Training Agents Inside of Scalable World Models

Danijar Hafner* Wilson Yan* Timothy Lillicrap

World models learn general knowledge from videos and simulate experience for training behaviors in imagination, offering a path towards intelligent agents. However, previous world models have been unable to accurately predict object interactions in complex environments. We introduce Dreamer 4, a scalable agent that learns to solve control tasks by reinforcement learning inside of a fast and accurate world model. In the complex video game Minecraft, the world model accurately predicts object interactions and game mechanics, outperforming previous world models by a large margin. The world model achieves real-time interactive inference on a single GPU through a shortcut forcing objective and an efficient transformer architecture. Moreover, the world model learns general action conditioning from only a small amount of data, allowing it to extract the majority of its knowledge from diverse unlabeled videos. We propose the challenge of obtaining diamonds in Minecraft from only offline data, aligning with practical applications such as robotics where learning from environment interaction can be unsafe and slow. This task requires choosing sequences of over 20,000 mouse and keyboard actions from raw pixels. By learning behaviors in imagination, Dreamer 4 is the first agent to obtain diamonds in Minecraft purely from offline data, without environment interaction. Our work provides a scalable recipe for imagination training, marking a step towards intelligent agents.

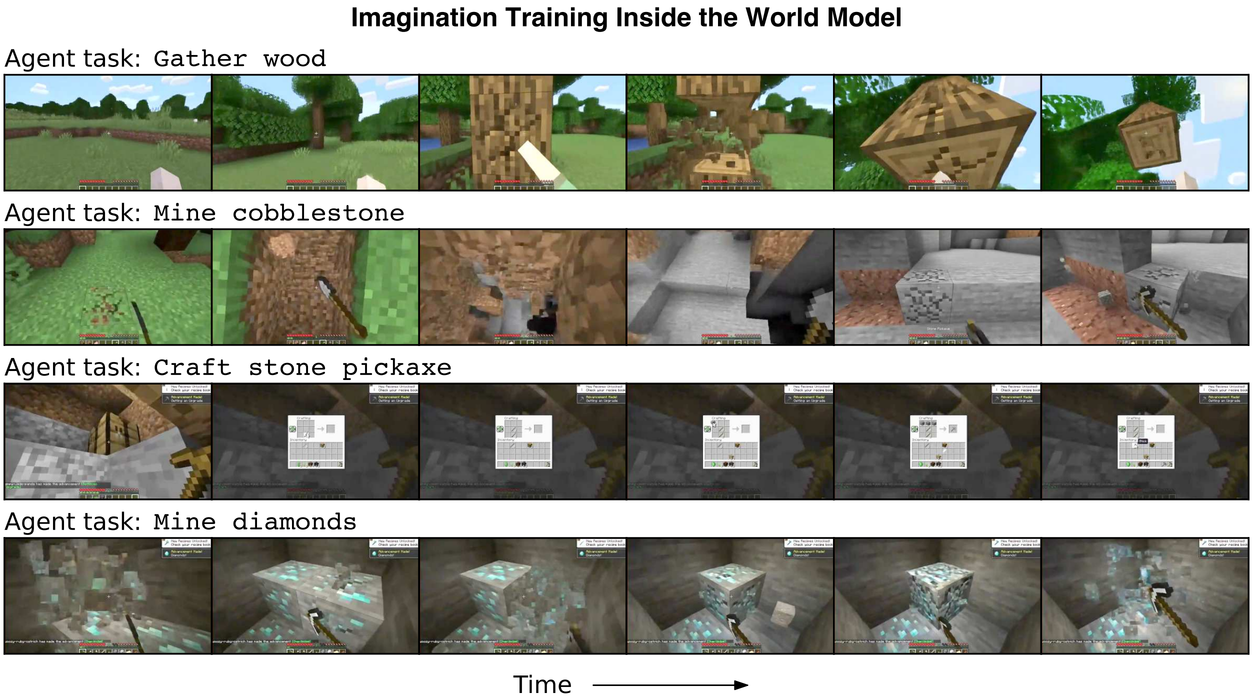

Imagination Training Inside the World Model
Agent task: Gather wood
Agent task: Mine cobblestone
Agent task: Craft stone pickaxe
Agent task: Mine diamonds
Time

Figure 1: Dreamer 4 learns to solve complex control tasks by reinforcement learning inside of its world model. We decode the imagined training sequences for visualization, showing that the world model has learned to simulate a wide range of game mechanics from low-level mouse and keyboard actions, including breaking blocks, using tools, and interacting with crafting tables.

* Equal contribution. Google DeepMind, San Francisco, USA. Website: danijar.com/dreamer4

1. Introduction

To solve complex tasks in embodied environments, intelligent agents need to deeply understand the world and choose successful actions. World models offer a promising approach towards this goal by learning to predict the future outcomes of potential actions from the perspective of an agent, such as a robot or a video game player. This way, world models equip agents with a deep understanding of the world and the ability to choose actions by planning or reinforcement learning in imagination. Moreover, world models can in principle learn from fixed datasets, allowing to train agents purely in imagination without the need for online interaction. Optimizing behaviors offline is valuable for many practical applications, such as robots in the physical world, where online interaction with a partially trained agent is often unsafe.

World model agents, such as Dreamer 3, are among the best-performing and most robust reinforcement learning algorithms for games and robotics to date[1–6]. While these models are fast and accurate for their narrow environments, their architecture lacks the ability to fit complex real world distributions. Controllable video models, such as Genie 3, have been trained on diverse real video and games and have accomplished diverse scene generation and simple interactions[7–12]. These models are based on scalable architectures, such as diffusion transformers[13,14]. However, they still struggle to learn the precise physics of object interactions and game mechanics, limiting their usefulness for training successful agents. Moreover, they often require many GPUs to simulate a single scene in real time, further reducing their practicality for imagination training.

We introduce Dreamer 4, a scalable agent that solves control tasks by imagination training inside of a fast and accurate world model. Dreamer 4 is the first agent to obtain diamonds in the challenging video game Minecraft purely from a standard offline dataset, without environment interaction. Dreamer 4 leverages a novel shortcut forcing objective and an efficient transformer architecture to accurately learn complex object interactions while enabling real-time human interaction and efficient imagination training. We show that the world model accurately predicts a wide range of semantic interactions in Minecraft, outperforming previous world models by a large margin. Moreover, Dreamer 4 can be trained on large amounts of unlabeled videos and requires only a small amount of videos paired with actions. This opens up the possibility of learning general world knowledge from diverse web videos in the future, for which action labels are not available.

Our contributions are summarized as follows:

- We introduce Dreamer 4, a scalable agent that learns to solve challenging control tasks by imagination training inside of a world model.
- Dreamer 4 is the first agent to collect diamonds in Minecraft from only offline data, substantially improving over OpenAI's VPT offline agent[15] despite using 100× less data.
- We introduce a high-capacity world model that achieves real-time inference on a single GPU through a shortcut forcing objective and an efficient transformer architecture.
- We show that the world model accurately predicts a wide range of object interactions and game mechanics in Minecraft, substantially outperforming previous world models.
- We show that the world model can learn from unlabeled videos and requires only a small amount of aligned data to learn action conditioning with strong generalization.
- An extensive ablation study measures the improvements of the objective and architecture.

2. Background

Flow matching Our world model is based on the paradigm of diffusion models[16,17], where the network f_θ is trained to restore the a data point x_1 given a corrupted version x_τ. The signal level $\tau \in [0, 1]$ determines the mixture of noise and data and is randomized during training, where $\tau = 0$ corresponds to pure noise and $\tau = 1$ means clean data. We build on the flow matching formulation[18,19] because of its simplicity, where the network predicts the velocity vector $v = x_1 - x_0$ that points towards the clean data:

$$\begin{gathered} x_\tau = (1-\tau)\,x_0 + \tau\,x_1 \qquad x_0 \sim \mathrm{N}(0, \mathbb{I}) \qquad x_1 \sim \mathcal{D} \qquad \tau \sim p(\tau) \\ \mathcal{L}(\theta) = \| f_\theta(x_\tau, \tau) - (x_1 - x_0) \|^2 \end{gathered} \tag{1}$$

The signal level is typically sampled from a uniform distribution or a logit-normal distribution[20]. At inference time, the sampling process starts with a pure noise vector x_0 and iteratively transforms it into a clean data point x_1 over K sampling steps with step size $d = 1/K$:

$$x_{\tau+d} = x_\tau + f_\theta(x_\tau, \tau)\,d \qquad x_0 \sim \mathrm{N}(0, \mathbb{I}) \tag{2}$$

Shortcut models Shortcut models[21] condition the neural network not only on the signal level τ but also on the requested step size d. This allows them to choose the step size at inference time and generate data points using only a few sampling steps and forward passes of the neural network. For the finest step size $d_{\min}$, shortcut models are trained using the flow matching loss. For larger step sizes $d_{\min} < d \leq 1$, shortcut models are trained using a bootstrap loss that distills two smaller steps, where $\mathrm{sg}(\cdot)$ stops the gradient:

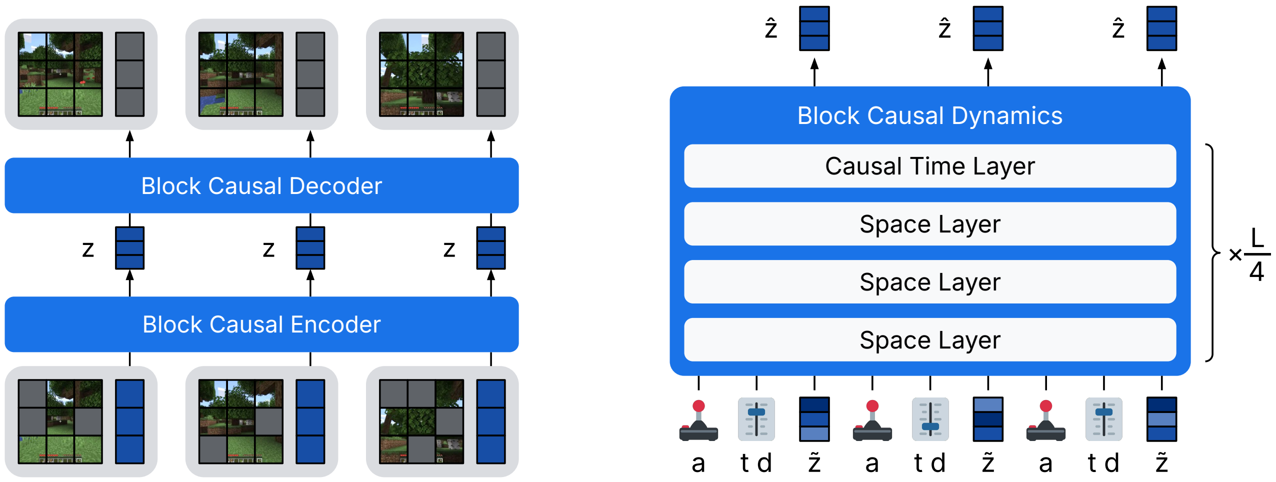
$$\begin{gathered} x_0 \sim \mathrm{N}(0, \mathbf{I}) \qquad x_1 \sim \mathcal{D} \qquad \tau, d \sim p(\tau, d) \\ b' = f_\theta(x_\tau, \tau, d/2) \qquad b'' = f_\theta(x', \tau + d/2, d/2) \qquad x' = x_\tau + b'\,d/2 \\ \mathcal{L}(\theta) = \| f_\theta(x_\tau, \tau, d) - v_{\text{target}} \|^2 \qquad v_{\text{target}} = \begin{cases} x_1 - x_0 & \text{if } d = d_{\min} \\ \mathrm{sg}(b' + b'')/2 & \text{else} \end{cases} \end{gathered} \tag{3}$$

The step size is sampled uniformly as a power of two, based on the maximum number of sampling steps $K_{\max}$, which defines the finest step size $d_{\min} = 1/K_{\max}$. The signal level is sampled uniformly over the grid that is reached by the current step size:

$$d \sim 1/\mathrm{U}(\{1, 2, 4, 8, \ldots, K_{\max}\}) \qquad \tau \sim \mathrm{U}(\{0, 1/d, \ldots, 1 - 1/d\}) \tag{4}$$

At inference time, one can condition the model on a step size $d = 1/K$ to target K sampling steps, without suffering from discretization error because the model has learned to predict the end point of each step. In practice, shortcut models generate high-quality samples with 2 or 4 sampling steps, compared to 64 or more steps for typical diffusion models.

Diffusion forcing For sequential data, diffusion forcing[14] assigns a different signal level to each time step of the data sequence, producing a corrupted sequence. This allows applying loss terms to all time steps in the sequence, where each time step serves both as denoising task and as history context for later time steps. At inference time, diffusion forcing supports flexible noise patterns, such as generating the next frame given clean or lightly noised history.



(a) Causal Tokenizer

(b) Interactive Dynamics

Figure 2: World model design. Dreamer 4 consists of a causal tokenizer and an interactive dynamics model, which both use the same block-causal transformer architecture. The tokenizer encodes partially masked image patches and latent tokens, squeezes the latents through a low-dimensional projection with tanh activation, and decodes the patches. It uses causal attention to achieve temporal compression while allowing frames to be decoded one by one. The dynamics model operates on the interleaved sequence of actions, shortcut noise levels and step sizes, and tokenizer representations. It denoises representations via a shortcut forcing objective. After pretraining, the world model is finetuned into an agent by inserting task tokens into the dynamics transformer and predicting actions, rewards, and values from them.

3. World Model Agent

We present Dreamer 4, a scalable agent that learns to solve complex control tasks by reinforcement learning inside of a fast and accurate world model. The agent consists of a tokenizer and a dynamics model, as shown in Figure 2. The tokenizer compresses video frames into continuous representations and the dynamics model predicts the representations given interleaved actions, both using the same efficient transformer architecture. The tokenizer is trained using masked autoencoding and the dynamics is trained using a shortcut forcing objective to enable interactive generations with a small number of forward passes and prevent accumulating errors over time. As outlined in Algorithm 1, we first pretrain the tokenizer and world model on videos and actions, then finetune the policy and reward model into the world model by interleaving task embeddings, and finally post-train the policy through imagination training. To train a single dynamics transformer with multiple modalities and output heads, we normalize all loss terms by running estimates of their root-mean-square (RMS).

Algorithm 1 Dreamer 4

Phase 1: World Model Pretraining

- Train tokenizer on videos using (5).
- Train world model on tokenized videos and optionally actions using (7).

Phase 2: Agent Finetuning

- Finetune world model with task inputs for policy and reward heads using (7) and (9).

Phase 3: Imagination Training

- Optimize policy head using (11) and value head using (10) on trajectories generated by the world model and the policy head.

3.1. Causal Tokenizer

The tokenizer compresses raw video into a sequence of continuous representations for the dynamics model to consume and generate. It consists of an encoder and a decoder with a bottleneck in between. Both components are causal in time, enabling temporal compression while maintaining the ability to decode frame by frame for interactive inference.

Architecture We use the efficient transformer architecture described later. Each time step consists of patch tokens of the current image and learned latent tokens. After applying the encoder, the representations are read out of the latent tokens using a linear projection to a smaller channel dimension followed by a `tanh` activation. For the decoder, this representation is projected back up to the model dimension and concatenated with learned tokens to read out the patches. To flexibly integrate multiple input modalities if available, the encoder allows the latent tokens to attend to all modalities, while each modality only attends within itself. Correspondingly, each decoder modality attends within itself and to the latents, while the latents only attend within themselves.

Masked autoencoding We train the tokenizer using a straightforward reconstruction objective, consisting of mean squared error and LPIPS[22] loss. To simplify weighing the two loss terms, we employ loss normalization as explained later.

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{MSE}}(\theta) + 0.2\,\mathcal{L}_{\text{LPIPS}}(\theta) \tag{5}$$

We drop out input patches to the encoder to improve its representations using masked autoencoding[23,24]. The dropout probability is randomized across images as $p \sim U(0, 0.9)$. Patches of each image are replaced with a learned embedding with this probability, so that the tokenizer is sometimes trained on the $p = 0$ case used during inference. We found MAE training to improve the spatial consistency of videos generated by the dynamics model.

3.2. Interactive Dynamics

The dynamics model operates on the interleaved sequence of actions and representations produced by the frozen tokenizer. It is trained using a shortcut forcing objective to enable fast interactive inference with $K = 4$ forward passes per generated frame.

Architecture The dynamics model uses our efficient transformer architecture on interleaved blocks of observations and actions. The representations are linearly projected into S_{z} spatial tokens and concatenated with S_{r} learned register tokens[25] and a single token for the shortcut signal level and step size. Since the signal level and step size are discrete, we encode each with a discrete embedding lookup and concatenate their channels. Actions can contain multiple components, such as mouse and keyboard. We encode each action component separately into S_{a} tokens and sum the results together with a learned embedding. Continuous actions components are linearly projected and categorical or binary components use an embedding lookup. When training unlabeled videos, only the learned embedding is used.

Shortcut forcing For efficient training and inference, we train the dynamics model using a shortcut forcing objective, which builds on diffusion forcing[14] and shortcut models[21], reviewed in Section 2. We formulate the objective in data space to prevent accumulating errors caused by high-frequency network outputs and introduce a simple loss weight to focus the model capacity on the loss terms with the most learning signal. The dynamics model takes the interleaved sequence of actions $a = \{a_t\}$, discrete signal levels $\tau = \{\tau_t\}$ and step sizes $d = \{d_t\}$, and corrupted representations $\tilde{z} = \{z_t^{(\tau_t)}\}$ as input and predicts the clean representations $z_1 = \{z_t^1\}$. Note that $t \in [1, T]$ is the sequence timestep while $\tau_t \in [0, 1]$ is the signal level at that step.

$$
\begin{gathered}
z_0 \sim \mathrm{N}(0, \mathbf{1}) \qquad z_1 \sim \mathcal{D} \qquad \tau, d \sim p(\tau, d) \qquad \tau, d \in [0, 1]^T \\
\hat{z}_1 = f_\theta(\tilde{z}, \tau, d, a) \qquad \tilde{z} = (1 - \tau)\, z_0 + \tau\, z_1
\end{gathered}
\tag{6}
$$

Shortcut models parameterize the network to predict velocities $v = x_1 - x_0$, called v-prediction[26]. This approach excels when generating the output jointly as one block, such as for image or video generation models. However, v-prediction trains the network to produce high-frequency outputs. When iteratively generating long videos frame by frame, this can cause subtle errors that accumulate over time. Instead, we found that parameterizing the network to predict clean representations, called x-prediction, enables high-quality rollouts of arbitrary length. Computing the flow loss term in x-space is straightforward[26]. To compute the bootstrap loss term, we convert the network output into v-space and scale the resulting loss back into x-space[*]:

$$
\begin{aligned}
& b' = (f_\theta(\tilde{z}, \tau, \tfrac{d}{2}, a) - z_\tau)/(1 - \tau) \qquad z' = \tilde{z} + b'\, \tfrac{d}{2} \\
& b'' = (f_\theta(z', \tau + \tfrac{d}{2}, \tfrac{d}{2}, a) - z')/(1 - (\tau + \tfrac{d}{2})) \\
\mathcal{L}(\theta) = & \begin{cases} \|\hat{z}_1 - z_1\|_2^2 & \text{if } d = d_{\min} \\ (1 - \tau)^2 \|(\hat{z}_1 - \tilde{z})/(1 - \tau) - \operatorname{sg}(b_1 + b_2)/2\|_2^2 & \text{else} \end{cases}
\end{aligned}
\tag{7}
$$

Low signal levels contain less learning signal, because the flow matching term degenerates to predicting the dataset mean, while the bootstrap term is generally easier to optimize because it has deterministic targets compared to the noisy flow matching term. To focus the model capacity on signal levels with the most learning signal, we propose a `ramp` loss weight that linearly increases with the signal level τ, where $\tau = 0$ corresponds to full noise and $\tau = 1$ to clean data:

$$
w(\tau) = 0.9\tau + 0.1 \tag{8}
$$

At inference time, the dynamics model supports different noise patterns. We sample autoregressively in time and generate the representations of each frame using the shortcut model with $K = 4$ sample steps with corresponding step size $d = 1/4$. We slightly corrupt the past inputs to the dynamics model to signal level $\tau_{\text{ctx}} = 0.1$ to make the model robust to small imperfections in its generations.

3.3. Imagination Training

To solve control tasks, we first adapt the pretrained world model to predict actions and rewards from the dataset conditioned on one of multiple tasks. For this, we insert agent tokens as an additional modality into the world model transformer and interleave it with the image representations, actions, and register tokens. The agent tokens receive task embeddings as input and we use them to predict the policy and reward model using MLP heads. While the agent tokens attend to themselves and all other modalities, no other modalities can attend back to the agent tokens. This is crucial for avoiding causal confusion of the world model—its future predictions can only be directly influenced

[*] The network output is converted as $\hat{v}_\tau = (\hat{x}_1 - x_\tau)/(1 - \tau)$. The MSE in x-space and v-space is related by $\|\hat{x}_1 - x_1\|_2^2 = (1 - \tau)^2 \|\hat{v}_\tau - v_\tau\|_2^2$, motivating a $(1 - \tau)^2$ multiplier to bring the bootstrap loss into a range similar to the x-space flow loss.

by actions, not by the current task. To improve beyond strategies displayed in the dataset, we then finetune the policy through imagination training by reinforcement learning on rollouts generated by the world model, using an additional value head.

Behavior cloning and reward model After pretraining the world model on action-conditioned video prediction, the second training phase involves learning a task-conditioned policy and reward model. Given a dataset of videos $x = \{x_t\}$ that are encoded into representations $z = \{z_t\}$, actions $a = \{a_t\}$, tasks $q = \{q_t\}$, and scalar rewards $r = \{r_t\}$, we train the policy and reward heads on the task output embeddings h_t using multi-token prediction (MTP)[27] of length $L = 8$:

$$\mathcal{L}(\theta) = -\sum_{n=0}^{L} \ln p_\theta(a_{t+n} \mid h_t) - \sum_{n=0}^{L} \ln p_\theta(r_{t+n} \mid h_t) \tag{9}$$

To preserve existing capabilities, we reuse the pretraining setting with this additional loss function, so the representations are noisy and we continue to apply the video prediction loss. We parameterize the policy and reward heads using small MLPs with one output layer per MTP distance. Following Dreamer 3, the reward head is parameterized as a symexp twohot output[1] to robustly learn stochastic rewards across varying orders of magnitude. The policy head is parameterized as categorical or vectorized binary distribution, depending on the action space of the dataset.

Reinforcement learning To improve the policy beyond behaviors displayed in the dataset, we continue training it with reinforcement learning on imagined rollouts to maximize the learned reward model. Unlike online reinforcement learning that requires interaction with the environment, our policy learns purely inside the world model, enabling it to improve offline. We initialize a value head and a frozen copy of the policy head that serves as a behavioral prior. We only update the policy and value heads and keep the transformer frozen.[†] Imagined rollouts start from contexts of the dataset that was used during the earlier training phases. Unlike previous generations of Dreamer, we start only one rollout from each context, prioritizing data diversity and reducing memory consumption. The rollouts are generated by unrolling the transformer with itself, sampling representations $z = \{z_t\}$ from the flow head and actions $a = \{a_t\}$ from the policy head. We annotate the resulting trajectories with rewards $r = \{r_t\}$ using the reward head and values $v = \{v_t\}$ using the value head.

The value head is trained to predict the discounted sum of future rewards, allowing the policy to maximize rewards beyond the imagination horizon. It uses a symexp twohot output to robustly learn across different scales of values[1]. We train the value head using temporal difference learning (TD-learning)[28] to predict λ-returns computed from the predicted rewards and values along the sequence, where $\gamma = 0.997$ is a discount factor and c_t indicates non-terminal states:

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \ln p_\theta(R^\lambda_t \mid s_t) \qquad R^\lambda_t = r_t + \gamma c_t\big((1-\lambda)v_t + \lambda R^\lambda_{t+1}\big) \qquad R^\lambda_T = v_T \tag{10}$$

Unlike previous generations of Dreamer, the policy head learns using PMPO[29], a robust reinforcement learning objective that uses the sign of the advantages $A_t = R^\lambda_t - v_t$ and ignores their magnitude. This property alleviates the need for normalizing returns or advantages and ensures equal focus on all tasks despite potentially differing return scales. PMPO balances the focus on positive and negative feedback by separately averaging a simple maximum likelihood loss over the states with positive and negative advantages, respectively. We assign all imagined states s_i across batch and time dimensions

[†] Finetuning the full transformer provides small additional benefits at higher computational cost. For that, the dynamics, policy prior, and reward losses need to be applied during imagination training to preserve their functions.

to the positive set $\mathcal{D}^+ = \{s_i \mid A_t \geq 0\}$ or the negative set $\mathcal{D}^- = \{s_i \mid A_t < 0\}$ and apply the following policy loss:

$$\mathcal{L}(\theta) = \frac{1-\alpha}{|\mathcal{D}^-|}\sum_{i\in\mathcal{D}^-}\ln\pi_\theta(a_i \mid s_i) - \frac{\alpha}{|\mathcal{D}^+|}\sum_{i\in\mathcal{D}^+}\ln\pi_\theta(a_i \mid s_i) + \frac{\beta}{N}\sum_{i=1}^{N}\mathrm{KL}[\pi_\theta(a_i \mid s_i) \,\|\, \pi_{\text{prior}}] \tag{11}$$

We set $\alpha = 0.5$ to balance the positive and negative sets equally and use a weaker scale of $\beta = 0.3$ for the behavioral prior. Unlike the original PMPO objective, we use a reverse direction for the prior KL to better constrain the policy to the space of reasonable behaviors. We find the scaling between the three objective terms to be highly robust in practice as they are all measured in nats[30].

3.4. Efficient Transformer

Scaling up world models to diverse data distributions while maintaining fast inference requires an efficient high capacity architecture. In this section, we introduce our efficient transformer architecture that is used for both the tokenizer and the dynamics model. The architecture is a 2D transformer[31] with time and space dimensions. To support interactive generation, the attention is masked to be causal in time, so that all tokens within a time step can attend to each other and to the past. We start from a standard transformer with pre-layer RMSNorm[32], RoPE[33], and SwiGLU[34]. We employ QKNorm[35] and attention logit soft capping[36,37] to increase training stability.

Efficiency Inference speed of block-causal transformer architectures is limited both by the FLOPs of the MLPs and the memory bandwidth needed to access the KV cache of a long context to attend into. We employ a sequence of improvements to speed up inference, some of which also accelerate training. First, we break up the cost of dense attention over all video tokens by using separate space-only and time-only attention layers[38]. Second, we find that only a relatively small number of temporal layers are needed and only use temporal attention once every 4 layers, in line with recent findings[39]. Third, we apply GQA[40] to all attention layers in the dynamics, where multiple query heads attend to the same key-value heads to reduce the KV cache size further.

Sequence length Increasing spatial tokens directly improves visual quality, whereas increasing temporal tokens allows training longer context lengths for more temporally consistent generation. To support efficient training, we alternate training on many short batches and occasional long batches, and finetune the model on only long batches afterwards. Alternating batch lengths produces intermediate training metrics and generations that are more indicative of final model performance than training only on short batches. The batch lengths need to be longer than the context length of the model to prevent the transformer from overfitting to always seeing a start frame at the beginning of its context, enabling length generalization to arbitrary generation lengths.

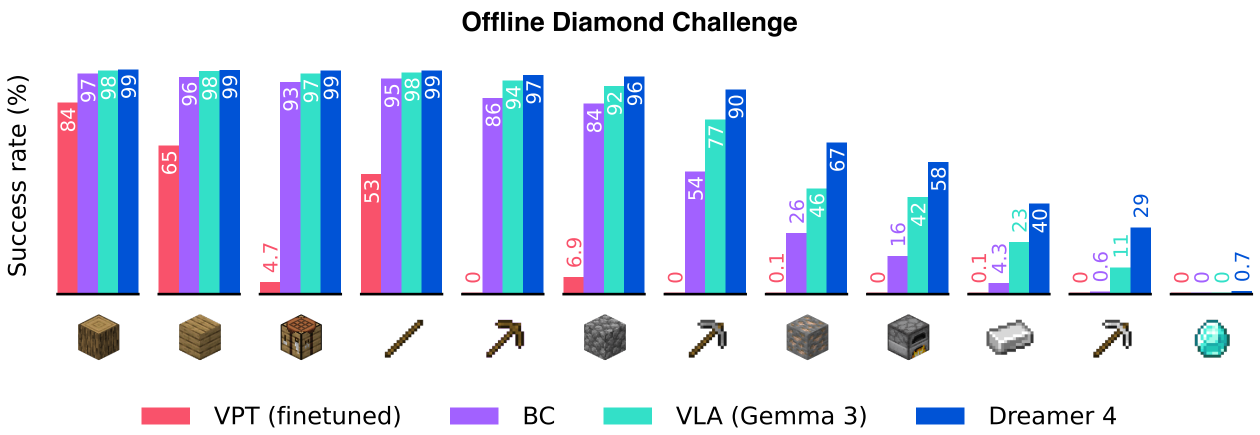


Figure 3: Agent performance in Minecraft without environment interaction. All methods have access to the same contractor dataset[15] with image inputs and low-level mouse and keyboard actions. We report the success rates of important items obtained during 60-minute episodes that start in random worlds and from empty inventory, computed over 1000 episodes. Leveraging imagination training, Dreamer 4 is the first agent to obtain diamonds in Minecraft purely from offline experience. Dreamer 4 substantially outperforms OpenAI's VPT offline agent[15] while using 100× less data. It also outperforms our VLA agent[41,42], which leverages the general knowledge of the Gemma 3 vision-language model[43], nearly tripling its success rate for crafting iron pickaxes.

4. Experiments

We perform a wide range of experiments to evaluate and explore the capabilities of Dreamer 4. The majority of our experiments focus on Minecraft, a complex video game that features infinite open worlds including monsters and hundreds of items that can be mined or crafted, with raw pixel observations and low-level mouse and keyboard actions. We primarily use the VPT dataset[15] that contains 2541 hours of contractor gameplay with 360p video and mouse and keyboard actions at 20 FPS. The experiments are designed to answer the following questions:

- Does Dreamer 4 learn to solve challenging control tasks purely by imagination training inside the world model, without online environment interaction? (Section 4.1)
- How well does Dreamer 4 learn to predict accurate object interactions and game mechanics in Minecraft compared to previous world models? (Section 4.2)
- How much action data does Dreamer 4 need for learning action conditioning, and how far does the learned action grounding generalize? (Section 4.3)
- To what extent does each component of its objective and architecture contribute to the performance of Dreamer 4? (Section 4.4)

We train models with 2B parameters—400M for the tokenizer and 1.6B for the dynamics model—on 256 to 1024 TPU-v5p with batch size 1 per device and FSDP sharding[44,45]. To improve generations without context, we treat 30% of the videos in the batch as separate images, effectively training the dynamics model to generate start frames. For Minecraft, we use 256 spatial tokens with 192 frames context length and 256 batch length. For the real world datasets, we use 512 spatial tokens with 96 frames context length and 128 batch length.

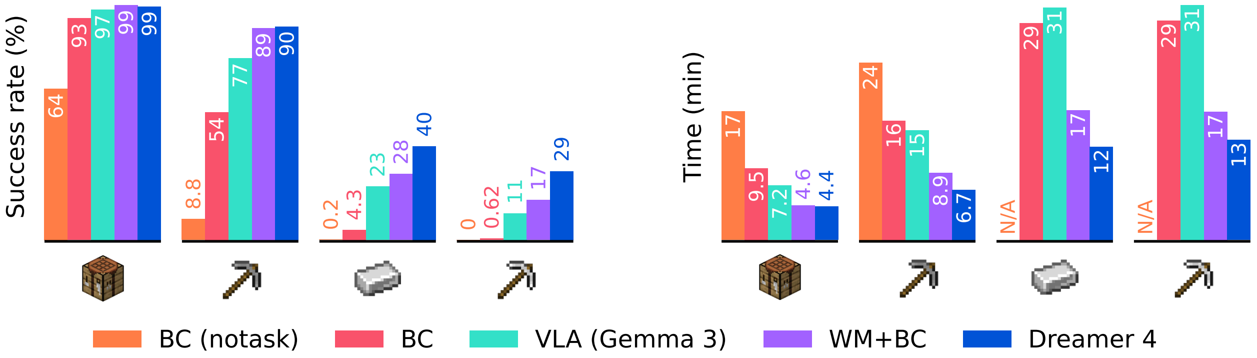


Figure 4: Agent ablations on the offline diamond challenge. We report success rates and time needed to reach an item for four milestone items. Dreamer 4 outperforms methods based on behavioral cloning in both metrics, demonstrating that imagination training improves both the robustness and efficiency of the policy. Moreover, using the world model representations for behavioral cloning outperforms using Gemma 3 or training from scratch.

4.1. Offline Diamond Challenge

We evaluate Dreamer 4 on the Minecraft diamond challenge, a long-horizon control task that requires solving several sub tasks, such as gathering materials and crafting tools in a complex procedurally generated 3D world from raw pixels and mouse and keyboard actions. Human players with Minecraft experience take 20 minutes to collect a diamond on average, corresponding to sequences of 24,000 mouse and keyboard actions.

Offline setting While previous agents have achieved diamonds in Minecraft through online interaction with the environment[1,15], deploying partially-trained policies is often infeasible for practical applications, such as physical robots, because ensuring safety, resetting the scene, and providing rewards in real time is difficult. Instead, we focus on the challenge of learning purely offline from a fixed experience dataset. We only use the VPT contractor dataset[15]—which contains 2.5K hours of videos, actions, and event annotations—without allowing the agent to interact with the environment for learning, and compare to baselines in this offline setting. We follow the VPT evaluation protocol of raw pixel inputs and low-level mouse and keyboard actions, requiring crafting through the in-game user interface. Episodes last 60 minutes, starting from an empty inventory in randomly generated Minecraft worlds.

Implementation Dreamer 4 learns a single transformer that predicts inputs, actions, rewards, and values. To build a steerable agent, we opt for a multi-task setting and condition the actions, rewards, and values on task embeddings. We annotate the tasks and their sparse binary rewards using the existing events in the VPT dataset. Table 4 lists the 20 tasks and Table 6 shows the linear prompt sequence that guides the agent to reach diamonds during evaluation in the environment. To amplify the signal in the dataset during behavior cloning, reward modeling, and reinforcement learning, we use data mixture of 50% uniform sequences and 50% relevant sequences that accomplish one of the tasks. The behavioral cloning loss is applied only on the relevant fraction, while the dynamics loss is applied only on the uniform sequences to avoid optimistic generations. We use one-hot task indicators but text embeddings could easily be used. We represent keyboard actions as 23 binary distributions and mouse actions as a categorical with 121 classes using foveated discretization[15].

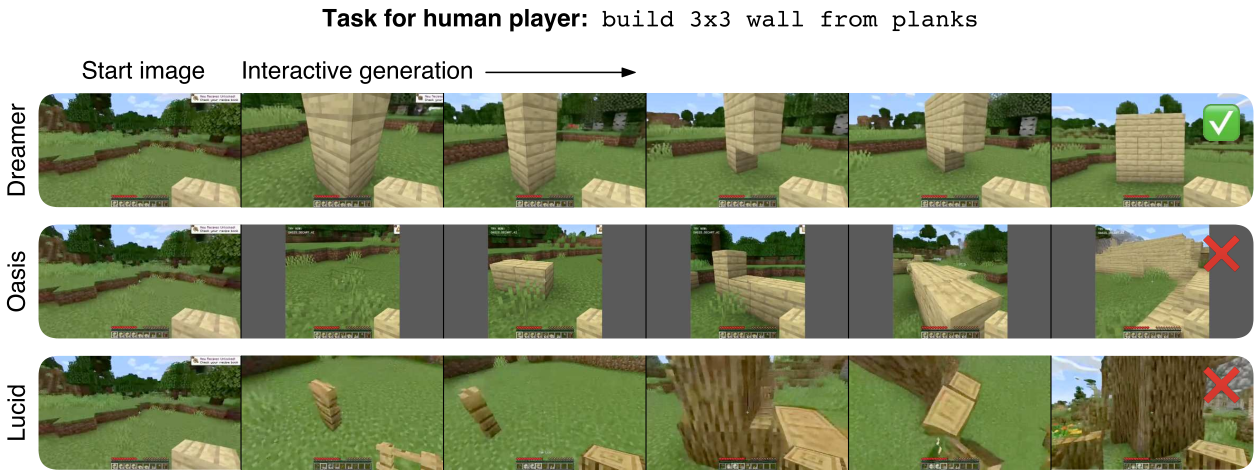


Figure 5: Human interaction. A human player counterfactually interacts with the world model in real time via mouse and keyboard to perform the same task from the same initial image. Dreamer 4 is the first world model to accurately predict the object interactions and game mechanics of placing the blocks in the correct shape. In contrast, previous Minecraft world models degrade visually, change the held item, and hallucinate structures that the player never built. The Dreamer 4 world model allows the player to accomplish the task (✅) while Oasis and Lucid do not (❌).

We compare the following agents:

- **VPT (finetuned)** The strongest Minecraft agent for mouse and keyboard control in the literature[15]. The VPT paper presents two unconditioned behavioral cloning policies in the offline setting, both trained on 270K hours of synthetically annotated YouTube gameplay videos, and one further finetuned on a filtered subset of "early game" data. We use the finetuned policy because it significantly outperforms the pretraining policy.
- **BC (notask)** Behavioral cloning from scratch using multi-token prediction (MTP), without task conditioning. While VPT uses the contractor data to train an action labeler and trains the policy on annotated YouTube videos, our BC agent trains directly and only on the relevant subset of the contractor actions. This agent is not task conditioned, making it directly comparable to VPT.
- **BC** Behavioral cloning from scratch on the filtered contractor dataset with task conditioning. This agent is trained on the same filtered contractor dataset as BC (notask) but the additional task input makes it steerable. When evaluating in the environment, the prompt sequence guides the agent through intermediate tasks towards mining diamonds.
- **VLA (Gemma 3)** Following the VLA recipe[41,42], we train a behavioral cloning policy by finetuning the vision-language model Gemma 3[43] on the relevant sequences using MTP. Gemma 3 has been pretrained using substantially more compute and data than our other models, including native pretraining on images for visual perception, making it a strong baseline.
- **WM+BC** The behavioral cloning policy of Dreamer 4, before applying imagination training. This policy is initialized from the world model pretrained on the full contractor dataset. It then undergoes agent finetuning using behavioral cloning, reward modeling, and dynamics losses.
- **Imagination RL** The full Dreamer 4 agent produced by finetuning the WM+BC agent through reinforcement learning in imagination, which we refer to as imagination training. Despite performing on-policy reinforcement learning inside of the world model, there is no actual environment interaction, making it an offline method.

Model	Parameters	Resolution	Context	FPS	Success
MineWorld	1.2B	384×224	0.8s	2	—
Lucid-v1	1.1B	640×360	1.0s	44	0/16
Oasis (small)	500M	640×360	1.6s	20	0/16
Oasis (large)	—	360×360	1.6s	~5	5/16
Dreamer 4	2B	640×360	9.6s	21	**14/16**

Table 1: Comparison of Minecraft world models. Dreamer 4 is the first world model to accurately simulate a wide range of object interactions and game mechanics in Minecraft. Moreover, Dreamer 4 pushes the limits of context length compared to previous models by 6×, while maintaining real-time interactive inference. We measure the inference speed of each model on a single H100 GPU, and translate the inference speed for the proprietary large Oasis model based on public information. The Minecraft dataset is recorded at 20 FPS, matching the update rate of the game.

Agent performance Figure 3 compares the agent performance on the diamond task. We report the success rates for several relevant items leading up to diamonds that are listed in Table 5. VPT (finetuned) progresses up to sticks, which it achieves 53% of the time. It also collects a small amount of stone, iron ore, and iron ingots through edge cases, such as exploding Creeper mobs and loot chests. Using the contractor actions directly instead of annotating YouTube videos, our modern BC baseline achieves higher performance than VPT (finetuned). VLA (Gemma 3) shows that initializing the policy from a pretrained models offers significant benefits, progressing up to the iron pickaxe with a success rate of 11%. Dreamer 4 achieves high success rates of over 90% up to the stone pickaxe, a success rate of 29% for the iron pickaxe, and obtains diamonds in 0.7% of episodes. Imagination training shows stronger improvements over the behavior cloning agents the more challenging the milestone is. Figure 4 compares additional agents on the diamond task, showing that the world model representations outperform the general representations of Gemma 3 for behavioral cloning. This indicates that video prediction implicitly learns an understanding of the world that is also useful for decision making. Finally, imagination training consistently improves not only the success rates but also makes the policy more efficient so that it reaches the milestones faster.

4.2. Human Interaction

To evaluate its ability to predict complex interactions, we train Dreamer 4 on the Minecraft VPT dataset[15] and compare its generations to previous world models on this dataset. For this evaluation, a human player tries to accomplish tasks by playing inside of the world model, as shown in Figure 5. The human player receives the task description and the world model is initialized to a start frame for the task. We select a diverse set of tasks that cover a broad range of object interactions and game mechanics. The tasks include digging a pit, building a wall, chopping a tree, placing and riding a boat, looking away and back at objects, interacting with crafting benches and furnaces, and more. We compare Dreamer 4 to the world models Oasis[46], Lucid-v1[47], and MineWorld[48]. Note that we cannot directly compare to Genie 3[7], because it only supports camera actions and one generic “interact” button, whereas Minecraft requires a more general mouse and keyboard action space. Table 1 summarizes the compared models. Full results are given in Figures 12 to 14.

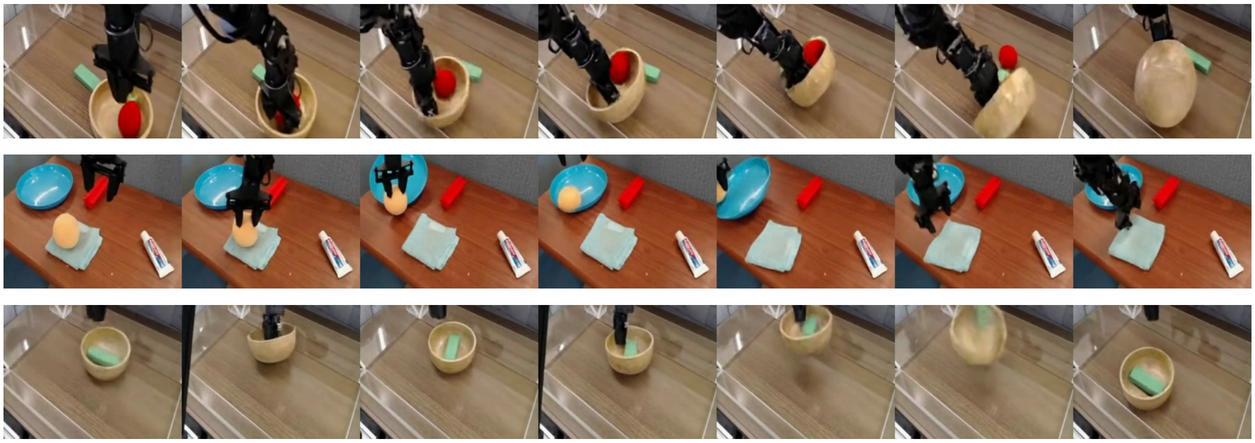


Figure 6: Robotics generations for counterfactual actions. Dreamer 4 learns an accurate real-time simulator of the environment, allowing human operators to control the imagined robot to pick up objects, flip over a bowl, press a ball onto a plate, move a towel, and throw a bowl.

Inference speed We measure the inference speed of each model on a single H100 GPU. Dreamer 4 and Lucid-v1 achieve real-time interactive inference by exceeding the 20 FPS of the Minecraft physics engine[‡] and the VPT dataset[15]. Dreamer 4 has a substantially longer context of 9.6 seconds compared to the 0.8–1.6 seconds of prior models. Oasis comes in two sizes, a 500M parameter version with open weights and a larger model of unknown size that is playable on the project website. The small model achieves 20 FPS on one H100. The large model is hosted on multiple H100s for interaction online and we estimate its inference speed on a single H100 around 5 FPS based on public information. MineWorld achieves 2 FPS with their parallel decoding approach and is even slower without. Additionally, parallel decoding requires knowing actions in advance, which is also required in the provided user interface. Thus, it does not support real-time interactions and we cannot evaluate it on our tasks, which require many hundreds of actions.

Complex interactions Video generations of human players attempting all tasks inside of the world models are shown in Figures 12 to 14. Lucid-v1 does not allow completing the tasks, with generations diverging or object interactions being ignored. The large Oasis model allows performing 5 out of 16, such as placing torches, filling a window with glass panes, and opening a door. However, it fails at building tasks because after placing a few blocks, it quickly hallucinates large structures into the world. This "autocompletion" failure mode reflects a lack of understanding of the game mechanics. We did not evaluate MineWorld because of its lack of interactive inference. Dreamer 4 achieves 14 out of 16 tasks, accurately generating complex interactions and game mechanics such as switching items, placing and breaking blocks, fighting monsters, placing and riding boats, entering portals, and more. Its temporal consistency is limited to a context of 9.6 seconds, albeit substantially longer than previous models. While it correctly generates the interfaces for inventory, crafting, and furnaces and predicts most mouse movement, inventory items are sometimes unclear or change over time, leaving room for future improvements.

As a step towards testing the applicability of Dreamer 4 to real world video, we also train the world model on a robotics dataset[49]. In Figure 6, we observe accurate physics and counterfactual interactions with real world objects, overcoming the causal confusion of existing video models. Additional details are included in the supplementary material.

[‡] https://minecraft.fandom.com/wiki/Tick

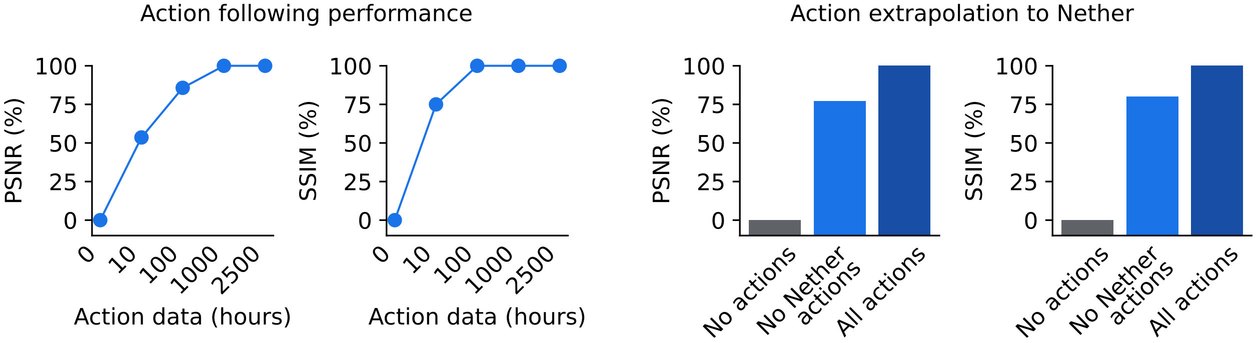


Figure 7: Action generalization. **(left)** Dreamer 4 learns accurate action conditioning from 2500 hours of video with only 100 hours of paired actions. It achieves over 80% of the action-conditioned generation accuracy, normalized within the range of training without any actions and using all actions. **(right)** When learned with only actions of the Minecraft Overworld, the action conditioning generalizes to the Nether and End dimensions of the game that are only seen in unlabeled videos. These environments are distinct from the Overworld in their textures, blocks, and items. The results indicate that Dreamer 4 learns action conditioning from small amounts of action data that generalize broadly, paving the way toward learning simulators from diverse unlabeled web videos.

4.3. Action Generalization

One promise of world models is to leverage diverse unlabeled videos to teach agents about the world. For example, a world model could learn general physics and object interactions from web videos where actions are not available. In this section, we investigate the amount of paired videos with actions that are needed for grounding an embodiment into the Dreamer 4 world model. Intuitively, the world model has to learn a broader distribution of possible outcomes when actions are missing, and can narrow the distribution down when actions are provided. Moreover, we investigate how well the action conditioning generalizes, not just within the same distribution, but also out of distribution to parts of the world specifically held out. To measure the accuracy of the action conditioning, we compare action-conditioned multi-step generations to ground truth videos on the holdout set. We report PSNR and SSIM for 16-step generations given 320 frames of context.

Amount of actions To understand the amount of video with actions needed to learn action conditioning, we train Dreamer 4 on all 2541 hours of videos in the VPT dataset, but only provide actions for a small subset of the videos. When actions are unavailable, the dynamics model is conditioned on a learned embedding, as described in Section 3.2. We train Dreamer 4 models with 0, 10, 100, 1000, and 2541 hours of actions. The available actions are the first in sequential order of the dataset, corresponding to fewer unique worlds and players than random shuffling would yield. Figure 7 show the quality of the action conditioning compared to training with no actions at all to training with all actions. With only 10 hours of actions, Dreamer 4 achieves 53% PSNR and 75% SSIM compared to a model trained with all actions. With 100 hours of actions, the performance increases further to 85% PSNR and 100% SSIM. This result demonstrates that world models absorb the majority of their knowledge from unlabeled videos, and require only a small amount of actions.

Action extrapolation In principle, world models may not only learn action ground from a few actions but also generalize their action conditioning to completely new scenarios. In the future, this could allow world models to absorb general knowledge from diverse web videos to simulate agents in diverse environments. We perform, to the best of our knowledge, the first controlled evaluation

Model	**Train step seconds**	**Inference FPS (↑)**	**Quality FVD (↓)**
Diffusion Forcing Transformer	9.8	0.8	306
+ Fewer sampling steps ($K = 4$)	9.8	9.1	875
+ Shortcut model	9.8	9.1	329
+ X-Prediction	9.8	9.1	326
+ X-Loss	9.8	9.1	151
+ Ramp weight	9.8	9.1	102
+ Alternating batch lengths	1.5	9.1	80
+ Long context every 4 layers	0.6	18.9	70
+ GQA	0.5	23.2	71
+ Time factorized long context	0.4	30.1	91
+ Register tokens	0.5	28.9	91
+ More spatial tokens ($N_z = 128$)	0.8	25.7	66
+ More spatial tokens ($N_z = 256$)	1.7	21.4	57

Table 2: Cascade of model design choices. Dreamer 4 is based on a shortcut forcing objective and an efficient transformer architecture, combining a range of known techniques to achieve accurate and fast interleaved generation. Starting from a naive diffusion forcing transformer with $N_z = 64$ spatial tokens and $K = 64$ sampling steps, we apply the objective and architecture modifications, and increase the number of spatial tokens once feasible. Inference speed measured on a single H100 GPU. The resulting world model achieves high model capacity and inference efficiency.

of this hypothesis. For this, we carefully split the VPT dataset into one portion that only contains videos of the Overworld and another portion that only contains the other two game dimensions, the Nether and End. Where the Overworld contains forests, deserts, oceans, and more, the Nether and End feature substantially different and unique visuals. The Nether is an underworld filled with lava and red blocks and the End is filled with yellow blocks and black towers unseen in the Overworld. We train Dreamer 4 on videos of both datasets but only provide actions for the Overworld. We then perform an action-conditioned evaluation of the resulting model on the Nether and End, for which it has never seen any actions. In a prior experiment, we observed that training only on the Overworld portion without any Nether videos results in poor generation scores for Nether start frames. Figure 7 reports the relative performance compared to training without any or with all actions. Surprisingly, the world model achieves 76% of the PSNR and 80% of the SSIM of the model trained with all actions. This demonstrates that the action conditioning of world models can generalize to scenarios known only from unlabeled videos.

4.4. Model Design

World models require high model capacity to predict complex object interactions and fast inference to support imagination training and human interaction for inspection. Moreover, interactive inference prompts different design choices compared to typical video models to enable fast generation of individual frames and prevent accumulating errors. In this section, we ablate the objective and architecture decisions of Dreamer 4 by applying a cascade of improvements to a naive diffusion forcing transformer baseline. To evaluate each model, we train for 48 hours and then generate 1024 videos of 384 frames (2̃0 seconds) each without any context, with interactive actions chosen by a fixed behavioral cloning policy. We then split the resulting videos into 16 frame chunks to compute the Frechét Video Distance (FVD)[50] to the holdout dataset.

Table 2 shows the progression of models and their generation quality and speed. We start with a standard block-causal transformer with dense attention and diffusion forcing with velocity parameterization. We target 20 FPS interactive inference on a single H100 GPU, matching the tick rate of Minecraft and the framerate of the VPT dataset. With 64 sampling steps per frame, the baseline falls short of real-time generations with only 0.8 FPS on one H100 GPU, while 4 sampling steps achieve 9.1 FPS but result in poor quality. Shortcut models[21] nearly recover the original visual quality with only 4 sampling steps. We then parameterize the model to make x-space predictions, compute the loss in x-space, and apply the ramp loss weight. These changes substantially improve generation quality over traditional v-space prediction, especially for long generations. We hypothesize that the prediction targets in x-space are more structured and thus reduce the risk of high-frequency errors that accumulate over time. Figure 8 compares the visual quality of shortcut forcing to diffusion forcing—both using the x-space loss with ramp weight—for a wider range of sampling steps.

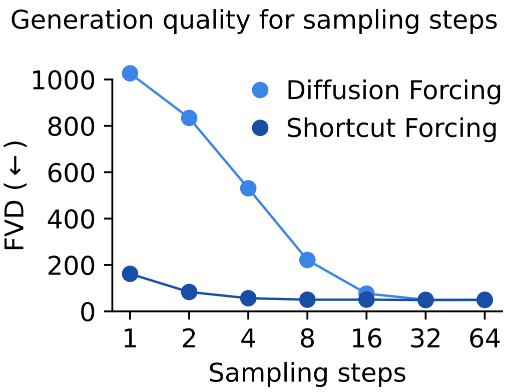


Figure 8: Generation quality of shortcut forcing compared to diffusion forcing. Shortcut forcing with only 4 sampling steps approaches the quality of diffusion forcing with 64 steps, resulting in 16× faster generations.

Training on alternating batch lengths is similar to progressive training and speeds up learning while allowing to generate long videos for inspection throughout training. Using temporal attention only every 4 layers not only speeds up training and inference[39] but also improves generation quality, possibly because of the inductive bias of spatial attention that focuses computation on the current frame. GQA further accelerates generations without degrading performance. Switching the long context layers from dense to time-only attention accelerates inference at a mild quality cost, and prepares us to increase the overall number of tokens subsequently. While the register tokens do not improve FVD measurably, we qualitatively notice that they improve temporal consistency. After these changes, training and inference is fast enough to increase model capacity through more spatial tokens, improving predictions of complex interactions. The complete model achieves an FVD of 57 compared to 306 for the naive diffusion forcing transformer baseline and 124 for the complete architecture with v-space prediction and losses.

Agent	**Inputs**	**Actions**	**Data (hours)**		
			Offline	Web	Online
Dreamer 3	64×64, inventory	keyboard, camera, abstract crafting	—	—	1.4K
VPT (RL)	128×128	keyboard, mouse	2.5K	270K	194K
VPT (BC)	128×128	keyboard, mouse	2.5K	270K	—
Dreamer 4	360×640	keyboard, mouse	2.5K	—	—

Table 3: Comparison of experimental setups for different Minecraft agents. Dreamer 4 learns purely from offline experience and requires 100× less data than previous keyboard and mouse agents. See Figure 10 for examples of input images.

5. Related Work

Minecraft agents Obtaining diamonds in the video game Minecraft has been a focus of intelligent agent research[51–54], a long-horizon task that requires gathering resources and crafting tools through thousands of low-level actions in complex procedurally generated 3D worlds. Different setups have been used in the literature, with Table 3 summarizing the ones relevant to our work. VPT[15] collects 2.5K hours of contractor gameplay and uses it to annotate 270K hours of web videos with synthetic mouse and keyboard actions. Behavioral cloning on this large offline dataset and targeted finetuning on "early-game" data yields a policy that occasionally obtains wooden pickaxes. Following up with 194K hours of online reinforcement learning results in a policy that obtains diamonds and diamond pickaxes. Dreamer 3[1] learns to collect diamonds from scratch from 1.4K hours of online interaction, without any human data. It uses the MineRL competition action space that includes abstract crafting actions. Other works have explored novel algorithms for easier tasks[55–58]. In this paper, we train an agent to achieve diamonds purely from the 2.5K hour contractor dataset, without any online interaction. We use low-level mouse and keyboard actions and high-resolution image inputs.

World model agents Learning behaviors based on learned models of the environment has been explored for a long time[59–61]. Visual Foresight[62], the work by Ha and Schmidhuber[63], and PlaNet[64] achieved world models accurate enough for planning from pixels. With Dreamer, world models have become the state-of-the-art approach for solving control problems with high-dimensional inputs, outperforming model-free reinforcement learning in robustness, efficiency, and final performance[1,65,66]. World models based on transformers or diffusion objectives have demonstrated high data efficiency for discrete control[67–69]. However, these world models have been limited in model capacity, restricting their applicability to relatively simple simulated environments.

Scalable world models Larger world models have been shown to simulate more complex data distributions[70]. World models like Genie 3 generate highly diverse scenes and simulate camera movement and simple interactions[7,9,10,12]. PlayerOne[8] and PEVA[11] condition on more detailed human movement. Oasis[46], Lucid[47], and MineWorld[48] learn Minecraft simulators from mouse and keyboard inputs. Whereas Oasis captures simple game mechanics and achieves real-time inference on specialized hardware, Lucid's predictions diverge quickly and MineWorld is slower than real time. GameNGen[71] finetunes Stable Diffusion into a playable simulator of a level of the game Doom. DIAMOND[4] learns a simulator of a level of the game CS GO, achieving short-term predictions. GAIA[72,73] generates driving scenarios from real world data. However, these world models still struggle to predict complex object interactions precisely enough for imagination training.

Fast generation Enabling fast and accurate generations has been a long-standing challenge in generating modeling. MaskGit generates discrete tokens in parallel to accelerate sampling over autoregressive models[74]. Diffusion models are often distilled to enable sampling using fewer forward passes[75,76], which requires two training phases. Consistency models learn straight paths to enable fast image generation but require careful schedules[77–79], with successful applications to video generation[80]. Shortcut models[21] condition flows on both noise level and step size, producing fast generative models in one training phase and without schedules. Mean Flow[81] extends the idea of conditioning on the step size to a continuous time formulation. Efficient and sparse architectures are used for video generation because of the large number of tokens[38,82,83]. These models do not support interactive inference and many of the techniques are complimentary to our work.

6. Discussion

We present Dreamer 4, a scalable agent that learns to solve challenging control tasks by imagination training inside of a fast and accurate world model. Dreamer 4 is the first agent to obtain diamonds in Minecraft purely from offline data, without online interaction. This achievement demonstrates its learning successful long-horizon strategies in complex environments. Learning purely from offline datasets enables applications where online interaction is impractical or unsafe.

The world model of Dreamer 4 rests on a shortcut forcing objective and an efficient transformer architecture to predict complex object interactions while supporting interactive inference in real time on a single GPU. We demonstrate that it significantly outperforms previous world models on Minecraft, where it accurately predicts a wide range of game mechanics from mouse and keyboard inputs. However, the world model is far from a full clone of the game, especially due to short memory and imprecise inventory predictions, leaving Minecraft as an ideal benchmark for future world model and agent research. We also show that Dreamer 4 learns accurate action conditioning from only a small amount of videos with actions, allowing it to absorb the majority of its knowledge from diverse unlabeled videos.

Promising future directions include pretraining on general internet videos, integrating long-term memory into the world model and agent, incorporating language understanding, leveraging small amounts of corrective online data, and automatically discovering goals to break up long tasks. Dreamer 4 offers a reliable and performant starting point for these explorations.

References

[1] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pages 1–7, 2025.

[2] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.

[3] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

[4] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos J Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.

[5] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

[6] Matteo Hessel, Ivo Danihelka, Fabio Viola, Arthur Guez, Simon Schmitt, Laurent Sifre, Theophane Weber, David Silver, and Hado Van Hasselt. Muesli: Combining improvements in policy optimization. In *International Conference on Machine Learning*, pages 4214–4226. PMLR, 2021.

[7] Philip J. Ball, Jakob Bauer, Frank Belletti, Bethanie Brownfield, Ariel Ephrat, Shlomi Fruchter, Agrim Gupta, Kristian Holsheimer, Aleksander Holynski, Jiri Hron, Christos Kaplanis, Marjorie Limont, Matt McGill, Yanko Oliveira, Jack Parker-Holder, Frank Perbet, Guy Scully, Jeremy Shar, Stephen Spencer, Omer Tov, Ruben Villegas, Emma Wang, Jessica Yung, Cip Baetu, Jordi Berbel, David Bridson, Jake Bruce, Gavin Buttimore, Sarah Chakera, Bilva Chandra, Paul Collins, Alex Cullum, Bogdan Damoc, Vibha Dasagi, Maxime Gazeau, Charles Gbadamosi, Woohyun Han, Ed Hirst, Ashyana Kachra, Lucie Kerley, Kristian Kjems, Eva Knoepfel, Vika Koriakin, Jessica Lo, Cong Lu, Zeb Mehring, Alex Moufarek, Henna Nandwani, Valeria Oliveira, Fabio Pardo, Jane Park, Andrew Pierson, Ben Poole, Helen Ran, Tim Salimans, Manuel Sanchez, Igor Saprykin, Amy Shen, Sailesh Sidhwani, Duncan Smith, Joe Stanton, Hamish Tomlinson, Dimple Vijaykumar, Luyu Wang, Piers Wingfield, Nat Wong, Keyang Xu, Christopher Yew, Nick Young, Vadim Zubov, Douglas Eck, Dumitru Erhan, Koray Kavukcuoglu, Demis Hassabis, Zoubin Gharamani, Raia Hadsell, Aäron van den Oord, Inbar Mosseri, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 3: A new frontier for world models. https://deepmind.google/discover/blog/genie-3-a-new-frontier-for-world-models/, 2025.

[8] Yuanpeng Tu, Hao Luo, Xi Chen, Xiang Bai, Fan Wang, and Hengshuang Zhao. Playerone: Egocentric world simulator. *arXiv preprint arXiv:2506.09995*, 2025.

[9] Xianglong He, Chunli Peng, Zexiang Liu, Boyang Wang, Yifan Zhang, Qi Cui, Fei Kang, Biao Jiang, Mengyin An, Yangyang Ren, et al. Matrix-game 2.0: An open-source, real-time, and streaming interactive world model. *arXiv preprint arXiv:2508.13009*, 2025.

[10] Wenqiang Sun, Fangyun Wei, Jinjing Zhao, Xi Chen, Zilong Chen, Hongyang Zhang, Jun Zhang, and Yan Lu. From virtual games to real-world play. *arXiv preprint arXiv:2506.18901*, 2025.

[11] Yutong Bai, Danny Tran, Amir Bar, Yann LeCun, Trevor Darrell, and Jitendra Malik. Whole-body conditioned egocentric video prediction. *arXiv preprint arXiv:2506.21552*, 2025.

[12] Yan Team. Yan: Foundational interactive video generation. *arXiv preprint arXiv:2508.08601*, 2025.

[13] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[14] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

[15] Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35: 24639–24654, 2022.

[16] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[18] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[19] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[20] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[21] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

[22] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[23] 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.

[24] Hao Chen, Yujin Han, Fangyi Chen, Xiang Li, Yidong Wang, Jindong Wang, Ze Wang, Zicheng Liu, Difan Zou, and Bhiksha Raj. Masked autoencoders are effective tokenizers for diffusion models. In *Forty-second International Conference on Machine Learning*, 2025.

[25] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

[26] Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36:65484–65516, 2023.

[27] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

[28] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[29] Abbas Abdolmaleki, Bilal Piot, Bobak Shahriari, Jost Tobias Springenberg, Tim Hertweck, Rishabh Joshi, Junhyuk Oh, Michael Bloesch, Thomas Lampe, Nicolas Heess, et al. Preference optimization as probabilistic inference. *arXiv e-prints*, pages arXiv–2410, 2024.

[30] Claude E Shannon. A mathematical theory of communication. *Bell system technical journal*, 27 (3):379–423, 1948.

[31] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[32] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

[33] J Su, H Zhang, X Li, J Zhang, and Y RoFormer Li. Enhanced transformer with rotary position embedding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP), Association for Computational Linguistics, Online*, pages 1–6, 2021.

[34] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[35] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR, 2023.

[36] Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[37] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[38] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

[39] Ajit Singh. Meta llama 4: The future of multimodal ai. *Available at SSRN 5208228*, 2025.

[40] Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[41] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[42] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. pi0.5: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[43] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[44] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

[45] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506, 2020.

[46] Decart and Etched. Oasis: A universe in a transformer. https://www.decart.ai/articles/oasis-interactive-ai-video-game-model, 2024.

[47] Rami Seid and Alberto Hojel. Lucid v1: Real-tiem latent world models. *International Journal of Current Research in Science, Engineering & Technology*, 2024.

[48] Junliang Guo, Yang Ye, Tianyu He, Haoyu Wu, Yushu Jiang, Tim Pearce, and Jiang Bian. Mineworld: a real-time and open-source interactive world model on minecraft. *arXiv preprint arXiv:2504.08388*, 2025.

[49] Zhiyuan Zhou, Pranav Atreya, Abraham Lee, Homer Walke, Oier Mees, and Sergey Levine. Autonomous improvement of instruction following skills via foundation models. *arXiv preprint arXiv:2407.20635*, 2024.

[50] Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. Fvd: A new metric for video generation. *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2019.

[51] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *IJCAI*, pages 4246–4247. Citeseer, 2016.

[52] William H Guss, Cayden Codel, Katja Hofmann, Brandon Houghton, Noboru Kuno, Stephanie Milani, Sharada Mohanty, Diego Perez Liebana, Ruslan Salakhutdinov, Nicholay Topin, et al. The minerl competition on sample efficient reinforcement learning using human priors. *arXiv e-prints*, pages arXiv–1904, 2019.

[53] Anssi Kanervisto, Stephanie Milani, Karolis Ramanauskas, Nicholay Topin, Zichuan Lin, Junyou Li, Jianing Shi, Deheng Ye, Qiang Fu, Wei Yang, et al. Minerl diamond 2021 competition: Overview, results, and lessons learned. *NeurIPS 2021 Competitions and Demonstrations Track*, pages 13–28, 2022.

[54] Ingmar Kanitscheider, Joost Huizinga, David Farhi, William Hebgen Guss, Brandon Houghton, Raul Sampedro, Peter Zhokhov, Bowen Baker, Adrien Ecoffet, Jie Tang, et al. Multi-task curriculum learning in a complex, visual, hard-exploration domain: Minecraft. *arXiv preprint arXiv:2106.14876*, 2021.

[55] Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *Advances in Neural Information Processing Systems*, 36:69900–69929, 2023.

[56] Shaofei Cai, Bowei Zhang, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Groot: Learning to follow instructions by watching gameplay videos. *arXiv preprint arXiv:2310.08235*, 2023.

[57] Enshen Zhou, Yiran Qin, Zhenfei Yin, Yuzhou Huang, Ruimao Zhang, Lu Sheng, Yu Qiao, and Jing Shao. Minedreamer: Learning to follow instructions via chain-of-imagination for simulated-world control. *arXiv preprint arXiv:2403.12037*, 2024.

[58] Juan José Nieto, Roger Creus, and Xavier Giro-i Nieto. Unsupervised skill-discovery and skill-learning in minecraft. *arXiv preprint arXiv:2107.08398*, 2021.

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

[60] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[61] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

[62] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[63] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[64] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

[65] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[66] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[67] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

[68] Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. *arXiv preprint arXiv:2303.07109*, 2023.

[69] Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. *Advances in Neural Information Processing Systems*, 36:27147–27166, 2023.

[70] Jiwen Yu, Yiran Qin, Haoxuan Che, Quande Liu, Xintao Wang, Pengfei Wan, Di Zhang, Kun Gai, Hao Chen, and Xihui Liu. A survey of interactive generative video. *arXiv preprint arXiv:2504.21853*, 2025.

[71] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. *arXiv preprint arXiv:2408.14837*, 2024.

[72] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

[73] Lloyd Russell, Anthony Hu, Lorenzo Bertoni, George Fedoseev, Jamie Shotton, Elahe Arani, and Gianluca Corrado. Gaia-2: A controllable multi-view generative world model for autonomous driving. *arXiv preprint arXiv:2503.20523*, 2025.

[74] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325, 2022.

[75] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[76] Akio Kodaira, Tingbo Hou, Ji Hou, Masayoshi Tomizuka, and Yue Zhao. Streamdit: Real-time streaming text-to-video generation. *arXiv preprint arXiv:2507.03745*, 2025.

[77] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *International Conference on Machine Learning*, 2023.

[78] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

[79] Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024.

[80] Xiang Wang, Shiwei Zhang, Han Zhang, Yu Liu, Yingya Zhang, Changxin Gao, and Nong Sang. Videolcm: Video latent consistency model. *arXiv preprint arXiv:2312.09109*, 2023.

[81] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

[82] Xuanlei Zhao, Xiaolong Jin, Kai Wang, and Yang You. Real-time video generation with pyramid attention broadcast. *arXiv preprint arXiv:2408.12588,* 2024.

[83] Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric Xing, and Hao Zhang. Vsa: Faster video diffusion with trainable sparse attention. *arXiv preprint arXiv:2505.13389*, 2025.

[84] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European conference on computer vision (ECCV)*, pages 720–736, 2018.

A. Datasets

Minecraft VPT We use the OpenAI VPT dataset of contractor gameplay[15] and combine the available subsets 6–10, resulting in 2541 hours of gameplay. We split the dataset into 90% training and 10% evaluation data, ensuring that the splits do not share any of the same underlying 5-min recording chunks. We encode keyboard actions as a vector of binary variables and process the mouse actions as in VPT by μ-law encoding, discretizing into 11 bins per coordinate, and enumerating all $11 \times 11 = 121$ combinations to obtain a categorical variable. The image resolution is 360×640 and the framerate is 20 FPS. We zero pad the frames to 384×640 and then patchify with patch size 16×16 into 960 tokens. We reshape the $(N_{\mathrm{b}} = 512) \times (D_{\mathrm{b}} = 16)$ bottleneck of the tokenizer to $(N_{\mathrm{z}} = 256) \times 32$ for the dynamics model. We train the dynamics model with $N_{\mathrm{z}} = 256$ spatial tokens, context length $C = 192$, and batch lengths $T_1 = 64$ and $T_2 = 256$.

Minecraft Overworld and Nether split To study out-of-distribution generalization of action conditioning in Dreamer 4, we carefully split the Minecraft dataset into videos of the Overworld versus the Nether dimension. We also include the End dimension into the Nether portion of the dataset. Both the Nether and the End feature unique visuals, blocks, and terrain shapes compared to the Overworld. The Overworld includes natural landscapes with forests, deserts, oceans, and more, whereas the Nether is underworld-themed with red blocks and lava and the End is a space-themed region. To separate the dataset, we want to ensure no leakage from players entering the Nether/End dimensions and bringing blocks from there back to the Overworld. For this reason, we exclude the VPT 6 and 7 subsets, which contain long free play. We then assigned each 5 min recording of the remaining dataset to either the Overworld or the Nether/End portion based on item events that are provided by the dataset. Whenever a Nether/End item was interacted with, we assign that video to the Nether/End split. This ensures that the Overworld split contains no Nether/End episodes, whereas the Nether/End split can sometimes contain some Overworld episodes, although this was rare in practice. We manually investigated the Overworld split obtained by this strategy and found no Nether/End trajectories in it.

SOAR Robotics The SOAR dataset[49] contains teleoperated demonstrations and online trajectories of a reinforcement learning policy, thus covering both successes and failures. We split the dataset into 90% training and 10% evaluation data. The dataset contains a total of 180 hours of videos with 7D relative end-effector actions. The image resolution is 256×256 and the framerate is 5 FPS. We patchify with patch size 16×16 into 256 tokens. We train the dynamics model with $N_{\mathrm{z}} = 512$ spatial tokens, context length $C = 96$, and batch lengths $T_1 = 32$ and $T_2 = 128$.

Epic Kitchens The Epic Kitchens 100 dataset[84] contains 100 hours of video from the first-person perspective of humans across 45 kitchens. The test set contains different tasks performed in the same kitchens. We use the dataset at 256×256 resolution and 10 FPS. We patchify with patch size 16×16 into 256 tokens. We train the dynamics model with $N_{\mathrm{z}} = 512$ spatial tokens, context length $C = 96$, and batch lengths $T_1 = 32$ and $T_2 = 128$.

B. Kitchen Generations

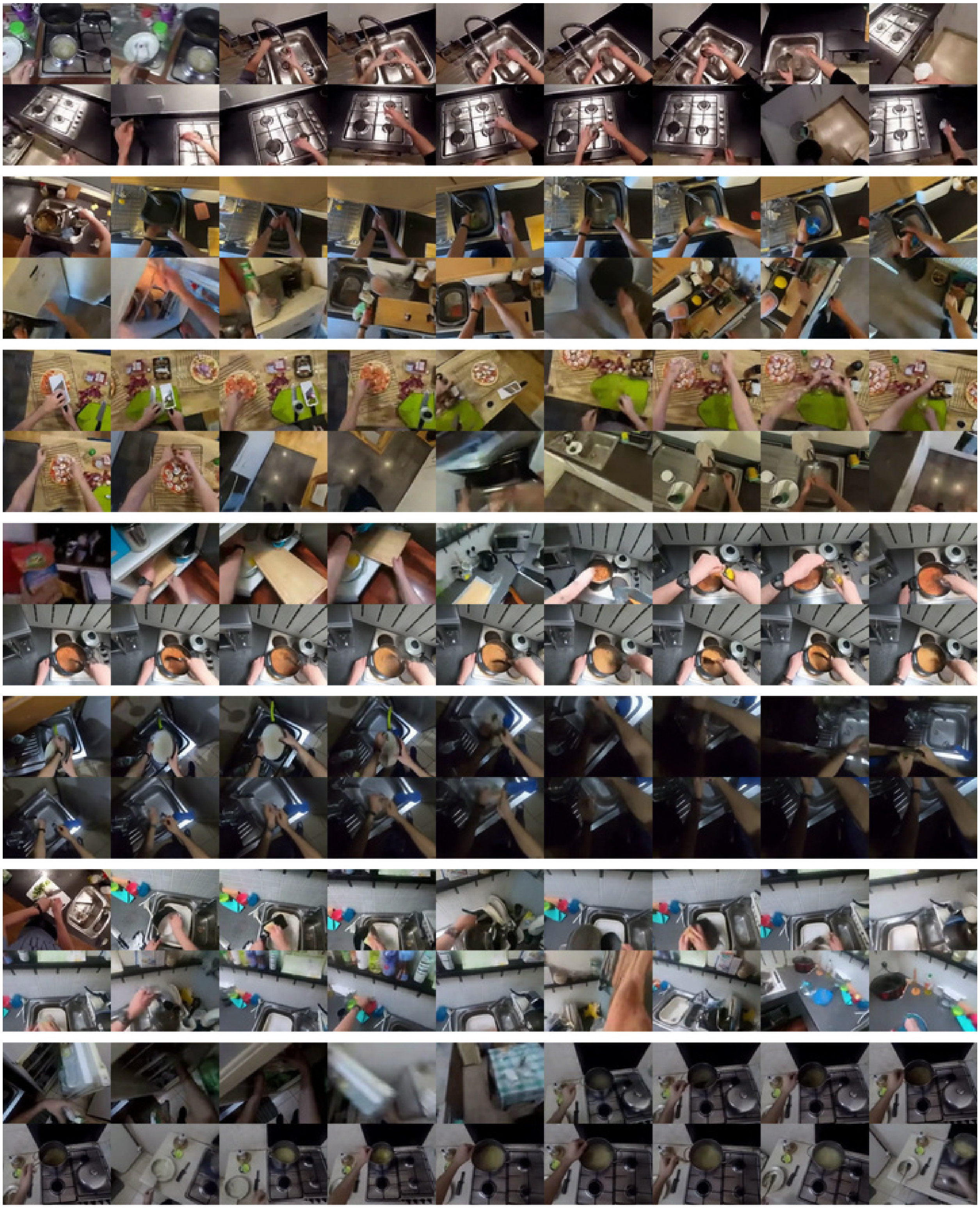


Figure 9: Kitchen video generations starting from holdout context.

C. Minecraft Tasks

Index	Task
1	mine_log
2	mine_cobblestone
3	mine_iron_ore
4	mine_coal
5	mine_diamond
6	craft_planks
7	craft_stick
8	craft_crafting_table
9	craft_furnace
10	craft_iron_ingot
11	craft_wooden_pickaxe
12	craft_stone_pickaxe
13	craft_iron_pickaxe
14	open_crafting_table
15	open_furnace
16	place_crafting_table
17	place_furnace
18	use_wooden_pickaxe
19	use_stone_pickaxe
20	use_iron_pickaxe

Table 4: Taskset for training the multi-task agent.

Icon	Item
	Log
	Planks
	Stick
	Crafting table
	Wooden pickaxe
	Cobblestone
	Stone pickaxe
	Iron ore
	Furnace
	Iron ingot
	Iron pickaxe
	Diamond

Table 5: Milestone items used for measuring progress during evaluation.

Task	Count
mine_log	10
craft_planks	20
craft_crafting_table	1
place_crafting_table	1
craft_stick	4
craft_wooden_pickaxe	1
use_wooden_pickaxe	1
mine_cobblestone	3
craft_planks	4
craft_crafting_table	1
place_crafting_table	1
craft_stick	4
craft_stone_pickaxe	1
use_stone_pickaxe	1
mine_iron_ore	3
mine_cobblestone	8
craft_planks	4
craft_crafting_table	1
place_crafting_table	1
craft_furnace	1
craft_iron_ingot	3
craft_stick	2
craft_iron_pickaxe	1
use_iron_pickaxe	1
mine_diamond	∞

Table 6: Prompt sequence for evaluation.

D. Offline Diamond Challenge

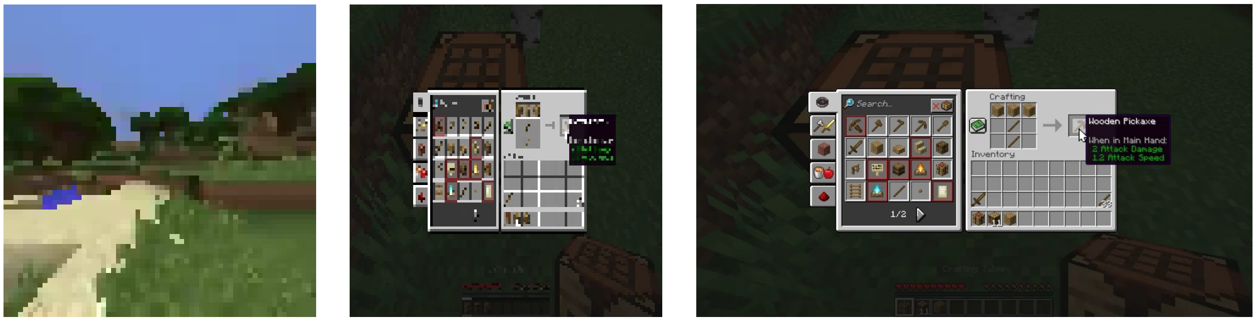
Item	VPT (pretrained)	VPT (finetuned)	BC (notask)	WM+BC (notask)	BC	VLA (Gemma 3)	WM+BC	Dreamer 4
Log	81.9	84.3	71.4	92.6	**97.3**	**98.5**	**99.6**	**99.1**
Planks	30.6	65.3	68.6	91.6	**95.7**	**98.3**	**99.6**	**98.9**
Crafting table	1.7	4.7	63.8	90.6	93.5	**97.2**	**99.1**	**98.5**
Stick	30.3	52.6	62.4	90.1	**95.0**	**97.7**	**98.9**	**98.7**
Wooden pickaxe	0.0	0.0	33.8	77.3	86.5	**94.1**	**97.3**	**96.6**
Cobblestone	4.8	6.9	32.0	77.4	83.9	91.6	**97.2**	**95.9**
Stone pickaxe	0.0	0.0	8.8	38.4	53.8	76.7	**89.4**	**90.1**
Iron ore	0.1	0.1	3.6	22.0	26.5	46.3	62.9	**66.7**
Furnace	0.0	0.0	4.0	28.0	16.2	42.4	51.1	**58.1**
Iron ingot	0.1	0.1	0.2	1.2	4.3	22.5	27.8	**39.5**
Iron pickaxe	0.0	0.0	0.0	0.1	0.6	11.2	16.9	**29.0**
Diamond	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**0.7**

Table 7: Success rates for each milestone item averaged over 1000 evaluation episodes. Scores within 5% of the highest recorded score are highlighted in bold.

Item	VPT (pretrained)	VPT (finetuned)	BC (notask)	WM+BC (notask)	BC	VLA (Gemma 3)	WM+BC	Dreamer 4
Log	9.1	6.3	11.9	5.4	1.8	2.2	1.2	**0.9**
Planks	25.2	14.2	12.2	5.9	4.3	3.4	2.1	**2.0**
Stick	32.0	24.0	13.3	6.7	6.4	5.0	**3.1**	**2.9**
Crafting table	41.4	27.5	17.1	8.0	9.5	7.2	**4.6**	**4.4**
Wooden pickaxe	—	—	18.8	11.6	11.4	9.8	5.7	**5.0**
Cobblestone	—	—	19.6	12.7	13.3	12.1	6.7	**5.6**
Stone pickaxe	—	—	23.5	15.7	15.8	14.5	8.9	**6.7**
Iron ore	—	—	28.9	17.5	20.9	23.5	14.3	**9.9**
Furnace	—	—	29.4	19.7	24.5	24.7	16.1	**11.0**
Iron ingot	—	—	—	30.5	28.8	30.8	17.2	**12.4**
Iron pickaxe	—	—	—	—	29.1	31.1	17.0	**13.3**
Diamond	—	—	—	—	—	—	—	**20.7**

Table 8: Time in minutes needed for reaching each milestone item, averaged over successful episodes. We omit timings for items below a success rate of 0.5% to ensure statistical significance. Scores within 5% of the fastest recorded score are highlighted in bold.

E. Minecraft Inputs



(a) Dreamer 3 (64×64) **(b)** VPT (128×128) **(c)** Dreamer 4 (360×640)

Figure 10: Comparison of input images for different agents. Dreamer 4 learns directly from high-resolution images reflecting the experience of human players.

F. Previous Dreamer Generations

Dreamer 3[1] learned to obtain diamonds in Minecraft from scratch by online interaction. Its inputs are low-resolution images and inventory states and the outputs are mouse, keyboard, and abstract crafting actions. Dreamer 3 uses a recurrent state-space model (RSSM)[64] as its world model, which is based on a recurrent neural network and a variational objective. This approach results in a lightweight world model with highly efficient inference but is difficult to scale to diverse data distributions. In contrast, Dreamer 4 learns to obtain diamonds in Minecraft purely from offline data. Its inputs are only high-resolution images and the outputs are low-level mouse and keyboard actions. Dreamer 4 uses a scalable world model based on an efficient transformer architecture and a shortcut forcing objective, allowing it to scale to diverse data distributions with many details. While Dreamer 3 uses return normalization and an entropy regularizer, Dreamer 4 uses PMPO with a KL to the behavioral cloning prior for imagination training, where no normalization is needed.

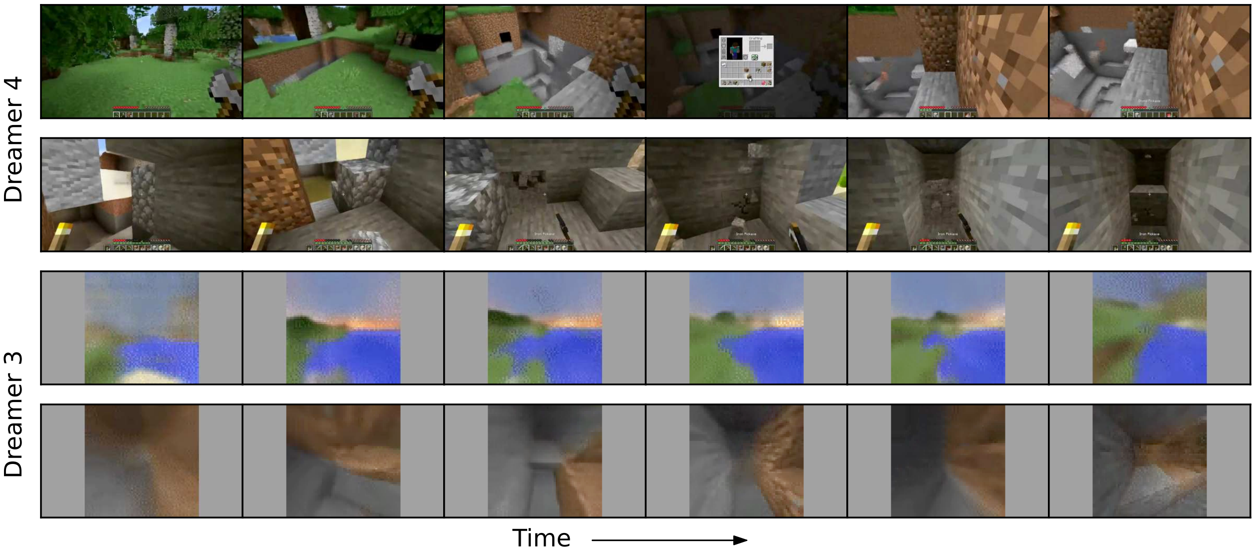


Figure 11: Comparison of multi-step video generations between Dreamer 3 and Dreamer 4.

G. Human Interaction: Lucid-v1

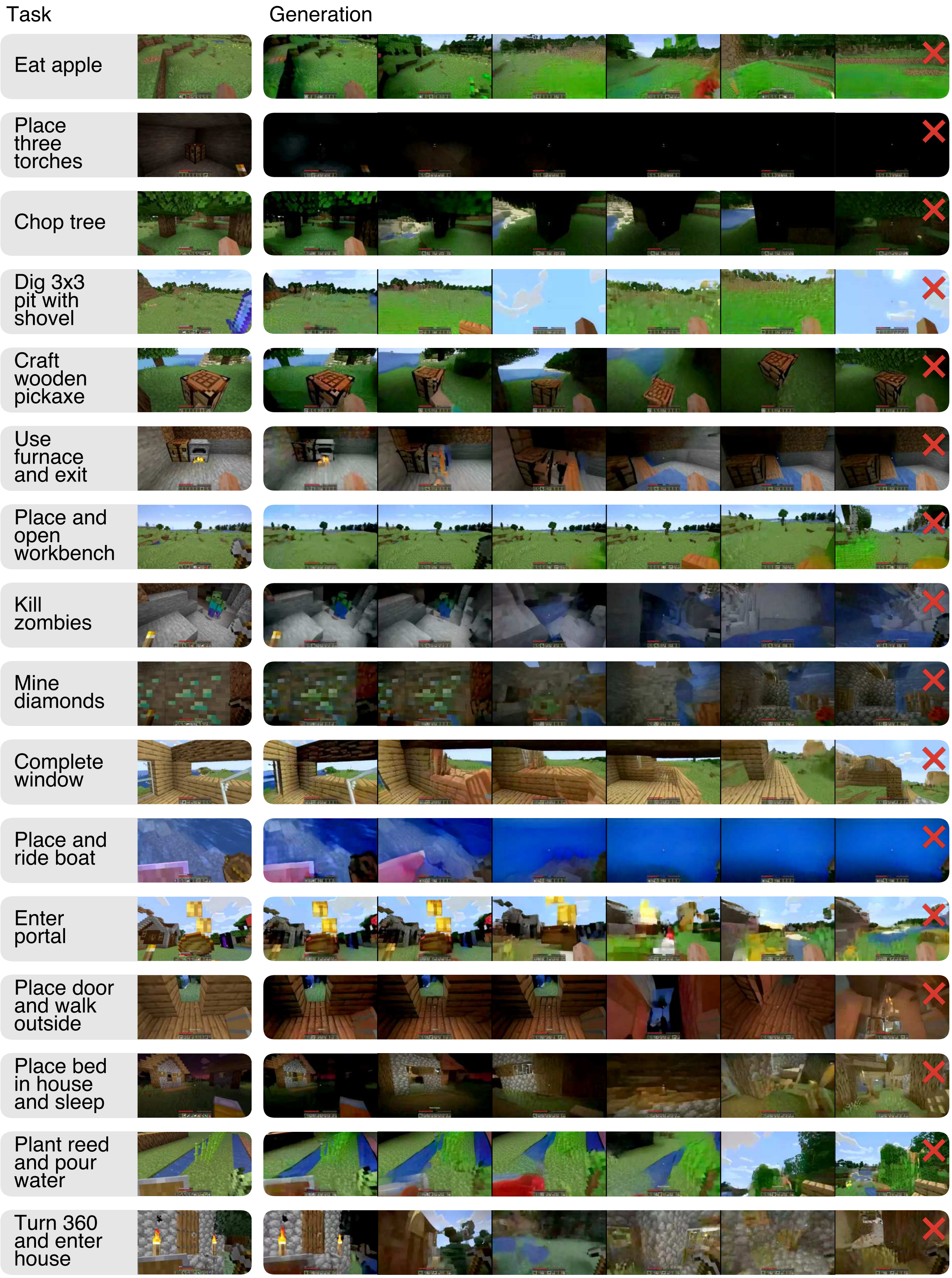


Figure 12: Lucid-v1

H. Human Interaction: OASIS (large)

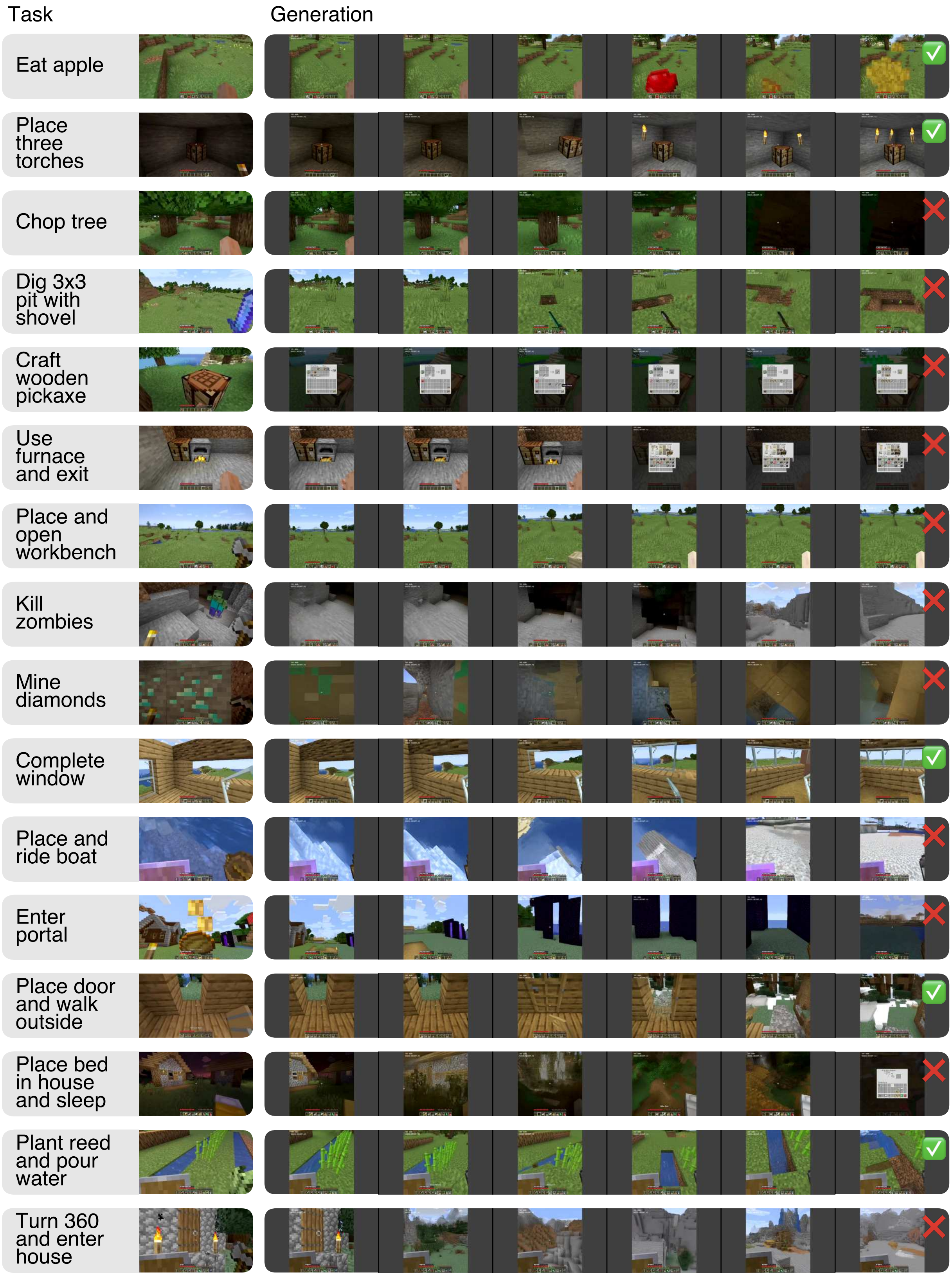


Figure 13: Oasis (large)

I. Human Interaction: Dreamer 4

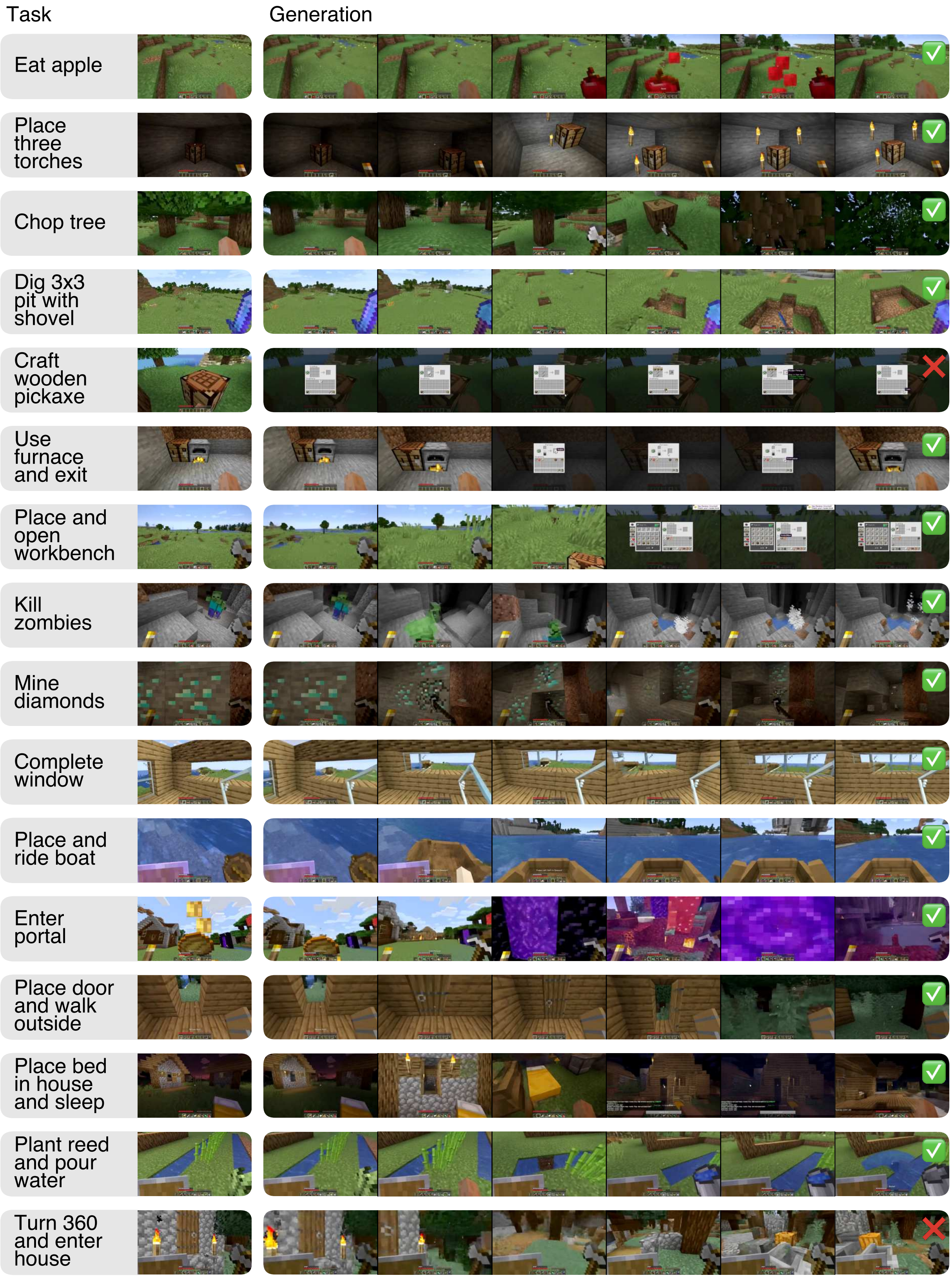


Figure 14: Dreamer 4