Discovering and Achieving Goals via World Models

Russell Mendonca*
Carnegie Mellon University

Oleh Rybkin*
University of Pennsylvania

Kostas Daniilidis
University of Pennsylvania

Danijar Hafner
University of Toronto

Deepak Pathak
Carnegie Mellon University

Abstract

How can artificial agents learn to solve many diverse tasks in complex visual environments without any supervision? We decompose this question into two challenges: discovering new goals and learning to reliably achieve them. Our proposed agent, Latent Explorer Achiever (LEXA), addresses both challenges by learning a world model from image inputs and using it to train an explorer and an achiever policy via imagined rollouts. Unlike prior methods that explore by reaching previously visited states, the explorer plans to discover unseen surprising states through foresight, which are then used as diverse targets for the achiever to practice. After the unsupervised phase, LEXA solves tasks specified as goal images zero-shot without any additional learning. LEXA substantially outperforms previous approaches to unsupervised goal reaching, both on prior benchmarks and on a new challenging benchmark with 40 test tasks spanning across four robotic manipulation and locomotion domains. LEXA further achieves goals that require interacting with multiple objects in sequence.

1 Introduction

How can we build an agent that learns to solve hundreds of tasks in complex visual environments, such as rearranging objects with a robot arm or completing chores in a kitchen? While traditional reinforcement learning (RL) has been successful for individual tasks, it requires a substantial amount of human effort for every new task. Specifying task rewards requires domain knowledge, access to object positions, is time-consuming, and prone to human errors. Moreover, traditional RL would require environment interaction to explore and practice in the environment for every new task. Instead, we approach learning hundreds of tasks through the paradigm of unsupervised goal-conditioned RL, where the agent learns many diverse skills in the environment in the complete absence of supervision, to later solve tasks via user-specified goal images immediately without further training [2, 27, 41].

Challenges Exploring the environment and learning to solve many different tasks is substantially more challenging than traditional RL with a dense reward function or learning from

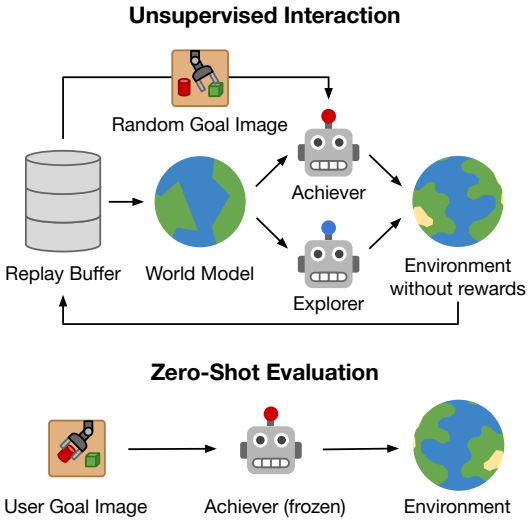

Figure 1: LEXA learns a world model without any supervision, and leverages it to train two policies in imagination. The *explorer* finds new images and the *achiever* learns to reliably reach them. Once trained, the achiever reaches user-specified goals zero-shot without further training at test time.

* Equal contribution. Ordering determined at random. Project page: https://orybkin.github.io/lexa/

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

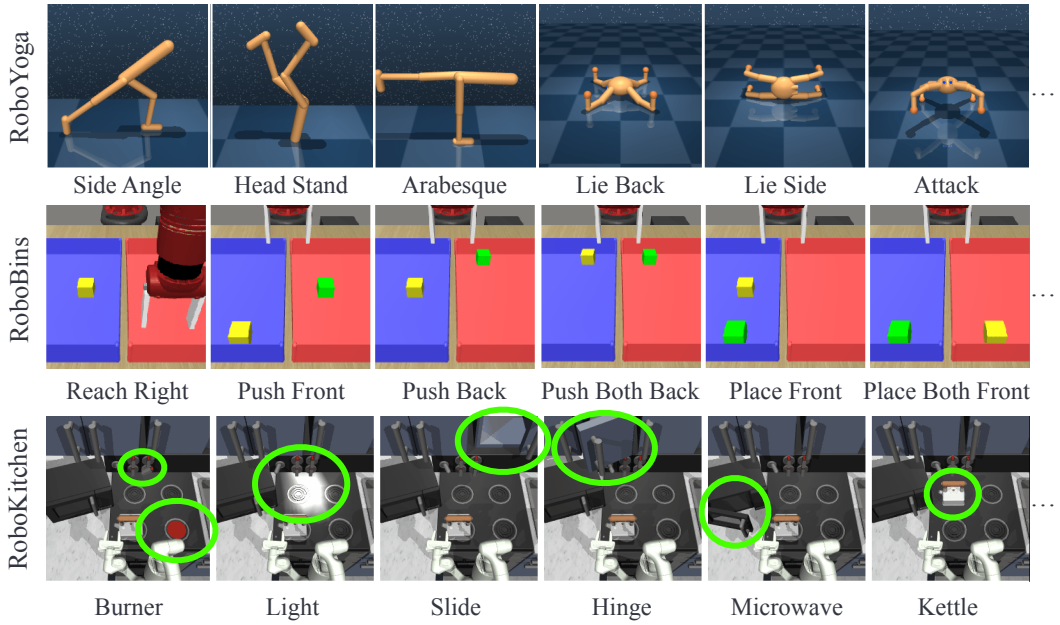

Figure 2: We benchmark LEXA across four visual control environments. A representative sample of the test-time goals is shown here. RoboYoga features complex locomotion and precise control of high-dimensional agents, RoboBins manipulation with multiple objects, and RoboKitchen a variety of diverse tasks that require complex control strategies such as opening a cabinet.

expert demonstrations. Existing methods are limited to simple tasks, such as picking or pushing a puck [13, 33, 38] or controlling simple 2D robots [51]. The key challenge in improving the performance of unsupervised RL is *exploration*. In particular, previous approaches explore by either revisiting previously seen rare goals [14, 18, 56] or sampling goals from a generative model [33, 38]. However, in both these approaches, the policy as well as the generative model are trained on previously visited states from the replay buffer, and hence the sampled goals are either within or near the frontier of agent's experience. Ideally, we would like the agent to discover goals much beyond its frontier for efficient exploration, but how does an agent generate goals that it is yet to encounter? This is an open question not just for AI but for cognitive science too [43].

Approach To rectify this issue, we leverage a learned world model to train a separate *explorer* and *achiever* policy in imagination. Instead of randomly sampling or generating goals, our explorer policy discovers distant goals by first planning a sequence of actions optimized in imagination of the world model to find novel states with high expected information gain [31, 44, 45]. It then executes those imagined actions in the environment to discover interesting states without the need to generate them. Note these actions are likely to lead the agent to states which are several steps outside the frontier because otherwise the model wouldn't have had high uncertainty or information gain. Finally, these discovered states are used as diverse targets for the achiever to practice. We train the achiever from on-policy imagination rollouts within the world model and without relying on experience relabeling, therefore leveraging *foresight over hindsight*. After this unsupervised training phase, the achiever solves tasks specified as goal images zero-shot without any additional learning at deployment. Unlike in the conventional RL paradigm [32, 48], our method is trained once and then used to achieve several tasks at test time without any supervision during training or testing.

Contributions We introduce Latent Explorer Achiever (LEXA), an unsupervised goal reaching agent that trains an explorer and an achiever within a shared world model. At training, LEXA unlocks diverse data for goal reaching in environments where exploration is nontrivial. At test time, the achiever solves challenging locomotion and manipulation tasks provided as user-specified goal images. Our contributions are summarized as follows:

- We propose to learn separate explorer and achiever policies as an approach to overcome the exploration problem of unsupervised goal-conditioned RL.
- We show that forward-looking exploration by planning with a learned world model substantially outperforms previous strategies for goal exploration.
- To evaluate on challenging tasks, we introduce a new goal reaching benchmark with a total of 40 diverse goal images across 4 different robot locomotion and manipulation environments.
- LEXA outperforms prior methods, being the first to show success in the Kitchen robotic manipulation environment, and achieves goal images where multiple objects need to be moved.

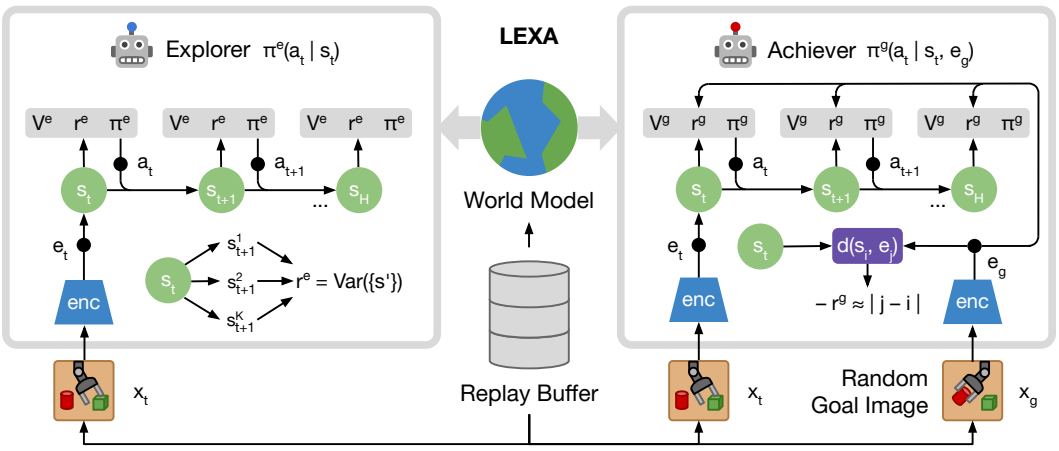

Figure 3: Latent Explorer Achiever (LEXA) learns a general world model that is used to train an explorer and a goal achiever policy. The explorer (left) is trained on imagined latent state rollouts of the world model $s_{t:T}$ to maximize the disagreement objective $r^e_t = \mathrm{Var}(s')$. The goal achiever (right) is conditioned on a goal g and is also trained on imagined rollouts to minimize a distance function $d(s_t, e_g)$. Goals are sampled randomly from replay buffer images. For training a temporal distance, we use the imagined rollouts of the achiever and predict the number of time steps between each two states. By combining forward-looking exploration and data-efficient training of the achiever, LEXA provides a simple and powerful solution for unsupervised reinforcement learning.

2 Latent Explorer Achiever (LEXA)

Our aim is to build an agent that can achieve arbitrary user-specified goals after learning in the environment without any supervision. This presents two challenges - collecting trajectories that contain diverse goals and learning to achieve these goals when specified as a goal image. We introduce a simple solution based on a world model and imagination training that addresses both challenges. The world model represents the agent's current knowledge about the environment and is used for training two policies, the explorer and the achiever. To explore novel situations, we construct an estimate of which states the world model is still uncertain about. To achieve goals, we train the goal-conditioned achiever in imagination, using the images found so far as unsupervised goals. At test time, the achiever is deployed to reach user-specified goals. The training procedure is in Algorithm 1.

2.1 World Model

To efficiently predict potential outcomes of future actions in environments with high-dimensional image inputs, we leverage a Recurrent State Space Model (RSSM) [23] that learns to predict forward using compact model states that facilitate planning [7, 52]. In contrast to predicting forward in image space, the model states enable efficient parallel planning with a large batch size and can reduce accumulating errors [40]. The world model consists of the following components:

$$\begin{aligned} &\text{Encoder:} && e_t = \mathrm{enc}_\phi(x_t) && \text{Posterior:} && q_\phi(s_t \mid s_{t-1}, a_{t-1}, e_t) \\ &\text{Dynamics:} && p_\phi(s_t \mid s_{t-1}, a_{t-1}) && \text{Image decoder:} && p_\phi(x_t \mid s_t) \end{aligned} \tag{1}$$

The model states s_t contain a deterministic component h_t and a stochastic component z_t with diagonal-covariance Gaussian distribution. h_t is the recurrent state of a Gated Recurrent Unit (GRU) [11]. The encoder and decoder are convolutional neural networks (CNNs) and the remaining components are multi-layer perceptrons (MLPs). The world model is trained end-to-end by optimizing the evidence lower bound (ELBO) via stochastic backpropagation [29, 39] with the Adam optimizer [28].

2.2 Explorer

To efficiently explore, we seek out surprising states imagined by the world model [6, 42, 44, 45, 47], as opposed to retrospectively exploring by revisiting previously novel states [4, 5, 8, 35]. As the world model can predict model states that correspond to unseen situations in the environment, the imagined trajectories contain more novel goals, compared to model-free exploration that is limited to the replay buffer. To collect informative novel trajectories in the environment, we train an exploration

Algorithm 1: Latent Explorer Achiever (LEXA)

initialize: World model $\mathcal{M}$, Replay buffer $\mathcal{D}$, Explorer $\pi^{\mathrm{e}}(a_t \mid z_t)$, Achiever $\pi^{\mathrm{g}}(a_t \mid z_t, g)$
while exploring **do**
  Train $\mathcal{M}$ on $\mathcal{D}$
  Train π^{e} in imagination of $\mathcal{M}$ to maximize exploration rewards $\sum_t r_t^{\mathrm{e}}$.
  Train π^{g} in imagination of $\mathcal{M}$ to maximize $\sum_t r_t^{\mathrm{g}}(z_t, g)$ for images $g \sim \mathcal{D}$.
  (Optional) Train $d(z_i, z_j)$ to predict distances $j - i$ on the imagination data from last step.
  Deploy π^{e} in the environment to explore and grow $\mathcal{D}$.
  Deploy π^{g} in the environment to achieve a goal image $g \sim \mathcal{D}$ to grow $\mathcal{D}$.
end while
while evaluating **do**
  given: Evaluation goal g
  Deploy π^{g} in the world to reach g.
end while

policy π^e from the model states s_t in imagination of the world model to maximize an exploration reward:

$$\text{Explorer:} \quad \pi^e(a_t \mid s_t) \qquad \text{Explorer Value:} \quad v^e(s_t) \tag{2}$$

To explore the most informative model states, we estimate the epistemic uncertainty as a disagreement of an ensemble of transition functions. We train an ensemble of 1-step models to predict the next model state from the current model state. The ensemble model is trained alongside the world model on model states produced by the encoder q_ϕ. Because the ensemble models are initialized at random, they will differ, especially for inputs that they have not been trained on [30, 37]:

$$\text{Ensemble:} \quad f(s_t, \theta^k) = \hat{z}_{t+1}^k \quad \text{for} \quad k = 1..K \tag{3}$$

Leveraging the ensemble, we estimate the epistemic uncertainty as the ensemble disagreement. The exploration reward is the variance of the ensemble predictions averaged across dimension of the model state, which approximates the expected information gain [3, 44]:

$$r_t^{\mathrm{e}}(s_t) \doteq \frac{1}{N} \sum_n \mathrm{Var}_{\{\mathrm{k}\}} \left[f(s_t, \theta_k) \right]_n \tag{4}$$

The explorer π^e maximizes the sum of future exploration rewards r_t^e using the Dreamer algorithm [24], which considers long-term rewards into the future by maximizing λ-returns under a learned value function. As a result, the explorer is trained to seek out situations are as informative as possible from imagined latent trajectories of the world model, and is periodically deployed in the environment to add novel trajectories to the replay buffer, so the world model and goal achiever policy can improve.

2.3 Achiever

To leverage the knowledge obtained by exploration for learning to reach goals, we train a goal achiever policy π^g that receives a model state and a goal as input. Our aim is to train a general policy that is capable of reaching many diverse goals. To achieve this in a data-efficient way, it is crucial that environment trajectories that were collected with one goal in mind are reused to also learn how to reach other goals. While prior work addressed this by goal relabeling which makes off-policy policy optimization a necessity [2], we instead leverage past trajectories via the world model trained on them that lets us generate an unlimited amount of new imagined trajectories for training the goal achiever on-policy in imagination. This simplifies policy optimization and can improve stability, while still sharing all collected experience across many goals.

$$\text{Achiever:} \quad \pi^g(a_t \mid s_t, e_g) \qquad \text{Achiever Value:} \quad v^g(s_t, e_g) \tag{5}$$

To train the goal achiever, we sample a goal image x_g from the replay buffer and compute its embedding $e_g = \mathrm{enc}_\phi(x_g)$. The achiever aims to maximize an unsupervised goal-reaching reward $r^g(s_t, e_g)$. We discuss different choices for this reward in Section 2.4. We again use the Dreamer algorithm [24] for training, where now the value function also receives the goal embedding as input.

In addition to imagination training, it can also be important to perform practice trials with the goal achiever in the true environment, so that any model inaccuracies along the goal reaching trajectories

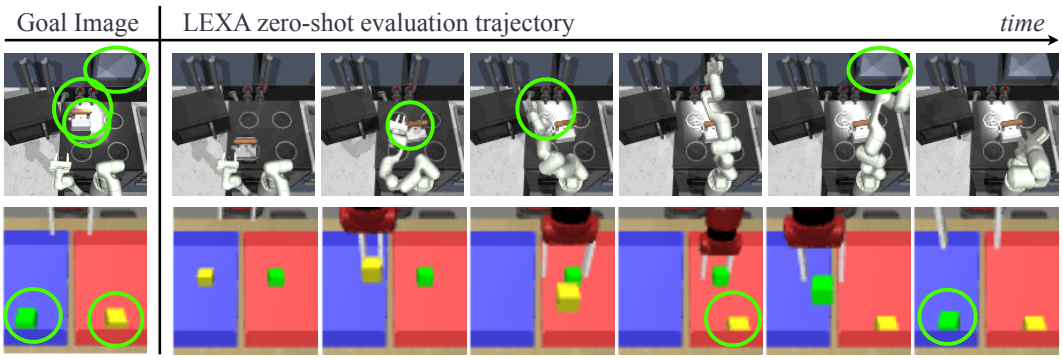

Figure 4: Successful LEXA trajectories. When given a goal image from the test set, LEXA's achiever is used in the environment to reach that image. On RoboKitchen, LEXA manipulates up to three different objects together from a single goal image (kettle, light switch, and cabinet). On RoboBins, LEXA performs temporally extended tasks such as picking and placing two objects in a row.

may be corrected. To perform practice trials, we sample a goal from the replay buffer and execute the goal achiever policy for that goal in the environment. These trials are interleaved with exploration episodes collected by the exploration policy in equal proportion. We note that the goal achiever learning is entirely unsupervised because the practice goals are simply images the agent encountered through exploration or during previous practice trails.

2.4 Latent Distances

Training the achiever policy requires us to define a goal achievement reward $r^g(s_t, e_g)$ that measures how close the latent state s_t should be considered to the goal e_g. One simple measure is the cosine distance in the latent space obtained by inputting image observations into the world-model. However, such a distance function brings *visually* similar states together even if they could be farther apart in *temporal* manner as measured by actions needed to reach from one to other. This bias makes this suitable only to scenarios where most of pixels in the observations are directly controllable, e.g., trying to arrange robot's body in certain shape, such as RoboYoga poses in Figure 2. However, many environments contain agent as well as the world, such as manipulation involves interacting with objects that are not directly controllable. The cosine distance would try matching the entire goal image, and thus places a large weight on both matching the robot and object positions with the desired goal. Since the robot position is directly controllable it is much easier to match, but this metric overly focuses on it, yielding poor policies that ignore objects. We address this is by using the number of timesteps it takes to move from one image to another as a distance measure [26, 27]. This ignores large changes in robot position, since these can be completed in very few steps, and will instead focus more on the objects. This temporal cost function can be learned purely in imagination rollouts from our world model allowing as much data as needed without taking any steps in the real world.

Cosine Distance To use cosine distance with LEXA, for a latent state s_t, and a goal embedding e^g, we use the latent inference network q to infer s^g, and define the reward as the cosine similarity [55]:

$$r^g_t(s_t, e_g) \doteq \sum_i \overline{s}_{ti}\overline{s}_{gi}, \quad \text{where} \quad \overline{s}_t = s_t/\|s_t\|_2, \quad \overline{s}_g = s_g/\|s_g\|_2, \tag{6}$$

i.e. the cosine of the angle between the two vectors s_t, s_g in the $N-$dimensional latent space.

Temporal Distance To use temporal distances with LEXA, we train a neural network d to predict the number of time steps between two embeddings. We train it by sampling pairs of states s_t, s_{t+k} from an imagined rollout of the achiever and predicting the distance k. We implement the temporal distance in terms of predicted image embeddings $\hat{e}_{t+k}$ in order to remove extra recurrent information:

$$\text{Predicted embedding:} \quad \text{emb}(s_t) = \hat{e}_t \approx e_t \quad \text{Temporal distance:} \quad d_\omega(\hat{e}_t, \hat{e}_{t+k}) \approx k/H, \tag{7}$$

where H is the maximum distance equal to the imagination horizon. Training distance function only on imagination data from the same trajectory would cause it to predict poor distance to far away states coming from other trajectories, such as images that are impossible to reach during one episode. In order to incorporate learning signal from such far-away goals, we include them by sampling images

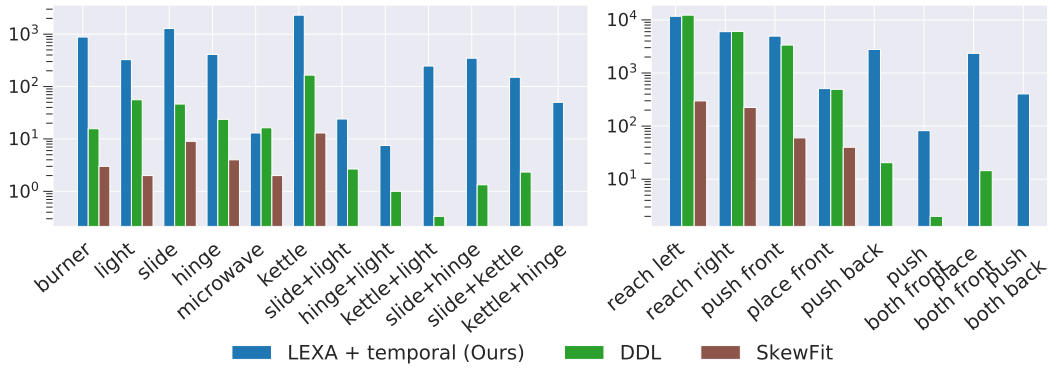

Figure 5: Coincidental goal success achieved during the unsupervised exploration phase. The forward-looking explorer policy of LEXA results in substantially better coverage compared to SkewFit, a popular method for goal based exploration.

from a different trajectory. We annotate these negative samples with the maximum possible distance, so that the agent always prefers images that were seen in the same trajectory.

$$r_t^g(s_t, e_g) = -d_\omega(\hat{e}_t, e_g), \quad \text{where} \quad \hat{e}_t = \text{emb}(s_t), \quad e_g = \text{enc}_\phi(x_g) \tag{8}$$

The learned distance function depends on the training data policy. However, as the policy becomes more competent, the distance estimates will be closer to the optimal number of time steps to reach a particular goal, and the policy converges to the optimal solution [26]. LEXA always uses the latest data to train the distance function using imagination, ensuring that the convergence is fast.

3 Experiments

Our evaluation focuses on the following scientific questions:

1. Does LEXA outperform prior work on previous benchmarks and a new challenging benchmark?
2. How does forward-looking exploration of goals compare to previous goal exploration strategies?
3. How does the distance function affect the ability to reach goals in different types of environments?
4. Can we train one general LEXA to control different robots across visually distinct environments?
5. What components of LEXA are important for performance?

We evaluate LEXA on prior benchmarks used by SkewFit [38], DISCERN [51], and Plan2Explore [44] in Section 3.3. Since these benchmarks are largely saturated, we also introduce a new challenging benchmark shown in Figure 2. We evaluate LEXA on this benchmark is Section 3.2.

3.1 Experimental setup

As not many prior methods have shown success on reaching diverse goals from image inputs, we perform an apples-to-apples comparison by implementing the baselines using the same world model and policy optimization as our method:

- **SkewFit** SkewFit [38] uses model-free hindsight experience replay and explores by sampling goals from the latent space of a variational autoencoder [29, 39]. Being one of the state-of-the-art agents, we use the original implementation that does not use a world model or explorer policy.
- **DDL** Dynamic Distance Learning [26] trains a temporal distance function similar to our method. Following the original algorithm, DDL uses greedy exploration and trains the distance function on the replay buffer instead of in imagination.
- **DIAYN** Diversity is All You Need [15] learns a latent skill space and uses mutual information between skills and reached states as the objective. We augment DIAYN with our explorer policy and train a learned skill predictor to obtain a skill for a given test image [12].
- **GCSL** Goal-Conditioned Supervised Learning [20] trains the goal policy on replay buffer goals and mimics the actions that previously led to the goal. We also augment GCSL with our explorer policy, as we found no learning success without it.

Our new benchmark defines goal images for a diverse set of four existing environments as follows:

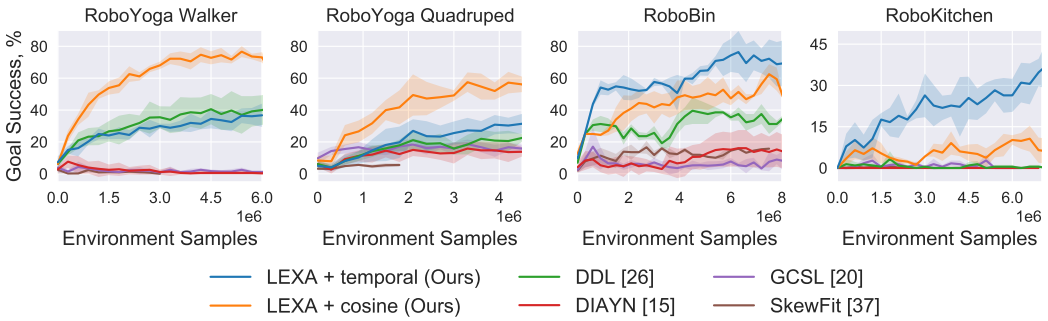

Figure 6: Evaluation of goal reaching agents on our four benchmarks. A single agent is trained from images without rewards and then evaluated on reaching goal images from the test set (see Figure 1). Both LEXA agents solve many of the tasks and significantly outperform prior work. SkewFit and DLL struggle with exploration, while DIAYN and GCSL use our explorer but still are not able to learn a good downstream policy. Refer table 1 for final success percentage (averaged across tasks) for each method and benchmark domain.

- **RoboYoga** We use the walker and quadruped domains of the DeepMind Control Suite [49] to define the RoboYoga benchmark, consisting of 12 goal images that correspond to different body poses for each of the two environments, such as lying down, standing up, and balancing.
- **RoboBins** Based on MetaWorld [54], we create a scene with a Sawyer robotic arm, two bins, and two blocks of different colors. The goal images specify tasks that include reaching, manipulating only one block, and manipulating both blocks.
- **RoboKitchen** The last benchmark involves the challenging kitchen environment from [22], where a franka robot can interact with various objects including a burner, light switch, sliding cabinet, hinge cabinet, microwave, or kettle. The goal images we include describe tasks that require interacting with only one object, as well as interacting with two objects.

3.2 Performance on New Benchmark

We show the results on our main benchmark in Figure 6 and include heatmaps that show per-task success on each of the evaluation tasks from the benchmarks in the Appendix. Further, we report success averaged across tasks for each domain at the end of training in Table 1. We visualize example successful trajectory executions for tasks that require manipulating multiple objects in Fig. 4.

RoboYoga The environments in this benchmark are directly controllable since they contain no other objects except the robot. We recall that for such settings we expect the cosine distance to be effective, as perceptual distance is quite accurate. Training is thus faster compared to using learned temporal distances, where the metric is learned from scratch. From Table 1 and Figure 6 we see that this is indeed the case for these environments (Walker and Quadruped), as LEXA with the cosine metric outperforms all prior approaches. Furthermore with temporal distances LEXA makes better progress compared to prior work on a much larger number of goals as can be seen from the per-task performance (Figures A.2, A.3), even though average success over goals looks similar to that of DDL.

RoboBins This environment involves interaction with block objects, and thus is not directly controllable, and so we expect LEXA to perform better with the temporal distance metric. From Table 1 and

Method	Kitchen	RoboBins	Quadruped	Walker
DDL	0.00	35.42	22.50	40.00
DIAYN	0.00	13.69	13.81	0.28
GCSL	0.00	7.94	15.83	1.11
SkewFit	0.23	15.77	5.52	0.01
LEXA + Temporal (Ours)	**37.50**	**69.44**	31.39	36.72
LEXA + Cosine (Ours)	6.02	45.83	**56.11**	**73.06**

Table 1: Performance on our new challenging benchmark, spanning across the four domains shown in Figure 2. The number are goal success rates, averaged over test goals within each environment.

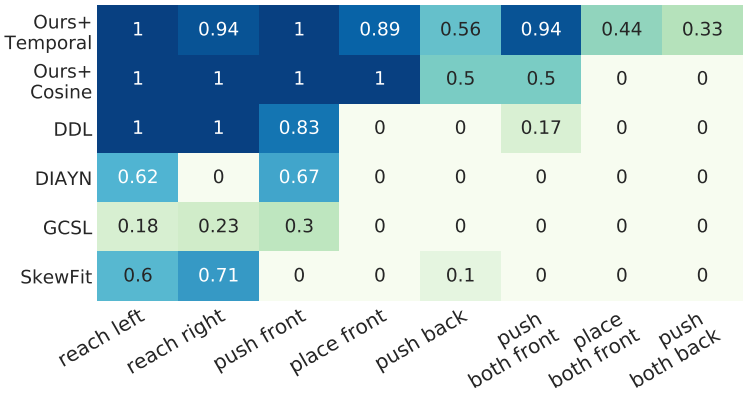

Figure 7: Success rates on RoboBin. In line with the prior literature, previous methods are successful at reaching and sometimes pushing. LEXA pushes the state-of-the-art by picking and placing multiple objects to reach challenging goal images. Analogous heat maps for the other domains are included in the appendix.

Figure 6, we see that LEXA gets higher average success than all prior approaches. Further from the per-task performance in 7, LEXA with the temporal distance metric is the only approach that makes progress on all goals in the benchmark. The main difference in performance between using temporal and cosine distance can be seen in the tasks involving two blocks, which are the most complex tasks in this environment (the last 3 columns of the per-task plot). The best performing prior method is DDL which solves reaching, and can perform simple pushing tasks. This method performs poorly due to poor exploration, as shown in Figure 5. We see that while other prior methods make some progress on reaching, they fail on harder tasks.

RoboKitchen This benchmark involves diverse objects that require different manipulation behavior. From Table 1 and Figures 6 and A.5 we find that LEXA with temporal distance is able to learn multiple RoboKitchen tasks, some of which require sequentially completing 2 tasks in the environment. All prior methods barely make progress due to the challenging nature of this benchmark, and furthermore using the cosine distance function makes very limited progress. The gap in performance between using the two distance functions is much larger in this environment compared to RoboBins since there are many more objects and they are not as clearly visible as the blocks.

Single Agent Across All Environments

In the previous sections we have shown that our approach can achieve diverse goals in different environments. However, we trained a new agent for every new environment, which doesn't scale well to large numbers of environments. Thus we investigate if we can train a train a single agent across four environments in the benchmark. From Figure A.6 we see that our approach with learned temporal distance is able to make progress on tasks from RoboKitchen, RoboBins Reaching, RoboBins Pick & Place and Walker, while the best prior method on the single-environment tasks (DDL) mainly solves walker tasks and reaching from RoboBin.

3.3 Performance on Prior benchmarks

To further verify the results obtained on our benchmark, we evaluate LEXA on previously used benchmarks. We observe that LEXA significantly outperforms prior work on these benchmarks, and is often close to the optimal policy. Additional details are provided in Appendix A.

Table 2: Goal distance for SkewFit goals [38].

Method	Pusher	Pickup
RIG [33]	7.7cm	3.7cm
RIG + HER [2]	7.5cm	3.5cm
Skew-Fit [38]	4.9cm	1.8cm
LEXA + Temporal	**2.3cm**	**1.4cm**

SkewFit Benchmark SkewFit [38] introduces a robotic manipulation benchmark for unsupervised methods with simple tasks like planar pushing or picking. We evaluate on this benchmark in Table 2. Baseline results are taken from [38]. LEXA significantly outperforms prior work on these tasks. Pushing and picking up blocks from images is largely solved and future work can focus on harder benchmarks such as those introduced in our paper.

DISCERN Benchmark We attempted to replicate the tasks described in [51] that are based on simple two-dimensional robots [49]. While the original tasks are not released, we followed the procedure for generating the goals described in the paper. Despite following the exact procedure, we were not able to obtain similar goals to the ones used in the original paper. Nevertheless, we show the goal completion percentage results obtained with our reproduced evaluation compared to DISCERN results from the original paper. LEXA results were obtained with early stopping. In Table 3 we see that our agent solves many of the tasks in this benchmark.

Table 3: Success for DISCERN goals [51].

Task	LEXA	DISCERN
Cup	**84.0%**	76.5%
Cartpole	**35.9%**	21.3%
Finger	**40.9%**	21.8%
Pendulum	**79.1%**	75.7%
Pointmass	**83.2%**	49.6%
Reacher	**100.0%**	87.1%

Plan2Explore Benchmark We provide a comparison on the standard reward-based DM control tasks [49] in Table 4. To compare on this benchmark, we create goal images that correspond to the reward functions. This setup is arguably harder for our agent, but is much more practical. Note our agent never observes the reward function and only observes the goal at test time. Plan2Explore adapts to new tasks but it needs the reward function to be known at test time, while DrQV2 is an oracle agent that observes the reward at training time. Baseline results are taken from [44, 53]. LEXA results were obtained with early stopping. LEXA outperforms Plan2Explore on most tasks and even performs comparably to state of the art oracle agents (DrQ, DrQv2, Dreamer) that use true task rewards during training.

Table 4: Zero-shot return on P2E tasks [44].

Task	LEXA	P2E	DrQv2
Zero-Shot	✓	✓*	✗
Walker Stand	**957**	331	968
Hopper Stand	**840**	**841**	957
Cartpole Balance	886	**950**	989
Cartpole Bal. Sparse	**996**	860	983
Pendulum Swing Up	**788**	**792**	837
Cup Catch	**969**	**962**	909
Reacher Hard	**937**	66	970

3.4 Analysis

Prior work Most work we compared against struggles with exploration, such as SkewFit and DLL methods. DIAYN is augmented with our explorer, but still fails to leverage the exploration data to learn a diverse set of skills. GCSL struggles to fit the exploration data and produces behavior that does not solve the task, perhaps because the exploration data is too diverse. We observed that all baselines make progress on the simple reaching, but struggle with other tasks. We have experimented with several versions and improvements to the baselines and report the best obtained performance.

Ablation of different components We ablated components of LEXA on the RoboBins environment in Figure 8. Using a separate explorer policy crucial as without it the agent does not discover the more interesting tasks. Without negative sampling the agent learns slower, perhaps because the distance function doesn't produce reasonable outputs when queried on images that are more than horizon length apart. Training the distance function with real data converges to slightly lower success than using imagination data, since real data is sampled in an off-policy manner due to its limited quantity.

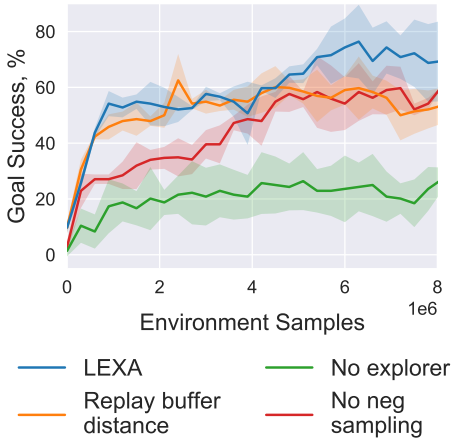

Figure 8: Ablations on RoboBins. A separate explorer is crucial for most tasks. Training temporal distance on negative samples speeds up learning, and both negative sampling and training in imagination as opposed to real data are important for the hardest tasks.

Exploration performance Due to importance of exploration, we further examine the diversity of the data collected during training. We log the instances where the agent coincidentally solves an evaluation task during exploration, for the RoboKitchen and RoboBins environments. In Figure 5, we see that our method encounters harder tasks involving multiple objects much more often.

4 Related Work

Learning to Achieve Goals The problem of learning to reach many different goals has been commonly addressed with model-free methods that learn a single goal-conditioned policy [2, 27, 41]. Recent work has combined these approaches with various ways to generate training goals, such as asymmetric self-play [34, 46] or by sampling goals of intermediate difficulty [14, 18]. These approaches can achieve remarkable performance in simulated robotic domains, however, they focus on the settings where the agent can directly perceive the low-dimensional environment state.

A few works have attempted to scale these model-free methods to visual goals by using contrastive [51] or reconstructive [33, 38] representation learning. However, these approaches struggle to perform meaningful exploration as no clear reward signal is available to guide the agent toward solving interesting tasks. Some works [10, 50] avoid this challenge by using a large dataset of interesting behaviors. Other works [38, 56] attempt to explore by generating goals similar to those that have already been seen, but do not try to explore truly novel states.

A particularly relevant set of approaches used model-based methods to achieve goals via planning [13, 17] or learning model-regularized policies [36]. However, these approaches are limited by short planning horizons. In contrast, we learn long-horizon goal-conditioned value functions which allows us to solve more challenging tasks. More generally, most of the above approaches are limited by simplistic exploration, while our method leverages model imagination to search for novel states, which significantly improves exploration and in turn the downstream capabilities of the agent.

Learning Distance Functions A crucial challenge for visual goal reaching is the choice of the reward or the cost function for the goal achieving policy. Several approaches use representation learning to create a distance in the feature space [9, 33, 51, 52]. However, this naive distance may not be most reflective of how hard a particular goal is to reach. One line of research has proposed using the mutual information between the current state and the goal as the distance metric [1, 12, 15, 21], however, it remains to be seen whether this approach can scale to more complex tasks.

Other works proposed temporal distances that measure the amount of time it takes to reach the goal. One approach is to learn the distance with approximate dynamic programming using Q-learning methods [16, 19, 27]. Our distance function is most similar to Hartikainen et al. [26], who learn a temporal distance with supervised learning on recent policy experience. In contrast to [26], we always train the distance on-policy in imagination, and we further integrate this achiever policy into our latent explorer achiever framework to discover novel goals for the achiever to practice on.

5 Conclusion

We presented Latent Explorer Achiever (LEXA), an agent for unsupervised RL that explores its environment, learns to achieve the discovered goals, and solves image-based tasks at test time in zero-shot way. By searching for novelty in imagination, LEXA explores better and discovers meaningful behaviors in substantially more diverse environments than considered by prior work. Further, LEXA is able to solve challenging downstream tasks specified as images without any supervision such as rewards or demonstrations. By proposing a challenging benchmark and the first agent to achieve meaningful performance on these tasks, we hope to stimulate future research on unsupervised agents, which we believe are fundamentally more scalable than traditional agents that require a human to design the tasks and rewards for learning.

Many challenges remain for building effective unsupervised agents. Many of the tasks in our proposed benchmark are still unsolved and there remains room for progress on the algorithmic side both for the world model and for the policy optimization parts. Further, it is important to demonstrate the benefits of unsupervised agents on real-world systems to verify their scalability. Finally, for widespread adoption, it is crucial to consider the problem of goal specification and design methods that act on goals that are easy to specify, such as via natural language. We believe LEXA will enable future work to tackle these goals effectively.

Acknowledgements We thank Ben Eysenbach, Stephen Tian, Sergey Levine, Dinesh Jayaraman, Karl Pertsch, Ed Hu and the members of GRASP lab and Pathak lab for insightful discussions. We also thank Murtaza Dalal and Chuning Zhu for help with MuJoCo environments. Finally, DP would like to thank Laura Schulz, Josh Tenenbaum and Alison Gopnik for seeding the idea of goal-setting in children, and Pulkit Agrawal for several crucial discussions (over gelato) since then. This work was supported by DARPA Machine Common Sense grant.

References

[1] J. Achiam, H. Edwards, D. Amodei, and P. Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018. 10

[2] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017. 1, 4, 8, 10, 15, 18

[3] P. Ball, J. Parker-Holder, A. Pacchiano, K. Choromanski, and S. Roberts. Ready policy one: World building through active learning. In *International Conference on Machine Learning*, pages 591–601. PMLR, 2020. 4

[4] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016. 3, 15

[5] L. Beyer, D. Vincent, O. Teboul, S. Gelly, M. Geist, and O. Pietquin. Mulex: Disentangling exploitation from exploration in deep rl. *arXiv preprint arXiv:1907.00868*, 2019. 3

[6] B. Bucher, K. Schmeckpeper, N. Matni, and K. Daniilidis. Adversarial curiosity. *arXiv preprint arXiv:2003.06082*, 2020. 3

[7] L. Buesing, T. Weber, S. Racaniere, S. Eslami, D. Rezende, D. P. Reichert, F. Viola, F. Besse, K. Gregor, D. Hassabis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018. 3

[8] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018. 3, 15

[9] V. Campos, A. Trott, C. Xiong, R. Socher, X. Giró-i Nieto, and J. Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327. PMLR, 2020. 10

[10] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. Julian, C. Finn, et al. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *arXiv preprint arXiv:2104.07749*, 2021. 10, 15

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 3

[12] J. Choi, A. Sharma, S. Levine, H. Lee, and S. S. Gu. Variational empowerment as representation learning for goal-based reinforcement learning. In *Deep Reinforcement Learning workshop at the Conference on Neural Information Processing Systems (DRL)*, 2020. 6, 10

[13] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018. 2, 10, 15

[14] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019. 2, 10, 15

[15] 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. 6, 10, 15

[16] B. Eysenbach, R. Salakhutdinov, and S. Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *arXiv preprint arXiv:1906.05253*, 2019. 10

[17] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2786–2793. IEEE, 2017. 10

[18] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018. 2, 10, 18

[19] C. Florensa, J. Degrave, N. Heess, J. T. Springenberg, and M. Riedmiller. Self-supervised learning of image embedding for continuous control. *arXiv preprint arXiv:1901.00943*, 2019. 10

[20] D. Ghosh, A. Gupta, J. Fu, A. Reddy, C. Devin, B. Eysenbach, and S. Levine. Learning to reach goals without reinforcement learning. *arXiv preprint arXiv:1912.06088*, 2019. 6, 14

[21] K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016. 10

[22] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019. 7

[23] 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. 3

[24] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019. 4

[25] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020. 14

[26] K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. *ICLR*, 2020. 5, 6, 10

[27] L. P. Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993. 1, 5, 10

[28] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[29] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3, 6

[30] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016. 4

[31] D. V. Lindley et al. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956. 2

[32] 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. 2

[33] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018. 2, 8, 10, 18

[34] O. OpenAI, M. Plappert, R. Sampedro, T. Xu, I. Akkaya, V. Kosaraju, P. Welinder, R. D'Sa, A. Petron, H. P. d. O. Pinto, et al. Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv preprint arXiv:2101.04882*, 2021. 10, 15

[35] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017. 3, 15

[36] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 2050–2053, 2018. 10

[37] D. Pathak, D. Gandhi, and A. Gupta. Self-supervised exploration via disagreement. In *International Conference on Machine Learning*, pages 5062–5071, 2019. 4

[38] 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. 2, 6, 8, 10, 15, 18

[39] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014. 3, 6

[40] V. Saxena, J. Ba, and D. Hafner. Clockwork variational autoencoders. *arXiv preprint arXiv:2102.09532*, 2021. 3

[41] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015. 1, 10

[42] J. Schmidhuber. Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463. IEEE, 1991. 3

[43] L. Schulz. Finding new facts; thinking new thoughts. *Advances in child development and behavior*, 43:269–294, 2012. 2

[44] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. *ICML*, 2020. 2, 3, 4, 6, 9, 15, 19

[45] P. Shyam, W. Jaśkowski, and F. Gomez. Model-based active exploration. *arXiv preprint arXiv:1810.12162*, 2018. 2, 3

[46] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *ICLR*, 2018. 10

[47] Y. Sun, F. Gomez, and J. Schmidhuber. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *International Conference on Artificial General Intelligence*, pages 41–51. Springer, 2011. 3

[48] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 2

[49] 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. 7, 9, 19

[50] S. Tian, S. Nair, F. Ebert, S. Dasari, B. Eysenbach, C. Finn, and S. Levine. Model-based visual planning with self-supervised functional distances. *arXiv preprint arXiv:2012.15373*, 2020. 10

[51] 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. 2, 6, 9, 10, 18

[52] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. 2015. 3, 10

[53] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021. 9, 19

[54] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020. 7

[55] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. 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. 5

[56] Y. Zhang, P. Abbeel, and L. Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 10

A Experimental Details

Environments The episode length is 150 for RoboBin and RoboKitchen and 1000 for RoboYoga. We show all goals in Figure A.1. For both *Walker* and *Quadruped*, the success criterion is based on the largest violation across all joints. The global rotation of the Quadruped is expressed as the three independent Euler angles. Global position is not taken into account for the success computation. *RoboBin*. The success criterion is based on placing all objects in the correct position within 10 cm. For reaching task, the success is based on placing the arm in the correct position within 10 cm. *RoboKitchen* uses 6 degrees of freedom end effector control implemented with simulation-based inverse kinematics. The success criterion is based on placing all objects in the correct position with a threshold manually determined by visual inspection. Note that this is a strict criterion: the robot needs to place the object in the correct position, while not perturbing any other objects.

Evaluation We reported success percentage at the final step of the episode. All experiments on our benchmark as well as on the SkewFit benchmark were ran 3 seeds. Due to large required compute, DISCERN and Plan2Explore results for LEXA were only run with one seed. The DISCERN and Plan2Explore results should therefore not be used for rigorous comparisons, but are nevertheless indicative of the simplicity of these benchmarks. Plots were produced by binning every 3e5 samples. Heatmap shows performance at the best timestep. Each model was trained on a single high-end GPU provided by either an internal cluster or a cloud provider. The training took 2 to 5 days. The final experiments required approximately 100 training runs, totalling approximately 200 GPU-days of used resources.

Implementation We base our agent on the Dreamer implementation. For sampling goals to train the achiever, we sample a batch of replay buffer trajectories and sample both the initial and the goal state from the same batch, therefore creating a mix of easy and hard goals. To collect data in the real environment with the achiever, we sample the goal uniformly from the replay buffer. We include code in the supplementary material. The code to reproduce all experiments will be made public upon the paper release under an open license.

Hyperparameters LEXA hyperparameters follow Dreamer V2 hyperparameters for DM control (which we use for all our environments). For the explorer, we use the default hyperparameters from the Dreamer V2 codebase [25]. We use action repeat of 2 following Dreamer. LEXA includes only one additional hyperparameter, the proportion of negative sampled goals for training the distance function. It is specified in Table A.5. The hyperparameters were chosen by manual tuning due to limited compute resources. The base hyperparameters are shared across all methods for fairness.

DIAYN baseline We found that this baseline performs best when the reverse predictor is conditioned on the single image embedding e rather than latent state s. We use a skill space dimension of 16 with uniform prior and Gaussian reverse predictor with constant variance. For training, we produce the embedding using the embedding prediction network from Section 2.4. We observed that DIAYN can successfully achieve simple reaching goals using the skill obtained by running the reverse predictor on the goal image. However, it struggles with more complex tasks such as pushing, where it only matches the robot arm.

GCSL baseline We found that this baseline performs best when the policy is conditioned on the single image embedding e rather than latent state s. This baseline is trained on the replay buffer images and only uses imagined rollouts to train an explorer policy. For training, we sample a random image from a trajectory and sample the goal image from the uniform distribution over the images later in the trajectory following [20]. We similarly observe that this baseline can perform simple reaching goals, but struggles with more complex goals.

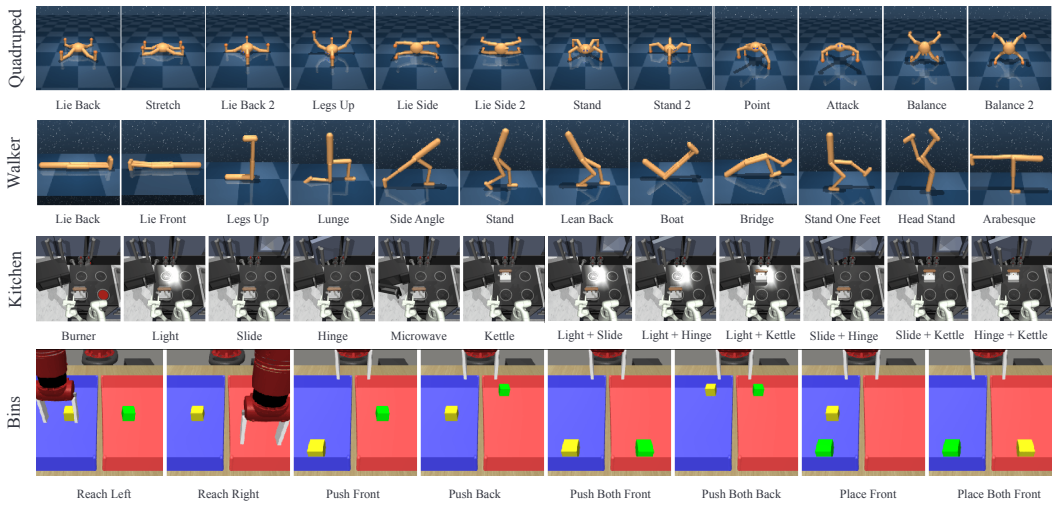

Figure A.1: All goals for the four environments that we consider. Our benchmark further includes an additional set of even harder goals, available in the repository.

Algorithm	**From Pixels**	**Zero-Shot**	**Exploration**	**Planning**
CTS [4], Curiosity [35], RND [8]	✓	✗	✓	✗
Plan2Explore [44]	✓	✗	✓	✓
HER [2]	✗	✓	✗	✗
Visual Foresight [13]	✓	✓	✗	✓
Actionable Models [10]	✓	✓	✗	✗
DIAYN [15]	✗	✓	✓	✗
Asymmetric Self-Play [34]	✗	✓	✓	✗
SkewFit [38]	✓	✓	✓	✗
Go-Explore [14]	✓	✓	✓	✗
LEXA (Ours)	✓	✓	✓	✓

Table A.1: Conceptual comparison of unsupervised reinforcement learning methods. LEXA combines forward-looking exploration by planning with achieving downstream tasks zero-shot while learning purely from pixels without any privileged information.

Table A.5: Hyperparameters for LEXA over the Dreamer default hyperparameters.

Hyperparameter	Value	Considered values
Action repeat (all environments)	2	2
Proportion of negative samples	0.1	0, 0.1, 0.5, 1
Proportion of explorer:achiever data collected in real environment	1:1	1:1
Proportion of explorer:achiever training imagination rollouts	1:1	1:1

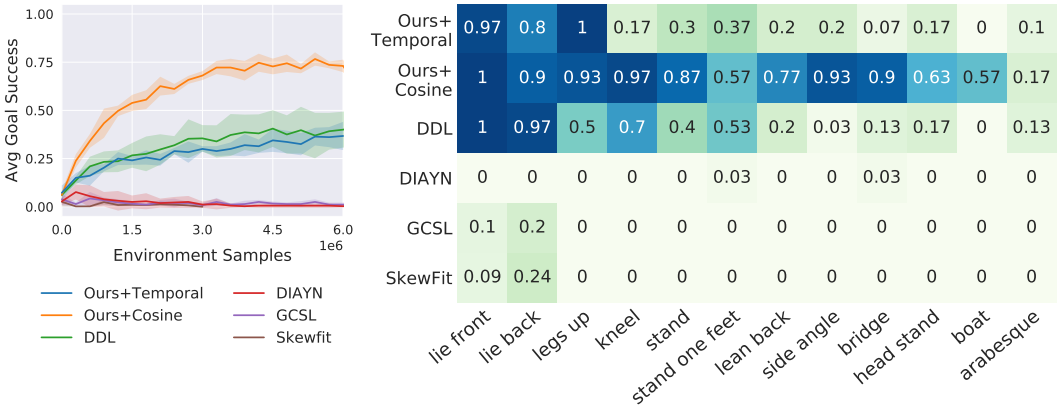

Figure A.2: **RoboYoga Walker Benchmark.** Left: success rates averaged across all 12 tasks. Right: final performance on each specific task, ranging from light green (0) to dark blue (100%). We observe that the simple latent cosine distance function works well on this task, substantially outperforming other competing agents. In the heatmap, most agents can solve the easy tasks, but only LEXA makes progress on solving a majority of the tasks and achieves good performance.

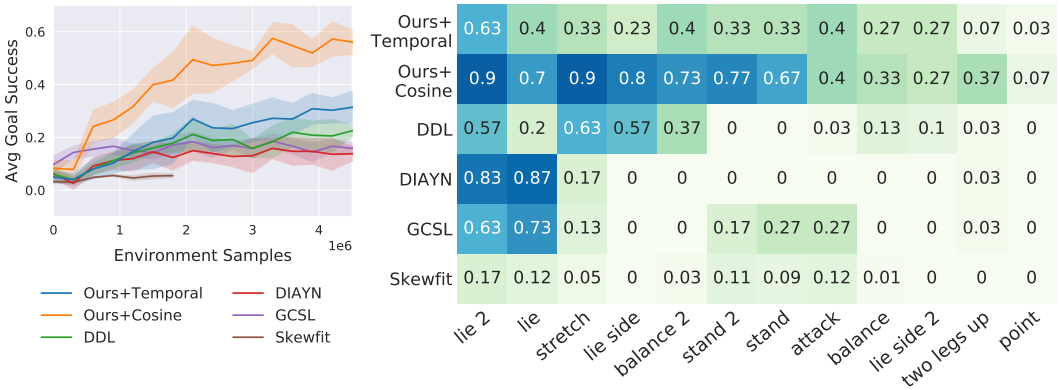

Figure A.3: **RoboYoga Quadruped Benchmark.** Left: success rates averaged across all 12 tasks. Right: final performance on each specific task, ranging from light green (0) to dark blue (100%). We observe that the simple latent cosine distance function works well on this task, substantially outperforming other competing agents. In the heatmap, most agents can solve the easy tasks, but only LEXA makes progress on solving a majority of the tasks and achieves good performance.

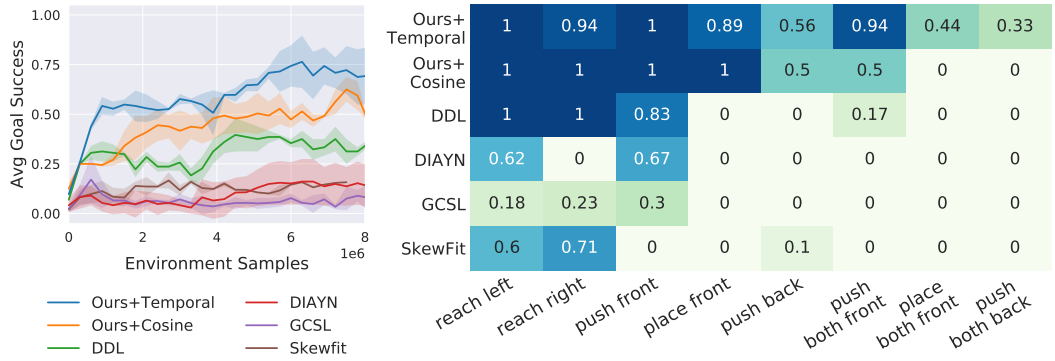

Figure A.4: **RoboBin Benchmark.** Left: success rates averaged across all 8 tasks. Right: final performance on each specific task. While cosine distance works on simple goals, temporal distance outperforms it on tasks requiring manipulating several blocks (last three columns), as this distance focuses on the part of the environment that's hardest to manipulate. Prior agents only solve the easiest reaching tasks, struggling either with exploration or learning the downstream policy.

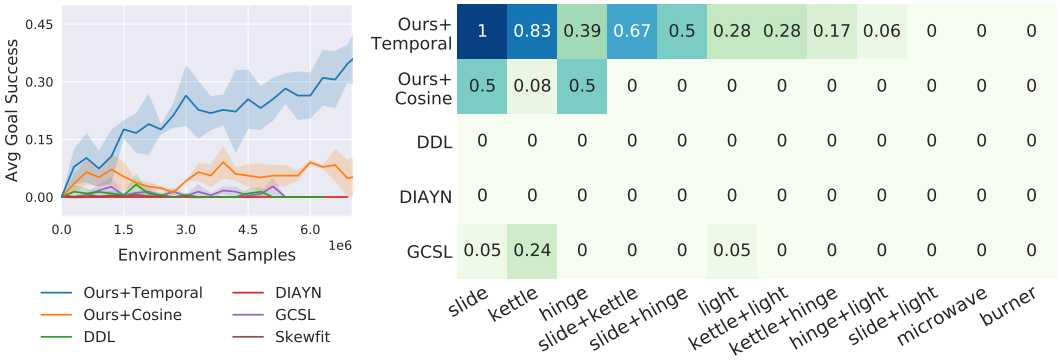

Figure A.5: **RoboKitchen Benchmark.** Left: success rates averaged across all 12 tasks. Right: final performance on each specific task. RoboKitchen is challenging both for exploration and downstream control, with most prior agents failing all tasks. In contrast, LEXA is able to learn both an effective explorer and achiever policy. Temporal distance helps LEXA focus on small parts such as the light switch, necessary to solve these tasks. LEXA makes progress on four out of six base tasks, and is even able to solve combined goal images requiring e.g. both moving the kettle and opening a cabinet.

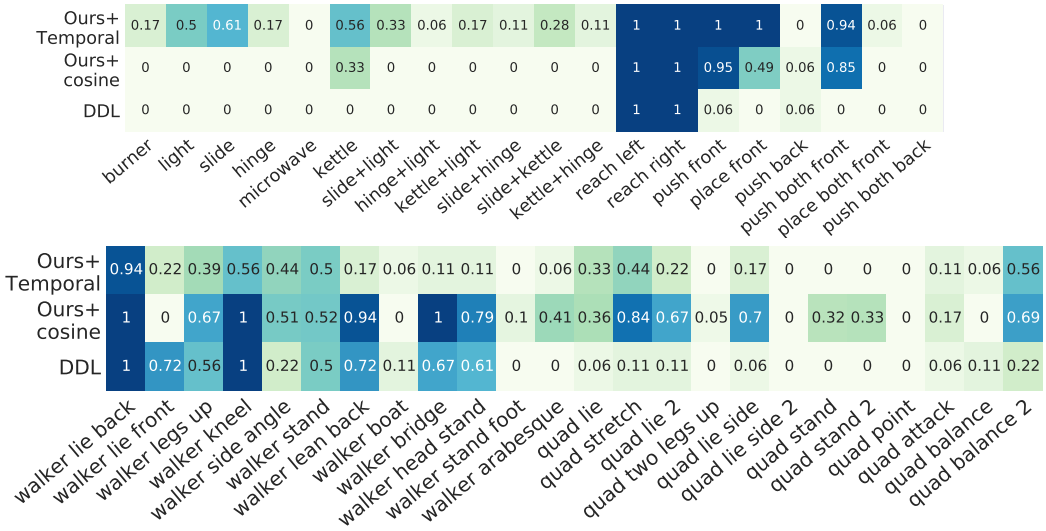

Figure A.6: **Single agent** trained across Kitchen, RoboBin, Walker, with final performance on each specific task. LEXA with temporal distance is able to make progress on tasks from all environments, while LEXA+cosine and DDL don't make progress on the kitchen tasks.

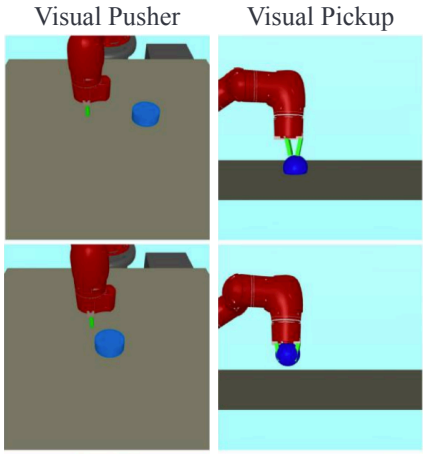

Table A.2: Results on SkewFit tasks [38].

Method	Visual Pusher	Visual Pickup
LEXA + temporal	**0.023**	**0.014**
Skew-Fit [38]	0.049	0.018
RIG [33]	0.077	0.037
RIG + Hazan et al.	0.059	0.039
RIG + HER [2]	0.075	0.035
DISCERN [51]	0.094	0.039
RIG + Goal GAN [18]	0.088	0.039
RIG + DISCERN-g	0.07	0.032
RIG + # Exploration	0.088	0.04
RIG + Rank-Based	0.067	0.035

Figure A.7: Final goal reaching error in meters on tasks from SkewFit [38]. Example observations are provided on the left. Baseline results are taken from [38]. LEXA significantly outperforms prior work on these tasks. Pushing and picking up blocks from visual observations is largely solved, so future work will likely focus on harder benchmarks such as the one proposed in our paper.

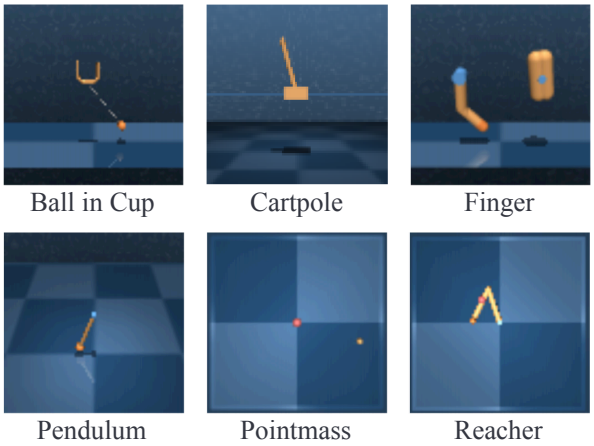

Table A.3: Results on DISCERN tasks* [51].

	LEXA	DISCERN* [51]
Ball in cup	**84%**	76.5%
Cartpole	**35.9%**	21.3%
Finger	**40.9%**	21.8%
Pendulum	**79.1%**	75.7%
Pointmass	**83.2%**	49.6%
Reacher	**100%**	87.1%

Figure A.8: Goal success rate on the tasks replicated from [51]. Example observations are provided on the left. LEXA results were obtained with early stopping. *While the original tasks are not released, we followed the procedure for generating the goals described in [51]. Despite following the exact procedure, we were not able to obtain similar goals to the ones used in the original paper. Nevertheless, we show the goal completion percentage results obtained with our reproduced evaluation compared to DISCERN results from the original paper. We see that our agent solves many of the tasks in this benchmark and performs better on this comparison. We further suspect that the goals that we generated are harder than the ones used in the original paper, such as in the cartpole environment where our goals require swinging the pole higher up. Future work will likely focus on harder benchmarks such as our RoboYoga benchmark, rather than these simple robots with one or two degrees of freedom.

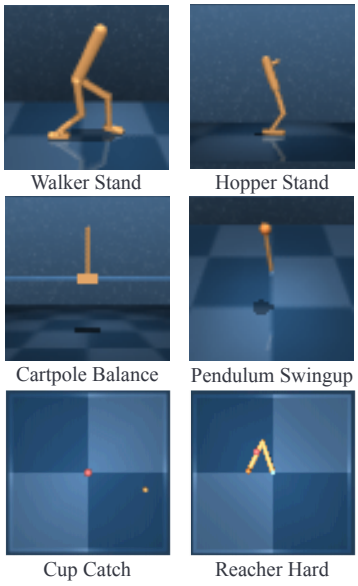

Table A.4: Results on zero-shot DeepMind control tasks from Plan2Explore [44].

Method	LEXA	P2E [44]	DrQv2 [53]
Zero-Shot	✓	✓*	✗
Walker Stand	**957**	331	968
Hopper Stand	**840**	**841**	957
Cartpole Balance	886	**950**	989
Cartpole Balance Sparse	**996**	860	983
Pendulum Swingup	**788**	**792**	837
Cup Catch	**969**	**962**	909
Reacher Hard	**937**	66	970

Figure A.9: Final return on DM control tasks [49]. Example goals achieved by LEXA are provided to the left. Baseline results taken from [44, 53]. LEXA results were obtained with early stopping. *Plan2Explore adapts to new tasks but it needs the reward function to be known at test time while LEXA does not require access to rewards. To compare on the same benchmark, we create goal images that correspond to the reward functions. This setup is arguably harder for our agent, but is much more practical. Our agent outperforms Plan2Explore on most tasks and even performs comparably to state of the art oracle agents (DrQ, DrQv2, Dreamer) that use true task rewards during training.