Deep Hierarchical Planning from Pixels

Anonymous Author(s)
Affiliation
Address
email

Abstract

Intelligent agents need to select long sequences of actions to solve complex tasks. While humans easily break down tasks into subgoals and reach them through millions of muscle commands, current artificial intelligence is limited to tasks with horizons of a few hundred decisions, despite large compute budgets. Research on hierarchical reinforcement learning aims to overcome this limitation but has proven to be challenging, current methods rely on manually specified goal spaces or subtasks, and no general solution exists. We introduce Director, a practical method for learning hierarchical behaviors directly from pixels by planning inside the latent space of a learned world model. The high-level policy maximizes task and exploration rewards by selecting latent goals and the low-level policy learns to achieve the goals. Despite operating in latent space, the decisions are interpretable because the world model can decode goals into images for visualization. Director learns successful behaviors across a wide range of environments, including visual control, Atari games, and DMLab levels and outperforms exploration methods on tasks with very sparse rewards, including 3D maze traversal with a quadruped robot from an egocentric camera and proprioception, without access to the global position or top-down view used by prior work. Videos and code are available on the project website: https://anonymous-website.github.io/director/

1 Introduction

Artificial agents have achieved remarkable performance on reactive video games (Mnih et al., 2015; Badia et al., 2020) or board games that last for a few hundred moves (Silver et al., 2017). However, solving complex control problems can require millions of time steps. For example, consider a robot that needs to navigate along the sidewalk and cross streets to buy groceries and then return home and cook a meal with those groceries. Manually specifying subtasks or dense rewards for such complex tasks would not only be expensive but also prone to errors and require tremendous effort to capture

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

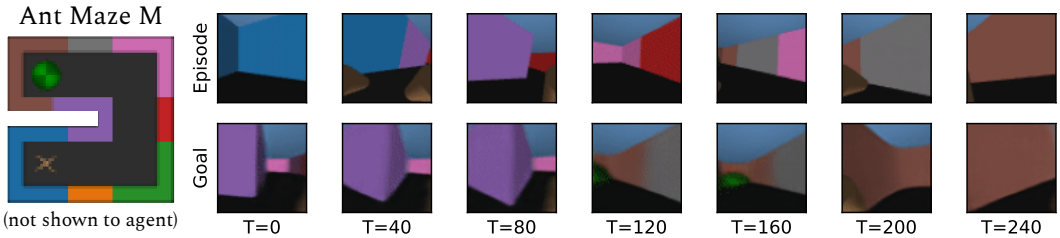


Figure 1: Director on Ant Maze M from egocentric camera inputs. The top row shows agent inputs. The bottom row shows the internal subgoals of the agent. The goals are latent vectors that Director's world model can decode into images for human inspection. Director solves this sparse reward task by breaking it down into internal goals. It first targets the purple wall in the middle of the maze. Once reached, it targets the reward object, and then the brown wall behind it to step onto the reward object.

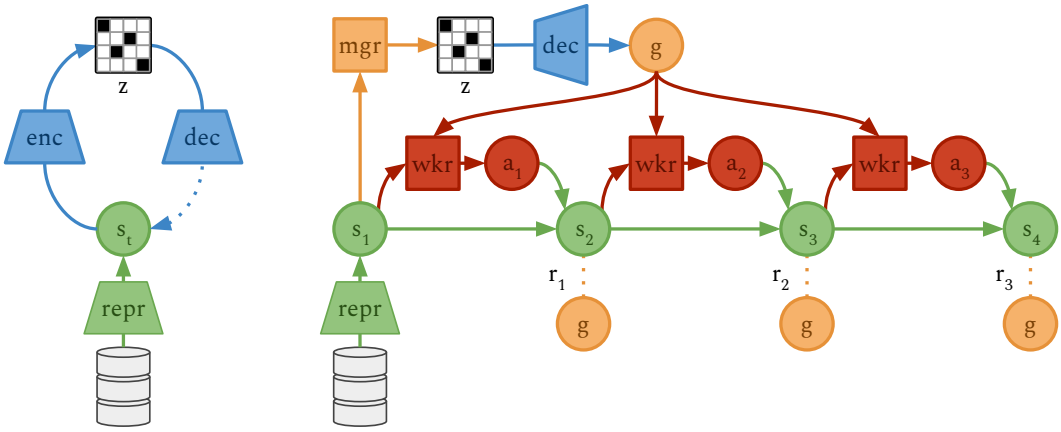


Figure 2: Director is based on the **world model** of PlaNet (Hafner et al., 2018) that predicts ahead in a compact representation space. The world model is trained by reconstructing images using a neural network not shown in the figure. Director then learns three additional components, which are all optimized concurrently. On the left, the **goal autoencoder** compresses the feature vectors s_t into vectors of discrete codes $z \sim \mathrm{enc}(z \mid s_t)$. On the right, the **manager** policy $\mathrm{mgr}(z \mid s_t)$ selects abstract actions in this discrete space every $K = 8$ steps, which the goal decoder then turns into feature space goals $g = \mathrm{dec}(z)$. The **worker** policy $\mathrm{wkr}(a_t \mid s_t, g)$ receives the current feature vector and goal as input to decide primitive actions that maximize the similarity rewards r_t to the goal. The manager maximizes the task reward and an exploration bonus based on the autoencoder reconstruction error, implementing temporally-extended exploration.

special cases (Chen et al., 2021; Ahn et al., 2022). Even training a robot to simply walk forward can require specifying ten different reward terms (Kumar et al., 2021), making reward engineering a critical component of such systems. Humans naturally break long tasks into subgoals, each of which is easy to achieve. In contrast, most current reinforcement learning algorithms reason purely at the clock rate of their primitive actions. This poses a key bottleneck of current reinforcement learning methods that could be challenging to solve by simply increasing the computational budget.

Hierarchical reinforcement learning (HRL) (Dayan and Hinton, 1992; Parr and Russell, 1997; Sutton et al., 1999) aims to automatically break long-horizon tasks into subgoals or commands that are easier to achieve, typically by learning high-level controllers that operate at more abstract time scales and provide commands to low-level controllers that select primitive actions. However, most HRL approaches require domain knowledge to break down tasks, either through manually specified subtasks (Tessler et al., 2017) or semantic goal spaces such as global XY coordinates for navigation tasks (Andrychowicz et al., 2017; Nachum et al., 2018a) or robot poses (Gehring et al., 2021). Attempts at learning hierarchies directly from sparse rewards have had limited success (Vezhnevets et al., 2017) and required providing task reward to the low-level controller, calling into question the benefit of their high-level controller.

In this paper, we present Director, a practical method for learning hierarchical behaviors directly from pixels by planning inside the latent space of a learned world model. We observe the effectiveness of Director on long-horizon tasks with very sparse rewards and demonstrate its generality by learning successfully in a wide range of domains. The key insights of Director are to leverage the representations of the world model, select goals in a compact discrete space to aid learning for the high-level policy, and to use a simple form of temporally-extended exploration in the high-level policy.

Contributions The key contributions of this paper are summarized as follows:

- We describe a practical, general, and interpretable algorithm for learning hierarchical behaviors within a world model trained from pixels, which we call Director (Section 2).
- We introduce two sparse reward benchmarks that underscore the limitations of traditional flat RL approaches and find that Director solves these challenging tasks (Section 3.1).
- We demonstrate that Director successfully learns in a wide range of traditional RL environments, including Atari, Control Suite, DMLab, and Crafter (Section 3.2).
- We visualize the latent goals that Director selects for breaking down various tasks, providing insights into its decision making (Section 3.3).

2 Director

Director is a reinforcement learning algorithm that learns hierarchical behaviors directly from pixels. As shown in Figure 2, Director learns a world model for representation learning and planning, a goal autoencoder that discretizes the possible goals to make them easier for the manager to choose, a manager policy that selects goals every fixed number of steps to maximize task and exploration rewards, and a worker policy that learns to reach the goals through primitive actions. All components are optimized throughout the course of learning by performing one gradient step every fixed number of environment steps. The world model is trained from a replay buffer whereas the goal autoencoder is trained on the world model representations and the policies are optimized from imagined rollouts. For the pseudo code of Director, refer to Appendix E.

2.1 World Model

Director learns a world model that compresses the history of observations into a compact feature space and enables planning in this space (Watter et al., 2015; Zhang et al., 2019). We use the Recurrent State Space Model (RSSM) model of PlaNet (Hafner et al., 2018), which we briefly review here to introduce notation. The world model consists of four neural networks that are optimized jointly:

$$\begin{array}{llll} \text{Model representation:} & \mathrm{repr}_\theta(s_t \mid s_{t-1}, a_{t-1}, x_t) & \text{Model decoder:} & \mathrm{rec}_\theta(s_t) \approx x_t \\ \text{Model dynamics:} & \mathrm{dyn}_\theta(s_t \mid s_{t-1}, a_{t-1}) & \text{Reward predictor:} & \mathrm{rew}_\theta(s_{t+1}) \approx r_t \end{array} \tag{1}$$

The representation model integrates actions a_t and observations x_t into the latent states s_t. The dynamics model predicts future states without the corresponding observations. The decoder reconstructs observations to provide a rich learning signal. The reward predictor later allows learning policies by planning in the compact latent space, without decoding images. The world model is optimized end-to-end on subsequences from the replay buffer by stochastic gradient descent on the variational objective (Hinton and Van Camp, 1993; Kingma and Welling, 2013; Rezende et al., 2014):

$$\begin{aligned} \mathcal{L}(\theta) \doteq \sum_{t=1}^{T} \Big(& \beta \, \mathrm{KL}\big[\mathrm{repr}_\theta(s_t \mid s_{t-1}, a_{t-1}, x_t) \,\big\|\, \mathrm{dyn}_\theta(s_t \mid s_{t-1}, a_{t-1})\big] \\ & + \|\mathrm{rec}_\theta(s_t) - x_t\|^2 + (\mathrm{rew}_\theta(s_{t+1}) - r_t)^2 \Big) \quad \text{where} \quad s_{1:T} \sim \mathrm{repr}_\theta \end{aligned} \tag{2}$$

The variational objective encourages learning a Markovian sequence of latent states with the following properties: The states should be informative of the corresponding observations and rewards, the dynamics model should predict future states accurately, and the representations should be formed such that they are easy to predict. The hyperparameter β trades off the predictability of the latent states with the reconstruction quality (Beattie et al., 2016; Alemi et al., 2018).

2.2 Goal Autoencoder

The world model representations s_t are 1024-dimensional continuous vectors. Selecting such representations as goals would be challenging for the manager because this constitutes a very high-dimensional continuous action space. To avoid a high-dimensional continuous control problem for the manager, Director compresses the representations s_t into smaller discrete codes z using a goal autoencoder that is trained on replay buffer model states from Equation 2:

$$\text{Goal Encoder:} \quad \mathrm{enc}_\phi(z \mid s_t) \qquad \text{Goal Decoder:} \quad \mathrm{dec}_\phi(z) \approx s_t \tag{3}$$

Simply representing each model state s_t by a class in one large categorical vector would require roughly one category per distinct state in the environment. It would also prevent the manager from generalizing between its different outputs. Therefore, we opt for a factorized representation of multiple categoricals. Specifically, we choose the vector of categoricals approach introduced in DreamerV2 (Hafner et al., 2020a). As visualized in Figure G.1, the goal encoder takes a model state as input and predicts a matrix of $8{\times}8$ logits, samples a one-hot vector from each row, and flattens the results into a sparse vector with 8 out of 64 dimensions set to 1 and the others to 0. Gradients are backpropagated through the sampling by straight-through estimation (Bengio et al., 2013). The goal autoencoder is optimized end-to-end by gradient descent on the variational objective:

$$\mathcal{L}(\phi) \doteq \big\| \mathrm{dec}_\phi(z) - s_t \big\|^2 + \beta \, \mathrm{KL}\big[\mathrm{enc}_\phi(z \mid s_t) \,\big\|\, p(z)\big] \quad \text{where} \quad z \sim \mathrm{enc}_\phi(z \mid s_t) \tag{4}$$

The first term is a mean squared error that encourages the encoder to compute informative codes from which the input can be reconstructed. The second term encourages the encoder to use all available codes by regularizing the distribution towards a uniform prior $p(z)$. The autoencoder is trained at the same time as the world model but does not contribute gradients to the world model.

2.3 Manager Policy

Director learns a manager policy that selects a new goal for the worker every fixed number of $K = 8$ time steps. The manager is free to choose goals that are much further than 8 steps away from the current state, and in practice, it often learns to choose the most distant goals that the worker is able to achieve. Instead of selecting goals in the high-dimensional continuous latent space of the world model, the manager outputs abstract actions in the discrete code space of the goal autoencoder (Section 2.2). The manager actions are then decoded into world model representations before they are passed on to the worker as goals. To select actions in the code space, the manager outputs a vector of categorical distributions, analogous to the goal encoder in Section 2.2:

$$\text{Manager Policy:} \quad \mathrm{mgr}_\psi(z \mid s_t) \tag{5}$$

The objective for the manager is to maximize the discounted sum of future task rewards and exploration rewards. The exploration encourages the manager to choose novel goals for the worker, resulting in temporally-abstract exploration. This is important because the worker is goal-conditioned, so without an explicit drive to expand the state distribution, it could prefer going back to previously common states it has been trained on the most, and thus hinder exploration of new states in the environment. Because the goal autoencoder is trained from the replay buffer, it tracks the current state distribution of the agent and we can reward novel states as those that have a high reconstruction error under the goal autoencoder:

$$r_t^{\mathrm{expl}} \doteq \left\| \mathrm{dec}_\phi(z) - s_{t+1} \right\|^2 \quad \text{where} \quad z \sim \mathrm{enc}_\phi(z \mid s_{t+1}) \tag{6}$$

Both manager and worker policies are trained from the same imagined rollouts and optimized using Dreamer (Hafner et al., 2019; 2020a), which we summarize in Appendix H. The manager learns two state-value critics for the extrinsic and exploration rewards, respectively. The critics are used to fill in rewards beyond the imagination horizon and as baseline for variance reduction (Williams, 1992). We normalize the extrinsic and exploration returns by their exponential moving standard deviations before summing them with weights $w^{\mathrm{extr}} = 1.0$ and $w^{\mathrm{expl}} = 0.1$. For updating the manager, the imagined trajectory is temporally abstracted by selecting every K-th model state and by summing rewards within each non-overlapping subsequence of length K. No off-policy correction (Schulman et al., 2017; Nachum et al., 2018a) is needed because the imagined rollouts are on-policy.

2.4 Worker Policy

The worker is responsible for reaching the goals chosen by the manager. Because the manager outputs codes z in the discrete space of the goal autoencoder, we first decode the goals into the state space of the world model $g \doteq \mathrm{dec}(z)$. Conditioning the worker on decoded goals rather than the discrete codes has the benefit that its learning becomes approximately decoupled from the goal autoencoder. The worker policy is conditioned on the current state and goal, which changes every $K = 8$ time steps, and it produces primitive actions a_t to reach the feature space goal:

$$\text{Worker Policy:} \quad \mathrm{wkr}_\xi(a_t \mid s_t, g) \tag{7}$$

To reach its latent goals, we need to choose a reward function for the worker that measures the similarity between the current state s_t and the current goal g, both of which are 1024-dimensional continuous activation vectors. Natural choices would be the negative L2 distance or the cosine similarity and their choice depends on the underlying feature space, which is difficult to reason about. Empirically, we found the cosine similarity to perform better. Cosine similarity would usually normalize both vectors, but normalizing the state vector encourages the worker to remain near the origin of the latent space, so that it can quickly achieve any goal by moving a small amount in the right direction. We thus incorporate the idea that the magnitude of both vectors should be similar into the cosine similarity, resulting in our *max-cosine* reward:

$$r_t^{\mathrm{goal}} \doteq \big(g/m\big)^T \big(s_{t+1}/m\big) \quad \text{where} \quad m \doteq \max\big(\, \|g\|,\, \|s_{t+1}\| \,\big) \tag{8}$$

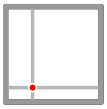
Method	Task	Observations	Training	Evaluation
HIRO		 • Proprioception • Goal XY pos • Robot XY pos	Random XY goals, dense L2 reward	Fixed XY goal, success rate
NORL		• Proprioception • Goal XY pos • Top view (5×5)	Random XY goals, dense L2 reward	Fixed XY goal, success rate
This paper		• Proprioception • First-person camera (64×64×3)	End-to-end reinforcement learning, sparse reward when the robot is touching the goal	

Figure 3: Comparison of Ant Mazes in the literature and this paper. HIRO (Nachum et al., 2018a) provided global XY coordinates of the goal and robot position to the agent and trained with dense rewards on uniformly sampled training goals. NORL (Nachum et al., 2018b) replaced the robot XY position with a global top-down view of the environments, downsampled to 5×5 pixels. In this paper, we tackle the more challenging problem of learning directly from an egocentric camera without global information provided to the agent, and only give a sparse reward for time steps where the robot is at the goal. To succeed at these tasks, an agent has to autonomously explore the environment and identify landmarks to localize itself and navigate the mazes.

When the state and goal vectors are of the same magnitude, the reward simplifies to the cosine similarity. When their magnitudes differ, the vectors are both normalized by the larger of the two magnitudes, and thus the worker receives a down-scaled cosine similarity as the reward. As a result, the worker is incentivized to match the angle and magnitude of the goal. Unlike the L2 similarity, the reward scale of our goal reward is not affected by the scale of the underlying feature space.

The worker maximizes only the goal rewards. We make this design choice to demonstrate that the interplay between the manager and the worker is successful across many environments, although we also include an ablation experiment where the worker additionally receives task reward, which further improves performance. The worker is optimized by Dreamer with a goal-conditioned state-value critic. For updating the worker, we cut the imagined rollouts into distinct trajectories of length K within which the goal is constant. The state-critic estimates goal rewards beyond this horizon under the same goal, allowing the worker to learn to reach far-away goals.

3 Experiments

We evaluate Director on two challenging benchmark suites with visual inputs and very sparse rewards, which we expect to be challenging to solve using a flat policy without hierarchy (Section 3.1). We further evaluate Director on a wide range of standard tasks from the literature to demonstrate its generality and ensure that the hierarchy is not harmful in simple settings (Section 3.2). We use a fixed set of hyperparameters not only across tasks but also across domains, detailed in Table F.1. Finally, we offer insights into the learned hierarchical behaviors by visualizing the latent goals selected during environment episodes (Section 3.3). Ablations and additional results are included in the appendix.

Implementation We implemented Director on top of the public source code of DreamerV2 (Hafner et al., 2020a), reusing its default hyperparameters. We additionally increased the number of environment instances to 4 and set the training frequency to one gradient step per 16 policy steps, which drastically reduced wall-clock time and decreased sample-efficiency mildly. Implementing Director in the code base amounts to about 250 lines of code. The computation time of Director is 20% longer than that of DreamerV2. Each training run used a single V100 GPU with XLA and mixed precision enabled and completed in less than 24 hours. All our agents and environments will be open sourced upon publication to facilitate future research in hierarchical reinforcement learning.

Baselines To fairly compare Director to the performance of non-hierarchical methods, we compare to the DreamerV2 agent on all tasks. DreamerV2 has demonstrated strong performance on Atari (Hafner et al., 2020a), Crafter (Hafner, 2021) and continuous control tasks (Yarats et al., 2021) and outperforms top model-free methods in these domains. We kept the default hyperparameters that the

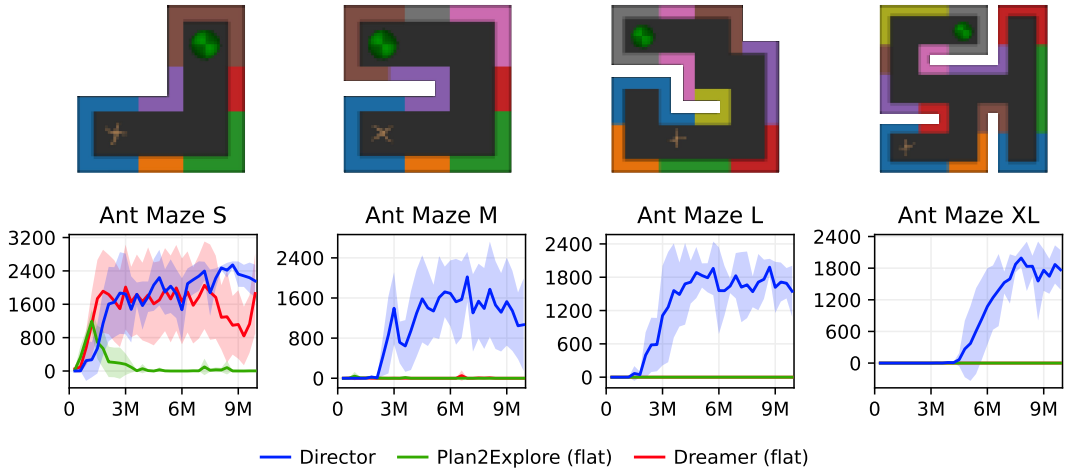


Figure 4: Egocentric Ant Maze benchmark. A quadruped robot is controlled through joint torques to navigate to a fixed location in a 3D maze, given only first-person camera and proprioceptive inputs. This is in contrast to prior benchmarks where the agents received their global XY coordinate or top-down view. The only reward is given at time steps where the agent touches the reward object. Plan2Explore fails in the small maze because the robot flips over too much, a common limitation of low-level exploration methods. Director solves all four tasks by breaking them down into manageable subgoals that the worker can reach, while learning in the end-to-end reinforcement learning setting.

authors tuned for DreamerV2 and did not change them for Director. In addition to its hierarchical policy, Director employs an exploration bonus at the top-level. To isolate the effects of hierarchy and exploration, we compare to Plan2Explore (Sekar et al., 2020), which uses ensemble disagreement of forward models as a directed exploration signal. We combined the extrinsic and exploration returns after normalizing by their exponential moving standard deviation with weights 1.0 and 0.1, as in Director. We found Plan2Explore to be effective across both continuous and discrete control tasks.

3.1 Sparse Reward Benchmarks

Egocentric Ant Mazes Learning navigation tasks directly from joint-level control has been a long-standing milestone for reinforcement learning with sparse rewards, commonly studied with quadruped robots in maze environments (Florensa et al., 2017; Nachum et al., 2018a). However, previous attempts typically required domain-specific inductive biases to solve such tasks, such as providing global XY coordinates to the agent, easier practice goals, and a ground-truth distance reward, as summarized in Figure 3. In this paper, we instead attempt learning directly from first-person camera inputs, without privileged information, and a single sparse reward that the agent receives while in the fixed goal zone. The control frequency is 50Hz and episodes end after a time limit of 3000 steps. There are no early terminations that could leak task-information to the agent (Laskin et al., 2022). To help the agent localize itself from first-person observations, we assign different colors to different walls of the maze.

As shown in Figure 4, we evaluate the agents in four mazes that span varying levels of difficulty. Because of the sparse reward, the episode returns correspond to the number of time steps for which the agent remains at the goal after reaching it. Curves show the mean and standard deviation across 5 independent seeds. We find that the flat Dreamer agent succeeds at the smallest of the four mazes, and the flat exploration policy Plan2Explore makes some initial learning progress but fails to converge to the optimal solution. Inspecting the trajectories revealed that Plan2Explore chooses too chaotic actions that often result in the robot flipping over. None of the baselines learn successful behaviors in the larger mazes, demonstrating that the benchmark pushes the limits of current approaches. In contrast, Director discovers and learns to reliably reach the goal zone at all four difficulty levels, with the larger maze taking longer to master.

Visual Pin Pads The Pin Pad suite of environments is designed to evaluate an agent's ability to explore and assign credit over long horizons, isolated from the complexity of 3D observations or sophisticated joint-level control. As shown in Figure 5, each environment features a moving black square that the agent can control in all four directions and fixed pads of different colors that the agent can activate by walking over to the pad and stepping on it. The task requires discovering the correct sequence of activating all pads, at which point the agent receives a sparse reward of 10 points and the

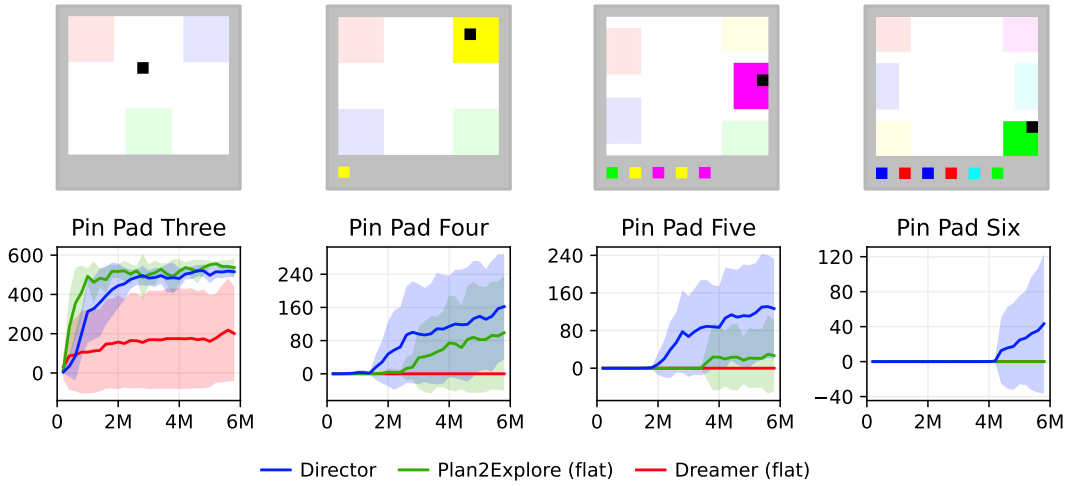


Figure 5: Visual Pin Pad benchmark. The agent controls the black square to move in four directions. Each environment has a different number of pads that can be activated by walking to and stepping on them. A single sparse reward is given when the agent activates all pads in the correct sequence. The history of previously activated pads is shown at the bottom of the screen. Plan2Explore uses low-level exploration and performs well in this environment, but struggles for five and six pads, which requires more abstract exploration and longer credit assignment. Director learns successfully across all these environments, demonstrating its benefit on this long-horizon benchmark.

agent position is randomly reset. Episodes last for 2000 steps and there are no intermediate rewards for activating the pads. To remove the orthogonal challenge of learning long-term memory (Gregor et al., 2019), the history of previously activated pads is displayed at the bottom of the image.

The agent performance is shown in Figure 5, which displays mean and standard deviation across five independent seeds. The easiest environment contains three pads, so the agent only has to decide whether to activate the pads in clockwise or counter-clockwise sequence. The flat Dreamer agent sometimes discovers the correct sequence. The exploration bonus of Plan2Explore offers a significant improvement over Dreamer on this task. Dreamer fails to discover the correct sequence in the harder environments that contain more pads. Plan2Explore still achieves some reward with four pads, struggles with five pads, and completely fails with six pads. In contrast, Director discovers the correct sequence in all four environments, demonstrating the benefit of hierarchy over flat exploration.

3.2 Standard Benchmarks

To evaluate the robustness of Director, we train on a wide range of standard benchmarks, which typically require no long-term reasoning. We choose Atari games (Bellemare et al., 2013), the Control Suite from pixels (Tassa et al., 2018), Crafter (Hafner, 2021), and tasks from DMLab (Beattie et al., 2016) to cover a spectrum of challenges, including continuous and discrete actions and 2D and 3D environments. We compare two versions of Director. In its standard variant, the worker learns purely from goal rewards. This tests the ability of the manager to propose successful goals and the ability of the worker to achieve them. In the second variant, the worker learns from goal and task returns with weights 1.0 and 0.5, allowing the worker to fill in low-level details in a task-specific manner, which the manager may be too coarse-grained to provide. Ideally, we would like to see Director not perform worse than Dreamer on any task when giving task reward to the worker.

The results of this experiment are summarized in Appendix A due to space constraints, with the full training curves for Atari and the Control Suite included in Appendices J and K. We observe that Director indeed learns successfully across many environments, showing broader applicability than most prior hierarchical reinforcement learning methods. In addition, providing task reward to the worker is not as important as expected — the hierarchy solves a wide range of tasks purely by following goals at the low level. Additionally providing task reward to the worker completely closes the gap to the state-of-the-art DreamerV2 agent. Figure K.1 in the appendix further shows that Director achieves a higher human-normalized median score than Dreamer on the 55 Atari games.

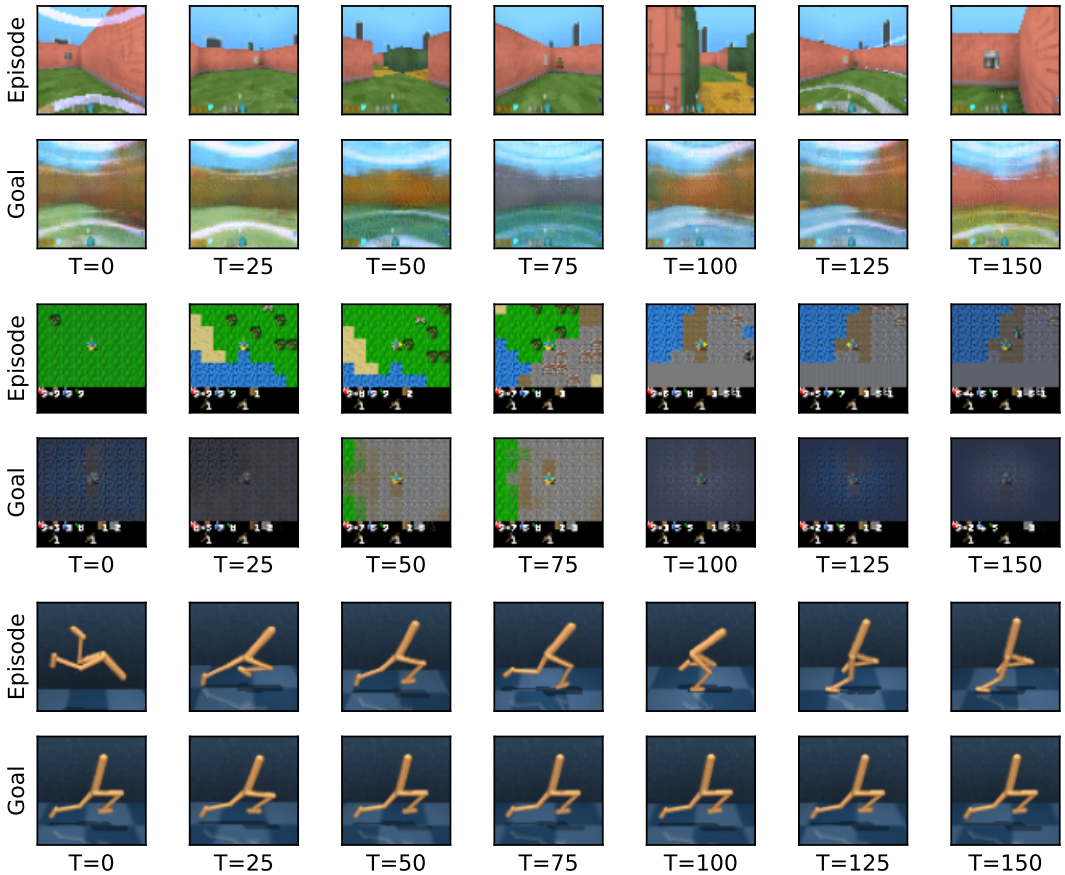


Figure 6: Subgoals discovered by Director. For interpretability, we decode the latent goals into images using the world model. **Top:** In DMLab, the manager chooses goals of the teleport animation that occurs every time the agent has collected the reward object, which can be seen in the episode at time step 125. **Middle:** In Crafter, the manager first directs the worker via the inventory display to collect wood and craft a pickaxe and a sword, with the worker following command. The worker then suggests a cave via the terrain image to help the worker find stone. As night breaks, it suggests hiding from the monsters in a cave or on a small island. **Bottom:** In Walker, the manager abstracts away the detailed leg movements by suggesting a forward leaning pose and a shifting floor pattern, with the worker successfully filling in the joint movements. Fine-grained subgoals are not required for this task, because the horizon needed for walking is short enough for the worker. Videos are available on the project website: `https://anonymous-website.github.io/director/`

3.3 Goal Interpretations

To gain insights into the decision making of Director, we visualize the sequences of goals the manager selects during environment interaction. While the goals are uninterpretable latent features, the world model allows us to decode them into images for human inspection. Figures 1 and 6 show episodes with the environment frames (what the agent sees) at the top and the decoded subgoals (what the manager wants the worker to achieve) at the bottom. Visualizations for additional environments are included in Appendix L.

- **Ant Maze M** The manager chooses goals that direct the agent through the different sections of the maze until the reward object is reached. We also observed that initially, the manager chooses more intermediate subgoals whereas later during training, the worker learns to achieve further away goals and thus the manager can select fewer intermediate goals.
- **DMLab Goals Small** The manager directs the worker to the teleport animation, which occurs every time the agent collects the reward object and gets teleported to a random location in the 3D maze. Time step $T = 125$ shows an example of the worker reaching the animation in the environment. Unlike the Ant Maze benchmark, navigating to the goal in DMLab requires no joint-level control and is simple enough for the worker to achieve without fine-grained subgoals.

- **Crafter** The manager requests higher inventory counts of wood and stone materials, as well as wooden tools, via the inventory display at the bottom of the screen. It also directs the worker to a cave to find stone and coal but generally spends less effort on suggesting what the world around the agent should look like. As night breaks, the manager tasks the worker with finding a small cave or island to hide from the monsters that are about to spawn.
- **Walker Walk** The manager successfully abstracts away the detail of leg movement, steering the worker using a forward leaning pose with both feet off the ground and a shifting floor pattern. While the images cannot show this, the underlying goal vectors are Markovian states and can thus contain information about velocity, so it is likely that the manager additionally requests high forward velocity. The worker fills in the details of standing up and moving the legs to pass through the sequence of goals.

Across domains, Director chooses semantically meaningful goals that are appropriate for the task, despite using the same hyperparameters across all tasks and receiving no domain-specific knowledge.

4 Related Work

Approaches to hierarchical reinforcement learning include learning low-level policies on a collection of easier pre-training tasks (Heess et al., 2016; Tessler et al., 2017; Frans et al., 2017; Rao et al., 2021; Veeriah et al., 2021), discovering latent skills via mutual-information maximization (Gregor et al., 2016; Florensa et al., 2017; Hausman et al., 2018; Eysenbach et al., 2018; Achiam et al., 2018; Merel et al., 2018; Laskin et al., 2022; Sharma et al., 2019; Xie et al., 2020; Hafner et al., 2020b; Strouse et al., 2021), or training the low-level as a goal-conditioned policy (Andrychowicz et al., 2017; Levy et al., 2017; Nachum et al., 2018a;b; Co-Reyes et al., 2018; Warde-Farley et al., 2018; Nair et al., 2018; Pong et al., 2019; Hartikainen et al., 2019; Gehring et al., 2021; Shah et al., 2021). These approaches are described in more detail in Appendix I.

Relatively few works have demonstrated successful learning of hierarchical behaviors directly from pixels without domain-specific knowledge, such as global XY positions, manually specified pre-training tasks, or precollected diverse experience datasets. HSD-3 (Gehring et al., 2021) showed transfer benefits for low-dimensional control tasks. HAC (Levy et al., 2017) learned interpretable hierarchies but required semantic goal spaces. FuN (Vezhnevets et al., 2017) learned a two-level policy where both levels maximize task reward and the lower level is regularized by a goal reward but did not demonstrate clear benefits over an LSTM baseline (Hochreiter and Schmidhuber, 1997). We leverage explicit representation learning and temporally-abstract exploration, demonstrate substantial benefits over flat policies on sparse reward tasks, and underline the generality of our method by showing successful learning without giving task reward to the worker across many domains.

5 Discussion

We present Director, a reinforcement learning agent that learns hierarchical behaviors from pixels by planning in the latent space of a learned world model. To simplify the control problem for the manager, we compress goal representations into compact discrete codes. Our experiments demonstrate the effectiveness of Director on two benchmark suites with very sparse rewards from pixels. We also show that Director learns successfully across a wide range of different domains without giving task reward to the worker, underlining the generality of the approach. Decoding the latent goals into images using the world model makes the decision making of Director more interpretable and we observe it learning a diverse range of strategies for breaking tasks down into subgoals.

Limitations and future work We designed Director to be as simple as possible, at the expense of design choices that could restrict its performance. The manager treats its action space as a black box, without any knowledge that its actions correspond to states. For example, one could imagine regularizing the manager to choose goals that have a high value under its learned critic. Lifting the assumption of changing goals every fixed number of time steps, for example by switching based on a separate classifier or as soon as the previous goal has been reached, could enable the hierarchy to adapt to the environment and perform better on tasks that require precise timing. Moreover, goals are points in latent space, whereas distributional goals or masks would allow the manager to only specify targets for the parts of the state that are currently relevant. Learning temporally-abstract dynamics would allow efficiently learning hierarchies of more than two levels without having to use exponentially longer batches. We suspect that these ideas will improve the long-term reasoning of the agent. Besides improving the capabilities of the agent, we see disentangling the reasons for why Director works well, beyond the ablation experiments in the appendix, as promising future work.

References

J. Achiam, H. Edwards, D. Amodei, and P. Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

A. Alemi, B. Poole, I. Fischer, J. Dillon, R. A. Saurous, and K. Murphy. Fixing a broken elbo. In *International Conference on Machine Learning*, pages 159–168. PMLR, 2018.

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pages 507–517. PMLR, 2020.

C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

A. S. Chen, S. Nair, and C. Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*, 2021.

J. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *International Conference on Machine Learning*, pages 1009–1018. PMLR, 2018.

P. Dayan and G. E. Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

C. Florensa, Y. Duan, and P. Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

J. Gehring, G. Synnaeve, A. Krause, and N. Usunier. Hierarchical skills for efficient exploration. *Advances in Neural Information Processing Systems*, 34, 2021.

K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

K. Gregor, D. Jimenez Rezende, F. Besse, Y. Wu, H. Merzic, and A. van den Oord. Shaping belief states with generative environment models for rl. *Advances in Neural Information Processing Systems*, 32, 2019.

D. Hafner. Benchmarking the spectrum of agent capabilities. *arXiv preprint arXiv:2109.06780*, 2021.

D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

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

D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020a.

D. Hafner, P. A. Ortega, J. Ba, T. Parr, K. Friston, and N. Heess. Action and perception as divergence minimization. *arXiv preprint arXiv:2009.01791*, 2020b.

K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. *arXiv preprint arXiv:1907.08225*, 2019.

K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. *International Conference on Learning Representations*, 2018.

N. Heess, G. Wayne, Y. Tassa, T. Lillicrap, M. Riedmiller, and D. Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

G. E. Hinton and D. Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

M. Laskin, H. Liu, X. B. Peng, D. Yarats, A. Rajeswaran, and P. Abbeel. Cic: Contrastive intrinsic control for unsupervised skill discovery. *arXiv preprint arXiv:2202.00161*, 2022.

A. Levy, G. Konidaris, R. Platt, and K. Saenko. Learning multi-level hierarchies with hindsight. *arXiv preprint arXiv:1712.00948*, 2017.

R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, and D. Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34, 2021.

J. Merel, A. Ahuja, V. Pham, S. Tunyasuvunakool, S. Liu, D. Tirumala, N. Heess, and G. Wayne. Hierarchical visuomotor control of humanoids. *arXiv preprint arXiv:1811.09656*, 2018.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

O. Nachum, S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:1805.08296*, 2018a.

O. Nachum, S. Gu, H. Lee, and S. Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018b.

A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.

R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, 10, 1997.

V. H. Pong, M. Dalal, S. Lin, A. Nair, S. Bahl, and S. Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

D. Rao, F. Sadeghi, L. Hasenclever, M. Wulfmeier, M. Zambelli, G. Vezzani, D. Tirumala, Y. Aytar, J. Merel, N. Heess, et al. Learning transferable motor skills with hierarchical latent mixture policies. *arXiv preprint arXiv:2112.05062*, 2021.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

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

R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. *arXiv preprint arXiv:2005.05960*, 2020.

D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine. Rapid exploration for open-world navigation with latent goal models. *arXiv preprint arXiv:2104.05859*, 2021.

A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676): 354, 2017.

D. Strouse, K. Baumli, D. Warde-Farley, V. Mnih, and S. Hansen. Learning more skills through optimistic exploration. *arXiv preprint arXiv:2107.14226*, 2021.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

C. Tessler, S. Givony, T. Zahavy, D. Mankowitz, and S. Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

V. Veeriah, T. Zahavy, M. Hessel, Z. Xu, J. Oh, I. Kemaev, H. P. van Hasselt, D. Silver, and S. Singh. Discovery of options via meta-learned subgoals. *Advances in Neural Information Processing Systems*, 34, 2021.

A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

D. Warde-Farley, T. Van de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

M. Watter, J. Springenberg, J. Boedecker, and M. 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.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

K. Xie, H. Bharadhwaj, D. Hafner, A. Garg, and F. Shkurti. Skill transfer via partially amortized hierarchical planning. *arXiv preprint arXiv:2011.13897*, 2020.

D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine. Solar: deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, 2019.

Societal Impact

Developing hierarchical reinforcement learning methods that are useful for real-world applications will still require further research. However, in the longer term future, it has the potential to help humans automate more tasks by reducing the need to specify intermediate rewards. Learning autonomously from sparse rewards has the benefit of reduced human effort and less potential for reward hacking, but reward hacking is nonetheless a possibility that will need to be addressed once hierarchical reinforcement learning systems become more powerful and deployed around humans.

As reinforcement learning techniques become deployed in real-world environments, they will be capable of causing direct harm, intentional or unintentional. Director does not attempt to directly address such safety issues, but the ability to decode its latent goals into images for human inspection provides some transparency to the decisions the agent is making. This transparency may permit auditing of failures after-the-fact, and may additionally support interventions in some situations, if a human or automated observer is in a position to monitor the high-level goals Director generates.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] As stated in the text, we will provide code and training curves upon publication.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Table F.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figures 4, 5, A.1, J.1 and K.1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Standard Benchmarks

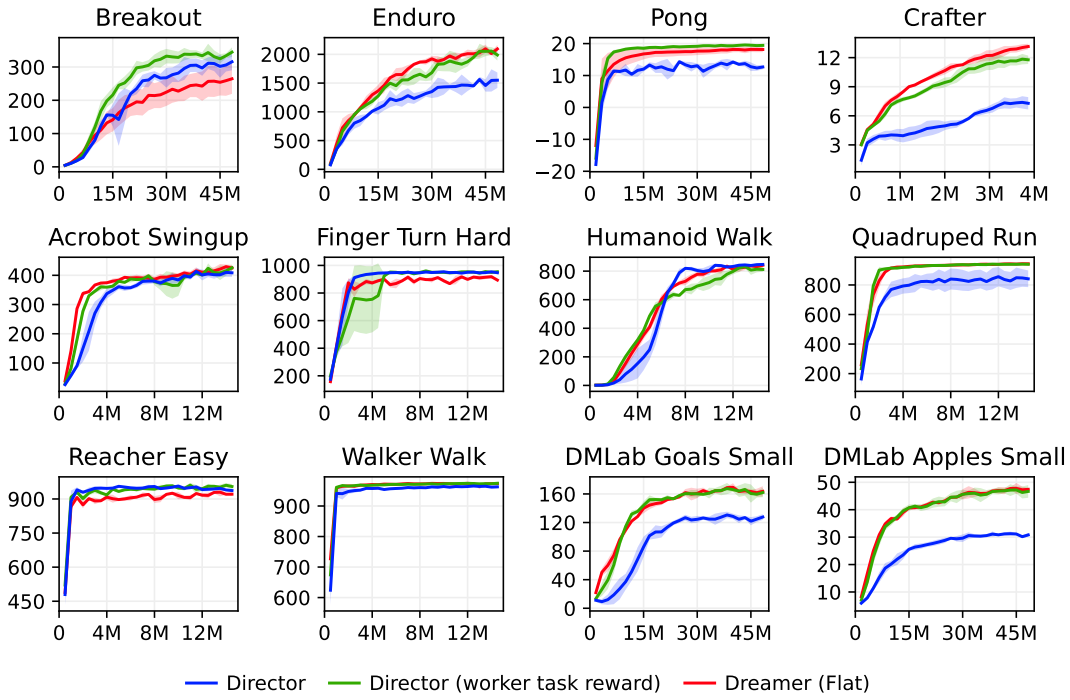


Figure A.1: Evaluation of Director on standard benchmarks, showing that Director is not limited to sparse reward tasks but is generally applicable. Director learns successfully across Atari, Crafter, Control Suite, and DMLab. This is an accomplishment because the worker receives no task reward and is purely steered through sub-goals selected by the manager. When additionally providing task reward to the worker, performance matches that of the state-of-the-art model-based agent Dreamer.

B Ablation: Goal Autoencoder

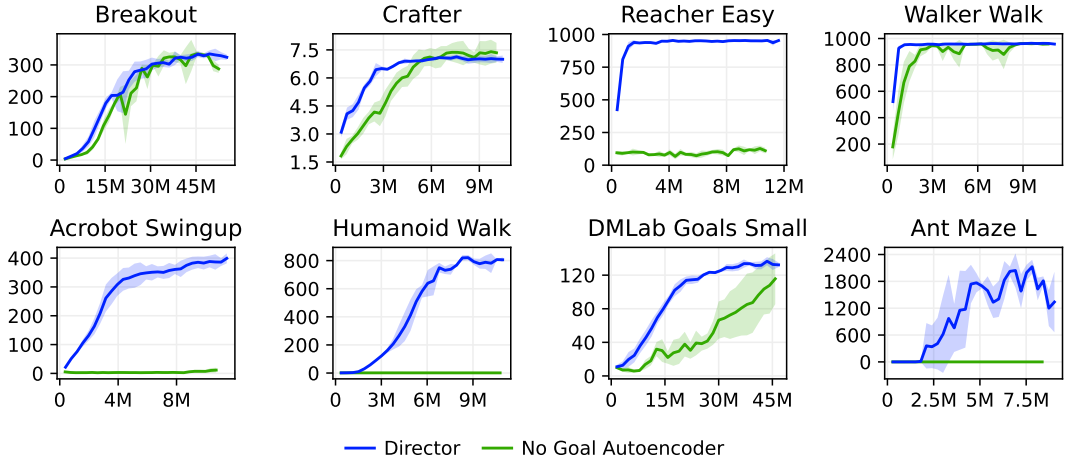


Figure B.1: Ablation of the Goal Autoencoder used by Director (Section 2.2). We compare the performance of Director to that of a hierarchical agent where the manager directly chooses 1024-dimensional goals in the continuous representation space of the world model. We observe that this simplified approach works surprisingly well in some environments but fails at environments with sparser rewards, likely because the control problem becomes too challenging for the manager.

C Ablation: Goal Reward

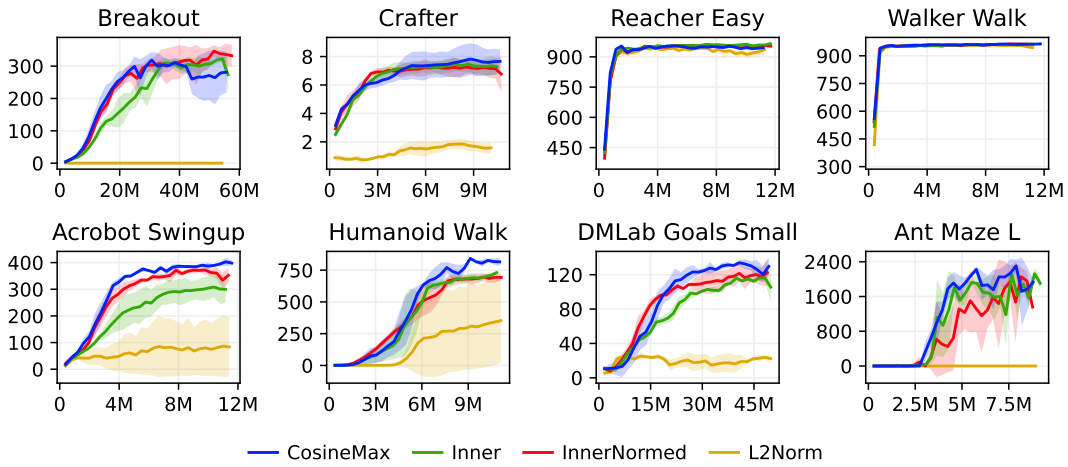


Figure C.1: Ablation of the goal reward used by Director. We compare the performance of Director with cosine-max reward (Equation 8) to alternative goal similarity functions. *Inner* refers to the inner product $g^T s_{t+1}$ without any normalization or clipping, which results in different reward scale based on the goal magnitude and encourages the worker to overshoot its goals in magnitude. *InnerNormed* refers to $(g/\|g\|)^T(s_{t+1}/\|g\|)$ where the goal and state are normalized by the goal magnitude, which normalizes the reward scale across goals but still encourages the worker to overshoot its goals. *L2Norm* is the negative euclidean distance $-\|g - s_{t+1}\|$. We observe that Director is robust to the precise goal reward, with the three rewards based on inner products performing well across tasks. The L2 reward works substantially worse. We hypothesize the reason to be that inner products allow goals to ignore some state dimensions by setting them to zero, whereas setting dimensions to zero for the L2 reward still requires the worker to care about them.

D Ablation: Exploration

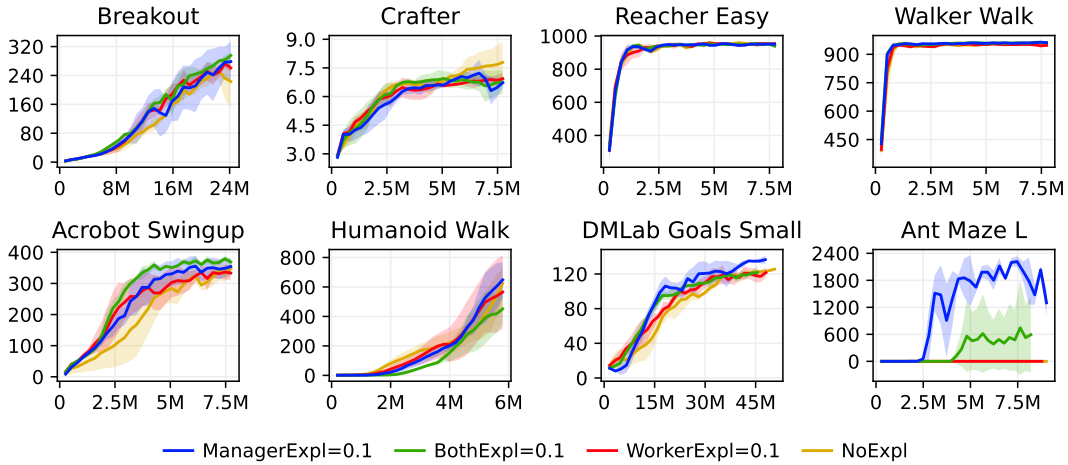


Figure D.1: Ablation of where to apply the exploration bonus in Director (Equation 6). We compare the performance of Director with exploration reward for only the manager, for only the worker, for both, and for neither of them. We find giving the exploration bonus to the manager, which results in temporally-abstract exploration, is required for successful learning in the Ant Maze, and that additional low-level exploration hurts because it results in too chaotic leg movements. In standard benchmarks that typically require short horizons, the exploration bonus is not needed.

E Pseudocode

Algorithm 1: Director

Initialize replay buffer and neural networks.
while *not converged* **do**
  `// Acting`
  Update model state $s_t \sim \mathrm{repr}(s_t \mid s_{t-1}, a_{t-1}, x_t)$.
  if $t \bmod K = 0$ **then**
    `// Update internal goal`
    Sample abstract action $z \sim \mathrm{mgr}(z \mid s_t)$.
    Decode into model state $g = \mathrm{dec}(z)$.
  Sample action $a_t \sim \mathrm{wkr}(a_t \mid s_t, g)$.
  Send action to environment and observe r_t and x_{t+1}.
  Add transition (x_t, a_t, r_t) to replay buffer.
  `// Learning`
  if $t \bmod E = 0$ **then**
    `// World Model`
    Draw sequence batch $\{(x, a, r)\}$ from replay buffer.
    Update world model on batch (Equation 2) and get states $\{s\}$.
    `// Goal Autoencoder`
    Update goal autoencoder on $\{s\}$ (Equation 4).
    `// Policies`
    Imagine trajectory $\{(\hat{s}, \hat{a}, \hat{g}, \hat{z})\}$ under model and policies starting from $\{s\}$.
    Predict extrinsic rewards $\{\mathrm{rew}(s)\}$.
    Compute exploration rewards $\{r^{\mathrm{expl}}\}$ (Equation 6).
    Compute goal rewards $\{r^{\mathrm{goal}}\}$ (Equation 8).
    Abstract trajectory to update manager (Equations 11 and 12).
    Split trajectory to update worker (Equations 11 and 12).

F Hyperparameters

Name	Symbol	Value
Parallel envs	—	4
Training every	E	16
MLP size	—	4×512
Activation	—	$\mathrm{LayerNorm} + \mathrm{ELU}$
Imagination horizon	H	16
Discount factor	γ	0.99
Goal duration	K	8
Goal autoencoder latents	L	8
Goal autoencoder classes	C	8
Goal autoencoder beta	β	1.0
Learning rate	—	10^{-4}
Weight decay	—	10^{-2}
Adam epsilon	ϵ	10^{-6}

Table F.1: Hyperparameters of Director. We use the same hyperparameters across all experiments. The hyperparameters for training the world model and optimizing the policies were left unchanged compared to DreamerV2 (Hafner et al., 2020a).

G Vector of Categoricals

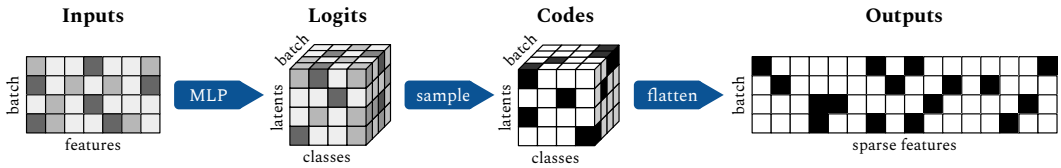


Figure G.1: Director uses a variational autoencoder to turn its state representations into discrete tokens that are easy to select between for the manager policy. For this, we use the vector of categoricals approach of DreamerV2 (Hafner et al., 2020a). For a given input vector, the encoder outputs a matrix of logits of L latent dimensions with C classes each. We sample from the logits and one-hot encode the result to obtain a sparse matrix of the same shape as the logits. To backpropagate gradients through this step, we simply use the gradient with respect to the one-hot matrix as the gradient with respect to the categorical probabilities (Bengio et al., 2013). The matrix is then flattened, resulting in a sparse representation with L out of the $L \times C$ feature dimensions set to one.

H Policy Optimization

Both the worker and manager policies of Director are optimized using the actor critic algorithm of Dreamer (Hafner et al., 2019; 2020a). For this, we use the world model to imagine a trajectory in its compact latent space. The actor network is optimized via policy gradients (Williams, 1992) with a learned state-value critic for variance reduction and to estimate rewards beyond the rollout:

$$\text{Actor:} \quad \pi(a_t \mid s_t) \qquad \text{Critic:} \quad v(s_t) \tag{9}$$

The world model allows cheaply generating as much on-policy experience as needed, so no importance weighting or clipping applies (Schulman et al., 2017). To train both actor and critic from an imagined trajectory of length H, λ-returns are computed from the sequence of rewards and predicted values (Sutton and Barto, 2018):

$$V_t^\lambda \doteq r_t + \gamma\Big((1-\lambda)v(s_{t+1}) + \lambda V_{t+1}^\lambda\Big), \quad V_H^\lambda \doteq v(s_H). \tag{10}$$

The λ-returns are averages over multi-step returns of different lengths, thus finding a trade-off between incorporating further ahead rewards quickly and reducing the variance of long sampled returns. The critic is learned by regressing the λ-returns via a squared loss, where $\mathrm{sg}(\cdot)$ indicates stopping the gradient around the targets:

$$\mathcal{L}(v) \doteq \mathrm{E}_{p_\phi,\pi}\Big[\textstyle\sum_{t=1}^{H-1} \frac{1}{2}\big(v(s_t) - \mathrm{sg}(V_t^\lambda)\big)^2\Big]. \tag{11}$$

The actor is updated by policy gradients on the same λ-returns, from which we subtract the state-value $v(s_t)$ as a baseline that does not depend on the current action for variance reduction. The second term in the actor objective, weighted by the scalar hyperparameter η, encourages policy entropy to avoid overconfidence and ensures that the actor explores different actions:

$$\mathcal{L}(\pi) \doteq -\mathrm{E}_{\pi,p_\phi}\Big[\textstyle\sum_{t=1}^{H} \ln \pi(a_t \mid s_t)\,\mathrm{sg}(V_t^\lambda - v(s_t)) + \eta\,\mathrm{H}\big[\pi(a_t \mid s_t)\big]\Big] \tag{12}$$

When there are multiple reward signals, such as the task and exploration rewards for the manager, we learn separate critics (Burda et al., 2018) for them and compute separate returns, which we normalize by their exponential moving standard deviation with decay rate 0.999. The baselines are normalized by the same statistics and the weighted average of the advantages is used for updating the policy.

I Further Related Work

Pretraining tasks One way to integrate domain knowledge into hierarchical agents is to learn primitives on simpler tasks and then compose them to solve more complex tasks afterwards. Learning primitives from manually specified tasks simplifies learning but requires human effort and limits the generality of the skills. DSN (Tessler et al., 2017) explicitly specifies reward functions for the low-level policies and then trains a high-level policy on top to solve Minecraft tasks. MLSH (Frans et al., 2017) pretrains a hierarchy with separate low-level policies on easier tasks, alternating update phases between the two levels. HeLMS (Rao et al., 2021) learns reusable robot manipulation skills from a given diverse dataset. MODAC (Veeriah et al., 2021) uses meta gradients to learn low-level policies that are helpful for solving tasks. However, all these approaches require manually specifying a diverse distribution of training tasks, and it is unclear whether generalization beyond the training tasks will be possible. Instead of relying on task rewards for learning skills, Director learns the worker as a goal-conditioned policy with a dense similarity function in feature space.

Mutual information Mutual information approaches allow automatic discovery of skills that lead to future states. VIC (Gregor et al., 2016) introduced a scalable recipe for discovering skills by rewarding the worker policy for reaching states from which the latent skill can be accurately predicted, effectively clustering the trajectory space. Several variations of this approach have been developed with further improvements in stability and diversity of skills, including SSN4HRL (Florensa et al., 2017), DIAYN (Eysenbach et al., 2018) and VALOR (Achiam et al., 2018). DISCERN (Warde-Farley et al., 2018) and CIC (Laskin et al., 2022) learn a more flexible similarity function between skills and states through contrastive learning. DADS (Sharma et al., 2019) estimates the mutual information in state-space through a contrastive objective with a learned model. DISDAIN overcomes a collapse problem by incentivizing exploration through ensemble disagreement (Strouse et al., 2021). LSP (Xie et al., 2020) learns a world model to discover skills that have a high influence on future representations rather than inputs. While these approaches are promising, open challenges include learning more diverse skills without dropping modes and learning skills that are precise enough for solving tasks.

Goal reaching Learning the low-level controller as a goal-conditioned policy offers a stable and general learning signal. HER (Andrychowicz et al., 2017) popularized learning goal-conditioned policies by combining a sparse reward signal with hindsight, which HAC (Levy et al., 2017) applied to learn an interpretable hierarchy with three levels by manually designing task-relevant goal spaces for each task. Instead of relying on hindsight with a sparse reward, HIRO (Nachum et al., 2018a) employs a dense reward in observation space based on the L2 norm and solves Ant Maze tasks given privileged information. NORL (Nachum et al., 2018b) introduces a representation learning objective to learn a goal space from a low-resolution top-down image instead but still uses a dense ground-truth reward for learning the manager. HSD-3 (Gehring et al., 2021) also learns a hierarchy with three levels and uses the robot joint space combined with a mask to specify partial goals. While not learning a high-level policy, RIG (Nair et al., 2018) computes goal rewards in the latent space of an autoencoder, allowing them to reach simple visual goals. FuN (Vezhnevets et al., 2017) provides task and goal rewards to its low-level policy and is trained from pixels but provides limited benefits over a flat policy trained on task reward. SecTAR (Co-Reyes et al., 2018) learns a sequence autoencoder that serves to propose goals and compute distances for low-dimensional environments and shows fast learning in low-dimensional environments with sparse rewards by high-level planning. DDL (Hartikainen et al., 2019) learns to reach goals from pixels by learning a temporal distance function. LEXA (Mendonca et al., 2021) learns a goal conditioned policy inside of a world model that achieves complex multi-object goal images, but assumes goals to be specified by the user. By comparison, Director solves difficult sparse reward tasks end-to-end from pixels without requiring domain-specific knowledge by learning a world model.

J Full Visual Control Suite

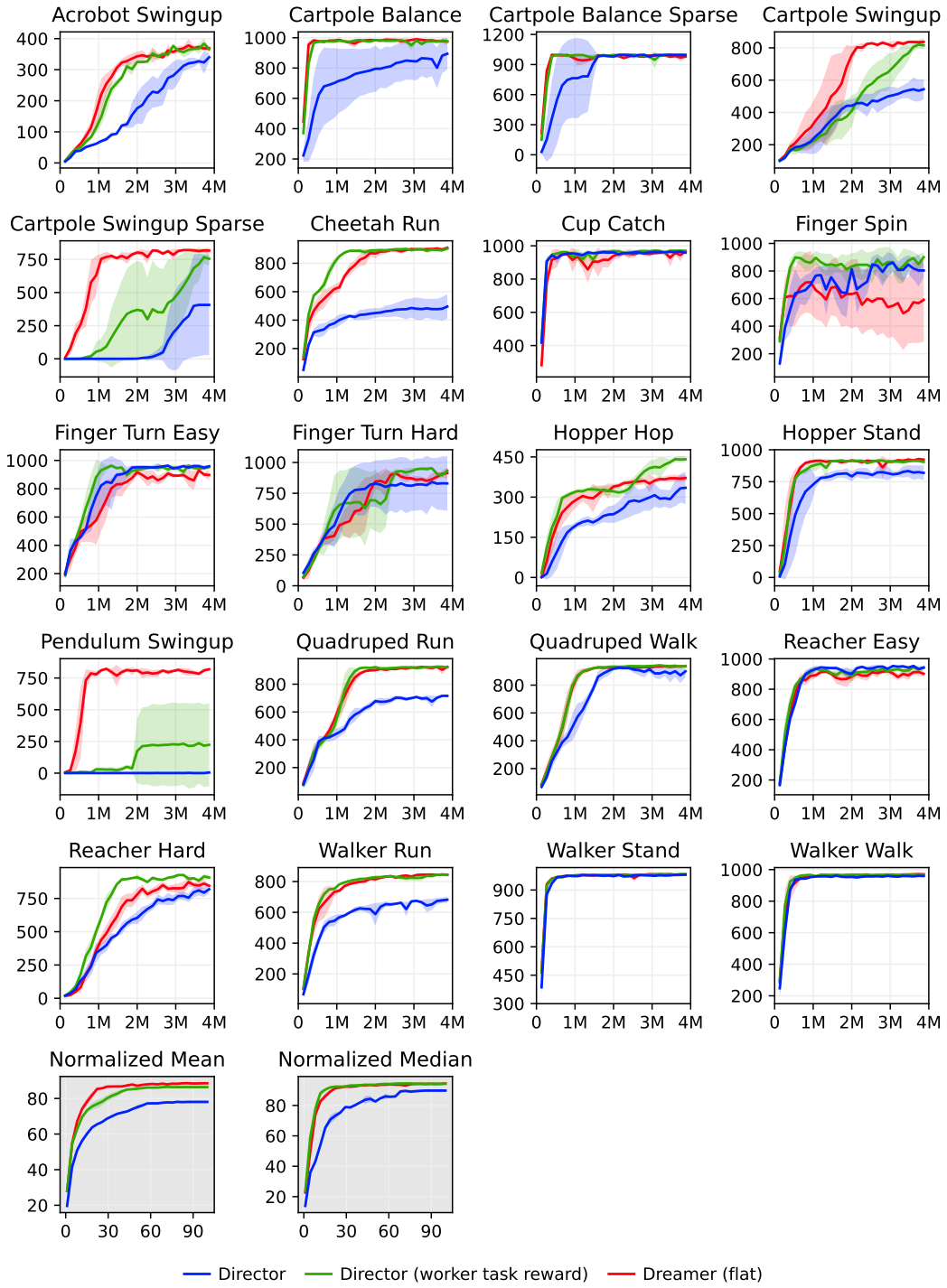


Figure J.1: To test the generality of Director, we evaluate it on a diverse set of 20 visual control tasks (Tassa et al., 2018) without action repeat. We find that Director solves a wide range of tasks despite giving no task reward to the worker, for the first time in the literature demonstrating a hierarchy with task-agnostic worker performing reliably across many tasks. When additionally providing task reward to the worker, performance reaches the state-of-the-art of Dreamer and even exceeds it on some tasks. These experiments use no action repeat and train every 16 environment steps, resulting in faster wall clock time and lower sample efficiency than the results reported by Hafner et al. (2019).

K Full Atari Suite

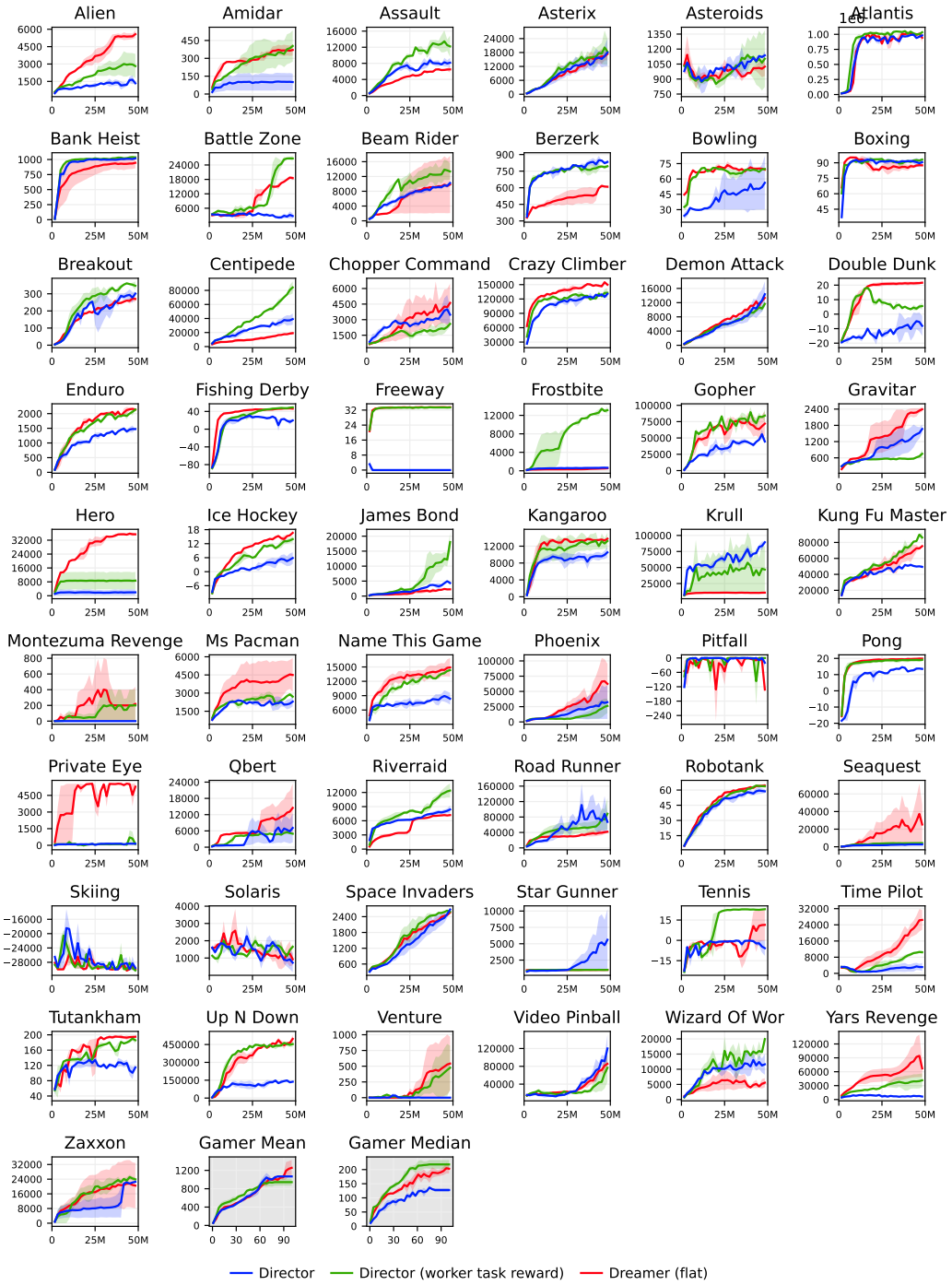
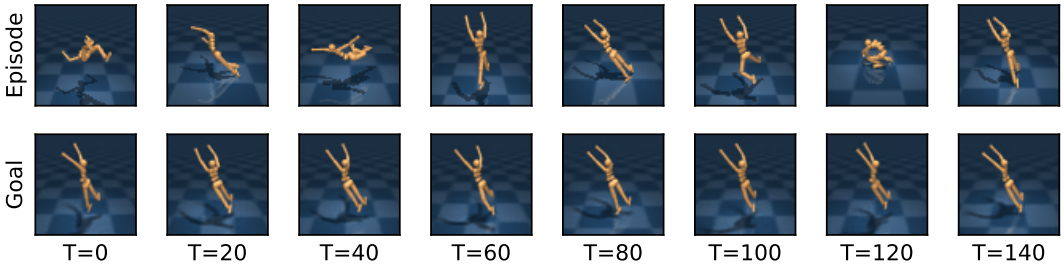


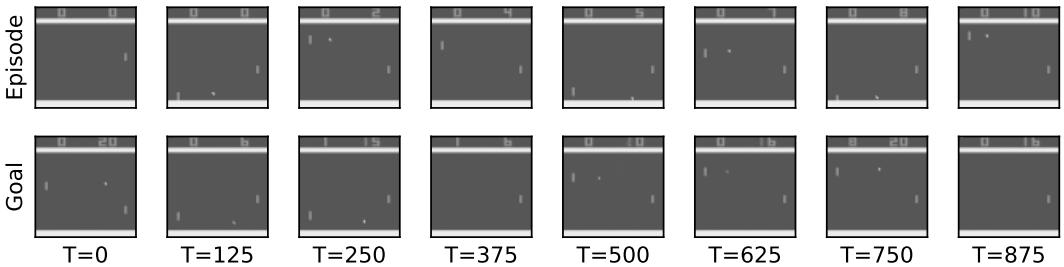
Figure K.1: To test the generality of Director, we evaluate it on 55 Atari games (Bellemare et al., 2013). We find that Director solves a wide range of tasks despite giving no task reward to the worker. Director is the first approach in the literature that demonstrates a hierarchy with task-agnostic worker performing reliably across many tasks. When additionally providing task reward to the worker, performance reaches that of Dreamer and even exceeds its human-normalized median score. These experiments use no action repeat and train every 16 environment steps, resulting in faster wall clock time and lower sample efficiency than the results reported by Hafner et al. (2019).

L Additional Goal Visualizations

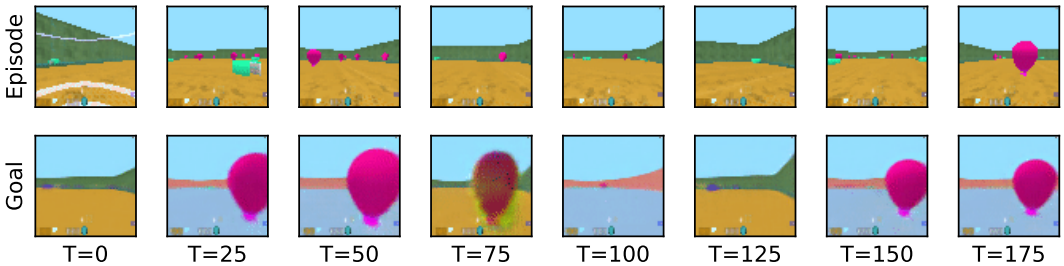
(a) Quadruped Walk. The manager learns to abstract away leg movements by requesting a forward leaning pose with shifting floor pattern, and the worker fills in the leg movements. The underlying latent goals are Markov states and thus likely contain a description of high forward velocity.



(b) Humanoid Walk. The manager learns to direct the worker via an extended pose with open arms, which causes the worker to perform energetic forward jumps that are effective at moving the robot forward. While the visualizations cannot show this, the underlying latent goals are Markovian and can contain velocity information.



(c) Pong. The manager directs the worker simply by requesting a higher score via the score display at the top of the screen. The worker follows the command by outplaying the opponent. We find that for tasks that are easy to learn for the worker, the manager frequently chooses this hands-off approach to managing.



(d) DMLab Collect Good. The manager steers through the arena by targeting the good objects and avoiding the bad ones. The worker succeeds at reaching the balloons regardless of the wall textures, showing that it learns to focus on achievable aspects of the goals without being distracted by non-achievable aspects.

Figure L.1: Additional goal visualizations.