# WORLD MODELS MADE OF LATENT SPACE TRANSFORMERS ARE HAPPY WITH 100K INTERACTIONS

**Jan Robine**[*], **Marc Höftmann**, **Tobias Uelwer**, **Stefan Harmeling**
Department of Computer Science
Technical University of Dortmund

## ABSTRACT

Deep reinforcement learning methods are very successful while being overly data hungry compared to human learners. To build a sample-efficient world model, we apply a Transformer to real-world episodes in an autoregressive manner: not only the compact latent states and the taken actions, but also the experienced or predicted rewards are fed into the Transformer, so that it can attend flexibly to all three modalities at various time steps. Hereby, we create a powerful world model on which a policy can be trained that compares favourably with previous sample-efficient reinforcement learning algorithms on the Atari 100k benchmark.

## 1 INTRODUCTION

Deep reinforcement learning methods have shown great success on many difficult challenging decision making problems. Notable successes include DQN (Mnih et al., 2015), PPO (Schulman et al., 2017), and MuZero (Schrittwieser et al., 2019). However, most algorithms require hundreds of millions of interactions with the environment, while humans can achieve similar results with less than 1% of these interactions. The large amount of data that is necessary renders a lot of potential real world applications of reinforcement learning impossible.

Recent works have made a lot of progress in advancing the sample efficiency. Model-free methods have been improved with auxiliary objectives (Laskin et al., 2020b), data augmentation (Yarats et al., 2021, Laskin et al., 2020a), or both (Schwarzer et al., 2021). Model-based methods have been successfully applied to complex image-based environments and have either been used for planning, such as EfficientZero (Ye et al., 2021), or for learning behaviors in imagination, such as SimPLe (Kaiser et al., 2020).

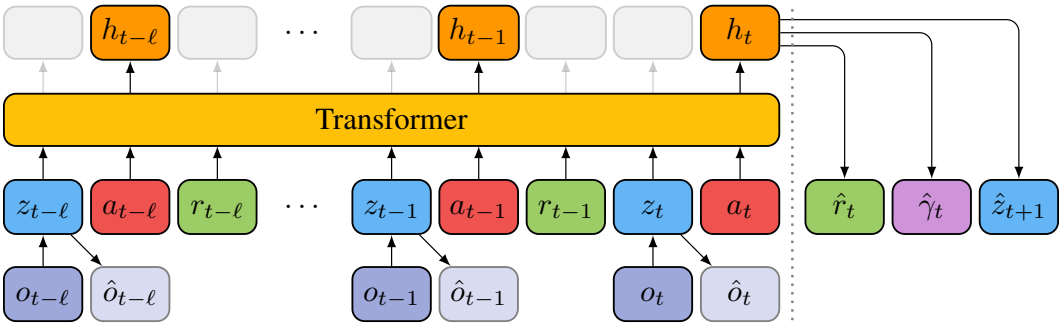


Figure 1: World model architecture. Observations $o_{t-\ell:t}$ are encoded using a CNN. Linear embeddings of stochastic discrete latent states $z_{t-\ell:t}$, actions $a_{t-\ell:t}$, and rewards $r_{t-\ell:t}$ are fed into a Transformer, that computes a deterministic state $h_t$ at each time step. Predictions of the reward $r_t$, discount factor $\gamma_t$, and next latent state $z_{t+1}$ are computed based on $h_t$ using MLPs. The policy is conditioned on the compact latent states, enabling efficient imagination without decoding the states.

[*]Correspondence to: jan.robine@tu-dortmund.de

The concept of learning in imagination (Ha & Schmidhuber, 2018; Kaiser et al., 2020; Hafner et al., 2020; Hafner et al., 2021) is promising: instead of learning behaviors from the collected experience directly, a generative model of the environment dynamics is learned in a (self-)supervised manner. Such a *world model* can create new trajectories by iteratively predicting the next state and reward. This allows for potentially indefinite training data for the reinforcement learning algorithm, without further interaction with the real environment. Due to the nature of deep neural networks, a world model can generalize to new, unseen situations, which has the potential to drastically increase the sample efficiency. This can be illustrated by a simple example: In the game of Pong the paddles and the ball move independently. A successfully trained world model is able to imagine trajectories with paddle and ball configurations that have never been observed before, which enables learning of improved behaviors.

Transformers (Vaswani et al., 2017) have significantly advanced the field of NLP and have been successfully applied to computer vision tasks (Dosovitskiy et al., 2021). A Transformer is a sequence model consisting of multiple self-attention layers with residual connections. In each self-attention layer, the inputs are mapped to keys, queries, and values. The outputs are computed by weighting the values by the similarity of keys and queries. Combined with causal masking, which prevents the self-attention layers from seeing future time steps, Transformers can be used as autoregressive generative models. The Transformer-XL architecture (Dai et al., 2019) is much more efficient than vanilla Transformers at inference time and introduces relative positional encodings, which remove the dependence on absolute time steps.

**Our contributions:** In this paper, we propose a new model-based reinforcement learning method that builds a world model based on Transformers, and trains a model-free agent in imagination. Our world model is autoregressive and even the predicted rewards are fed back into the Transformer. This allows for more accurate reward predictions, since the world model knows exactly when and how much reward it has already emitted. By utilizing the Transformer-XL architecture (Dai et al., 2019), our world model combines the benefits of Transformers and RNNs, most notably, accessing previous states directly instead of viewing them through a compressed recurrent state and learning long-term dependencies, while staying computationally efficient. Since our policy is conditioned on the compact latent states, decoding is not necessary and imagination can be performed efficiently with large batch sizes. Further, evaluation in the real environment remains lightweight, as the Transformer is not needed at inference time. We also propose a simple yet effective sampling procedure for the growing dataset of experience, which balances the training distribution. We report interval estimates of the aggregate metrics suggested by Agarwal et al. (2021).

## 2 METHOD

We consider a partially observable Markov decision process (POMDP) with discrete time steps $t \in \mathbb{N}$, scalar rewards $r_t \in \mathbb{R}$, high-dimensional image observations $o_t \in \mathbb{R}^{h \times w \times c}$ and discrete actions $a_t \in \{1, \dots, m\}$ that are generated by some policy $a_t \sim \pi(a_t \mid o_{\leq t}, a_{<t})$, where $o_{<t}$ and $o_{\leq t}$ denote the sequences of actions and observations up to timestep $t-1$. Episode ends are indicated by a boolean variable $d_t \in \{0, 1\}$. Observations, rewards and episode ends are jointly generated by the unknown environment dynamics $o_t, r_t, d_t \sim p(o_t, r_t, d_t \mid o_{<t}, a_{<t})$. The goal is to find a policy $\pi$ that maximizes the expected sum of discounted rewards $\mathbb{E}_\pi\left[\sum_{t=1}^{\infty} \gamma^t r_t\right]$, where $\gamma \in [0, 1)$ is the discount factor.

Learning in imagination consists of three steps that are repeated iteratively: learning the dynamics, learning a policy, and interacting in the real environment. In this section, we describe our world model and policy, concluding with the training procedure.

### 2.1 WORLD MODEL

Our world model consists of two models with separate parameters: The observation model and the dynamics model. Figure 1 illustrates our combined world model architecture.

**Observation Model:** The observation model is a variational autoencoder (Kingma & Welling, 2014) that encodes observations $o_t$ into compact stochastic latent states $z_t$ and reconstructs the

observations with a decoder, which in our case is only required to obtain a learning signal:

$$\begin{aligned} &\text{Observation Encoder:} \quad z_t \sim p_\phi(z_t \mid o_t) \\ &\text{Observation Decoder:} \quad \hat{o}_t \sim p_\phi(\hat{o}_t \mid z_t) \end{aligned} \tag{1}$$

We adopt a slightly modified version of the neural network architecture of DreamerV2 (Hafner et al., 2021) for our observation model. Thus, the latent states $z_t$ are discrete and consist of a vector of 32 categorical variables with 32 categories. The observation decoder reconstructs the observation and predicts the means of independent standard normal distributions for all pixels. In contrast to Hafner et al. (2021), the role of the observation model is to only capture non-temporal information about the current time step (but we use frame stacking, see Section 2.2). This stabilizes training since the learning signal is independent of other time steps.

**Autoregressive Dynamics Model:** The dynamics model predicts the next time step conditioned on the history of its past predictions. The backbone is a deterministic aggregation model $f_\psi$ which computes a deterministic hidden state $h_t$ based on the history of the $\ell$ previously generated latent states, actions, and rewards. Stochastic predictors for the reward, discount, and next latent state are conditioned on the hidden state. This leads to the following components:

$$\begin{aligned} &\text{Aggregation Model:} && h_t = f_\psi(z_{t-\ell:t}, a_{t-\ell:t}, r_{t-\ell:t-1}) \\ &\text{Reward Predictor:} && \hat{r}_t \sim p_\psi(\hat{r}_t \mid h_t) \\ &\text{Discount Predictor:} && \hat{\gamma}_t \sim p_\psi(\hat{\gamma}_t \mid h_t) \\ &\text{Latent State Predictor:} && \hat{z}_{t+1} \sim p_\psi(\hat{z}_{t+1} \mid h_t) \end{aligned} \tag{2}$$

The aggregation model is implemented as a causally masked Transformer-XL (Dai et al., 2019), which enhances vanilla Transformers (Vaswani et al., 2017) with a recurrence mechanism and relative positional encodings. With these encodings our world model learns the dynamics independent of absolute time steps. Following Chen et al. (2021), the latent states and rewards are fed into modality-specific linear embeddings before being passed to the Transformer. The discrete actions are replaced with learned embedding vectors. Due to the three modalities (states, actions, rewards), the number of input tokens is $3\ell - 1$, as the last reward $r_t$ is not part of the input. We consider the outputs of the action modality as the hidden states and disregard the outputs of the other two modalities (see top row in Figure 1).

The state, reward, and discount predictors are implemented as multilayer perceptrons (MLPs) and compute the parameters of a vector of independent categorical distributions, a normal distribution, and a Bernoulli distribution, respectively, conditioned on the deterministic hidden state. The next state is determined by sampling from $p_\psi(\hat{z}_{t+1} \mid h_t)$. The reward and discount are determined by the mean of $p_\psi(\hat{r}_t \mid h_t)$ and $p_\psi(\hat{\gamma}_t \mid h_t)$, respectively.

Due to these design choices, our world model combines the benefits of Transformers and recurrent neural networks, i.e.:

1. The dynamics model is autoregressive and has direct access to its previous outputs.
2. Training is more efficient compared with RNNs, since sequences are processed in parallel.
3. Inference is efficient due to the caching of previous outputs.
4. Long-term dependencies can be captured by the recurrence mechanism.

We want to provide an intuition on why a fully autoregressive dynamics model is beneficial. First, the direct access to previous representations enables to model more complex dependencies between them, compared with RNNs, which only see them indirectly through a compressed state. This also has the potential to make inference more robust, since bad predictions can be ignored more easily and be "skipped". Second, because the model sees which rewards it has produced previously, it can react to its own predictions. For example, consider a specific event in an environment which causes a reward. Since the dynamics model is not a perfect replication of the environment, it might base its reward prediction on other features than the true underlying cause. In this case, a non-autoregressive dynamics model might erroneously generate too much reward over multiple time steps for the same event. An autoregressive dynamics model can learn to only emit a single reward for this event and stop after that. This is even more significant when the rewards are sampled from a probability distribution, since the introduced noise cannot be observed without autoregression.

**Loss Functions:** Since the observation model is non-temporal, it is implemented as a variational autoencoder (VAE) with a stationary prior. The prior $p(z)$ follows a discrete uniform distribution. The observation decoder is optimized via negative log-likelihood. We stop the gradients of the dynamics model, so they do not influence the observation model. This leads to the self-supervised loss function for the observation model (negative ELBO)

$$\mathcal{L}_{\phi}^{\text{Repr.}} \coloneqq \mathbb{E}\left[\sum_{t=1}^{T} \underbrace{\beta D_{\text{KL}}(p_\phi(z_t \mid o_t) \parallel p(z))}_{\text{encoder regularizer}} \underbrace{- \ln p_\phi(o_t \mid z_t)}_{\text{decoder}}\right], \tag{3}$$

where $\beta > 0$ is a hyperparameter.

The reward predictor and discount predictor are optimized via negative log-likelihood. The transition predictor is optimized by minimizing the KL divergence to the outputs of the observation encoder, while the parameters of the encoder are held fixed, which reduces to the cross-entropy. This leads to the self-supervised loss function for the dynamics model

$$\mathcal{L}_{\psi}^{\text{Dyn.}} \coloneqq \mathbb{E}\left[\sum_{t=1}^{T} \underbrace{D_{\text{KL}}(p_\phi(z_{t+1} \mid o_{t+1}) \parallel p_\psi(\hat{z}_{t+1} \mid h_t))}_{\text{state predictor}} \underbrace{- \alpha_1 \ln p_\psi(r_t \mid h_t)}_{\text{reward predictor}} \underbrace{- \alpha_2 \ln p_\psi(\gamma_t \mid h_t)}_{\text{discount predictor}}\right], \tag{4}$$

where $\gamma_t = 0$ for episode ends ($d_t = 1$), and otherwise $\gamma_t = \gamma$.

### 2.2 Policy

Our policy $\pi_\theta(a_t \mid \hat{z}_t)$ is trained using a standard actor-critic style approach. We train two separate networks: an actor $a_t \sim \pi_\theta(a_t \mid \hat{z}_t)$ with parameters $\theta$ and a critic $v_\xi(\hat{z}_t)$ with parameters $\xi$. We penalize the objective of the actor with a slightly modified version of the usual entropy regularization term (Mnih et al., 2016). The penalty normalizes the entropy and only takes effect when the entropy falls below a certain threshold

$$\mathcal{L}^{\text{Ent.}}(\theta) \coloneqq \max\left(0, \Gamma - \frac{H(\pi_\theta)}{\ln(m)}\right), \tag{5}$$

where $\Gamma$ is a scalar threshold parameter, $H(\pi_\theta)$ is the entropy of the policy, $m$ is the number of discrete actions, and $\ln(m)$ is the maximum possible entropy. By doing this, we explicitly control the percentage of entropy that is preserved independent of the number of actions. This allows us to sample actions from the stochastic policy $\pi_\theta$ without changing the temperature and without $\epsilon$-greedy action selection in all scenarios: (i) collection of real data, (ii) imagination with the world model, and (iii) evaluating the policy.

We compute the advantages via Generalized Advantage Estimation (Schulman et al., 2016), while incorporating the discount factors predicted by the world model $\hat{\gamma}_t$ instead of a fixed discount factor for all time steps. As in DreamerV2 (Hafner et al., 2021), we weight the losses of the actor and the critic by the cumulative product of the discount factors, in order to softly account for episode ends.

**Algorithm 1** Training the world model and the actor-critic agent.

```
function train_world_model()
  // sample sequences of observations,
  // rewards, actions and discounts
  o,a,r,d = sample_from_dataset()
  z = encode(o)
  o_hat = decode(z)
  h = transformer(z,a,r)
  r_hat,d_hat,z_hat = predict(h)

  // optimize world model via
  // self-supervised learning
  optim_observation(o,z,o_hat,z_hat)
  optim_dynamics(r,d,z,r_hat,d_hat,z_hat)

  // z will be used for imagination
  return z
```

```
function train_actor_critic(z)
  // imagine trajectories of states,
  // rewards, actions and discounts,
  // use z as starting point
  imag = [z]
  for t = 0 until H do
    a = actor(z)
    imag.append(a)
    h = transformer(imag)
    r,d,z = predict(h)
    imag.extend([r,d,z])

  // optimize actor-critic via
  // reinforcement learning
  optim_actor_critic(imag)
```

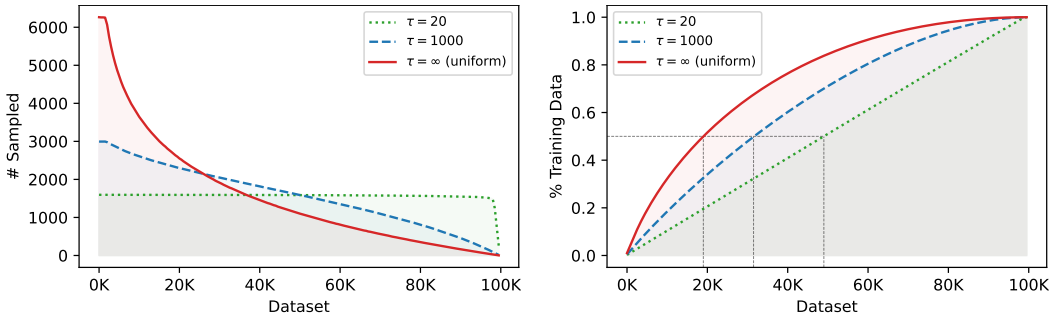


Figure 2: Comparison of different sampling procedures based on Equation (6). The x-axes correspond to the entries in dataset $\mathcal{D}$. The left plot shows the number of times the entries have been sampled after training. The right plot shows the relative amount of training that is spent up to that entry. With uniform sampling, $50\%$ of the training time is used for the first 19K entries, whereas $\tau = 20$ spends approximately the same training time on both halves of the dataset.

**Choice of Policy Input:** The policy computes an action distribution $\pi_\theta(a_t \mid x_t)$ given some view $x_t$ of the state. This view has to be chosen carefully, since it can have a significant impact on the performance of the policy, and it affects the design choices for the world model. Some evident choices for $x_t$ are: (i) the reconstructed observation $\hat{o}_t$, (ii) the latent state $\hat{z}_t$, (iii) the hidden state $h_t$, or (iv) both $\hat{z}_t$ and $h_t$.

Using $\hat{o}_t$ is stable even with imperfect predictions, since the underlying distribution of observations $p(o)$ does not change during training, but is also much less computationally efficient, since it requires decoding of the latent states and additional convolutional layers for the policy. Using $\hat{z}_t$ is slightly less stable, since the distribution $p(z|o)$ changes during training. Nevertheless, the regularizer term in Equation (3) prevents the distribution from diverging too heavily from the uniform distribution. Using $h_t$ has a nice interpretation, as it summarizes the recent history of experience and should provide meaningful information to the policy, but we found in our experiments that it degrades performance. One reason might be that $h_t$ is very instable, since it has to adapt to the newly encountered dynamics and latent states during training. It even degrades performance when using both $\hat{z}_t$ and $h_t$, compared with only $\hat{z}_t$.

For these reasons, we chose $x_t = \hat{z}_t$. To incorporate short-time temporal information into $\hat{z}_t$, we build $o_t$ by stacking the four most recent frames (as usual for model-free reinforcement learning). This also means that we do not need the dynamics model at inference time, but only the observation encoder and the actor, which is as efficient as small model-free models.

### 2.3 Training

As is usual for learning with world models, we repeatedly (i) collect experience in the real environment with the current policy, (ii) improve the world model using the past experience, (iii) improve the policy using new experience generated by the world model.

During training we build a dataset $\mathcal{D} = [(o_1, a_1, r_1, d_1), \ldots, (o_T, a_T, r_T, d_T)]$ of the collected experience. To train the world model, $N$ sequences of length $\ell$ are sampled from $\mathcal{D}$, and are used to estimate the loss functions in Equation (3) and Equation (4). After each update of the world model, we generate $M$ new trajectories of length $H$ with the dynamics model and update the policy using standard model-free objectives. In Algorithm 1 we present pseudocode for training the world model and the actor-critic agent.

**Pretraining for Better Initialization:** During training we need to correctly balance the amount of world model training and policy training, since the policy has to "keep up" with the distributional shift of the latent space. However, at the beginning of training we can spend some extra time on training the world model with pre-collected data (included in the 100k interactions), in order to obtain a reasonable initialization for the latent representations.

**Balanced Dataset Sampling:** Since the dataset grows slowly during training, uniform sampling of trajectories puts too much weight on early experience and could lead to overfitting, especially in the low data regime. Therefore, we keep visitation counters $v_1, \dots, v_T$, which are incremented every time an entry is sampled as start of a trajectory. Then, each entry $i$ has the following probability of being selected:

$$p_i = \mathrm{softmax}(-\tfrac{v_1}{\tau}, \dots, -\tfrac{v_T}{\tau})_i \tag{6}$$

where $\tau > 0$ is a scalar temperature parameter. With our sampling procedure, new entries in the dataset are oversampled and are selected more often than old ones. Setting $\tau = \infty$ restores uniform sampling as a special case, while reducing $\tau$ increases the amount of oversampling. See Figure 2 for a comparison.

## 3 Experiments

We evaluate our method on the Atari 100k benchmark, which was proposed by Kaiser et al. (2020) and is used by many data-efficient reinforcement learning algorithms. It contains 26 of the Atari games from the Arcade Learning Environment (Bellemare et al., 2013) and the agent is allowed to interact with each game 100 thousand times. This corresponds to 400 thousand frames (due to frame skipping) or roughly 2 hours of real-time gameplay, which is 500 times less than the commonly used 200 million frames.

We compare our method with five strong baselines on the Atari 100k benchmark: (i) SimPLe (Kaiser et al., 2020) implements a world model as an action-conditional video prediction model and trains a policy with PPO (Schulman et al., 2017), (ii) DER (van Hasselt et al., 2019) is a variant of Rainbow (Hessel et al., 2018) fine-tuned for sample efficiency, (iii) CURL (Laskin et al., 2020b) improves representations using contrastive learning as an auxiliary task and is combined with DER, (iv) DrQ (Yarats et al., 2021) improves DQN by averaging Q-value estimates over multiple data augmentations of observations, and (v) SPR (Schwarzer et al., 2021) forces representations to be consistent across multiple time steps and data augmentations by extending Rainbow with a self-supervised consistency loss. Additionally, we report human performance and the performance of a random agent.

TODO evaluate with stochastic policy (not deterministic and no epsilon-greedy, (mention normalized entropy?))

TODO hardware (single-gpu), model size, training time

### 3.1 Results

Table 1 shows the results of our method on the Atari 100k benchmark. Agarwal et al. (2021) found significant discrepancies between point estimates such as mean and median scores, which ignore statistical uncertainty, and thorough statistical analysis. We follow their advice and report the aggregate metrics Interquartile Mean (IQM) and Optimality Gap in Table 1, as well as interval estimates of the mean, median, IQM, and Optimality Gap in Figure 3 and performance profiles in Figure 4. The aggregate metrics are computed on human-normalized scores, which are calculated as $(\text{score}_{\text{agent}} - \text{score}_{\text{random}})/(\text{score}_{\text{human}} - \text{score}_{\text{random}})$. Agarwal et al. (2021) also provide scores for DER, CURL, DrQ, and SPR on 100 new runs, which we use for our comparison. To evaluate DrQ, Agarwal et al. (2021) used standard $\varepsilon$-greedy parameters, which they call DrQ($\varepsilon$).

TODO we achieve ...

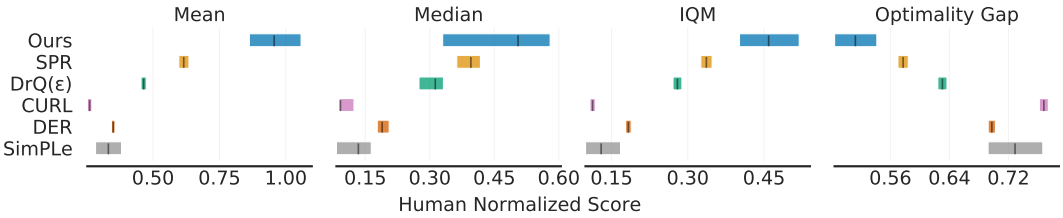


Figure 3: TODO

Table 1: Scores on the Atari 100k benchmark for each game as well as human-normalized aggregate metrics (Agarwal et al., 2021). We perform 5 runs per game and compute the average over 100 episodes at the end of training for each run. For mean, median, and IQM higher values are better, for the Optimality Gap lower values are better. TODO bold values indicate... TODO short summary

| Game | Random | Human | SimPLe | DER | CURL | DrQ($\varepsilon$) | SPR | Ours |
|---|---|---|---|---|---|---|---|---|
| Alien | 227.8 | 7127.7 | 616.9 | 802.3 | 711.0 | 865.2 | 841.9 | *674.6* |
| Amidar | 5.8 | 1719.5 | 74.3 | 125.9 | 113.7 | 137.8 | 179.7 | 121.8 |
| Assault | 222.4 | 742.0 | 527.2 | 561.5 | 500.9 | 579.6 | 565.6 | **682.6** |
| Asterix | 210.0 | 8503.3 | 1128.3 | 535.4 | 567.2 | 763.6 | 962.5 | *1116.6* |
| BankHeist | 14.2 | 753.1 | 34.2 | 185.5 | 65.3 | 232.9 | 345.4 | ***466.7*** |
| BattleZone | 2360.0 | 37187.5 | 4031.2 | 8977.0 | 8997.8 | 10165.3 | 14834.1 | 5068.0 |
| Boxing | 0.1 | 12.1 | 7.8 | -0.3 | 0.9 | 9.0 | 35.7 | ***77.5*** |
| Breakout | 1.7 | 30.5 | 16.4 | 9.2 | 2.6 | 19.8 | 19.6 | **20.0** |
| ChopperCommand | 811.0 | 7387.8 | 979.4 | 925.9 | 783.5 | 844.6 | 946.3 | ***1697.4*** |
| CrazyClimber | 10780.5 | 35829.4 | 62583.6 | 34508.6 | 9154.4 | 21539.0 | 36700.5 | ***71820.4*** |
| DemonAttack | 152.1 | 1971.0 | 208.1 | 627.6 | 646.5 | 1321.5 | 517.6 | 350.2 |
| Freeway | 0.0 | 29.6 | 16.7 | 20.9 | 28.3 | 20.3 | 19.3 | 24.3 |
| Frostbite | 65.2 | 4334.7 | 236.9 | 871.0 | 1226.5 | 1014.2 | 1170.7 | ***1475.58*** |
| Gopher | 257.6 | 2412.5 | 596.8 | 467.0 | 400.9 | 621.6 | 660.6 | **1674.8** |
| Hero | 1027.0 | 30826.4 | 2656.6 | 6226.0 | 4987.7 | 4167.9 | 5858.6 | ***7253.99*** |
| Jamesbond | 29.0 | 302.8 | 100.5 | 275.7 | 331.0 | 349.1 | 366.5 | 362.4 |
| Kangaroo | 52.0 | 3035.0 | 51.2 | 581.7 | 740.2 | 1088.4 | 3617.4 | *1240.0* |
| Krull | 1598.0 | 2665.5 | 2204.8 | 3256.9 | 3049.2 | 4402.1 | 3681.6 | **6349.2** |
| KungFuMaster | 258.5 | 22736.3 | 14862.5 | 6580.1 | 8155.6 | 11467.4 | 14783.2 | ***24554.6*** |
| MsPacman | 307.3 | 6951.6 | 1480.0 | 1187.4 | 1064.0 | 1218.1 | 1318.4 | ***1588.4*** |
| Pong | -20.7 | 14.6 | 12.8 | -9.7 | -18.5 | -9.1 | -5.4 | ***18.8*** |
| PrivateEye | 24.9 | 69571.3 | 35.0 | 72.8 | 81.9 | 3.5 | 86.0 | **86.6** |
| Qbert | 163.9 | 13455.0 | 1288.8 | 1773.5 | 727.0 | 1810.7 | 866.3 | ***3330.9*** |
| RoadRunner | 11.5 | 7845.0 | 5640.6 | 11843.4 | 5006.1 | 11211.4 | 12213.1 | 9109.0 |
| Seaquest | 68.4 | 42054.7 | 683.3 | 304.6 | 315.2 | 352.3 | 558.1 | ***774.4*** |
| UpNDown | 533.4 | 11693.2 | 3350.3 | 3075.0 | 2646.4 | 4324.5 | 10859.2 | ***15981.7*** |
| Mean | 0.000 | 1.000 | 0.332 | 0.350 | 0.261 | 0.465 | 0.616 | **0.956** |
| Median | 0.000 | 1.000 | 0.134 | 0.189 | 0.092 | 0.313 | 0.396 | **0.505** |
| IQM | 0.000 | 1.000 | 0.130 | 0.183 | 0.113 | 0.280 | 0.337 | **0.459** |
| Optimality Gap | 1.000 | 0.000 | 0.729 | 0.698 | 0.768 | 0.631 | 0.577 | **0.513** |

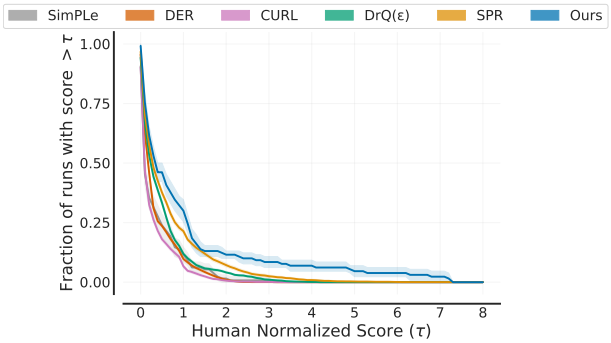


Figure 4: TODO

### 3.2 Analysis

- TODO Attention weights (see Trajectory Transformer), how many timesteps are useful?

| Game | Uniform | No Rewards | Full |
|---|---|---|---|
| Breakout | 9.7 | **20.8** | 20.0 |
| CrazyClimber | 70781.7 | **77716.6** | 71820.4 |
| KungFuMaster | 16453.3 | 23680.6 | **24554.6** |
| MsPacman | 1178.1 | 986.3 | **1588.4** |
| Pong | 2.7 | 4.3 | **18.8** |

### 3.3 Ablation Study

**1. No Rewards** TODO

**2. Uniform Sampling** TODO

TODO

- No temporal prior?
- Choice of policy input?
- No pretraining?

## 4 Related Work

The Dyna architecture (Sutton, 1991) introduced the idea of training a model of the environment and using it to further improve the value function or the policy. Ha & Schmidhuber (2018) introduced the notion of a *world model*, which tries to completely imitate the environment and is used to generate experience to train a model-free agent. They implement a world model as a VAE (Kingma & Welling, 2014) and an RNN and learn a policy in latent space with an evolution strategy. SimPLe (Kaiser et al., 2020) introduces an iterative training procedure that alternates between training the world model and the policy. Their policy operates on pixel-level and is trained using PPO (Schulman et al., 2017). Dreamer (Hafner et al., 2020) implements the world model as a stochastic recurrent neural network that splits the latent state in a stochastic state and a deterministic state (first introduced by Hafner et al., 2019). This allows their world model to capture the stochasticity of the environment and simultaneously facilitates to remember information over multiple time steps. DreamerV2 (Hafner et al., 2021) is achieves great performance on the Atari benchmark, after making some changes to Dreamer, the most important ones being Categorical latent variables and an improved objective. Robine et al. (2020) use a VQ-VAE to construct a world model with drastically lower number of parameters.

Another direction of model-based reinforcement learning is *planning*, where the model is used at inference time to improve the action selection by looking ahead in the future for multiple time steps. The most prominent work is MuZero (Schrittwieser et al., 2019), where a learned sequence model of rewards and values is combined with Monte-Carlo Tree Search (Coulom, 2006), without learning explicit representations of the observations. MuZero achieves impressive performance on the Atari benchmark, but is also computationally expensive and requires significant engineering effort. EfficientZero (Ye et al., 2021) improves MuZero and achieves great performance on the Atari 100k benchmark.

Transformers (Vaswani et al., 2017) advanced the effectiveness of sequence model in multiple domains, such as NLP and computer vision (Dosovitskiy et al., 2021). Recently, they have also been applied to reinforcement learning tasks. The Decision Transformer (Chen et al., 2021) and the Trajectory Transformer (Janner et al., 2021) are trained on an offline dataset of trajectories. The Decision Transformer is conditioned on states, actions, and returns, and outputs optimal actions. The Trajectory Transformer trains a sequence model of states, actions, and rewards, and is used for planning. Chen et al. (2022)....

## 5 DISCUSSION

We propose a new model-based reinforcement learning agent that builds an autoregressive world model based on Transformers, and trains a model-free agent by efficient imagination in a compact state space. TODO summary of results

TODO Limitations and future work

### ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

## REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 29304–29320, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47:253–279, 2013. doi: 10.1613/jair.3912. URL https://doi.org/10.1613/jair.3912.

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

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 15084–15097, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. Donkers (eds.), *Computers and Games, 5th International Conference, CG 2006, Turin, Italy, May 29-31, 2006. Revised Papers*, volume 4630 of *Lecture Notes in Computer Science*, pp. 72–83. Springer, 2006. doi: 10.1007/978-3-540-75538-8_7. URL https://doi.org/10.1007/978-3-540-75538-8_7.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 2978–2988. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1285. URL https://doi.org/10.18653/v1/p19-1285.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 2455–2467, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/2de5d16682c3c35007e4e92982f1a2ba-Abstract.html.

Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019. URL http://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3215–3222. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17204.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 1273–1286, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/099fe6b0b444c23836c4a5d07346082b-Abstract.html.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=S1xCPJHtDB.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/e615c82aba461681ade82da2da38004a-Abstract.html.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of*

*Machine Learning Research*, pp. 5639–5650. PMLR, 2020b. URL http://proceedings.mlr.press/v119/laskin20a.html.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria-Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/mniha16.html.

Jan Robine, Tobias Uelwer, and Stefan Harmeling. Smaller world models for reinforcement learning. *arXiv preprint arXiv:2010.05767*, 2020.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent 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. *CoRR*, abs/1911.08265, 2019. URL http://arxiv.org/abs/1911.08265.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron C. Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=uCQfPZwRaUu.

Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, 1991. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

Hado van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 14322–14333, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1b742ae215adf18b75449c6e272fd92d-Abstract.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 25476–25488, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/d5eca8dc3820cad9fe56a3bafda65ca1-Abstract.html.

# A APPENDIX

Table 2: Hyperparameters.

| Description | Symbol | Value |
|---|---|---|
| Environment steps | — | 100K |
| Dataset sampling temperature | $\tau$ | 20 |
| Discount factor | $\gamma$ | 0.99 |
| GAE parameter | $\lambda$ | 0.95 |
| World model batch size | $N$ | 100 |
| History length | $\ell$ | 16 |
| Imagination batch size | $M$ | 800 |
| Imagination horizon | $H$ | 15 |
| Gradient clipping | — | 100 |
| Encoder coefficient | $\beta$ | ? |
| Reward coefficient | $\alpha_1$ | 10.0 |
| Discount coefficient | $\alpha_2$ | 50.0 |