Diffusion for World Modeling: Visual Details Matter in Atari[†]

Eloi Alonso[*] University of Geneva
Adam Jelley[*] University of Edinburgh
Vincent Micheli University of Geneva
Anssi Kanervisto Microsoft Research

Amos Storkey University of Edinburgh
Tim Pearce[‡] Microsoft Research
François Fleuret[‡] University of Geneva

Abstract

World models constitute a promising approach for training reinforcement learning agents in a safe and sample-efficient manner. Recent world models predominantly operate on sequences of discrete latent variables to model environment dynamics. However, this compression into a compact discrete representation may ignore visual details that are important for reinforcement learning. Concurrently, diffusion models have become a dominant approach for image generation, challenging well-established methods modeling discrete latents. Motivated by this paradigm shift, we introduce DIAMOND (DIffusion As a Model Of eNvironment Dreams), a reinforcement learning agent trained in a diffusion world model. We analyze the key design choices that are required to make diffusion suitable for world modeling, and demonstrate how improved visual details can lead to improved agent performance. DIAMOND achieves a mean human normalized score of 1.46 on the competitive Atari 100k benchmark; a new best for agents trained entirely within a world model. We further demonstrate that DIAMOND's diffusion world model can stand alone as an interactive neural game engine by training on static *Counter-Strike: Global Offensive* gameplay. To foster future research on diffusion for world modeling, we release our code, agents, videos and playable world models at `https://diamond-wm.github.io`.

1 Introduction

Generative models of environments, or "world models" (Ha and Schmidhuber, 2018), are becoming increasingly important as a component for generalist agents to plan and reason about their environment (LeCun, 2022). Reinforcement Learning (RL) has demonstrated a wide variety of successes in recent years (Silver et al., 2016; Degrave et al., 2022; Ouyang et al., 2022), but is well-known to be sample inefficient, which limits real-world applications. World models have shown promise for training reinforcement learning agents across diverse environments (Hafner et al., 2023; Schrittwieser et al., 2020), with greatly improved sample-efficiency (Ye et al., 2021), which can enable learning from experience in the real world (Wu et al., 2023).

Recent world modeling methods (Hafner et al., 2021; Micheli et al., 2023; Robine et al., 2023; Hafner et al., 2023; Zhang et al., 2023) often model environment dynamics as a sequence of discrete latent variables. Discretization of the latent space helps to avoid compounding error over multi-step time horizons. However, this encoding may lose information, resulting in a loss of generality and reconstruction quality. This may be problematic for more real-world scenarios where the information

[†]To prevent confusion, this is the final version of (Alonso et al., 2023) and is not related to (Ding et al., 2024).
[*]Equal contribution. [‡]Equal supervision. Contact: `eloi.alonso@unige.ch` and `adam.jelley@ed.ac.uk`

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

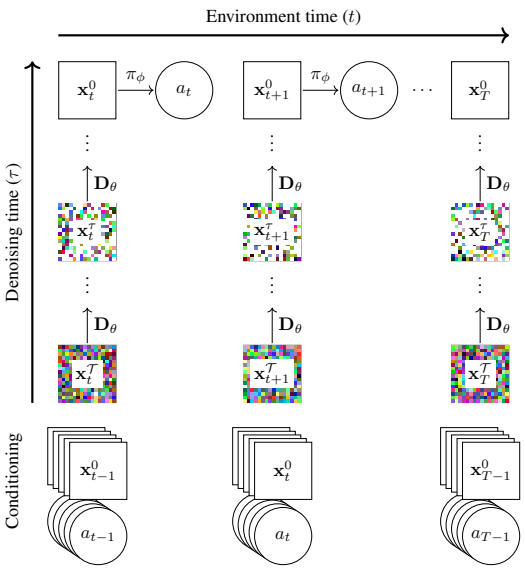


Figure 1: Unrolling imagination of DIAMOND over time. The top row depicts a policy π_ϕ taking a sequence of actions in the imagination of our learned diffusion world model $\mathbf{D}_\theta$. The environment time t flows along the horizontal axis, while the vertical axis represents the denoising time τ flowing backward from $\mathcal{T}$ to 0. Concretely, given (clean) past observations $\mathbf{x}^0_{<t}$, actions $a_{<t}$, and starting from an initial noisy sample $\mathbf{x}^\mathcal{T}_t$, we simulate a reverse noising process $\{\mathbf{x}^\tau_t\}^{\tau=0}_{\tau=\mathcal{T}}$ by repeatedly calling $\mathbf{D}_\theta$, and obtain the (clean) next observation $\mathbf{x}^0_t$. The imagination procedure is autoregressive in that the predicted observation $\mathbf{x}^0_t$ and the action a_t taken by the policy become part of the conditioning for the next time step. Animated visualizations of this procedure can be found at `https://diamond-wm.github.io`.

required for the task is less well-defined, such as training autonomous vehicles (Hu et al., 2023). In this case, small details in the visual input, such as a traffic light or a pedestrian in the distance, may change the policy of an agent. Increasing the number of discrete latents can mitigate this lossy compression, but comes with an increased computational cost (Micheli et al., 2023).

In the meantime, diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) have become a dominant paradigm for high-resolution image generation (Rombach et al., 2022; Podell et al., 2023). This class of methods, in which the model learns to reverse a noising process, challenges well-established approaches modeling discrete tokens (Esser et al., 2021; Ramesh et al., 2021; Chang et al., 2023), and thereby offers a promising alternative to alleviate the need for discretization in world modeling. Additionally, diffusion models are known to be easily conditionable and to flexibly model complex multi-modal distributions without mode collapse. These properties are instrumental to world modeling, since adherence to conditioning should allow a world model to reflect an agent's actions more closely, resulting in more reliable credit assignment, and modeling multi-modal distributions should provide greater diversity of training scenarios for an agent.

Motivated by these characteristics, we propose DIAMOND (DIffusion As a Model Of eNvironment Dreams), a reinforcement learning agent trained in a diffusion world model. Careful design choices are necessary to ensure our diffusion world model is efficient and stable over long-time horizons, and we provide a qualitative analysis to illustrate their importance. DIAMOND achieves a mean human normalized score of 1.46 on the well-established Atari 100k benchmark; a new state of the art for agents trained entirely within a world model. Additionally, operating in image space has the benefit of enabling our diffusion world model to be a drop-in substitute for the environment, which provides greater insight into world model and agent behaviors. In particular, we find the improved performance in some games follows from better modeling of critical visual details. To further demonstrate the effectiveness of our world model in isolation, we train DIAMOND's diffusion world model on 87 hours of static *Counter-Strike: Global Offensive* (CSGO) gameplay (Pearce and Zhu, 2022) to produce an interactive neural game engine for the popular in-game map, *Dust II*. We release our code, agents and playable world models at `https://diamond-wm.github.io`.

2 Preliminaries

2.1 Reinforcement learning and world models

We model the environment as a standard Partially Observable Markov Decision Process (POMDP) (Sutton and Barto, 2018), $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, \gamma)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of discrete actions, and $\mathcal{O}$ is a set of image observations. The transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ describes the environment dynamics $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$, and the reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ maps transitions to scalar rewards. Agents cannot directly access states s_t and only see the environment through image observations $x_t \in \mathcal{O}$, emitted according to observation probabilities $p(\mathbf{x}_t \mid \mathbf{s}_t)$,

described by the observation function $O : \mathcal{S} \times \mathcal{O} \to [0, 1]$. The goal is to obtain a policy π that maps observations to actions in order to maximize the expected discounted return $\mathbb{E}_\pi[\sum_{t \geq 0} \gamma^t r_t]$, where $\gamma \in [0, 1]$ is a discount factor. World models (Ha and Schmidhuber, 2018) are generative models of environments, i.e. models of $p(s_{t+1}, r_t \mid s_t, a_t)$. These models can be used as simulated environments to train RL agents (Sutton, 1991) in a sample-efficient manner (Wu et al., 2023). In this paradigm, the training procedure typically consists of cycling through the three following steps: collect data with the RL agent in the real environment; train the world model on all the collected data; train the RL agent in the world model environment (commonly referred to as "in imagination").

2.2 Score-based diffusion models

Diffusion models (Sohl-Dickstein et al., 2015) are a class of generative models inspired by non-equilibrium thermodynamics that generate samples by reversing a noising process.

We consider a diffusion process $\{\mathbf{x}^\tau\}_{\tau \in [0, \mathcal{T}]}$ indexed by a continuous time variable $\tau \in [0, \mathcal{T}]$, with corresponding marginals $\{p^\tau\}_{\tau \in [0, \mathcal{T}]}$, and boundary conditions $p^0 = p^{data}$ and $p^{\mathcal{T}} = p^{prior}$, where p^{prior} is a tractable unstructured prior distribution, such as a Gaussian. Note that we use τ and superscript for the diffusion process time, in order to keep t and subscript for the environment time.

This diffusion process can be described as the solution to a standard stochastic differential equation (SDE) (Song et al., 2020),

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, \tau)d\tau + g(\tau)d\mathbf{w}, \tag{1}$$

where $\mathbf{w}$ is the Wiener process (Brownian motion), $\mathbf{f}$ a vector-valued function acting as a drift coefficient, and g a scalar-valued function known as the diffusion coefficient of the process.

To obtain a generative model, which maps from noise to data, we must reverse this process. Remarkably, Anderson (1982) shows that the reverse process is also a diffusion process, running backwards in time, and described by the following SDE,

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, \tau) - g(\tau)^2 \nabla_{\mathbf{x}} \log p^\tau(\mathbf{x})]d\tau + g(\tau)d\bar{\mathbf{w}}, \tag{2}$$

where $\bar{\mathbf{w}}$ is the reverse-time Wiener process, and $\nabla_{\mathbf{x}} \log p^\tau(\mathbf{x})$ is the (Stein) score function, the gradient of the log-marginals with respect to the support. Therefore, to reverse the forward noising process, we are left to define the functions f and g (in Section 3.1), and to estimate the unknown score functions $\nabla_{\mathbf{x}} \log p^\tau(\mathbf{x})$, associated with marginals $\{p^\tau\}_{\tau \in [0, \mathcal{T}]}$ along the process. In practice, it is possible to use a single time-dependent score model $\mathbf{S}_\theta(\mathbf{x}, \tau)$ to estimate these score functions (Song et al., 2020).

At any point in time, estimating the score function is not trivial since we do not have access to the true score function. Fortunately, Hyvärinen (2005) introduces the *score matching* objective, which surprisingly enables training a score model from data samples without the knowledge of the underlying score function. To access samples from the marginal p^τ, we need to simulate the forward process from time 0 to time τ, as we only have clean data samples. This is costly in general, but if f is affine, we can analytically reach any time τ in the forward process in a single step, by applying a Gaussian perturbation kernel $p^{0\tau}$ to clean data samples (Song et al., 2020). Since the kernel is differentiable, score matching simplifies to a *denoising score matching* objective (Vincent, 2011),

$$\mathcal{L}(\theta) = \mathbb{E}\left[\|\mathbf{S}_\theta(\mathbf{x}^\tau, \tau) - \nabla_{\mathbf{x}^\tau} \log p^{0\tau}(\mathbf{x}^\tau \mid \mathbf{x}^0)\|^2\right], \tag{3}$$

where the expectation is over diffusion time τ, noised sample $\mathbf{x}^\tau \sim p^{0\tau}(\mathbf{x}^\tau \mid \mathbf{x}^0)$, obtained by applying the τ-level perturbation kernel to a clean sample $\mathbf{x}^0 \sim p^{data}(\mathbf{x}^0)$. Importantly, as the kernel $p^{0\tau}$ is a known Gaussian distribution, this objective becomes a simple L_2 reconstruction loss,

$$\mathcal{L}(\theta) = \mathbb{E}\left[\|\mathbf{D}_\theta(\mathbf{x}^\tau, \tau) - \mathbf{x}^0\|^2\right], \tag{4}$$

with reparameterization $\mathbf{D}_\theta(\mathbf{x}^\tau, \tau) = \mathbf{S}_\theta(\mathbf{x}^\tau, \tau)\sigma^2(\tau) + \mathbf{x}^\tau$, where $\sigma(\tau)$ is the variance of the τ-level perturbation kernel.

2.3 Diffusion for world modeling

The score-based diffusion model described in Section 2.2 provides an unconditional generative model of p_{data}. To serve as a world model, we need a conditional generative model of the environment dynamics, $p(\mathbf{x}_{t+1} \mid \mathbf{x}_{\leq t}, a_{\leq t})$, where we consider the general case of a POMDP, in which the Markovian state s_t is unknown and can be approximated from past observations and actions. We can condition a diffusion model on this history, to estimate and generate the next observation directly, as shown in Figure 1. This modifies Equation 4 as follows,

$$\mathcal{L}(\theta) = \mathbb{E}\left[\|\mathbf{D}_\theta(\mathbf{x}^\tau_{t+1}, \tau, \mathbf{x}^0_{\leq t}, a_{\leq t}) - \mathbf{x}^0_{t+1}\|^2\right]. \tag{5}$$

During training, we sample a trajectory segment $\mathbf{x}^0_{\leq t}, a_{\leq t}, \mathbf{x}^0_{t+1}$ from the agent's replay dataset, and we obtain the noised next observation $\mathbf{x}^\tau_{t+1} \sim p^{0\tau}(\mathbf{x}^\tau_{t+1} \mid \mathbf{x}^0_{t+1})$ by applying the τ-level perturbation kernel. In summary, this diffusion process for world modeling resembles the standard diffusion process described in Section 2.2, with a score model conditioned on past observations and actions.

To sample the next observation, we iteratively solve the reverse SDE in Equation 2, as illustrated in Figure 1. While we can in principle resort to any ODE or SDE solver, there is an inherent trade-off between sampling quality and Number of Function Evaluations (NFE), that directly determines the inference cost of the diffusion world model (see Appendix A for more details).

3 Method

3.1 Practical choice of diffusion paradigm

Building on the background provided in Section 2, we now introduce DIAMOND as a practical realization of a diffusion-based world model. In particular, we now define the drift and diffusion coefficients $\mathbf{f}$ and g introduced in Section 2.2, corresponding to a particular choice of diffusion paradigm. While DDPM (Ho et al., 2020) is an example of one such choice (as described in Appendix B) and would historically be the natural candidate, we instead build upon the EDM formulation proposed in Karras et al. (2022). The practical implications of this choice are discussed in Section 5.1. In what follows, we describe how we adapt EDM to build our diffusion-based world model.

We consider the perturbation kernel $p^{0\tau}(\mathbf{x}^\tau_{t+1} \mid \mathbf{x}^0_{t+1}) = \mathcal{N}(\mathbf{x}^\tau_{t+1}; \mathbf{x}^0_{t+1}, \sigma^2(\tau)\mathbf{I})$, where $\sigma(\tau)$ is a real-valued function of diffusion time called the noise schedule. This corresponds to setting the drift and diffusion coefficients to $\mathbf{f}(\mathbf{x}, \tau) = \mathbf{0}$ (affine) and $g(\tau) = \sqrt{2\dot{\sigma}(\tau)\sigma(\tau)}$.

We use the network preconditioning introduced by Karras et al. (2022) and so parameterize $\mathbf{D}_\theta$ in Equation 5 as the weighted sum of the noised observation and the prediction of a neural network $\mathbf{F}_\theta$,

$$\mathbf{D}_\theta(\mathbf{x}^\tau_{t+1}, y^\tau_t) = c^\tau_{\text{skip}}\, \mathbf{x}^\tau_{t+1} + c^\tau_{\text{out}}\, \mathbf{F}_\theta\big(c^\tau_{\text{in}}\, \mathbf{x}^\tau_{t+1}, y^\tau_t\big), \tag{6}$$

where for brevity we define $y^\tau_t \coloneqq (c^\tau_{\text{noise}}, \mathbf{x}^0_{\leq t}, a_{\leq t})$ to include all conditioning variables.

The preconditioners c^τ_{in} and c^τ_{out} are selected to keep the network's input and output at unit variance for any noise level $\sigma(\tau)$, c^τ_{noise} is an empirical transformation of the noise level, and c^τ_{skip} is given in terms of $\sigma(\tau)$ and the standard deviation of the data distribution σ_{data}, as $c^\tau_{skip} = \sigma^2_{data}/(\sigma^2_{data} + \sigma^2(\tau))$. These preconditioners are fully described in Appendix C.

Combining Equations 5 and 6 provides insight into the training objective of $\mathbf{F}_\theta$,

$$\mathcal{L}(\theta) = \mathbb{E}\Big[\big\| \underbrace{\mathbf{F}_\theta\big(c^\tau_{\text{in}}\mathbf{x}^\tau_{t+1}, y^\tau_t\big)}_{\text{Network prediction}} - \underbrace{\frac{1}{c^\tau_{\text{out}}}\big(\mathbf{x}^0_{t+1} - c^\tau_{\text{skip}}\mathbf{x}^\tau_{t+1}\big)}_{\text{Network training target}} \big\|^2\Big]. \tag{7}$$

The network training target adaptively mixes signal and noise depending on the degradation level $\sigma(\tau)$. When $\sigma(\tau) \gg \sigma_{\text{data}}$, we have $c^\tau_{\text{skip}} \to 0$, and the training target for $\mathbf{F}_\theta$ is dominated by the clean signal $\mathbf{x}^0_{t+1}$. Conversely, when the noise level is low, $\sigma(\tau) \to 0$, we have $c^\tau_{\text{skip}} \to 1$, and the target becomes the difference between the clean and the perturbed signal, i.e. the added Gaussian noise. Intuitively, this prevents the training objective to become trivial in the low-noise regime. In practice, this objective is high variance at the extremes of the noise schedule, so Karras et al. (2022) sample the noise level $\sigma(\tau)$ from an empirically chosen log-normal distribution in order to concentrate the training around medium-noise regions, as described in Appendix C.

We use a standard U-Net 2D for the vector field $\mathbf{F}_\theta$ (Ronneberger et al., 2015), and we keep a buffer of L past observations and actions that we use to condition the model. We concatenate these past observations to the next noisy observation channel-wise, and we input actions through adaptive group normalization layers (Zheng et al., 2020) in the residual blocks (He et al., 2015) of the U-Net.

As discussed in Section 2.3 and Appendix A, there are many possible sampling methods to generate the next observation from the trained diffusion model. While our codebase supports a variety of sampling schemes, we found Euler's method to be effective without incurring the cost of additional NFE required by higher order samplers, or the unnecessary complexity of stochastic sampling.

3.2 Reinforcement learning in imagination

Given the diffusion model from Section 3.1, we now complete our world model with a reward and termination model, required for training an RL agent in imagination. Since estimating the reward and termination are scalar prediction problems, we use a separate model R_ψ consisting of standard CNN (LeCun et al., 1989; He et al., 2015) and LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) layers to handle partial observability. The RL agent involves an actor-critic network parameterized by a shared CNN-LSTM with policy and value heads. The policy π_ϕ is trained with REINFORCE with a value baseline, and we use a Bellman error with λ-returns to train the value network V_ϕ, similar to Micheli et al. (2023). We train the agent entirely in imagination as described in Section 2.1. The agent only interacts with the real environment for data collection. After each collection stage, the current world model is updated by training on all data collected so far. Then, the agent is trained with RL in the updated world model environment, and these steps are repeated. This procedure is detailed in Algorithm 1, and is similar to Kaiser et al. (2019); Hafner et al. (2020); Micheli et al. (2023). We provide architecture details, hyperparameters, and RL objectives in Appendices D, E, F, respectively.

4 Experiments

4.1 Atari 100k benchmark

For comprehensive evaluation of DIAMOND we use the established Atari 100k benchmark (Kaiser et al., 2019), consisting of 26 games that test a wide range of agent capabilities. For each game, an agent is only allowed to take 100k actions in the environment, which is roughly equivalent to 2 hours of human gameplay, to learn to play the game before evaluation. As a reference, unconstrained Atari agents are usually trained for 50 million steps, a 500 fold increase in experience. We trained DIAMOND from scratch for 5 random seeds on each game. Each run utilized around 12GB of VRAM and took approximately 2.9 days on a single Nvidia RTX 4090 (1.03 GPU years in total).

We compare with other recent methods training an agent entirely within a world model in Table 1, including STORM (Zhang et al., 2023), DreamerV3 (Hafner et al., 2023), IRIS (Micheli et al., 2023), TWM (Robine et al., 2023), and SimPle (Kaiser et al., 2019). A broader comparison to model-free and search-based methods, including BBF (Schwarzer et al., 2023) and EfficientZero (Ye et al., 2021), the current best performing methods on this benchmark, is provided in Appendix J. BBF and EfficientZero use techniques that are orthogonal and not directly comparable to our approach, such as using periodic network resets in combination with hyperparameter scheduling for BBF, and computationally expensive lookahead Monte-Carlo tree search for EfficientZero. Combining these additional components with our world model would be an interesting direction for future work.

4.2 Results on the Atari 100k benchmark

Table 1 provides scores for all games, and the mean and interquartile mean (IQM) of human-normalized scores (HNS) (Wang et al., 2016). Following the recommendations of Agarwal et al. (2021) on the limitations of point estimates, we provide stratified bootstrap confidence intervals for the mean and IQM in Figure 2, as well as performance profiles and additional metrics in Appendix H.

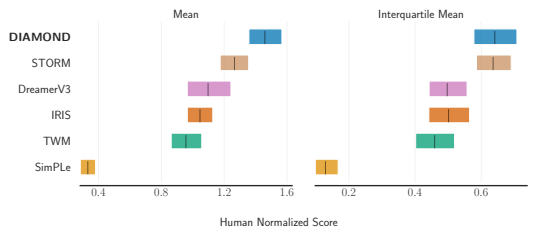


Figure 2: Mean and interquartile mean human normalized scores. DIAMOND, in blue, obtains a mean HNS of 1.46 and an IQM of 0.64.

Table 1: Returns on the 26 games of the Atari 100k benchmark after 2 hours of real-time experience, and human-normalized aggregate metrics. Bold numbers indicate the best performing methods. DIAMOND notably outperforms other world model baselines in terms of mean score over 5 seeds.

Game	Random	Human	SimPLe	TWM	IRIS	DreamerV3	STORM	DIAMOND (ours)
Alien	227.8	7127.7	616.9	674.6	420.0	959.0	**983.6**	744.1
Amidar	5.8	1719.5	74.3	121.8	143.0	139.0	204.8	**225.8**
Assault	222.4	742.0	527.2	682.6	1524.4	706.0	801.0	**1526.4**
Asterix	210.0	8503.3	1128.3	1116.6	853.6	932.0	1028.0	**3698.5**
BankHeist	14.2	753.1	34.2	466.7	53.1	**649.0**	641.2	19.7
BattleZone	2360.0	37187.5	4031.2	5068.0	13074.0	12250.0	**13540.0**	4702.0
Boxing	0.1	12.1	7.8	77.5	70.1	78.0	79.7	**86.9**
Breakout	1.7	30.5	16.4	20.0	83.7	31.0	15.9	**132.5**
ChopperCommand	811.0	7387.8	979.4	1697.4	1565.0	420.0	**1888.0**	1369.8
CrazyClimber	10780.5	35829.4	62583.6	71820.4	59324.2	97190.0	66776.0	**99167.8**
DemonAttack	152.1	1971.0	208.1	350.2	**2034.4**	303.0	164.6	288.1
Freeway	0.0	29.6	16.7	24.3	31.1	0.0	**33.5**	33.3
Frostbite	65.2	4334.7	236.9	**1475.6**	259.1	909.0	1316.0	274.1
Gopher	257.6	2412.5	596.8	1674.8	2236.1	3730.0	**8239.6**	5897.9
Hero	1027.0	30826.4	2656.6	7254.0	7037.4	**11161.0**	11044.3	5621.8
Jamesbond	29.0	302.8	100.5	362.4	462.7	445.0	**509.0**	427.4
Kangaroo	52.0	3035.0	51.2	1240.0	838.2	4098.0	4208.0	**5382.2**
Krull	1598.0	2665.5	2204.8	6349.2	6616.4	7782.0	8412.6	**8610.1**
KungFuMaster	258.5	22736.3	14862.5	24554.6	21759.8	21420.0	**26182.0**	18713.6
MsPacman	307.3	6951.6	1480.0	1588.4	999.1	1327.0	**2673.5**	1958.2
Pong	-20.7	14.6	12.8	18.8	14.6	18.0	11.3	**20.4**
PrivateEye	24.9	69571.3	35.0	86.6	100.0	882.0	**7781.0**	114.3
Qbert	163.9	13455.0	1288.8	3330.8	745.7	3405.0	**4522.5**	4499.3
RoadRunner	11.5	7845.0	5640.6	9109.0	9614.6	15565.0	17564.0	**20673.2**
Seaquest	68.4	42054.7	683.3	**774.4**	661.3	618.0	525.2	551.2
UpNDown	533.4	11693.2	3350.3	**15981.7**	3546.2	9234.0	7985.0	3856.3
#Superhuman (↑)	0	N/A	1	8	10	9	10	**11**
Mean (↑)	0.000	1.000	0.332	0.956	1.046	1.097	1.266	**1.459**
IQM (↑)	0.000	1.000	0.130	0.459	0.501	0.497	**0.636**	**0.641**

Our results demonstrate that DIAMOND performs strongly across the benchmark, outperforming human players on 11 games, and achieving a superhuman mean HNS of 1.46, a new best among agents trained entirely within a world model. DIAMOND also achieves an IQM on par with STORM, and greater than all other baselines. We find that DIAMOND performs particularly well on environments where capturing small details is important, such as *Asterix*, *Breakout* and *Road Runner*. We provide further qualitative analysis of the visual quality of the world model in Section 5.3.

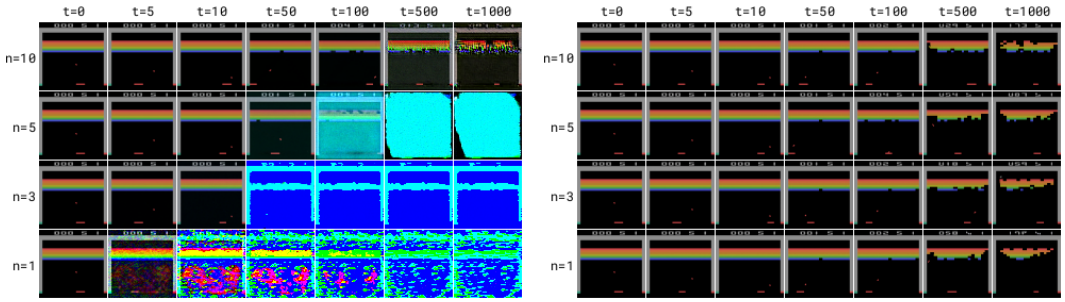
5 Analysis

5.1 Choice of diffusion framework

As explained in Section 2, we could in principle use any diffusion model variant in our world model. While DIAMOND utilizes EDM (Karras et al., 2022) as described in Section 3, DDPM (Ho et al., 2020) would also be a natural candidate, having been used in many image generation applications (Rombach et al., 2022; Nichol and Dhariwal, 2021). We justify this design decision in this section.

To provide a fair comparison of DDPM with our EDM implementation, we train both variants with the same network architecture, on a shared static dataset of 100k frames collected with an expert policy on the game *Breakout*. As discussed in Section 2.3, the number of denoising steps is directly related to the inference cost of the world model, and so fewer steps will reduce the cost of training an agent on imagined trajectories. Ho et al. (2020) use a thousand denoising steps, and Rombach et al. (2022) employ hundreds steps for Stable Diffusion. However, for our world model to be computationally comparable with other world model baselines (such as IRIS which requires 16 NFE for each timestep), we need at most tens of denoising steps, and preferably fewer. Unfortunately, if the number of denoising steps is set too low, the visual quality will degrade, leading to compounding error.

To investigate the stability of the diffusion variants, we display imagined trajectories generated autoregressively up to $t = 1000$ timesteps in Figure 3, for different numbers of denoising steps $n \leq 10$. We see that using DDPM (Figure 3a) in this regime leads to severe compounding error, causing the world model to quickly drift out of distribution. In contrast, the EDM-based diffusion world model (Figure 3b) appears much more stable over long time horizons, even for a single denoising step. A quantitative analysis of this compounding error is provided in Appendix K.



(a) DDPM-based world model trajectories.

(b) EDM-based world model trajectories.

Figure 3: Imagined trajectories with diffusion world models based on DDPM (left) and EDM (right). The initial observation at $t = 0$ is common, and each row corresponds to a decreasing number of denoising steps n. We observe that DDPM-based generation suffers from compounding error, and that the smaller the number of denoising steps, the faster the error accumulates. In contrast, our EDM-based world model appears much more stable, even for $n = 1$.

This surprising result is a consequence of the improved training objective described in Equation 7, compared to the simpler noise prediction objective employed by DDPM. While predicting the noise works well for intermediate noise levels, this objective causes the model to learn the identity function when the noise is dominant ($\sigma_{noise} \gg \sigma_{data} \implies \xi_\theta(\mathbf{x}^\tau_{t+1}, y^\tau_t) \to \mathbf{x}^\tau_{t+1}$), where ξ_θ is the noise prediction network of DDPM. This gives a poor estimate of the score function at the beginning of the sampling procedure, which degrades the generation quality and leads to compounding error.

In contrast, the adaptive mixing of signal and noise employed by EDM, described in Section 3.1, means that the model is trained to predict the clean image when the noise is dominant ($\sigma_{noise} \gg \sigma_{data} \implies \mathbf{F}_\theta(\mathbf{x}^\tau_{t+1}, y^\tau_t) \to \mathbf{x}^0_{t+1}$). This gives a better estimate of the score function in the absence of signal, so the model is able to produce higher quality generations with fewer denoising steps, as illustrated in Figure 3b.

5.2 Choice of the number of denoising steps

While we found that our EDM-based world model was very stable with just a single denoising step, as shown for *Breakout* in the last row of Figure 3b, we discuss here how this choice would limit the visual quality of the model in some cases. We provide more a quantitative analysis in Appendix L.

As discussed in Section 2.2, our score model is equivalent to a denoising autoencoder (Vincent et al., 2008) trained with an L_2 reconstruction loss. The optimal single-step prediction is thus the expectation over possible reconstructions for a given noisy input, which can be out of distribution if this posterior distribution is multimodal. While some games like *Breakout* have deterministic transitions that can be accurately modeled with a single denoising step (see Figure 3b), in some other games partial observability gives rise to multimodal observation distributions. In this case, an iterative solver is necessary to drive the sampling procedure towards a particular mode, as illustrated in the game *Boxing* in Figure 4. As a result, we therefore set $n = 3$ in all of our experiments.

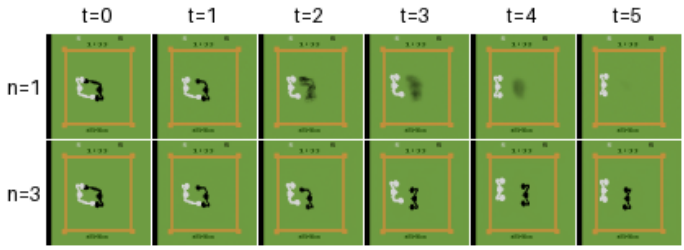
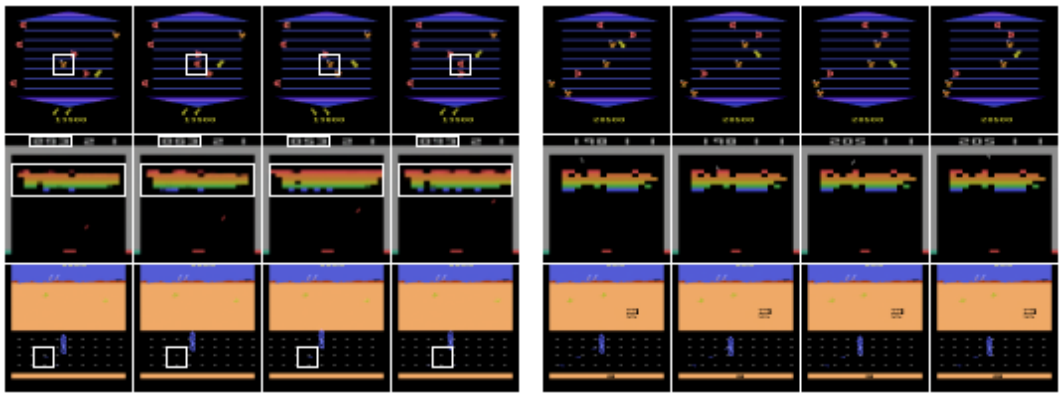


Figure 4: Single-step (top row) versus multi-step (bottom row) sampling in *Boxing*. Movements of the black player are unpredictable, so that single-step denoising interpolates between possible outcomes and results in blurry predictions. In contrast, multi-step sampling produces a crisp image by driving the generation towards a particular mode. Interestingly, the policy controls the white player, so his actions are known to the world model. This information removes any ambiguity, and so we observe that both single-step and multi-step sampling correctly predict the white player's position.

5.3 Qualitative visual comparison with IRIS

We now compare to IRIS (Micheli et al., 2023), a well-established world model that uses a discrete autoencoder (Van Den Oord et al., 2017) to convert images to discrete tokens, and composes these tokens over time with an autoregressive transformer (Radford et al., 2019). For fair comparison, we train both world models on the same static datasets of 100k frames collected with expert policies. This comparison is displayed in Figure 5 below.



(a) IRIS (b) DIAMOND

Figure 5: Consecutive frames imagined with IRIS (left) and DIAMOND (right). The white boxes highlight inconsistencies between frames, which we see only arise in trajectories generated with IRIS. In *Asterix* (top row), an enemy (orange) becomes a reward (red) in the second frame, before reverting to an enemy in the third, and again to a reward in the fourth. In *Breakout* (middle row), the bricks and score are inconsistent between frames. In *Road Runner* (bottom row), the rewards (small blue dots on the road) are inconsistently rendered between frames. None of these inconsistencies occur with DIAMOND. In *Breakout*, the score is even reliably updated by +7 when a red brick is broken[3].

We see in Figure 5 that the trajectories imagined by DIAMOND are generally of higher visual quality and more faithful to the true environment compared to the trajectories imagined by IRIS. In particular, the trajectories generated by IRIS contain visual inconsistencies between frames (highlighted by white boxes), such as enemies being displayed as rewards and vice-versa. These inconsistencies may only represent a few pixels in the generated images, but can have significant consequences for reinforcement learning. For example, since an agent should generally target rewards and avoid enemies, these small visual discrepancies can make it more challenging to learn an optimal policy.

These improvements in the consistency of visual details are generally reflected by greater agent performance on these games, as shown in Table 1. Since the agent component of these methods is similar, this improvement can likely be attributed to the world model.

Finally, we note that this improvement is not simply the result of increased computation. Both world models are rendering frames at the same resolution (64×64), and DIAMOND requires only 3 NFE per frame compared to 16 NFE per frame for IRIS. This is further reflected by the fact that DIAMOND has significantly fewer parameters and takes less time to train than IRIS, as provided in Appendix H.

6 Scaling the diffusion world model to *Counter-Strike: Global Offensive*[4]

To investigate the ability of DIAMOND's diffusion world model to learn to model more complex 3D environments, we train the world model in isolation on static data from the popular video game *Counter-Strike: Global Offensive* (CS:GO). We use the *Online* dataset of 5.5M frames (95 hours) of online human gameplay captured at 16Hz from the map *Dust II* by Pearce and Zhu (2022). We randomly hold out 0.5M frames (corresponding to 500 episodes, or 8 hours) for testing, and use the remaining 5M frames (87 hours) for training. There is no reinforcement learning agent or online data collection involved in these experiments.

[3] `https://en.wikipedia.org/wiki/Breakout_(video_game)#Gameplay`

[4] This section was added after NeurIPS acceptance, following community interest in later CS:GO experiments.

To reduce the computational cost, we reduce the resolution from (280×150) to (56×30) for world modeling. We then introduce a second, smaller diffusion model as an upsampler to improve the generated images at the original resolution (Saharia et al., 2022b). We scale the channels of the U-Net to increase the number of parameters from 4M for our Atari models to 381M for our CS:GO model (including 51M for the upsampler). The combined model was trained for 12 days on an RTX 4090.

Finally, we introduce stochastic sampling and increase the number of denoising steps for the upsampler to 10, which we found to improve the resulting visual quality of the generations, while keeping the dynamics model the same (in particular, still using only 3 denoising steps). This enables a reasonable tradeoff between visual quality and inference cost, with the model running at 10Hz on an RTX 3090. Typical generations of the model are provided in Figure 6 below.

Figure 6: Images captured from people playing with keyboard and mouse inside DIAMOND's diffusion world model. This model was trained on 87 hours of static *Counter-Strike: Global Offensive* (CS:GO) gameplay (Pearce and Zhu, 2022) to produce an interactive neural game engine for the popular in-game map, *Dust II*. Best viewed as videos at `https://diamond-wm.github.io`.

We find the model is able to generate stable trajectories over hundreds of timesteps, although is more likely to drift out-of-distribution in less frequently visited areas of the map. Due to the limited memory of the model, approaching walls or losing visibility may cause the model to forget the current state and instead generate a new weapon or area of map. Interestingly, we find the model wrongly enables successive jumps by generalizing the effect of a jump on the geometry of the scene, since multiple jumps do not appear often enough in the training gameplay for the model to learn that mid-air jumps should be ignored. We expect scaling the model and data to address many of these limitations, with the exception of the memory of the model. Quantitative measurements of the capabilities of the CS:GO world model and attempts to address these limitations are left to future work.

7 Related work

World models. The idea of reinforcement learning (RL) in the imagination of a neural network world model was introduced by Ha and Schmidhuber (2018). SimPLe (Kaiser et al., 2019) applied world models to Atari, and introduced the Atari 100k benchmark to focus on sample efficiency. Dreamer (Hafner et al., 2020) introduced RL from the latent space of a recurrent state space model (RSSM). DreamerV2 (Hafner et al., 2021) demonstrated that using discrete latents could help to reduce compounding error, and DreamerV3 (Hafner et al., 2023) was able to achieve human-level performance on a wide range of domains with fixed hyperparameters. TWM (Robine et al., 2023) adapts DreamerV2's RSSM to use a transformer architecture, while STORM (Zhang et al., 2023) adapts DreamerV3 in a similar way but with a different tokenization approach. Alternatively, IRIS (Micheli et al., 2023) builds a language of image tokens with a discrete autoencoder, and composes these tokens over time with an autoregressive transformer.

Generative vision models. There are parallels between these world models and image generation models which suggests that developments in generative vision models could provide benefits to world modeling. Following the rise of transformers in natural language processing (Vaswani et al.,

2017; Devlin et al., 2018; Radford et al., 2019), VQGAN (Esser et al., 2021) and DALL·E (Ramesh et al., 2021) convert images to discrete tokens with discrete autoencoders (Van Den Oord et al., 2017), and leverage the sequence modeling abilities of autoregressive transformers to build powerful text-to-image generative models. Concurrently, diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2020) gained traction (Dhariwal and Nichol, 2021; Rombach et al., 2022), and have become a dominant paradigm for high-resolution image generation (Saharia et al., 2022a; Ramesh et al., 2022; Podell et al., 2023).

The same trends have taken place in the recent developments of video generation methods. VideoGPT (Yan et al., 2021) provides a minimal video generation architecture by combining a discrete autoencoder with an autoregressive transformer. Godiva (Wu et al., 2021) enables text conditioning with promising generalization. Phenaki (Villegas et al., 2023) allows arbitrary length video generation with sequential prompt conditioning. TECO (Yan et al., 2023) improves upon autoregressive modeling by using MaskGit (Chang et al., 2022), and enables longer temporal dependencies by compressing input sequence embeddings. Diffusion models have also seen a resurgence in video generation using 3D U-Nets to provide high quality but short-duration video (Singer et al., 2023; Bar-Tal et al., 2024). Recently, transformer-based diffusion models such as DiT (Peebles and Xie, 2023) and Sora (Brooks et al., 2024) have shown improved scalability for both image and video generation, respectively.

Diffusion for reinforcement learning. There has also been much interest in combining diffusion models with reinforcement learning. This includes taking advantage of the flexibility of diffusion models as a policy (Wang et al., 2022; Ajay et al., 2022; Pearce et al., 2023), as planners (Janner et al., 2022; Liang et al., 2023), as reward models (Nuti et al., 2023), and trajectory modeling for data augmentation in offline RL (Lu et al., 2023; Ding et al., 2024; Jackson et al., 2024). DIAMOND represents the first use of diffusion models as world models for learning online in imagination.

Generative game engines. Playable games running entirely on neural networks have recently been growing in scope. *GameGAN* (Kim et al., 2020) learns generative models of games using a GAN (Goodfellow et al., 2014) while Bamford and Lucas (2020) use a Neural GPU (Kaiser and Sutskever, 2015). Concurrent work includes *Genie* (Bruce et al., 2024), which generates playable platformer environments from image prompts, and *GameNGen* (Valevski et al., 2024), which similarly leverages a diffusion model to obtain a high resolution simulator of the game DOOM, but at a larger scale.

8 Limitations

We identify three main limitations of our work for future research. First, our main evaluation is focused on discrete control environments, and applying DIAMOND to the continuous domain may provide additional insights. Second, the use of frame stacking for conditioning is a minimal mechanism to provide a memory of past observations. Integrating an autoregressive transformer over environment time, using an approach such as Peebles and Xie (2023), would enable longer-term memory and better scalability. We include an initial investigation into a potential cross-attention architecture in Appendix M, but found frame-stacking more effective in our early experiments. Third, we leave potential integration of the reward/termination prediction into the diffusion model for future work, since combining these objectives and extracting representations from a diffusion model is not trivial (Luo et al., 2023; Xu et al., 2023) and would make our world model unnecessarily complex.

9 Conclusion and Broader Impact

We have introduced DIAMOND, a reinforcement learning agent trained in a diffusion world model. We explained the key design choices we made to adapt diffusion for world modeling and to make our world model stable over long time horizons with a low number of denoising steps. DIAMOND achieves a mean human normalized score of 1.46 on the well-established Atari 100k benchmark; a new best among agents trained entirely within a world model. We analyzed our improved performance in some games and found that it likely follows from better modeling of critical visual details. We further demonstrated DIAMOND's diffusion world model can successfully model 3D environments and serve as a real-time neural game engine by training on static *Counter-Strike: Global Offensive* gameplay.

World models constitute a promising direction to address sample efficiency and safety concerns associated with training agents in the real world. However, imperfections in the world model may lead to suboptimal or unexpected agent behaviors. We hope that the development of more faithful and interactive world models will contribute to broader efforts to further reduce these risks.

Acknowledgments and Disclosure of Funding

We would like to thank Andrew Foong, Bálint Máté, Clément Vignac, Maxim Peter, Pedro Sanchez, Rich Turner, Stéphane Nguyen, Tom Lee, Trevor McInroe and Weipu Zhang for insightful discussions and comments. Adam and Eloi met during an internship at Microsoft Research Cambridge, and would like to thank the Game Intelligence team, including Anssi Kanervisto, Dave Bignell, Gunshi Gupta, Katja Hofmann, Lukas Schäfer, Raluca Georgescu, Sam Devlin, Sergio Valcarcel Macua, Shanzheng Tan, Tabish Rashid, Tarun Gupta, Tim Pearce, and Yuhan Cao, for their support in the early stages of this project, and a great summer.

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. (2021). Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34:29304–29320.

Ajay, A., Du, Y., Gupta, A., Tenenbaum, J., Jaakkola, T., and Agrawal, P. (2022). Is conditional generative modeling all you need for decision-making? *International Conference on Learning Representations*.

Alonso, E., Jelley, A., Kanervisto, A., and Pearce, T. (2023). Diffusion world models. *OpenReview*.

Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

Ascher, U. M. and Petzold, L. R. (1998). *Computer methods for ordinary differential equations and differential-algebraic equations*. Society for Industrial and Applied Mathematics.

Bamford, C. and Lucas, S. M. (2020). Neural game engine: Accurate learning of generalizable forward models from pixels. In *2020 IEEE Conference on Games (CoG)*, pages 81–88. IEEE.

Bar-Tal, O., Chefer, H., Tov, O., Herrmann, C., Paiss, R., Zada, S., Ephrat, A., Hur, J., Li, Y., Michaeli, T., et al. (2024). Lumiere: A space-time diffusion model for video generation. *arXiv preprint arXiv:2401.12945*.

Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., and Ramesh, A. (2024). Video generation models as world simulators.

Bruce, J., Dennis, M. D., Edwards, A., Parker-Holder, J., Shi, Y., Hughes, E., Lai, M., Mavalankar, A., Steigerwald, R., Apps, C., et al. (2024). Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*.

Chang, H., Zhang, H., Barber, J., Maschinot, A., Lezama, J., Jiang, L., Yang, M.-H., Murphy, K. P., Freeman, W. T., Rubinstein, M., Li, Y., and Krishnan, D. (2023). Muse: Text-to-image generation via masked generative transformers. In *International Conference on Machine Learning*, pages 4055–4075. PMLR.

Chang, H., Zhang, H., Jiang, L., Liu, C., and Freeman, W. T. (2022). Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. (2016). 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pages 424–432. Springer.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602:414–419.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.