Transformers are Sample Efficient World Models

Vincent Micheli*
University of Geneva
vincent.micheli@unige.ch

Eloi Alonso*
University of Geneva
eloi.alonso@unige.ch

François Fleuret
University of Geneva
francois.fleuret@unige.ch

Abstract

Deep reinforcement learning agents are notoriously sample inefficient, which considerably limits their application to real-world problems. Recently, many model-based methods have been designed to address this issue, with learning in the imagination of a world model being one of the most prominent approaches. However, while virtually unlimited interaction with a simulated environment sounds appealing, the world model has to be accurate over extended periods of time. Motivated by the success of Transformers in sequence modeling tasks, we introduce IRIS, a data-efficient agent that learns in a world model composed of a discrete autoencoder and an autoregressive Transformer. With the equivalent of only two hours of gameplay in the Atari 100k benchmark, IRIS achieves a mean human normalized score of 1.046, and outperforms humans on 10 out of 26 games. Our approach sets a new state of the art for methods without lookahead search, and even surpasses MuZero. To foster future research on Transformers and world models for sample-efficient reinforcement learning, we release our codebase at https://github.com/eloialonso/iris.

1 Introduction

Deep Reinforcement Learning (RL) has become the dominant paradigm for developing competent agents in challenging environments. Most notably, human experts were surpassed by deep RL algorithms in a multitude of arcade [1, 2, 3], real-time strategy [4, 5], board [6, 7, 2] and imperfect information [8, 9] games. However, a common drawback of these methods is their extremely low sample efficiency. Indeed, experience requirements range from months of gameplay for DreamerV2 [3] in Atari 2600 games [10] to thousands of years for OpenAI Five in Dota2 [5]. While some environments can be sped up for training agents, real-world applications often cannot. Besides, additional cost or safety considerations related to the number of environmental interactions may arise [11]. Hence, sample efficiency is a necessary condition to bridge the gap between research and the deployment of deep RL agents in the wild.

Model-based methods [12] constitute a promising direction to achieve data efficiency. Recently, world models were leveraged in several ways: pure representation learning [13], lookahead search [2, 14], and learning in imagination [15, 16, 17, 3]. The latter approach is particularly appealing because training an agent inside a world model frees it from sample efficiency constraints. Nevertheless, this framework relies heavily on accurate world models since the policy is purely trained in imagination. In a pioneering work, Ha and Schmidhuber [15] successfully built imagination-based agents in toy environments. SimPLe recently showed promise in the more challenging Atari 100k benchmark

*Equal contributions.

Preprint. Under review.

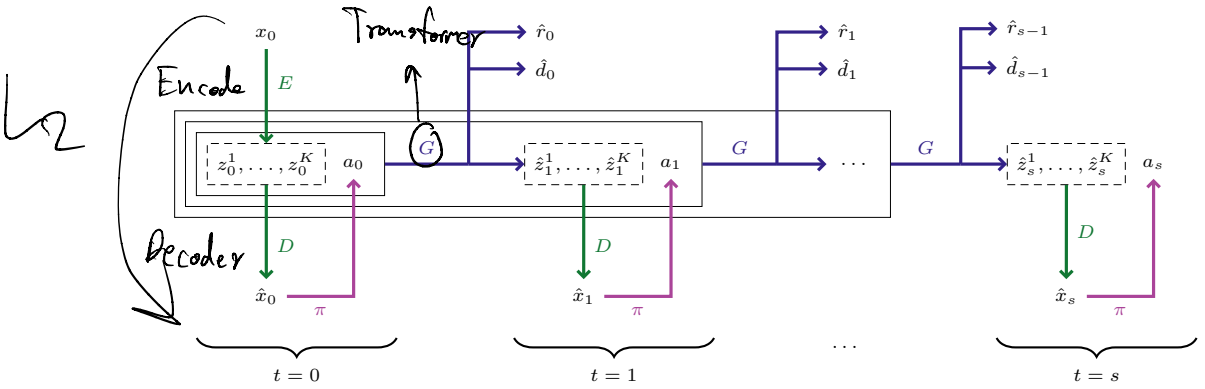


Figure 1: Unrolling imagination over time. This figure shows the policy π, depicted with purple arrows, taking a sequence of actions in imagination. The green arrows correspond to the encoder E and the decoder D of a discrete autoencoder, whose task is to represent frames in its learnt symbolic language. The backbone G of the world model is a GPT-like Transformer, illustrated with blue arrows. For each action that the policy π takes, G simulates the environment dynamics, by autoregressively unfolding new frame tokens that D can decode. G also predicts a reward and a potential episode termination. More specifically, an initial frame x_0 is encoded with E into tokens $z_0 = (z_0^1, \ldots, z_0^K) = E(x_0)$. The decoder D reconstructs an image $\hat{x}_0 = D(z_0)$, from which the policy π predicts the action a_0. From z_0 and a_0, G predicts the reward $\hat{r}_0$, episode termination $\hat{d}_0 \in \{0, 1\}$, and in an autoregressive manner $\hat{z}_1 = (\hat{z}_1^1, \ldots, \hat{z}_1^K)$, the tokens for the next frame. A dashed box indicates image tokens for a given time step, whereas a solid box represents the input sequence of G, *i.e.* (z_0, a_0) at $t = 0$, $(z_0, a_0, \hat{z}_1, a_1)$ at $t = 1$, etc. The policy π is purely trained with imagined trajectories, and is only deployed in the real environment to improve the world model (E, D, G).

[16]. Currently, the best Atari agent learning in imagination is DreamerV2 [3], although it was developed and evaluated with two hundred million frames available, far from the sample-efficient regime. Therefore, designing new world model architectures, capable of handling visually complex and partially observable environments with few samples, is key to realize their potential as surrogate training grounds.

The Transformer architecture [18] is now ubiquitous in Natural Language Processing [19, 20, 21, 22], and is also gaining traction in Computer Vision [23, 24], as well as in Offline Reinforcement Learning [25, 26]. In particular, the GPT [27, 20, 21] family of models delivered impressive results in language understanding tasks. Similarly to world models, these attention-based models are trained with high-dimensional signals and a self-supervised learning objective, thus constituting ideal candidates to simulate an environment.

Transformers particularly shine when they operate over sequences of discrete tokens [19, 21]. For textual data, one can easily build a vocabulary in multiple ways [28, 29], but this conversion is not straightforward with images. A naive approach would consist in treating pixels as image tokens, but standard Transformer architectures scale quadratically with sequence length, making this idea computationally intractable. To address this issue, VQGAN [30] and DALL-E [31] employ a discrete autoencoder [32] as a mapping from raw pixels to a much smaller amount of image tokens. Combined with an autoregressive Transformer, these methods demonstrate strong unconditional and conditional image generation capabilities. Such results suggest a new approach to design world models.

In the present work, we introduce IRIS (Imagination with auto-Regression over an Inner Speech), an agent trained in the imagination of a world model composed of a discrete autoencoder and a GPT-like autoregressive Transformer. IRIS learns behaviors by accurately simulating millions of trajectories. Our approach casts dynamics learning as a sequence modeling problem, where an autoencoder builds a language of image tokens and a Transformer composes that language over time. With minimal tuning and a drastically different architecture, IRIS outperforms a line of recent methods [16, 33, 34, 35, 13] for sample-efficient RL in the Atari 100k benchmark [16]. After only two hours of real-time experience, it achieves a mean human normalized score of 1.046, and reaches superhuman performance on 10 out of 26 games. We describe IRIS in Section 2 and present our results in Section 3.

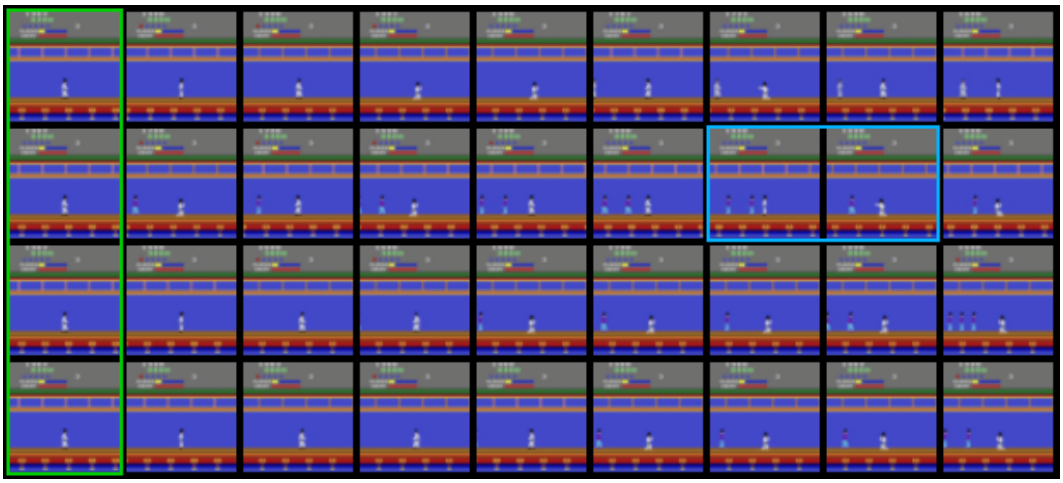


Figure 2: Four imagined trajectories in *KungFuMaster*. We use the same conditioning frame across the four rows, in green, and let the world model imagine the rest. As the initial frame only contains the player, there is no information about the enemies that will come next. Consequently, the world model generates different types and numbers of opponents in each simulation. It is also able to reflect an essential game mechanic, highlighted in the blue box, where the first enemy disappears after getting hit by the player.

2 Method

We formulate the problem as a Partially Observable Markov Decision Process (POMDP) with image observations $x_t \in \mathbb{R}^{h \times w \times 3}$, discrete actions $a_t \in \{1, \dots, A\}$, scalar rewards $r_t \in \mathbb{R}$, episode termination $d_t \in \{0, 1\}$, discount factor $\gamma \in (0, 1)$, initial observation distribution ρ_0, and environment dynamics $x_{t+1}, r_t, d_t \sim p(x_{t+1}, r_t, d_t \mid x_{\leq t}, a_{\leq t})$. The reinforcement learning objective is to train a policy π that yields actions maximizing the expected sum of rewards $\mathbb{E}_\pi[\sum_{t \geq 0} \gamma^t r_t]$.

Our method relies on the three standard components to learn in imagination [12]: experience collection, world model learning, and behavior learning. In the vein of Ha and Schmidhuber [15], Kaiser et al. [16], Hafner et al. [17, 3], our agent learns to act exclusively within its model of the world, and we only make use of real experience to learn the environment dynamics.

We repeatedly perform the three following steps:

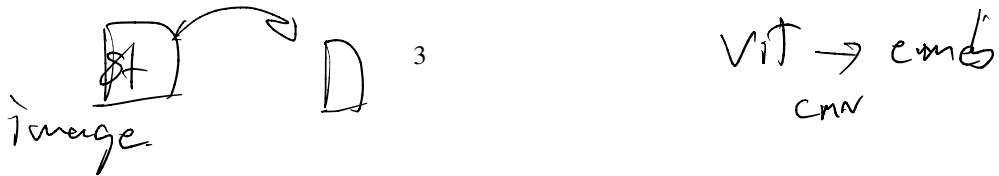
- `collect_experience`: gather experience in the real environment with the current policy.
- `update_world_model`: improve rewards, episode ends and next observations predictions.
- `update_behavior`: in imagination, improve the policy and value functions.

As outlined in Figure 1, we use a discrete autoencoder [32] to convert an image to tokens and back. As for the backbone of the world model, whose task is to capture environment dynamics, we rely on a GPT-like autoregressive Transformer [18, 20, 21]. We first describe the autoencoder and the Transformer in Sections 2.1 and 2.2, respectively. Section 2.3 then details the procedure to learn the policy and value functions in imagination. Appendix A provides a comprehensive description of model architectures and hyperparameters. Algorithm 1 summarizes the training protocol.

2.1 From image observations to tokens

The discrete autoencoder (E, D) learns a symbolic language of its own to represent high-dimensional images as a small number of tokens. The back and forth between frames and tokens is illustrated with green arrows in Figure 1.

More precisely, the encoder $E : \mathbb{R}^{h \times w \times 3} \to \{1, \dots, N\}^K$ converts an input image x_t into K tokens from a vocabulary of size N. Let $\mathcal{E} = \{e_i\}_{i=1}^N \in \mathbb{R}^{N \times d}$ be the corresponding embedding table of d-dimensional vectors. The input image x_t is first passed through a Convolutional Neural Network



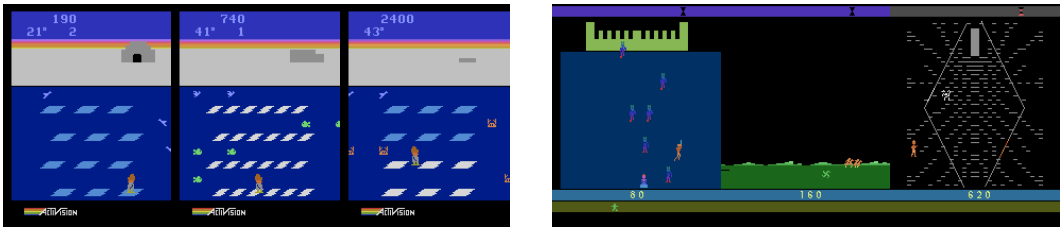


Figure 3: Three consecutive levels in the games *Frostbite* (left) and *Krull* (right). In our experiments, the world model struggles to simulate subsequent levels in *Frostbite*, but not in *Krull*. Indeed, exiting the first level in *Frostbite* requires a long and unlikely sequence of actions to first build the igloo, and then go back to it from the bottom of the screen. Such rare events prevent the world model from internalizing new aspects of the game, which will therefore not be experienced by the policy in imagination. While *Krull* features more diverse levels, the world model successfully reflects this variety, and IRIS even sets a new state of the art in this environment. This is likely due to more frequent transitions from one stage to the next in *Krull*, resulting in a sufficient coverage of each level.

(CNN) [36] producing output $y_t \in \mathbb{R}^{K \times d}$. We then obtain the output tokens $z_t = (z_t^1, \ldots, z_t^K) \in \{1, \ldots, N\}^K$ as $z_t^k = \text{argmin}_i \left\| y_t^k - e_i \right\|_2$, the index of the closest embedding vector in $\mathcal{E}$ [32, 30]. Conversely, the CNN decoder $D : \{1, \ldots, N\}^K \to \mathbb{R}^{h \times w \times 3}$ turns K tokens back into an image.

This discrete autoencoder is trained on previously collected frames, with an equally weighted combination of a L_2 reconstruction loss, a commitment loss [32, 30], and a perceptual loss [30, 37, 38]. We use a straight-through estimator [39] to enable backpropagation training.

2.2 Modeling dynamics

At a high level, the Transformer G captures the environment dynamics by modeling the language of the discrete autoencoder over time. Its central role of unfolding imagination is highlighted with the blue arrows in Figure 1.

Specifically, G operates over sequences of interleaved frame and action tokens. An input sequence $(z_0^1, \ldots, z_0^K, a_0, z_1^1, \ldots, z_1^K, a_1, \ldots, z_t^1, \ldots, z_t^K, a_t)$ is obtained from the raw sequence $(x_0, a_0, x_1, a_1, \ldots, x_t, a_t)$ by encoding the frames with E, as described in Section 2.1.

At each time step t, the Transformer models the three following distributions:

$$\text{Transition:} \quad \hat{z}_{t+1} \sim p_G\big(\hat{z}_{t+1} | \, z_{\leq t}, a_{\leq t}\big) \text{ with } \hat{z}_{t+1}^k \sim p_G\big(\hat{z}_{t+1}^k \mid z_{\leq t}, a_{\leq t}, z_{t+1}^{<k}\big) \tag{1}$$

$$\text{Reward:} \quad \hat{r}_t \sim p_G\big(\hat{r}_t \mid z_{\leq t}, a_{\leq t}\big) \tag{2}$$

$$\text{Termination:} \quad \hat{d}_t \sim p_G\big(\hat{d}_t \mid z_{\leq t}, a_{\leq t}\big) \tag{3}$$

Note that the conditioning for the k-th token also includes $z_{t+1}^{<k} := (z_{t+1}^1, \ldots, z_{t+1}^{k-1})$, the tokens that were already predicted, i.e. the autoregressive process happens at the token level.

We train G in a self-supervised manner on segments of L time steps, sampled from past experience. We use a cross-entropy loss for the transition and termination predictors, and a mean-squared error loss for the reward predictor.

2.3 Learning in imagination

Together, the discrete autoencoder (E, D) and the Transformer G form a world model, capable of imagination. The policy π, depicted with purple arrows in Figure 1, exclusively learns in this imagination MDP.

At time step t, the policy observes a reconstructed image observation $\hat{x}_t$ and samples action $a_t \sim \pi(a_t | \hat{x}_t)$. The world model then predicts the reward $\hat{r}_t$, the episode end $\hat{d}_t$, and the next observation $\hat{x}_{t+1} = D(\hat{z}_{t+1})$, with $\hat{z}_{t+1} \sim p_G(\hat{z}_{t+1} \mid z_0, a_0, \hat{z}_1, a_1, \ldots, \hat{z}_t, a_t)$. This imagination procedure is initialized with a real observation x_0 sampled from past experience, and is rolled out for H steps, the imagination horizon hyperparameter. We stop if an episode end is predicted before reaching the horizon. Figure 1 illustrates the imagination procedure.

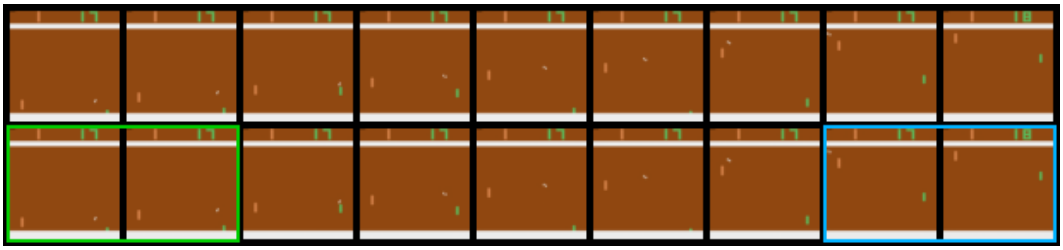


Figure 4: Pixel perfect predictions in *Pong*. The top row displays a test trajectory collected in the real environment. The bottom row depicts the reenactment of that trajectory inside the world model. More precisely, we condition the world model with the first two frames of the true sequence, in green. We then sequentially feed it the true actions and let it imagine the subsequent frames. After only 120 games of training, the world model perfectly simulates the ball's trajectory and players' movements. Notably, it also captures the game mechanic of updating the scoreboard after winning an exchange, as shown in the blue box.

As we roll out imagination for a fixed number of steps, we cannot simply use a Monte Carlo estimate for the expected return. Hence, to bootstrap the rewards that the agent would get beyond a given time step, we have a value network V that estimates $V(\hat{x}_t) \simeq \mathbb{E}_\pi\big[\sum_{\tau \geq t} \gamma^{\tau-t} \hat{r}_\tau\big]$.

Many actor-critic methods could be employed to train π and V in imagination [12, 16, 17]. For the sake of simplicity, we opt for the learning objectives and hyperparameters of DreamerV2 [3], that delivered strong performance in Atari games. We even use a simpler configuration, in that we do not include an additional dynamics backpropagation term in the actor's objective, and we do not have a target value network. Appendix B gives a detailed breakdown of the reinforcement learning objectives.

3 Experiments

Sample-efficient reinforcement learning is a growing field with multiple benchmarks in complex visual environments [40, 41]. In this work, we focus on the well established Atari 100k benchmark [16]. We present the benchmark and its baselines in Section 3.1. We describe the evaluation protocol and discuss the results in Section 3.2. Qualitative examples of the world model's capabilities are given in Section 3.3.

3.1 Benchmark and Baselines

Atari 100k consists of 26 Atari games [42], where an agent is only allowed 100k actions in each environment. This constraint is roughly equivalent to 2 hours of human gameplay. By way of comparison, unconstrained Atari agents are usually trained for 50 million steps, a 500 fold increase in experience.

Multiple baselines were benchmarked on Atari 100k. SimPLe [16] trains a policy with PPO [43] in a video generation model. CURL [34] develops off-policy agents from high-level image features obtained with contrastive learning. DrQ [35] augments input images and averages Q-value estimates over several transformations. SPR [13] enforces consistent representations of input images across augmented views and neighbouring time steps. The aforementioned baselines carry additional techniques to improve performance. Instead, IRIS does not rely on prioritized experience replay [44], epsilon-greedy scheduling, or data augmentations.

We make a distinction between methods with and without lookahead search. Indeed, algorithms such as Monte Carlo Tree Search (MCTS) [6, 7, 2] can vastly improve agent performance, but they come at a premium in computational resources and code complexity. On the contrary, IRIS and the baselines above do not require distributed infrastructures and multiple GPUs. Moreover, IRIS could be combined with MCTS, both in imagination and in the real environment. Therefore, methods involving lookahead search should not be seen as direct competitors but rather as potential extensions to learning-only methods.

Table 1: Returns on the 26 games of Atari 100k after 2 hours of real-time experience, and human-normalized aggregate metrics. Bold numbers indicate the top methods without lookahead search while underlined numbers specify the overall best methods. IRIS outperforms learning-only methods in terms of number of superhuman games, mean, interquartile mean (IQM), and optimality gap.

			Lookahead search		No lookahead search				
Game	Random	Human	MuZero	EfficientZero	SimPLe	CURL	DrQ	SPR	IRIS (ours)
Alien	227.8	7127.7	530.0	808.5	616.9	711.0	**865.2**	841.9	420.0
Amidar	5.8	1719.5	38.8	148.6	74.3	113.7	137.8	**179.7**	143.0
Assault	222.4	742.0	500.1	1263.1	527.2	500.9	579.6	565.6	**1524.4**
Asterix	210.0	8503.3	1734.0	25557.8	**1128.3**	567.2	763.6	962.5	853.6
BankHeist	14.2	753.1	192.5	351.0	34.2	65.3	232.9	**345.4**	53.1
BattleZone	2360.0	37187.5	7687.5	13871.2	4031.2	8997.8	10165.3	**14834.1**	13074.0
Boxing	0.1	12.1	15.1	52.7	7.8	0.9	9.0	35.7	**70.1**
Breakout	1.7	30.5	48.0	414.1	16.4	2.6	19.8	19.6	**83.7**
ChopperCommand	811.0	7387.8	1350.0	1117.3	979.4	783.5	844.6	946.3	**1565.0**
CrazyClimber	10780.5	35829.4	56937.0	83940.2	**62583.6**	9154.4	21539.0	36700.5	59324.2
DemonAttack	152.1	1971.0	3527.0	13003.9	208.1	646.5	1321.5	517.6	**2034.4**
Freeway	0.0	29.6	21.8	21.8	16.7	28.3	20.3	19.3	**31.1**
Frostbite	65.2	4334.7	255.0	296.3	236.9	**1226.5**	1014.2	1170.7	259.1
Gopher	257.6	2412.5	1256.0	3260.3	596.8	400.9	621.6	660.6	**2236.1**
Hero	1027.0	30826.4	3095.0	9315.9	2656.6	4987.7	4167.9	5858.6	**7037.4**
Jamesbond	29.0	302.8	87.5	517.0	100.5	331.0	349.1	366.5	**462.7**
Kangaroo	52.0	3035.0	62.5	724.1	51.2	740.2	1088.4	**3617.4**	838.2
Krull	1598.0	2665.5	4890.8	5663.3	2204.8	3049.2	4402.1	3681.6	**6616.4**
KungFuMaster	258.5	22736.3	18813.0	30944.8	14862.5	8155.6	11467.4	14783.2	**21759.8**
MsPacman	307.3	6951.6	1265.6	1281.2	**1480.0**	1064.0	1218.1	1318.4	999.1
Pong	-20.7	14.6	-6.7	20.1	12.8	-18.5	-9.1	-5.4	**14.6**
PrivateEye	24.9	69571.3	56.3	96.7	35.0	81.9	3.5	86.0	**100.0**
Qbert	163.9	13455.0	3952.0	13781.9	1288.8	727.0	**1810.7**	866.3	745.7
RoadRunner	11.5	7845.0	2500.0	17751.3	5640.6	5006.1	11211.4	**12213.1**	9614.6
Seaquest	68.4	42054.7	208.0	1100.2	**683.3**	315.2	352.3	558.1	661.3
UpNDown	533.4	11693.2	2896.9	17264.2	3350.3	2646.4	4324.5	**10859.2**	3546.2
#Superhuman (↑)	0	N/A	5	14	1	2	3	6	**10**
Mean (↑)	0.000	1.000	0.562	1.943	0.332	0.261	0.465	0.616	**1.046**
Median (↑)	0.000	1.000	0.227	1.090	0.134	0.092	0.313	**0.396**	0.289
IQM (↑)	0.000	1.000	N/A	N/A	0.130	0.113	0.280	0.337	**0.501**
Optimality Gap (↓)	1.000	0.000	N/A	N/A	0.729	0.768	0.631	0.577	**0.512**

MuZero [2] and EfficientZero [14] are the current standard for search-based methods. MuZero leverages MCTS as a policy improvement operator, by unrolling multiple hypothetical trajectories in the latent space of a world model. EfficientZero improves upon MuZero by introducing a self-supervised consistency loss, predicting returns over short horizons in one shot, and correcting off-policy trajectories with its world model.

3.2 Results

The human normalized score is the established measure of performance in Atari 100k. It is defined as $\frac{score\_agent - score\_random}{score\_human - score\_random}$, where *score_random* comes from a random policy, and *score_human* is obtained from human players [45].

Agarwal et al. [46] showed that substantial discrepancies may arise between standard point estimates and interval estimates in RL benchmarks. Besides, they discuss the limitations of mean and median scores, and call for more robust aggregate metrics.

Following their recommendations, we summarize in Figure 5 the human normalized scores with stratified bootstrapped confidence intervals for mean, median, and interquartile mean (IQM). Confidence intervals for probability of improvement and optimality gap are found in Appendix C. We also provide performance profiles in Figure 6 for finer comparisons. Table 1 displays unnormalized returns across games and aggregate metrics.

For MuZero and EfficientZero, we report the averaged results published by Ye et al. [14] (3 runs). We use results from the Atari 100k case study conducted by Agarwal et al. [46] for the other baselines (100 new runs for CURL, DrQ, SPR, and 5 existing runs for SimPLe). Finally, we evaluate IRIS by computing an average over 100 episodes collected at the end of training for each game (5 runs).

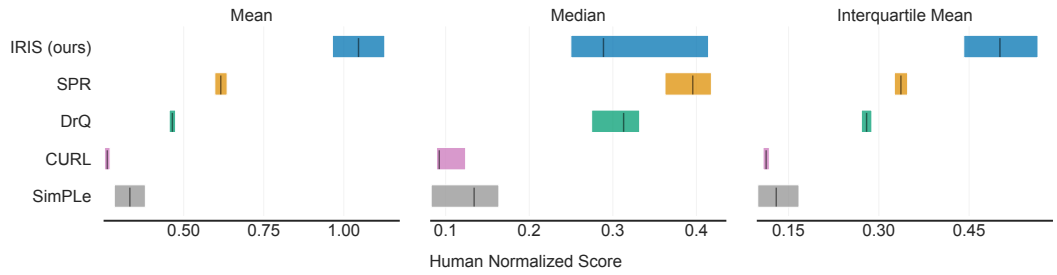


Figure 5: Mean, median, and interquartile mean human normalized scores, computed with stratified bootstrapped confidence intervals [46]. 5 runs for IRIS and SimPLe, 100 runs for SPR, CURL, and DrQ [46].

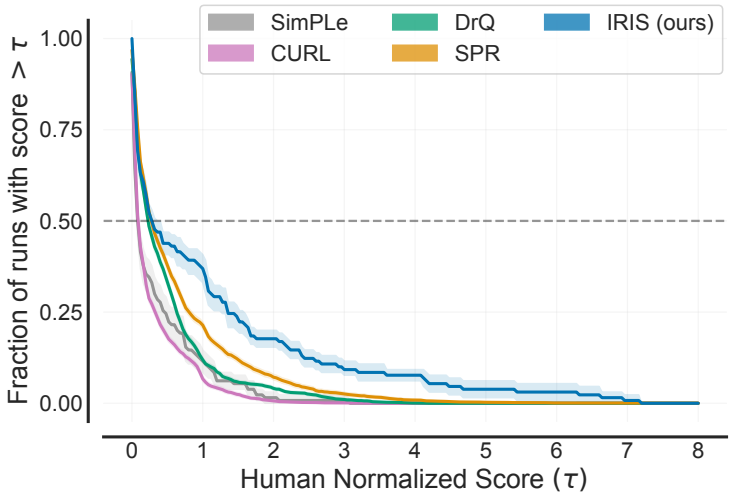


Figure 6: Performance profiles, i.e. fraction of runs above a given human normalized score [46].

With the equivalent of only two hours of gameplay, IRIS achieves a mean human normalized score of 1.046, an IQM of 0.501, and outperforms human players on 10 out of 26 games. These results constitute a new state of the art for methods without lookahead search in the Atari 100k benchmark, and IRIS even surpasses MuZero. In addition, performance profiles reveal that IRIS is on par with the strongest baselines for its bottom 50% of games, at which point it stochastically dominates [46, 47] the other methods.

IRIS is particularly strong in games that do not suffer from distributional shifts as the training progresses. Examples of such games include *Pong*, *Breakout*, and *Boxing*. On the contrary, the agent struggles when a new level or game mechanic is unlocked through an unlikely event. This sheds light on a double exploration problem. IRIS has to first discover a new aspect of the game for its world model to internalize it. Only then may the policy rediscover and exploit it. Figure 3 details this phenomenon in *Frostbite* and *Krull*, two games with multiple levels. In summary, as long as transitions between levels do not depend on low-probability events, we observe that the double exploration problem does not hinder performance.

Another kind of games difficult to simulate are mazes with moving enemies, such as *MsPacman*, *BankHeist*, and *Alien*. As discussed in Appendix E, this is due to the small number of tokens used to encode frames. When we increase that amount, the world model is able to situate enemies, albeit at the cost of increased computation.

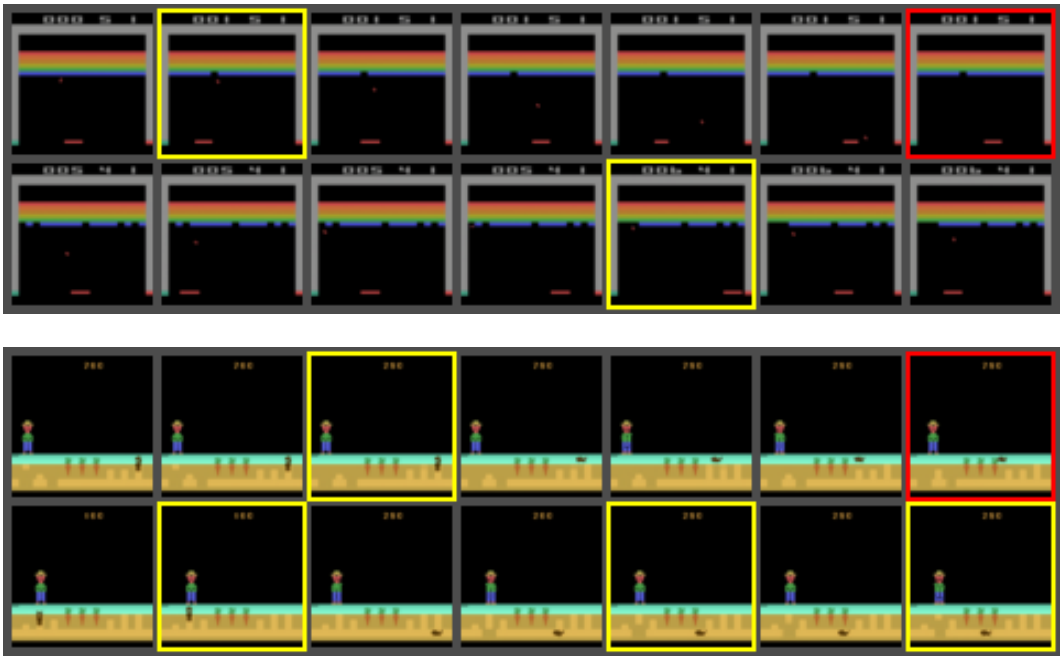


Figure 7: Imagining rewards and episode ends in *Breakout* (top) and *Gopher* (bottom). Each row depicts an imagined trajectory initialized with a single frame from the real environment. Yellow boxes indicate frames where the world model predicts a positive reward. In *Breakout*, it captures that breaking a brick yields rewards, and the brick is correctly removed from the following frames. In *Gopher*, the player has to protect the carrots from moles. The world model successfully internalizes that plugging a hole or killing a mole leads to rewards. Predicted episode terminations are highlighted with red boxes. The world model accurately reflects that missing the ball in *Breakout*, or letting a rodent reach the carrots in *Gopher*, will result in the end of an episode.

3.3 World Model Analysis

As IRIS learns behaviors entirely in its imagination, the quality of the world model is the cornerstone of our approach. For instance, it is key that the discrete autoencoder correctly reconstructs elements like a ball, a player, or an enemy. Similarly, the potential inability of the Transformer to capture important game mechanics, like reward attribution or episode termination, can severely hamper the agent's performance. Hence, no matter the amount of imagined trajectories, the agent will learn suboptimal policies if the world model is flawed.

While Section 3.2 provides a quantitative evaluation, we aim to complement the analysis with qualitative examples of the abilities of the world model. Figure 2 shows the generation of many plausible futures in the face of uncertainty. Figure 4 depicts pixel-perfect predictions in *Pong*. Finally, we illustrate in Figure 7 predictions for rewards and episode terminations, which are crucial to the reinforcement learning objective.

4 Related Work

Learning in the imagination of world models

The idea of training policies in a learnt model of the world was first investigated in tabular environments [12]. Ha and Schmidhuber [15] showed that simple visual environments could be simulated with autoencoders and recurrent networks. SimPLe [16] demonstrated that a PPO policy [43] trained in a video prediction model outperformed humans in some Atari games. Improving upon Dreamer [17], DreamerV2 [3] was the first agent learning in imagination to achieve human-level performance in the Atari 50M benchmark. Its world model combines a convolutional autoencoder with a recurrent state-space model (RSSM) [48] for latent dynamics learning. More recently, Chen et al. [49] explored a variant of DreamerV2 where a Transformer replaces the recurrent network in the RSSM.

Reinforcement Learning with Transformers

Following spectacular advances in natural language processing [50], the reinforcement learning community has recently stepped into the realm of Transformers. Parisotto et al. [51] make the observation that the standard Transformer architecture is difficult to optimize with RL objectives. The authors propose to replace residual connections by gating layers to stabilize the learning procedure. Our world model did not require such modifications, which is most likely due to its self-supervised learning objective. The Trajectory Transformer [25] and the Decision Transformer [26] represent offline trajectories as a static dataset of sequences. The Trajectory Transformer is trained to predict future returns, states and actions. At inference time, it can thus plan for the optimal action with a reward-driven beam search, yet the approach is limited to low-dimensional states. On the contrary, the Decision Transformer can handle image inputs but it cannot be easily extended as a world model. Ozair et al. [52] introduce an offline variant of MuZero [2] capable of handling stochastic environments by performing an hybrid search with a Transformer over both actions and trajectory-level discrete latent variables.

Video generation with discrete autoencoders and Transformers

VQGAN [30] and DALL-E [31] use discrete autoencoders to compress a frame into a small sequence of tokens, that a transformer can then model autoregressively. Other works extend the approach to video generation. GODIVA [53] models sequences of frames instead of a single frame for text conditional video generation. VideoGPT [54] introduces video-level discrete autoencoders, and Transformers with spatial and temporal attention patterns, for unconditional and action conditional video generation.

5 Ethics Statement

The development of autonomous agents for real-world environments raises many safety and environmental concerns. During its training period, an agent may cause serious harm to individuals and damage its surroundings. It is our belief that learning in the imagination of world models greatly reduces the risks associated with training new autonomous agents. Indeed, in this work, we propose a world model architecture capable of accurately modeling environments with very few samples. However, in a future line of research, one could go one step further and leverage existing data to eliminate the necessity of interacting with the real world.

6 Conclusion

We introduced IRIS, an agent that learns purely in the imagination of a world model composed of a discrete autoencoder and an autoregressive Transformer. IRIS sets a new state of the art in the Atari 100k benchmark for methods without lookahead search, and even outperforms MuZero. We showed that its world model acquires a deep understanding of game mechanics, resulting in pixel perfect predictions in some games. We also illustrated the generative capabilities of the world model, providing a rich gameplay experience when training in imagination. Ultimately, with minimal tuning compared to existing battle-hardened agents, IRIS opens a new path towards efficiently solving complex environments.

In the future, IRIS could be scaled up to computationally demanding and challenging tasks that would benefit from the speed of its world model. Besides, its policy currently learns from reconstructed frames, but it could probably leverage the internal representations of the world model. Another exciting avenue of research would be to combine learning in imagination with MCTS. Indeed, both approaches deliver impressive results, and their contributions to agent performance are orthogonal.

Acknowledgments and Disclosure of Funding

We would like to thank Maxim Peter, Bálint Máté, Daniele Paliotta, Atul Sinha, and Alexandre Dupuis for insightful discussions and comments. Vincent Micheli was supported by the Swiss National Science Foundation under grant number FNS-187494.

References

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

[2] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, L. Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

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

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

[5] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

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

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

[8] Martin Schmid, Matej Moravcik, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, Zach Holland, et al. Player of games. *arXiv preprint arXiv:2112.03178*, 2021.

[9] Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. *Advances in Neural Information Processing Systems*, 33:17057–17069, 2020.

[10] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[11] Roman V Yampolskiy. *Artificial Intelligence Safety and Security*. Chapman & Hall/CRC, 2018.

[12] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[13] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021.

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

[15] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

[16] Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020.

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

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

[20] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[21] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[22] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[24] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[25] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34, 2021.

[26] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.

[27] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[28] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE, 2012.

[29] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, 2018.

[30] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021.

[31] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

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

[33] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

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

[35] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.

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

[37] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[38] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

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

[40] Danijar Hafner. Benchmarking the spectrum of agent capabilities. In *International Conference on Learning Representations*, 2022.

[41] Anssi Kanervisto, Stephanie Milani, Karolis Ramanauskas, Nicholay Topin, Zichuan Lin, Junyou Li, Jianing Shi, Deheng Ye, Qiang Fu, Wei Yang, Weijun Hong, Zhongyue Huang, Haicheng Chen, Guangjun Zeng, Yue Lin, Vincent Micheli, Eloi Alonso, François Fleuret, Alexander Nikulin, Yury Belousov, Oleg Svidchenko, and Aleksei Shpilman. Minerl diamond 2021 competition: Overview, results, and lessons learned. *arXiv preprint arXiv:2202.10583*, 2022.

[42] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

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

[44] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

[45] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[46] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[47] Rotem Dror, Segev Shlomov, and Roi Reichart. Deep dominance-how to properly compare deep neural models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2785, 2019.

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

[49] Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

[50] Christopher Manning and Anna Goldie. Cs224n natural language processing with deep learning, 2022.

[51] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, pages 7487–7498. PMLR, 2020.

[52] Sherjil Ozair, Yazhe Li, Ali Razavi, Ioannis Antonoglou, Aaron Van Den Oord, and Oriol Vinyals. Vector quantized models for planning. In *International Conference on Machine Learning*, pages 8302–8313. PMLR, 2021.

[53] Chenfei Wu, Lun Huang, Qianxi Zhang, Binyang Li, Lei Ji, Fan Yang, Guillermo Sapiro, and Nan Duan. Godiva: Generating open-domain videos from natural descriptions. *arXiv preprint arXiv:2104.14806*, 2021.

[54] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

[55] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[56] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997.

[57] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.

[58] Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2019.

A Models and Hyperparameters

A.1 Discrete autoencoder

Our discrete autoencoder is based on the implementation of VQGAN [30]. We removed the discriminator, essentially turning the VQGAN into a vanilla VQVAE [32] with an additional perceptual loss [37, 38].

Table 2: Encoder / Decoder hyperparameters. We list the hyperparameters for the encoder, the same ones apply for the decoder.

Hyperparameter	Value
Frame dimensions (h, w)	64×64
Layers	4
Residual blocks per layer	2
Channels in convolutions	64
Self-attention layers at resolution	8 / 16

Table 3: Embedding table hyperparameters.

Hyperparameter	Value
Vocabulary size (N)	512
Tokens per frame (K)	16
Token embedding dimension (d)	512

Note that during experience collection in the real environment, frames still go through the autoencoder to keep the input distribution of the policy unchanged. See Algorithm 1 for details.

A.2 Transformer

The Transformer takes as input a sequence of $L(K+1)$ tokens and embeds it into a $L(K+1) \times D$ tensor using an $A \times D$ embedding table for actions, and a $N \times D$ embedding table for frames tokens. This tensor is forwarded through M Transformer blocks. We use GPT2-like blocks [20], i.e. each block consists of a self-attention module with layer normalization of the input, wrapped with a residual connection, followed by a per-position multi-layer perceptron with layer normalization of the input, wrapped with another residual connection.

Table 4: Transformer hyperparameters

Hyperparameter	Value
Timesteps (L)	20
Embedding dimension (D)	256
Layers (M)	10
Attention heads	4
Weight decay	0.01
Embedding dropout	0.1
Attention dropout	0.1
Residual dropout	0.1

A.3 Actor-Critic

The actor-critic takes as input a $64 \times 64 \times 3$ frame, and forwards it through a convolutional block followed by an LSTM cell [55, 56, 57]. The convolutional block consists of the same layer repeated four times: a 3x3 convolution with stride 1 and padding 1, a ReLU activation, and 2x2 max-pooling with stride 2. The dimension of the LSTM hidden state is 512. Before starting the imagination procedure from a given frame, we burn-in [58] the 20 previous frames to initialize the hidden state.

Table 5: Training loop & Shared hyperparameters

Hyperparameter	Value
Epochs	600
# Collection epochs	500
Environment steps per epoch	200
Collection epsilon-greedy	0.01
Eval sampling temperature	0.5
Start autoencoder after epochs	5
Start transformer after epochs	25
Start actor-critic after epochs	50
Autoencoder batch size	256
Transformer batch size	64
Actor-critic batch size	64
Training steps per epoch	1000
Learning rate	1e-4
Optimizer	Adam
Adam β_1	0.9
Adam β_2	0.999
Max gradient norm	10.0

B Actor-critic learning objectives

We follow Dreamer [17, 3] in using the generic λ-return, that balances bias and variance, as the regression target for the value network. Given an imagined trajectory $(\hat{x}_0, a_0, \hat{r}_0, \hat{d}_0, \ldots, \hat{x}_{H-1}, a_{H-1}, \hat{r}_{H-1}, \hat{d}_{H-1}, \hat{x}_H)$, the λ-return can be defined recursively as follows:

$$\Lambda_t = \begin{cases} \hat{r}_t + \gamma(1-\hat{d}_t)\Big[(1-\lambda)V(\hat{x}_{t+1}) + \lambda\Lambda_{t+1}\Big] & \text{if} \quad t < H \\ V(\hat{x}_H) & \text{if} \quad t = H \end{cases} \tag{4}$$

The value network V is trained to minimize $\mathcal{L}_V$, the expected squared difference with λ-returns over imagined trajectories.

$$\mathcal{L}_V = \mathbb{E}_\pi\Big[\sum_{t=0}^{H-1}\big(V(\hat{x}_t) - \mathrm{sg}(\Lambda_t)\big)^2\Big] \tag{5}$$

Here, $\mathrm{sg}(\cdot)$ denotes the gradient stopping operation, meaning that the target is a constant in the gradient-based optimization, as classically established in the literature [1, 33, 17].

As large amounts of trajectories are generated in the imagination MDP, we can use a straightforward reinforcement learning objective for the policy, such as REINFORCE [12]. To reduce the variance of REINFORCE gradients, we use the value $V(\hat{x}_t)$ as a baseline [12]. We also add a weighted entropy maximization objective to maintain a sufficient exploration. The actor is trained to minimize the following REINFORCE objective over imagined trajectories:

$$\mathcal{L}_\pi = -\mathbb{E}_\pi\Big[\sum_{t=0}^{H-1}\log(\pi(a_t|\hat{x}_t))\,\mathrm{sg}(\Lambda_t - V(\hat{x}_t)) + \eta\,\mathcal{H}(\pi(a_t|\hat{x}_t))\Big] \tag{6}$$

Note that, unlike Dreamer, we do not include an additional dynamics backpropagation term in the actor's objective.

Table 6: RL training hyperparameters

Hyperparameter	Value
Imagination horizon (H)	20
γ	0.995
λ	0.95
η	0.001

C Additional robust aggregate metrics

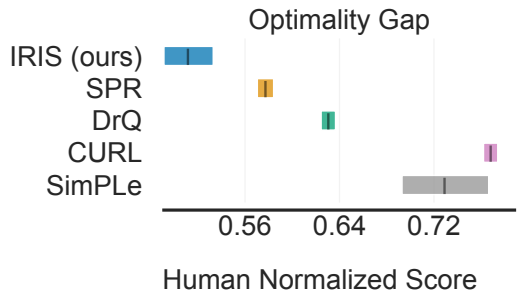


Figure 8: Optimality gap. The amount by which the algorithm fails to reach a human-level score [46], lower is better.

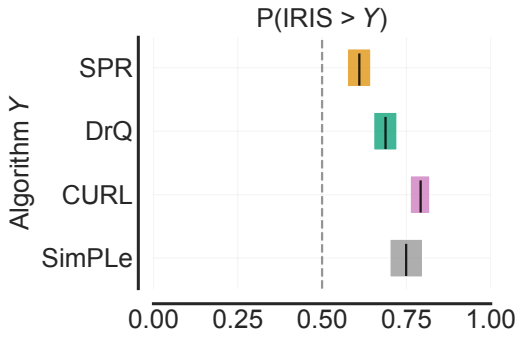


Figure 9: Probability of improvement. This metric shows how likely it is for IRIS to outperform Algorithm Y on a randomly selected game [46].

D IRIS Algorithm

Algorithm 1: IRIS

Procedure `training_loop()`:
- **for** *epochs* **do**
  - `collect_experience(`*steps_collect*`)`
  - **for** *steps_world_model* **do**
    - `update_world_model()`
  - **for** *steps_behavior* **do**
    - `update_behavior()`

Procedure `collect_experience(`n`)`:
- $x_0 \leftarrow$ `env.reset()`
- **for** $t = 0$ **to** $n - 1$ **do**
  - $\hat{x}_t \leftarrow D(E(x_t))$ `// forward frame through discrete autoencoder`
  - Sample $a_t \sim \pi(a_t|\hat{x}_t)$
  - $x_{t+1}, r_t, d_t \leftarrow$ `env.step(`a_t`)`
  - **if** $d_t = 1$ **then**
    - $x_{t+1} \leftarrow$ `env.reset()`
- $\mathcal{D} \leftarrow \mathcal{D} \cup \{x_t, a_t, r_t, d_t\}_{t=0}^{n-1}$

Procedure `update_world_model()`:
- Sample $\{x_t, a_t, r_t, d_t\}_{t=\tau}^{\tau+L-1} \sim \mathcal{D}$
- Compute $z_t \coloneqq E(x_t)$ and $\hat{x}_t \coloneqq D(z_t)$ for $t = \tau, \ldots, \tau + L - 1$
- Update E and D
- Compute $p_G(\hat{z}_{t+1}, \hat{r}_t, \hat{d}_t \mid z_\tau, a_\tau, \ldots, z_t, a_t)$ for $t = \tau, \ldots, \tau + L - 1$
- Update G

Procedure `update_behavior()`:
- Sample $x_0 \sim \mathcal{D}$
- $z_0 \leftarrow E(x_0)$
- $\hat{x}_0 \leftarrow D(z_0)$
- **for** $t = 0$ **to** $H - 1$ **do**
  - Sample $a_t \sim \pi(a_t|\hat{x}_t)$
  - Sample $\hat{z}_{t+1}, \hat{r}_t, \hat{d}_t \sim p_G(\hat{z}_{t+1}, \hat{r}_t, \hat{d}_t \mid z_0, a_0, \ldots, \hat{z}_t, a_t)$
  - $\hat{x}_{t+1} \leftarrow D(\hat{z}_{t+1})$
- Compute $V(\hat{x}_t)$ for $t = 0, \ldots, H$
- Update π and V

E Autoenconding frames with varying amounts of tokens

The length of the input sequence of G is determined by the number of tokens K used to encode a single frame and the number of timesteps L in memory. Increasing the number of tokens per frame results in better reconstructions, although it requires more compute and memory.

This tradeoff is particularly important in Atari games where enemies and players are moving in mazes with rewards to collect. Due to the high number of possible configurations, the discrete autoencoder struggles to properly encode frames with only $K = 16$ tokens. Indeed, sometimes the player, its enemies, or rewards are not correctly reconstructed, which severely hinders agent performance.

In Figure 10, we show that when increasing the number of tokens per frame to 64, the discrete autoencoder is perfectly capable of dealing with detailed environments such as *Alien*. However, this increases the sequence length of G from 340 to 1300. Therefore, with more computational resources, IRIS would most likely improve in these settings.

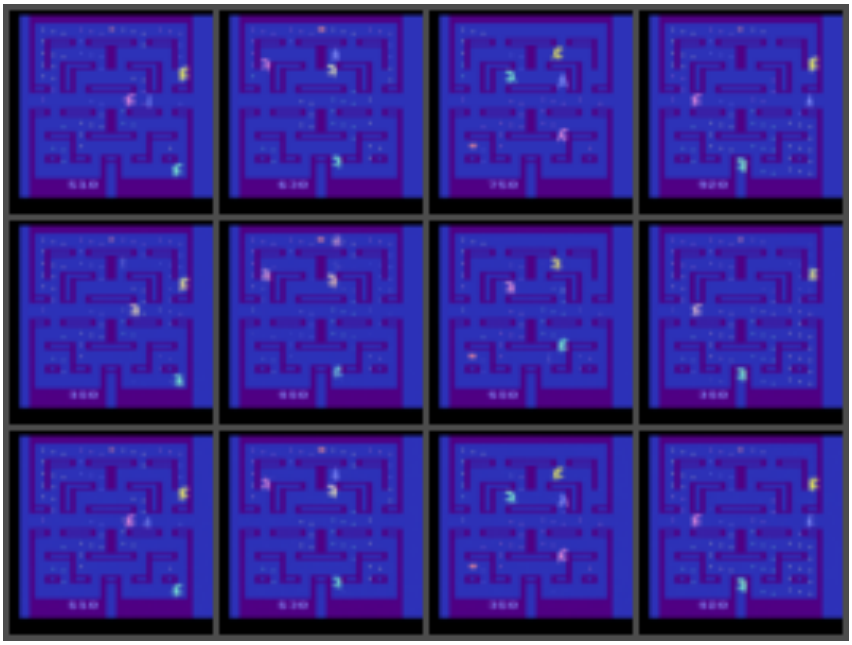


Figure 10: Tradeoff between the number of tokens per frame and reconstructions quality in *Alien*. Each column displays a 64×64 frame from the real environment (top), its reconstruction with a discrete encoding of 16 tokens (center), and its reconstruction with a discrete encoding of 64 tokens (bottom). In *Alien*, the player is the dark blue character, and the enemies are the large colored sprites. With 16 tokens per frame, the autoencoder often erases the player, switches colors, and misplaces rewards. When increasing the amount of tokens, it properly reconstructs the frame.

F Computational resources

For each Atari environment, we repeatedly trained IRIS with 5 different random seeds. We ran our experiments with 8 Nvidia A100 40GB GPUs. With two Atari environments running on the same GPU, training takes around 7 days, resulting in an average of 3.5 days per environment.

SimPLe [16], the only baseline that involves learning in imagination, trains for 3 weeks with a P100 GPU on a single environment. As for SPR [13], the strongest baseline without lookahead search, it trains notably fast in 4.6 hours with a P100 GPU.

Regarding baselines with lookahead search, MuZero [2] originally used 40 TPUs for 12 hours to train in a single Atari environment. Ye et al. [14] train both EfficientZero and their reimplementation of MuZero in 7 hours with 4 RTX 3090 GPUs. EfficientZero's implementation relies on a distributed infrastructure with CPU and GPU threads running in parallel, and a C++/Cython implementation of MCTS. By contrast, IRIS and the baselines without lookahead search rely on straightforward single GPU / single CPU implementations.

G Exploration in Freeway

The reward function in Freeway is sparse since the agent is only rewarded when it completely crosses the road. In addition, bumping into cars will drag it down, preventing it from smoothly ascending the highway. This poses an exploration problem for newly initialized agents because a random policy will almost surely never obtain a non-zero reward with a 100k frames budget.

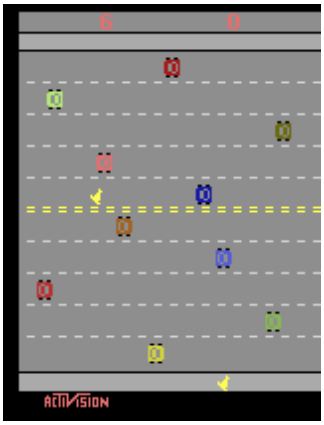


Figure 11: A game of *Freeway*. Cars will bump the player down, making it very unlikely to cross the road and be rewarded for random policies.

The solution to this problem is actually straightforward and simply requires stretches of time when the UP action is oversampled. Most Atari 100k baselines fix the issue with epsilon-greedy schedules and argmax action selection, where at some point the network configuration will be such that the UP action is heavily favored. In this work, we opted for the simpler strategy of having a fixed epsilon-greedy parameter and sampling from the policy. However, we lowered the sampling temperature from 1 to 0.01 for Freeway, in order to avoid random walks that would not be conducive to learning in the early stages of training. As a consequence, once it received its first few rewards through exploration, IRIS was able to internalize the sparse reward function in its world model.