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 wide ranges of tasks in complex visual environments in the absence of external supervision? We decompose this question into two problems - discovering new goals and learning to reliably reach them. We introduce the Latent Explorer Achiever (LEXA), a unified solution to these that learns a world model from image inputs and uses it to train an explorer and an achiever policy from imagined trajectories. 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. After the unsupervised phase, LEXA solves tasks specified as goal images zero-shot without any additional learning. We introduce a challenging benchmark spanning across four standard robotic manipulation and locomotion domains with a total of over 40 test tasks. LEXA substantially outperforms previous approaches to unsupervised goal reaching, achieving goals that require interacting with multiple objects in sequence. Finally, to demonstrate the scalability and generality of LEXA, we train a single general agent across four distinct environments.

1 Introduction

This paper tackles the question of how to build an autonomous agent that can achieve a diverse set of tasks specified by a user at test time, such as a legged robot standing in certain pose, a robotic arm rearranging blocks between bins, or performing chores in a kitchen. Such a general system would be difficult to realize within the traditional reinforcement learning (RL) paradigm, where a user specifies task rewards that the agent finds by sampling its environment. Designing task rewards requires domain knowledge, is time-consuming, and prone to human errors. Moreover, a traditional RL would have to explore and retrain for every new task. Instead, the problem of solving diverse tasks is often studied under the paradigm of unsupervised RL, where an agent builds a repertoire of skills without any supervision, which later enable solving human-specified goals with no or limited further training.

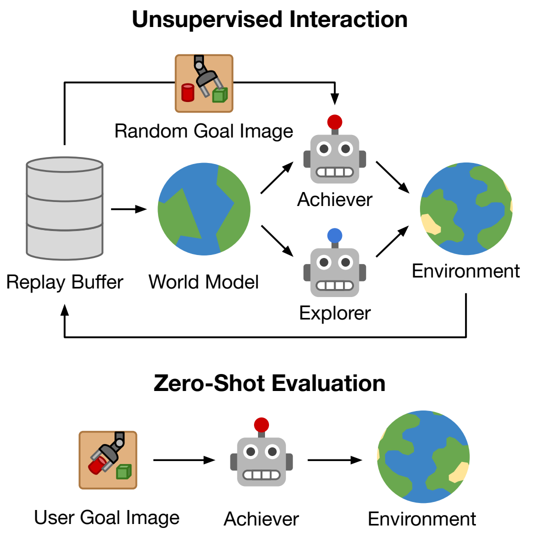


Figure 1: LEXA learns a world model without any supervision, and leverages it to train two policies in imagination. The *explorer* finds new images and the *achiever* learns to reliably reach them. Once trained, the achiever reaches user-specified goals zero-shot without further training at test time.

Two challenges Building a capable unsupervised agent for reaching arbitrary goals presents

[*] Equal contribution. Ordering determined by a coin flip. Project page: https://orybkin.github.io/lexa/

Preprint. Under review.

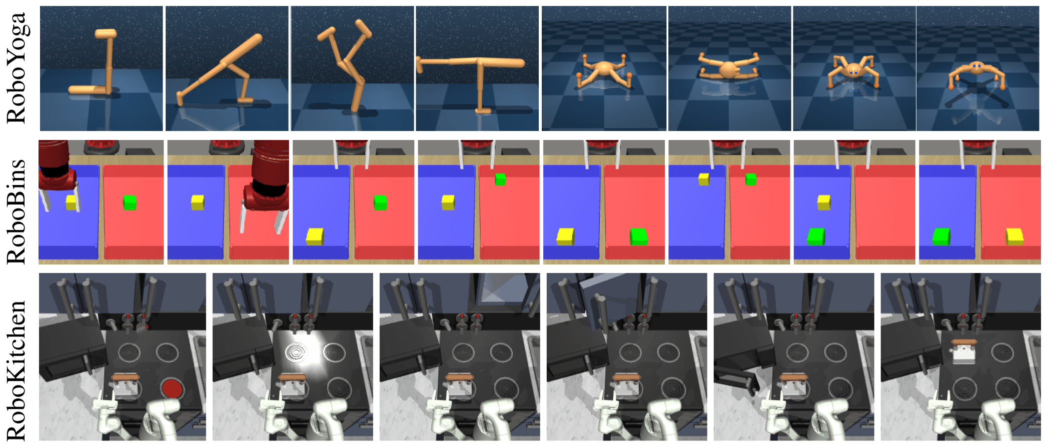


Figure 2: We benchmark our method across four visual control tasks of varying difficulty: RoboYoga (Walker, Quadruped), RoboBins, and RoboKitchen. A representative sample of the test-time goals is shown here. RoboYoga benchmark features complex locomotion and precise control of high-dimensional agents,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

two major challenges. First, the user-specified goals are often diverse and rare situations, hence the agent needs to globally explore its environment. Second, the agent then needs to learn to reliably achieve the diverse intrinsic goals it found during exploration to prepare itself for user-specified goals at test time. We propose a unified approach that learns a world model together with an explorer policy and a goal achiever policy in a fully unsupervised manner to address both challenges.

Goal discovery Prior methods for unsupervised goal reaching approach the exploration problem by relabeling trajectories with previously visited states as goals [2, 40] or by sampling goals from a density model of previous inputs [18, 32]. The goals can be sampled uniformly [32], or rarely visited goals can be oversampled [14, 37, 52]. These approaches have in common that they explore by visiting goals that either have been reached before or are interpolations of previously reached states. They do not explore far beyond the frontier and suffer from a chicken and egg problem: the policy does not yet know how to reach interesting goals and thus repeats already known behaviors. We observe that this leads to poor exploration performance in such methods. To rectify this issue, we leverage a learned world model to train an *explorer* policy in imagination. Instead of "generating" a goal, explorer "discovers" goals by executing a sequence of actions optimized in imagination to find novel states with high information gain [42, 43]. These states can be several steps away from the frontier unlocking diverse data for goal reaching in environments where exploration is nontrivial.

Goal reaching The diverse experience collected by the explorer provides start and goal states for training the goal *achiever* policy. Because our approach does not rely on goal relabeling [2], we are free to train the achiever using on-policy trajectories generated by the world model [24, 30, 46]. Training the achiever requires measuring the distance between an the states along an imagined trajectory and the goal, a non-trivial problem when inputs are high-dimensional images. We empirically analyze two choices for the distance measures, namely the cosine distance in latent space and a learned temporal distance and provide recommendations on which distance function is appropriate for different settings.

Contributions We introduce the Latent Explorer Achiever (LEXA), an unsupervised goal reaching agent that trains an *explorer* and an *achiever* within a shared world model. At test time, the achiever solves challenging locomotion and manipulation tasks provided as user-specified goal images. For a thorough evaluation, we introduce a new challenging goal reaching benchmark by defining a total of 40 diverse goal images across 4 different robot environments. In contrast to common RL benchmarks such as Atari [5] that require training over 50 different agents and thus enormous computational resources, our unsupervised RL benchmark only requires training 4 agents, which are then evaluated across many tasks, allowing for faster iteration time and making the research more accessible. Using this benchmark, we experimentally study the following scientific questions:

- Does the separation into explorer and achiever policies enable reaching more challenging goals than were previously possible and outperform state-of-the-art approaches?
- How does forward-looking exploration of goals compare to previous goal exploration strategies?
- How does the distance function affect the ability to reach goals in different types of environments?
- Can we train one general LEXA to control different robots across visually distinct environments?

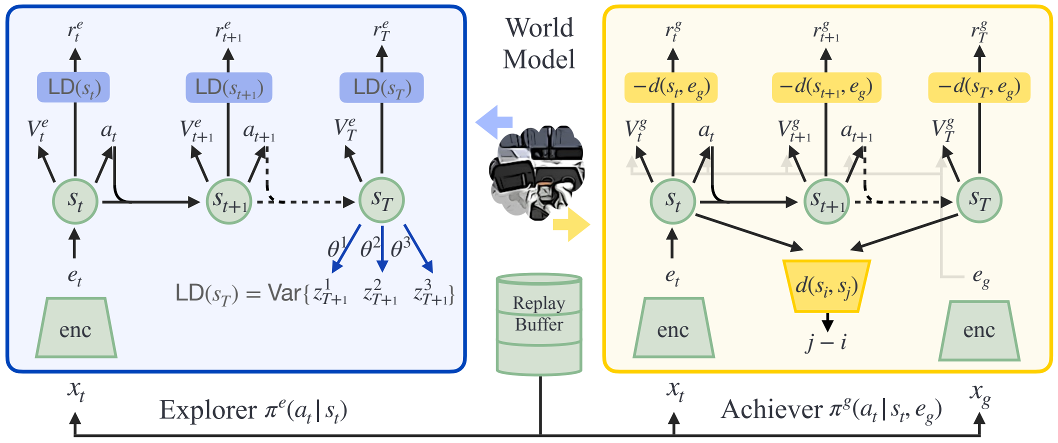


Figure 3: Explore Achieve Networks learn a single general world model that is used to train an explorer and a goal achiever policy. The explorer (π^e, left) is trained on imagined latent state rollouts of the world model $s_{t:T}$ to maximize the disagreement objective $r^e_t = \mathrm{LD}(s_t)$. The goal achiever (π^g, right) is conditioned on a goal g and is also trained on imagined rollouts to minimize a distance function $d(s_{t+1}, e_g)$. Goals are sampled randomly from replay buffer images. For training a temporal distance function, 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 goal achiever in imagination, Explore Achieve Network 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 problem presents two challenges, collecting trajectories that contain diverse goals and learning to reach 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 reach goals, we train the goal-conditioned achiever in imagination, using the images found so far as unsupervised goals. At test time, the agent reaches user-specified goals by deploying the achiever. A summary of the training procedure is given 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, 50]. In contrast to predicting forward in image space, the model states enables efficient parallel planning with a large batch size and can reduce accumulating errors [39]. The world model consists of the following components:

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

The model states s_t contain a deterministic component h_t and a stochastic component z_t with diagonal Gaussian distribution. The deterministic component is implemented as 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, 38] with the Adam optimizer [28].

Algorithm 1: Latent Explorer Achiever (LEXA)

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

2.2 Explorer

To efficiently explore, we seek out surprising states imagined by the world model [41–43, 45], as opposed to retrospectively exploring by revisiting previously novel states [4, 6, 8, 34]. 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 policy π^e from the model states s_t in imagination of the world model to maximize an exploration reward:

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

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 [31, 36]:

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

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, 42]:

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

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 reuse and amplify past trajectories via the world model that is trained on past trajectories 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.

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

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

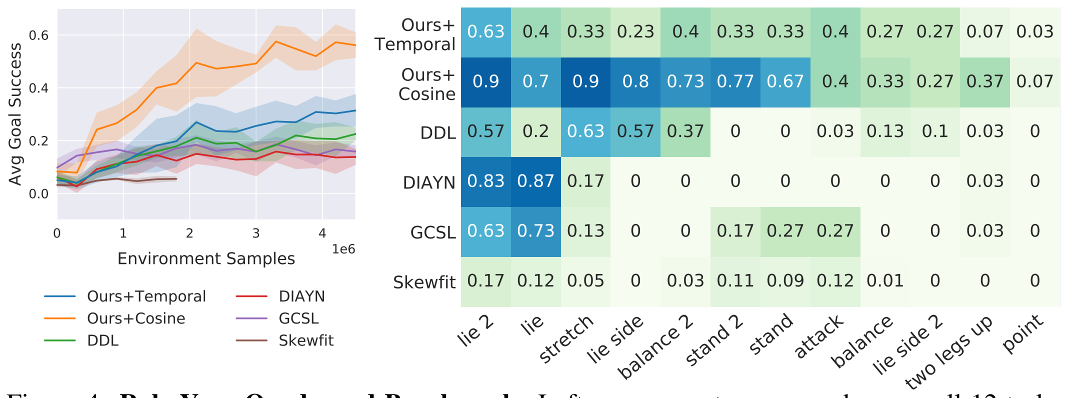


Figure 4: **RoboYoga Quadruped Benchmark.** Left: success rates averaged across all 12 tasks. Right: final performance on each specific task, ranging from light green (0) to dark blue (100%). We observe that the simple latent cosine distance function works well on this task, substantially outperforming 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.

$r^g(s_t, e_g)$. 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, we found it important to perform practice trials with the goal achiever in the true environment, so that any model inaccuracies along the goal reaching trajectories 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:

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

This metric is the cosine of the angle between the two vectors s_t, s_g in the $N-$dimensional latent space. Using this simple metric in our Explore-Achieve framework, we obtain an effective agent for unsupervised goal reaching, especially for environments with a single controllable agent.

Temporal Distance To use temporal distances with LEXA, we train a neural network d to predict the number of time steps between two image predicted embeddings of an imagination trajectory. We sample a trajectory containing s_i by running the current goal-reaching policy in imagination using a random initial and goal images sampled from the replay buffer. We select the second state s_j to be a random state later in the same trajectory and regress towards the ground truth number of time steps

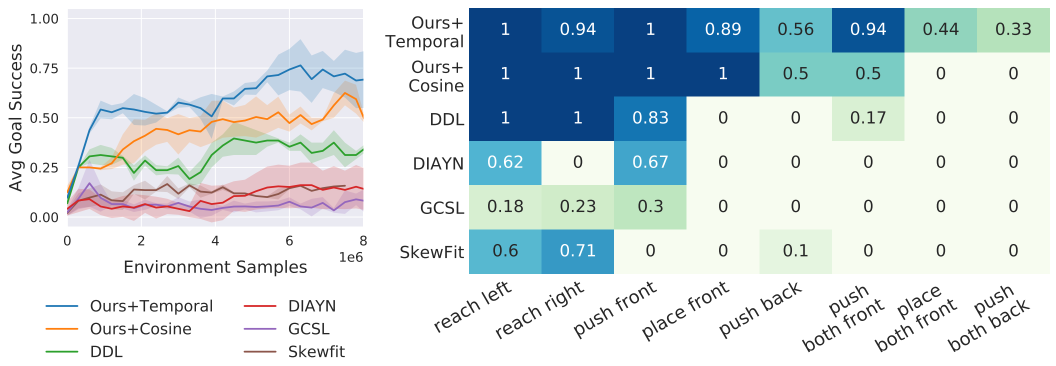


Figure 5: **RoboBin Goal Benchmark.** Left: success rates averaged across all 8 tasks. Right: final performance on each specific task, ranging from light green (0) to dark blue (100%). We observe that while the simple latent cosine distance function works on simple goals, temporal distance functions outperform it on the more challenging tasks requiring manipulations of several blocks (last three columns in the heatmap), as this distance metric is able to focus on the part of the environment that's hardest to manipulate. We further observe that other competing agents perform poorly and only solve the easiest reaching tasks, struggling either with exploration or learning the downstream policy.

between two states. We implement the temporal distance in terms of predicted image embeddings $\hat{e}_{t+k}$ in order to remove extra recurrent information:

$$\begin{aligned} &\text{Predicted CNN embedding:} \qquad \operatorname{emb}(s_t) = \hat{e}_t \approx e_t \\ &\text{Temporal distance:} \qquad d_\omega(\hat{e}_t, \hat{e}_{t+k}) \approx k/H, \end{aligned} \tag{7}$$

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 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.

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

We note this 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 [26]. In our method, we always use the data from the latest policy to train the distance function via the imagination training, ensuring that the convergence is fast.

3 Experiments

We compare LEXA to several goal reaching approaches to evaluate its empirical performance and understand the contributions of the individual components. 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:

- **DDL** Dynamic Distance Learning Hartikainen et al. [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 reach 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.
- **SkewFit** SkewFit [37] uses model-free hindsight experience replay and explores by sampling goals from the latent space of a variational autoencoder [29, 38]. Being one of the state-of-the-art agents, we use the original implementation that does not use a world model or explorer policy.

We introduce three benchmarks for unsupervised goal reaching by defining goal images for a diverse set of four existing environments, shown in Figure 2:

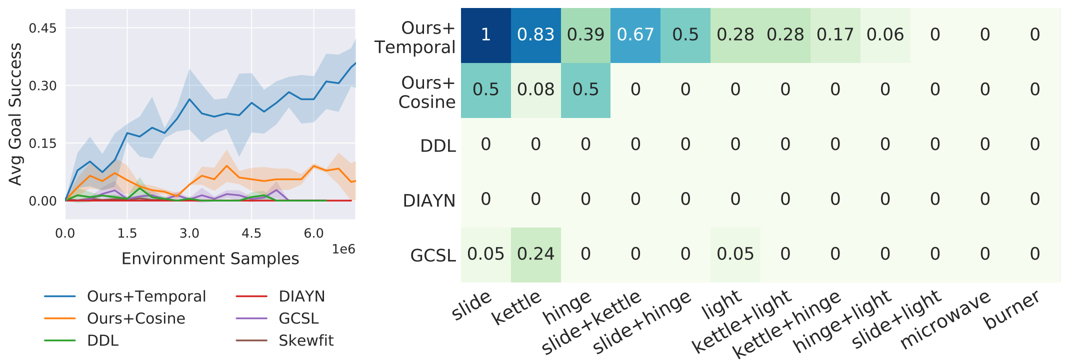


Figure 6: **RoboKitchen Benchmark.** Left: success rates averaged across all 12 tasks. Right: final performance on each specific task, ranging from light green (0) to dark blue (100%). This enviroment presents both extremely challenging exploration and downstream control, with most prior agents never solving any tasks. In contrast, LEXA is able to learn both an effective explorer and achiever policy. Temporal distance functions help 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.

- **RoboYoga** We use the walker and quadruped domains of the DeepMind Control Suite [47] to defined 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 [51], 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.

3.1 RoboYoga Benchmark

The environments in this benchmark are directly controllable, since they only contain the robot and no other objects. We recall that for such settings we expect the cosine distance to be a very effective metric, since success requires exactly matching the goal image. Training is thus faster compared to using learned temporal distances, where the metric is learned from scratch.

From Figure 4 we see that this is indeed the case for the Quadruped environment, and LEXA with the cosine metric outperforms all prior approaches on both the Walker and Quadruped environments. 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, even though average success over goals looks similar to that of DDL. We found similar results on the Walker environment, and include them in the appendix.

3.2 RoboBins Benchmark

This environment nvolves interaction with block objects, and thus not directly controllable, and so we expect LEXA to perform better with the temporal distance metric. We see that LEXA beats all prior approaches, and is also able to make progress on all goals in the benchmark.From the per-task performance in Figure 5 we see that the main difference in performance between using temporal distance and cosine 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 completely solves reaching, and is about 3 times worse on pushing. This is because the the method sees a lot less data relevant to the harder tasks owing to poorer exploration since it doesn't have the disagreement objective. We see that while skewfit make some progress on reaching, it completely fails on harder tasks involving manipulation.

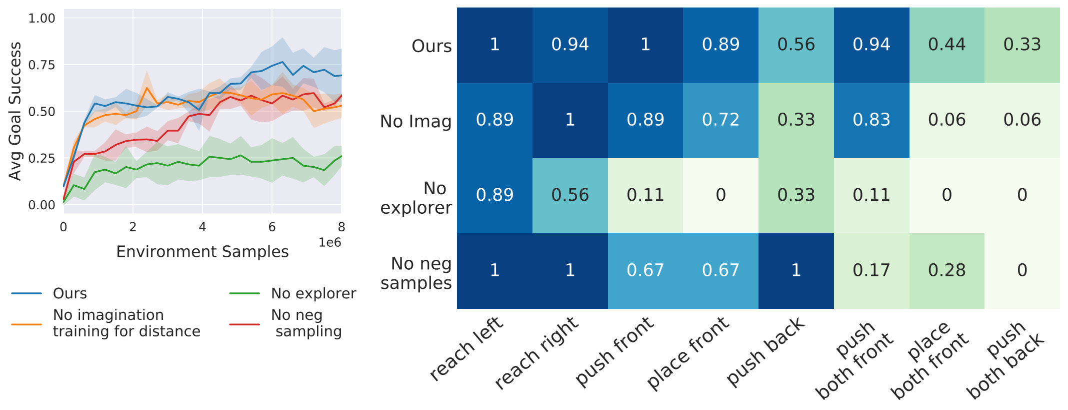


Figure 7: **Ablations** Testing different components of LEXA with temporal distances on the RoboBins benchmark. We observe that training a separate exploration policy is crucial for solving most tasks, as the agent never discovers them in the *no explorer* version. Training temporal distance on negative samples significantly speeds up learning, and both negative sampling and training in imagination as opposed to real data is important for performance on the hardest tasks.

3.3 RoboKitchen Benchmark

This benchmark involoves a very diverse set of objects, that require different manipulation behavior. We see from Figure 6 that our approach with learned 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.

3.4 Single Agent Across All Environments

In the previous sections we have shown that our approach can reach 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 all the environments in the benchmark (i.e RoboKitchen, RoboBins, Walker and Quadruped). From Figure 8 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.

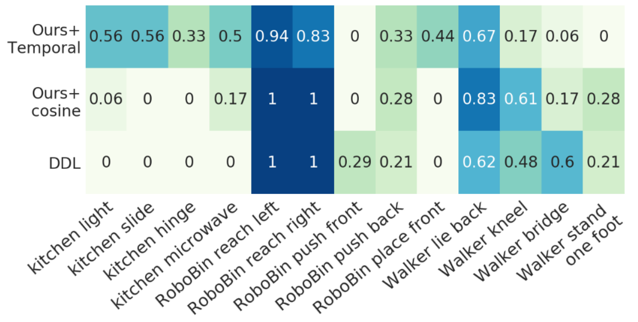


Figure 8: **Single agent** trained across Kitchen, RoboBin, Walker, with final performance on each specific task, ranging from light green (0) to dark blue (100%). LEXA with temporal distance is able to make progress on tasks from all environments, while LEXA+cosine and DDL don't make progress on the kitchen tasks.

3.5 Ablation Study

Exploration Quality Effective exploration is a necessary condition for goal reaching, since without diverse data it is not possible to train a good controller. In order to examine the diversity of the data collected during training, we also log the 'coincidental success', i.e whether in the process of collecting data, the agent exhibits behavior that is need to solve an evaluation task. We include the Figure in the appendix, and see that while dynamical distances encounter reaching goals, our method encounters the harder tasks like place much more often.

Ablation of different components We ran ablations of our approach on the RoboBins environment, where we examined the effect of removing negative sampling while training the distance function, removing disagreement for the exploration policy, and training the distance function with real world data from the replay buffer instead of imagination data. From the plots in Figure 7 we see that using a separate explorer policy is the most critical component, and without the explore the agent does not collect good data from which to learn from. Without negative sampling the agent learns slower, and this is probably because the distance function doesn't produce reasonable outputs when queried on images that are more than horizon length apart, since it is never trained on such data. 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.

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, 40]. Recent work has combined these approaches with various ways to generate training goals, such as asymmetric self-play [33, 44] 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 [49] or reconstructive [32, 37] 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. Chebotar et al. [10], Tian et al. [48] avoid this challenge by using a large dataset of interesting behaviors. Pong et al. [37], Zhang et al. [52] 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 learn to reach goals via explicit planning [13, 17] or learning model-regularized policies [35]. 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, 32, 49, 50]. 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 goal achiever is most similar to Hartikainen et al. [26], who learn a temporal distance with simple supervised learning on recent policy experience. In contrast to [26], we always train this distance function on the most recent policy data in imagination, and we further integrate this achiever policy into our explore-achieve framework and leverage world models to discover novel goals for the achiever to practice on.

5 Conclusion

We presented Explorer Achiever Network (LEXA), a unified agent for unsupervised RL that explores its environment, learns to reach the discovered goals, and solves image-based tasks at test time in zero-shot way. By searching for novelty in imagination, LEXA explores better than prior approaches 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, even being able to train a single policy in several different environments together. 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 largely 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 diverse 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 our framework provides a solid foundation for future work to make concrete progress on these goals.

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.

References

[1] J. Achiam, H. Edwards, D. Amodei, and P. Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018. 9

[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. 2, 4, 9

[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. 4

[5] 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. 2

[6] 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. 4

[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. 4

[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. 9

[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. 9

[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, 9

[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. 9

[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, 9

[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, 9

[16] B. Eysenbach, R. Salakhutdinov, and S. Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *arXiv preprint arXiv:1906.05253*, 2019. 9

[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. 9

[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, 9

[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. 9

[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, 13

[21] K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016. 9

[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. 2, 4, 5

[25] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020. 13

[26] K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. *ICLR*, 2020. 5, 6, 9

[27] L. P. Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993. 5, 9

[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] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018. 2

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

[32] 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, 9

[33] 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. 9

[34] 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. 4

[35] 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. 9

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

[37] 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, 9

[38] 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

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

[40] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015. 2, 9

[41] J. Schmidhuber. Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463. IEEE, 1991. 4

[42] 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. 2, 4

[43] P. Shyam, W. Jaśkowski, and F. Gomez. Model-based active exploration. *arXiv preprint arXiv:1810.12162*, 2018. 2, 4

[44] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *ICLR*, 2018. 9

[45] 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. 4

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

[47] 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

[48] 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. 9

[49] 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. 9

[50] 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, 9

[51] 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

[52] Y. Zhang, P. Abbeel, and L. Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33, 2020. 2, 9

A Experimental Details

Environments The episode length is 150 for RoboBin and RoboKitchen and 1000 for RoboYoga. We show all goals in Figure 10 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 were ran 3 seeds. 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 1. 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.

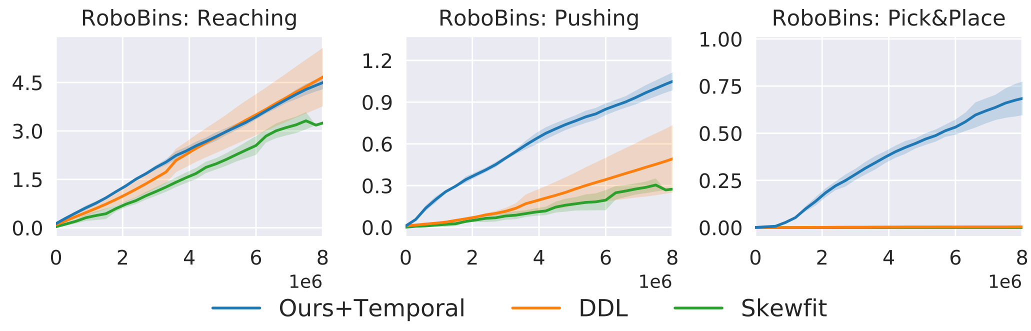


Figure 9: Coincidental success rate on RoboBins

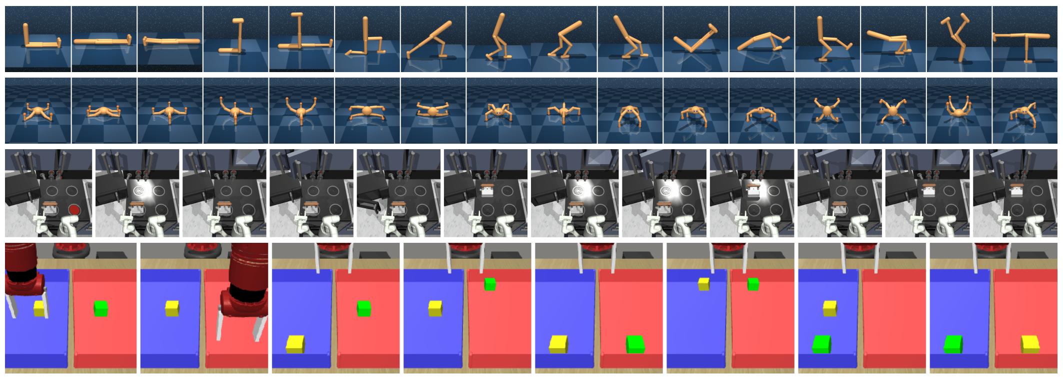
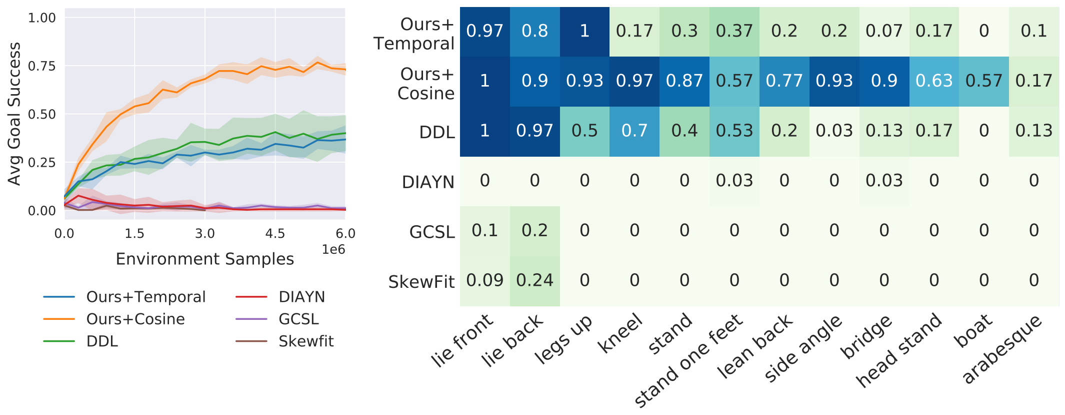


Figure 10: All goals for the four environments.



	lie front	lie back	legs up	kneel	stand	stand one feet	lean back	side angle	bridge	head stand	boat	arabesque
Ours+ Temporal	0.97	0.8	1	0.17	0.3	0.37	0.2	0.2	0.07	0.17	0	0.1
Ours+ Cosine	1	0.9	0.93	0.97	0.87	0.57	0.77	0.93	0.9	0.63	0.57	0.17
DDL	1	0.97	0.5	0.7	0.4	0.53	0.2	0.03	0.13	0.17	0	0.13
DIAYN	0	0	0	0	0	0.03	0	0	0.03	0	0	0
GCSL	0.1	0.2	0	0	0	0	0	0	0	0	0	0
SkewFit	0.09	0.24	0	0	0	0	0	0	0	0	0	0

Figure 11: RoboYoga Walker Benchmark

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	0.5:0.5	0.5:0.5
Proportion of explorer:achiever training imagination rollouts	0.5:0.5	0.5:0.5

Table 1: Hyperparameters for LEXA