al. 2020).

# 5 Conclusion

*DreamerV2* is a state-of-the-art model-based reinforcement learning agent that learns from pixel observations and surpasses human-level performance on the *ALE* benchmark. *DreamerV2* empirically demonstrates the benefits of world-models over model-free RL algorithms, such as improved sample efficiency, stability, and generalization (see section 2.2.1). It also shows that model-based reinforcement learning can surpass model-free methods. *DreamerV2* is a promising direction for developing more scalable and robust reinforcement learning agents, as its core concepts build on unsupervised model learning, and its core architecture allows for further improving its world-model architecture and the use of newer model-free algorithms as agent.

## 5.1 DreamerV3

Very recently (Jan 2023), the follow-up paper *Mastering Diverse Domains through World Models* (Hafner et al. 2023) was released, presenting the *DreamerV3* agent. It outperforms model-free and model-based algorithms in a wide range of benchmarks, both in terms of performance and in terms of data-efficiency (Hafner et al. 2023), see figure 16. Ablation studies from the authors show that the word-model has excellent scaling properties, see figure 15. It is the first algorithm that was able to learn to solve the difficult *Minecraft Diamond* task from pixel inputs.




(a) training ratio (b) model size

Figure 15: Ablation studies on the scaling properties of *DreamerV3*. a) The training ratio is the ratio of replayed steps to environment steps, an increase in replayed steps increases the data-efficiency. b) Larger models achieve not only higher final performance but also higher data-efficiency. (Hafner et al. 2023)

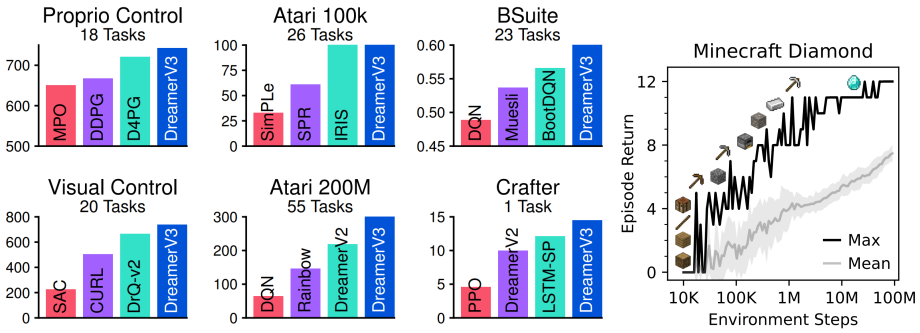



Figure 16: *DreamerV3* performance on different environments compared to other state-of-the-art model-free and model-based algorithms. (Hafner et al. 2023)

## 5.2 Possible future applications

By achieving state-of-the-art performance, *DreamerV2* enables efficient and performant model learning. Having access to a fast and reasonably accurate world-model trained from pixel inputs only, provides the ability to simulate and plan future events. This can also be effective in other research areas, most notably vision based robotic RL. There, multi-task learning could be greatly accelerated with an accurate world-model (see section 2.2.1).
Another interesting application could be using world-models for generating explanations for the decisions of RL agents. Generated explanations for the agent's decisions/suggestions can enhance human-agent collaboration by building trust and aiding the human to understand an agent's decisions, which increases the performance of their team (Silva et al. 2022). By simulating the agent's suggestions using the world-model, and then showing it to a human, or providing the human with the world-model directly for evaluating and testing its actions inside a simulation could be an effective novel approach to *XAI*.

# References

G. Barth-Maron, M. W. Hoffman, D. Budden, W. Dabney, D. Horgan, D. Tb, A. Muldal, N. Heess, and T. Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

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, jun 2013.

M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

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

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 tranferable latent dynamics models. In *Conference on Robot Learning*, pages 566–589. PMLR, 2020.

W. Dabney, G. Ostrovski, D. Silver, and R. Munos. Implicit quantile networks for distributional reinforcement learning. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1096–1105. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/dabney18a.html`.

D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evaluation @ONLINE, Mar. 2018a. URL `https://worldmodels.github.io/`. Last accessed 06 Mar 2023.

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

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

D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019b.

D. Hafner, T. P. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *CoRR*, abs/2010.02193, 2020. URL `https://arxiv.org/abs/2010.02193`.

D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

M. Henaff, W. F. Whitney, and Y. LeCun. Model-based planning in discrete action spaces. *CoRR*, abs/1705.07177, 2017. URL `http://arxiv.org/abs/1705.07177`.

O. Henaff. Data-efficient image recognition with contrastive predictive coding. In *International conference on machine learning*, pages 4182–4192. PMLR, 2020.

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.

L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, R. Sepassi, G. Tucker, and H. Michalewski. Model-based reinforcement learning for atari. *CoRR*, abs/1903.00374, 2019. URL `http://arxiv.org/abs/1903.00374`.

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.

M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL `http://arxiv.org/abs/1312.5602`.

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):529–533, 2015. doi: 10.1038/nature14236. URL `https://doi.org/10.1038/nature14236`.

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. PMLR, 2016.

N. Nortmann, S. Rekauzke, S. Onat, P. König, and D. Jancke. Primary visual cortex represents the difference between past and present. *Cerebral Cortex*, 25(6):1427–1440, 2015.

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

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

A. Silva, M. Schrum, E. Hedlund-Botti, N. Gopalan, and M. Gombolay. Explainable artificial intelligence: Evaluating the objective and subjective impacts of xai on human-agent interaction. *International Journal of Human–Computer Interaction*, pages 1–15, 2022.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. The MIT Press, 2 edition, 2018. ISBN 9780262039246. URL `https://lccn.loc.gov/2018023826`.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

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.

J. urgen Schmidhuber. Making the world differentiable: On using self-supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments (tr fki-126-90). 1990.