

# Accelerating Visual Sparse-Reward Learning with Latent Nearest-Demonstration-Guided Explorations

Anonymous Author(s)

Affiliation

Address

email

**Abstract:** Recent progress in deep reinforcement learning (RL) and computer vision enables artificial agents to solve complex tasks, including locomotion, manipulation, and video games from high-dimensional pixel observations. However, RL usually relies on domain-specific reward functions for sufficient learning signals, requiring expert knowledge. While vision-based agents could learn skills from only sparse rewards, exploration challenges arise. We present **Latent Nearest-demonstration-guided Exploration (LaNE)**, a novel and efficient method to solve sparse-reward robot manipulation tasks from image observations and a few demonstrations. First, LaNE builds on the pre-trained DINOv2 feature extractor to learn an embedding space for forward prediction. Next, it rewards the agent for exploring near the demos, quantified by quadratic control costs in the embedding space. Finally, LaNE optimizes the policy for the augmented rewards with RL. Experiments demonstrate that our method achieves state-of-the-art sample efficiency in Robosuite simulation and enables *under-an-hour* RL training from scratch on a Franka Panda robot, using only a few demonstrations.

**Keywords:** Computer Vision, Sparse Reward, RL from Demonstrations

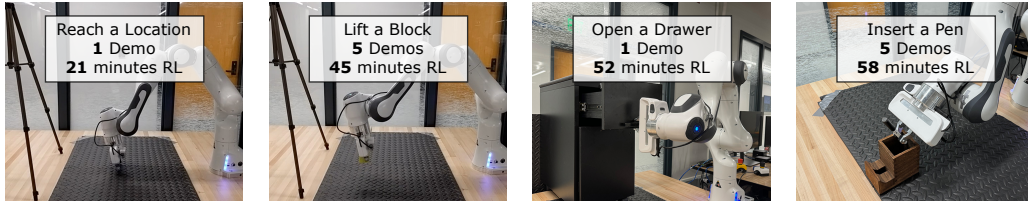


Figure 1: LaNE enables under-an-hour RL training from scratch on a Franka Panda arm from image observations and sparse rewards, utilizing only a few demonstrations. LaNE achieves unparalleled sample efficiency by learning an embedding space to quantify state proximity and reward explorations close to the demonstrations.

## 1 Introduction

Deep reinforcement learning (RL) is a versatile approach that learns from interaction data, often without an explicit, hand-coded dynamics model. Through environmental interactions, RL agents can learn optimal policies from dense or sparse reward feedback. State-of-the-art approaches allow learning policies for discrete actions and continuous action spaces while taking either low-dimensional state vectors or high-dimensional sensor readings [1, 2, 3, 4].

However, applying deep RL to real-life domains, including real-hardware robot learning, remains difficult. One challenge is the need to reliably track the complete system state [5]. A policy that directly maps images to optimal actions could alleviate such engineering challenges: data augmentation and self-supervised learning have enabled policy learning from image observations with high

sample efficiency [6, 7, 8, 9]. Meanwhile, it is also hard to assign informative rewards in a scalable way. Reward engineering requires domain-specific knowledge: popular simulated environments provide optional dense reward functions based on heuristics [10, 11] but rely on state readings not readily available in the real physical system. Thus, there has been a lot of effort to help RL agents explore more effectively in environments with sparse or no rewards [12, 13, 14].

We present **Latent Nearest-demonstration-guided Exploration (LaNE)**, a novel approach to tackle the exploration challenge in image-based control tasks with sparse rewards. LaNE builds on the idea of learning from demonstrations (LfD) and draws inspiration from reward shaping [15, 16, 17, 18, 19, 20, 21, 22, 23]. Our main insight is that each step in the demonstration can be considered a subgoal, and the agent should be credited for reaching a similar state (see Fig. 2). Our approach presents two main contributions:

1. We define a distance measure among image observations by learning a lower-dimensional latent space. Specifically, we train a Variational Autoencoder (VAE) so that the resulting latent space’s forward dynamics are locally linear. A quadratic control cost in this space effectively identifies nearby states, whereas pre-trained embeddings from even a state-of-the-art computer vision model are insufficient.
2. We propose a systematic way to provide dense reward signals in sparse-reward tasks under the LfD paradigm. When exploring near the demonstrations, the RL agent receives additional *task-progress-informed* rewards. The augmented reward function also derives bounded value functions, significantly improving training stability.

LaNE is independent of the underlying RL algorithm, and we use Soft Actor-Critic [1] in this paper. In Robosuite simulation [11], our method significantly improves sample efficiency when learning long-horizon, sparse-reward visual manipulation tasks. On real hardware, LaNE enables learning various manipulation tasks from scratch with a Franka Panda arm, each with under an hour of training and only one to five demonstrations (see Fig. 1).

## 2 Related Work

Exploration is a known challenge in deep RL, especially in sparse-reward environments [24]. One solution is to guide exploration with domain knowledge. In reward shaping, intermediate rewards can be added at important checkpoint locations [11], derived from physics knowledge [25], or learned from human annotations [26]. Aside from external guidance, other works aim to improve the agents’ intrinsic exploration behavior. Maximum entropy RL balances exploration and exploitation by encouraging high policy entropy [1]. Hierarchical RL and intrinsically motivated skill learning are also potential remedies [12, 27]. Our work aims to alleviate the need for domain expertise in the reward-shaping approach by using a few demonstrations.

Learning from demonstrations (LfD) has proven helpful in expediting RL, especially in sparse-reward settings. Prior works have introduced various auxiliary training objectives. Behavior cloning [21, 23] and supervised  $Q$ -value updates [18] can effectively learn from optimal demonstrations. [28] employ an information theoretic approach to guide policy distributions. GAIL [29] and AIRL [30] perform adversarial training to distinguish expert and policy rollouts. [17] and [20] learn potential functions from value estimates, allowing imperfect or mismatched demonstrations to be used. CoDER [31] performs contrastive learning to pre-train the image encoders. Other methods perform warm start with scripted or behavior-cloned policies [14, 22]. Our method adopts the LfD paradigm for efficient learning and is compatible with sub-optimal demonstrations.

Aside from more efficient exploration and LfD, model-based RL methods aim to improve efficiency by generating new synthetic experiences with learned world models. DreamerV2 [6] learns an accurate discrete world model from high-dimensional image inputs, enabling human-level RL performance in Atari games. Modem [32] combines model-based learning with demonstrations by over-sampling demonstrated data to form a behavior prior. Our method LaNE also learns a forward

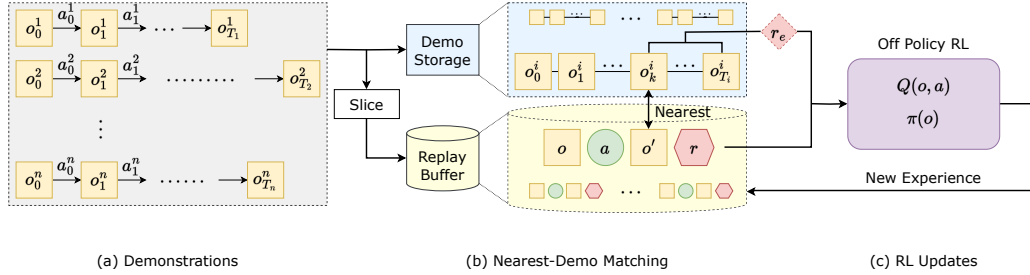


Figure 2: **Latent Nearest-demonstration-guided Exploration (LaNE)** augments the sparse task reward with a dense exploration reward in vision-based RL. **(Left)** We utilize variable-length demonstrations, each consisting of observations  $o_t^i$  and actions  $a_t^i$ . **(Middle)** A dense exploration reward  $r_e$  is given when a transition lands sufficiently close to a demonstration and is discounted based on its distance to the goal. **(Right)** Using the combined reward signal, the RL agent learns to map a sensor observation  $o$  to an action  $a$ .

dynamics model but only as an auxiliary task for representation learning. Hence, it can be integrated with both model-based and model-free RL methods.

Finally, a rich collection of prior work has studied ways to learn better representations for RL from high-dimensional image observations. RAD and DrQ [8, 9, 33] perform data augmentation on image observations to promote task-relevant features. CURL and CoDER [7, 31] use contrastive learning as a self-supervised auxiliary objective for feature learning. Recently, using foundation models as feature extractors has also shown to be beneficial for learning generalizable agents [34]. Our method builds on a state-of-the-art foundation model, DINOv2 [35], and uses data augmentation to learn a robust embedding space.

### 3 Problem Setting

We tackle the challenges of vision-based RL in long-horizon, sparse reward tasks, where the agent only receives a positive reward  $r_{\text{done}}$  at task completion while getting a constant negative reward  $r_{\text{live}}$  everywhere else. The interpretation of such a reward function is that the agent gets a high reward only at task completion but is penalized for the trajectory length. Formally, letting  $\mathcal{G}$  denote the set of goal states, we define the reward function as follows:

$$r(s, a, s') = \begin{cases} r_{\text{done}} > 0 & s' \in \mathcal{G} \\ r_{\text{live}} < 0 & \text{otherwise.} \end{cases} \quad (1)$$

The sparse reward function reduces the need for expert knowledge or human intervention, making it much easier to implement in a real-world environment, but makes exploration hard in training.

## 4 Method

We present **Latent Nearest-demonstration-guided Exploration**, an efficient RL algorithm centered around a few demonstrations to tackle the exploration challenge with sparse-reward learning. The core idea is to provide additional task-progress-aware dense rewards when the agent is close to the demonstrations. We learn a structured embedding space to quantify state proximity from image observations by learning a latent dynamics model as an auxiliary task. The cleverly designed augmented reward function also derives bounded value functions, enabling us to perform value clipping and greatly enhance training stability.

### 4.1 Reinforcement Learning from Demonstrations

LaNE utilizes a demonstration set  $\mathcal{D}$  consisting of  $n$  successful trajectories of observations and actions. Each demonstration trajectory  $i$  may have a different length  $T_i$ , but must terminate in the goal set  $\mathcal{G}$ . We assume the demonstrations to come from a human operator or a heuristic controller

and, thus, can be sub-optimal. We formalize the notations as follows:  $\mathcal{T}^i$  denotes the trajectory for demonstration  $i$ .  $s_t$  is the underlying true environment state, and  $o_t$  is the high-dimensional observation, such as images.  $a_t$  is the action taken by the demonstrator. Note that the RL algorithm cannot access the ground truth state  $s_t$  in the demonstrations, but only observations, as shown below:

$$\begin{aligned}\mathcal{D} &= \{\mathcal{T}^1, \mathcal{T}^2 \dots \mathcal{T}^n\} \\ \mathcal{T}^i &= (o_0^i, a_0^i, o_1^i, a_1^i, \dots, o_{T_i-1}^i, a_{T_i-1}^i, o_{T_i}^i) \\ \forall i, s_{T_i}^i &\in \mathcal{G}.\end{aligned}\tag{2}$$

The demonstrations are stored in two forms. First, the trajectory form records the steps to success from each state, allowing us to discount the exploration reward according to task progress, as detailed in Section 4.3. Next, they are sliced into experience tuples  $(o, a, o', r, d)$  and placed in a replay buffer  $\mathcal{B}$  for representation learning and off-policy RL updates. Here,  $o'$  denotes the next observation after  $o$ .  $r$  is the sparse reward as defined in Equation 1, and  $d$  is a Boolean variable indicating episode termination. LaNE uses a first-in-first-out replay buffer  $\mathcal{B}$  with capacity for a limited number of transitions, but the demonstrations are always retained to ensure sufficient reward signal.

## 4.2 Augmentation-Invariant Distance Measure

The key technical challenge behind LaNE is to find a demonstration state closest to the agent’s current state and quantify its proximity. Computing the distance between two states from their respective image observations is nontrivial: two drastically different states might only differ by a few pixels. Conversely, the same underlying state might appear very different in two images due to task-irrelevant background features. To solve this, we embed the images into a low-dimensional latent space to obtain a viable distance measure. Inspired by Embed to Control (E2C) [36], we train a VAE [37] and enforce a locally linear dynamics model to regularize the structure of the latent space. The locally-linear dynamics model captures our goal for the latent space to be temporally consistent since we have a multi-step control task.

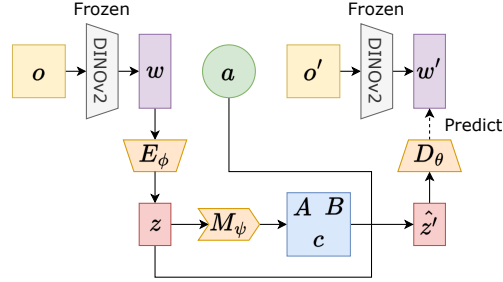


Figure 3: LaNE learns a latent space with locally linear dynamics. Given a transition tuple  $(o, a, o')$ , the observations are first encoded by a frozen DINOv2 model into  $w$  and  $w'$ . Next, the encoder  $E_\phi$  further embeds  $w$  into a low-dimensional latent state  $z$ . The forward model  $M_\psi$  predicts the transition matrices  $A$ ,  $B$  and offset  $c$ . Finally, the decoder  $D_\theta$  reconstructs  $w'$  from the predicted  $\hat{z}'$ , where  $\hat{z}' = Az + Ba + c$ . The trainable modules  $E_\phi$ ,  $M_\psi$ , and  $D_\theta$  are colored in orange.

Our method differs from E2C in three key ways: **1.** LaNE leverages the feature extractor from a strong pre-trained image model by embedding and predicting the frozen DINOv2 [35] features. As we find with ablation studies in Section 5.2, learning on top of DINOv2 features  $w$  is superior to learning directly from pixels  $o$ . **2.** We learn a latent space robust to pixel-space perturbations. It has been shown that data augmentation is crucial for efficient and robust image-based reinforcement learning [8, 9, 31, 33]. Hence, we apply a random data-augmentation function  $f(\cdot)$  during representation learning. **3.** Our RL policy network uses a separate CNN encoder to learn in an unconstrained embedding space and benefit from low inference latency. Overall, the trainable components include the encoder  $E_\phi$ , the decoder  $D_\theta$ , and the transition model  $M_\psi$ .

LaNE optimizes a variational lower bound (ELBO) objective across transition tuples  $(o, a, o')_i$  sampled from the replay buffer. We assume the latent states  $z$  form a unit Gaussian prior  $p(\mathbf{z}) \sim \mathcal{N}(0, \mathcal{I})$ . The encoded distributions  $q(\mathbf{z} | w)$  and decoded distribution  $p(\mathbf{w} | z)$  are also modeled with Gaussian distributions. During training, the data-augmented observation features are encoded into their latent distributions whose mean  $\mu$  and diagonal covariance matrix  $\Sigma$  are predicted by the encoder network  $E_\phi$ :

$$\begin{aligned}z &\sim q_\phi(\mathbf{z} | w) = \mathcal{N}(\mu, \Sigma), \text{ where } (\mu, \Sigma) = E_\phi(w), \quad w = \text{DINOv2}(f(o)) \\ z' &\sim q_\phi(\mathbf{z}' | w') = \mathcal{N}(\mu', \Sigma'), \text{ where } (\mu', \Sigma') = E_\phi(w'), \quad w' = \text{DINOv2}(f(o')).\end{aligned}\tag{3}$$

149 The one-step forward model in the latent space is locally linear in the state and action, whose param-  
 150 eters (matrices  $A$ ,  $B$  and offset  $c$ ) depend on the starting state, as predicted by the latent transition  
 151 model  $M_\psi$ . Prior work shows that a latent linear dynamics model is tractable to learn but provides  
 152 modeling flexibility through local linearity [36]. The linear transition model allows the prediction  
 153 of the next step latent distribution using the current distribution and action as follows:

$$\hat{z}' = Az + Ba + c, \text{ where } (A, B, c) = M_\psi(z) \quad (4)$$

$$q_\psi(\hat{z}' | z, a) = \mathcal{N}(\hat{\mu}', \hat{\Sigma}'), \text{ where } \hat{\mu}' = A\mu + Ba + c, \hat{\Sigma}' = A\Sigma A^T. \quad (5)$$

154 Finally, the decoder  $D_\theta$  reconstructs the next step observation embedding from the predicted next  
 155 step latent vector:

$$\hat{w}' = D_\theta(\hat{z}'). \quad (6)$$

156 The encoder  $E_\phi$ , decoder  $D_\theta$ , and transition model  $M_\psi$  are updated jointly using a combined loss  
 157 with three terms. First, we want the sampled starting latent state  $z$  to be reconstructed back to  
 158 the original image features  $w$ . Similarly, as we pass the sample through the dynamics model, the  
 159 resulting latent state prediction  $\hat{z}'$  should be reconstructed back to  $w'$ . Finally, to ensure the latent  
 160 dynamics model is consistent over multiple steps, we want the predicted distribution  $q_\psi(\hat{z}' | w, a)$   
 161 and encoded distribution  $q_\phi(z' | w')$  to be similar. Formally, we write the overall training objective  $\mathcal{L}$   
 162 as follows, where  $\lambda$  is a hyper-parameter for weighing the two loss terms:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{z \sim q_\phi, \hat{z}' \sim q_\psi} \left[ -\log p(w|z) - \log p(w'|\hat{z}') \right] + D_{\text{KL}} \left( q_\phi(z | w) \parallel p(z) \right) \quad (7)$$

$$\mathcal{L}_{\text{dynamics}} = \mathbb{E}_{z \sim q_\phi} \left[ D_{\text{KL}} \left( q_\psi(\hat{z}' | z, a) \parallel q_\phi(z' | w') \right) \right] \quad (8)$$

$$\mathcal{L} = \mathbb{E}_{(o, a, o') \in \mathcal{B}} \left[ \mathcal{L}_{\text{ELBO}} + \lambda \mathcal{L}_{\text{dynamics}} \right]. \quad (9)$$

163 In essence, we minimize the reconstruction error for the VAE using the ELBO objective (term 1)  
 164 and the forward prediction error in the latent space using KL divergence (term 2).

165 The learned latent space allows us to define a dynamics-aware distance measure between observa-  
 166 tions. Specifically, for two observations  $o_1$  and  $o_2$ , we define the **Augmentation-invariant Distance**  
 167 **Measure (ADM)** to be the root quadratic cost between the augmented and *encoded* states  $z_1$  and  $z_2$ :

$$d(o_1, o_2) := ((z_1 - z_2)^T Q (z_1 - z_2))^{\frac{1}{2}}. \quad (10)$$

168 Our design echoes the quadratic cost function commonly used in optimal control. Using an identity  
 169 weighting matrix  $Q = I$  further simplifies ADM to the Euclidean distance in the latent space. We  
 170 apply this simplification in our experiments, but other weighting matrices could be useful when the  
 171 agent observes both images and proprioceptive states.

### 172 4.3 Demonstration-Guided Exploration

173 We propose a systematic reward-engineering approach to credit the agent for staying close to demon-  
 174 strations. Given an experience tuple  $(o, a, o', r, d)$ , we assign an additional exploration reward  $r_e^*$  if  
 175  $o'$  is sufficiently close to a demonstrated state, up to a distance threshold  $\epsilon$ , which is dynamically  
 176 computed. We define  $\epsilon$  as the average distance between consecutive demonstration observations:

$$\epsilon := \mathbb{E}_{i, t} [d(o_t^i, o_{t+1}^i)], \quad o_t^i, o_{t+1}^i \in \mathcal{D}. \quad (11)$$

177 The threshold  $\epsilon$  approximates the distance of one environment step. As the agent gathers new ex-  
 178 periences, we re-compute  $\epsilon$ . This is necessary because the encoder  $E_\phi$ , decoder  $D_\theta$ , and dynamics  
 179 model parameters  $M_\psi$  are constantly updated with the latest experience to ensure that ADM is not  
 180 overfitted to only the demonstration data. In addition, we find the trajectory index  $i$  and time step  $t$   
 181 of the nearest demonstration using the ADM  $d$ :

$$i^*, t^* = \arg \min_{i, t} d(o', o_t^i), \quad o_t^i \in \mathcal{D}. \quad (12)$$

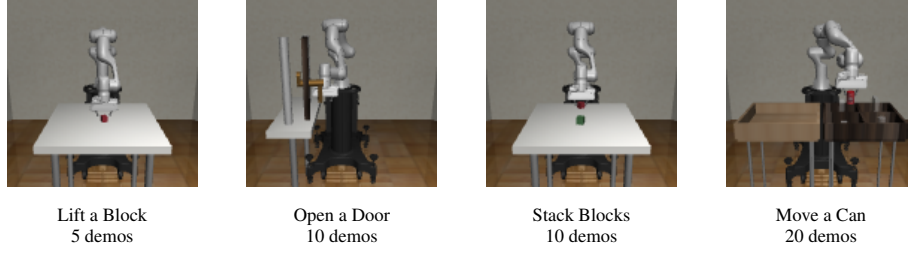


Figure 4: LaNE achieves state-of-the-art sample efficiency in four Robosuite visual manipulation tasks. The RL agent observes RGB images from 2 cameras, one in the front (shown above) and the other on the gripper.

After we compute  $\epsilon$ ,  $i^*$ , and  $t^*$ , we assign a dense reward  $r_{\text{dense}}$  by augmenting the environment reward with a *task-progress-informed* exploration reward:

$$r_{\text{dense}} = \begin{cases} r + \alpha^{T_{i^*} - t^*} r_e & [d(o', o_{t^*}^*) \leq \epsilon] \wedge [o' \notin \mathcal{G}] \\ r & \text{otherwise.} \end{cases} \quad (13)$$

Here, the exploration reward  $r_e^* = \alpha^{T_{i^*} - t^*} r_e$  is modulated by the expected step to success, which is the difference between the demo trajectory length  $T_{i^*}$  and  $t^*$ . In the context of LfD,  $r_e^*$  can be interpreted as a point estimate of the potential function at  $o'$ , echoing prior work in this domain [17, 20]. The discounting factor  $\alpha$  is a hyper-parameter chosen independently from the RL discounting factor  $\gamma$ , and the nominal exploration reward  $r_e$  is a constant.

Inspecting the augmented reward  $r_{\text{dense}}$ , we see that when  $o'$  finds its nearest neighbor close to one of the successful terminal states,  $o'$  is awarded almost the full nominal exploration reward  $r_e$ . When  $o'$  is close to one of the earlier steps in a demonstration,  $r_e$  is heavily discounted. Finally, if we are very far from any demonstration observation (relative to the distance threshold  $\epsilon$ ), or if we are at the goal, the RL agent gets only the environment reward  $r$  (case 2 in Eq. 13). LaNE is versatile because we can train a control policy to maximize  $r_{\text{dense}}$  using any off-the-shelf RL algorithm.

#### 4.4 Prioritized Replay and Value Clipping

We improve training efficiency and stability by performing prioritized replay and  $Q$ -value clipping. Prioritized replay is a standard tool when learning from demonstrations [19]: in each batch of  $b$  transitions, we sample at least  $p_d$  fraction from the demonstrations, where  $b$  and  $p_d$  are hyper-parameters. Conservative  $q$ -value estimates are also widely used to stabilize training [1, 38]. LaNE stands out because  $r_{\text{dense}}$  derives upper and lower bounds on the  $q$ -value landscape, allowing us to use a clipped value target when performing temporal difference updates.

The definition of  $r_{\text{dense}}$  in Eq. 13 contains the nominal exploration reward  $r_e$ , a constant hyper-parameter. We pick  $|r_e| \leq |r_{\text{live}}|$  to obtain bounded  $Q$ -values. Under  $r_{\text{dense}}$ , a transition either receives a positive reward  $r_{\text{done}}$  and terminates the episode or receives a non-positive reward  $r_{\text{live}} + \mathbb{1} \cdot r_e^*$ . The highest  $Q$ -value is achieved at task completion. On the other hand, in the worst case where the agent receives  $r_{\text{live}}$  all the time and never succeeds, the  $Q$ -value is bounded below by  $\sum_{t=0}^{\infty} \gamma^t r_{\text{live}} = (1 - \gamma)r_{\text{live}}$ . Thus, we obtain the following bound:

$$(1 - \gamma)r_{\text{live}} \leq Q(o, a) \leq r_{\text{done}}.$$

## 5 Experiments

### 5.1 Simulated Manipulation Tasks

We solve four robot manipulation tasks from the Robosuite simulator [11]: lifting a block, opening a door, stacking blocks, and moving a soda can. The RL agent observes  $128 \times 128$  RGB images from two cameras, one mounted on the robot gripper and one in front of the robot. The robot uses Operational Space Control – the agent predicts actions to change the robot’s hand displacement,

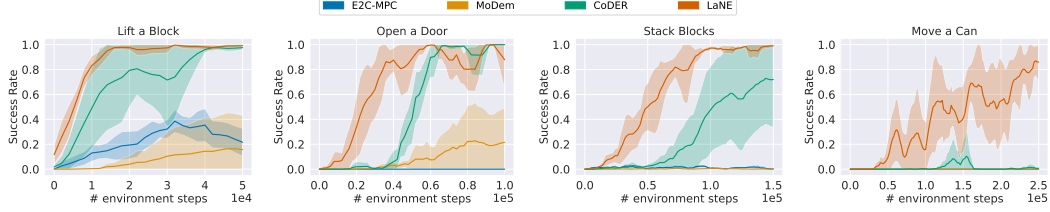


Figure 5: We compare LaNE with optimal control and state-of-the-art RL methods: E2C [36], MoDem [32] and CoDER [31]. Our method (red) consistently learns faster and converges to higher success rates than all three baseline methods.

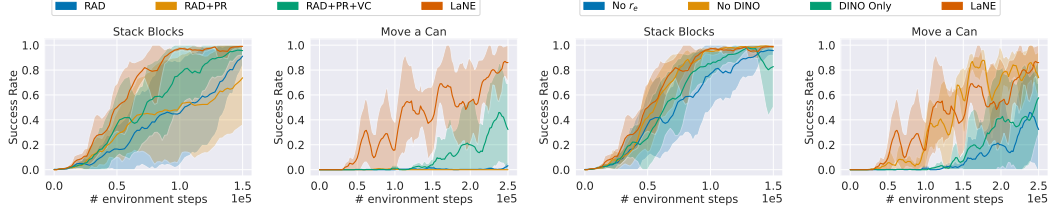


Figure 6: **Left:** Starting from RAD [8], we add PR (prioritized replay), VC (value clipping), and exploration reward  $r_e$  to understand their contributions. PR and VC are necessary for efficient learning, but the exploration reward is the most crucial and significantly improves. **Right:** We experiment with LaNE variants without pre-training (No DINO) or fine-tuning (DINO Only). While either variant improves from the baseline where no exploration reward is given, combining both results in the quickest exploration.

214 rotation, and gripper width for 7 degrees of freedom. Figure 4 shows one of the camera angles.  
 215 The demonstrations come from a hand-coded controller that utilizes state information. The state is  
 216 unavailable for the RL agent, who must learn from images alone.

217 We compare LaNE with optimal control, model-free, and model-based RL. For E2C [36], we run  
 218 an MPC controller in the latent space to minimize a quadratic state cost towards the demonstrated  
 219 goal state. CoDER [31] and MoDem [32] are state-of-the-art vision-based RL algorithms in the LfD  
 220 setting. We initiate all methods with the same demonstrations and measure the evaluation success  
 221 rates during training. Figure 5 shows the mean and standard deviation across five random seeds as a  
 222 function of training environment steps. Our method outperforms all three baselines across all tasks  
 223 while showing major advantages in the two more challenging tasks.

## 224 5.2 Ablation Studies

225 Our approach utilizes multiple techniques to maximize the utility of demonstrations and to speed  
 226 up learning. We perform ablation studies to answer two key questions: **1.** Is reward augmentation  
 227 really necessary, or are regularization tricks like prioritized replay (PR) and value clipping (VC) by  
 228 themselves sufficient? **2.** LaNE learns an embedding space by fine-tuning a pre-trained computer  
 229 vision model. Are both the fine-tuning step and the pre-trained DINOv2 model necessary?

230 For question 1, we start from a standard image-based RL algorithm RAD [8] (initialized with demon-  
 231 strations) and introduce our key components one at a time, namely prioritized replay (PR), value  
 232 clipping (VC), and the exploration reward  $r_{\text{dense}}$ . As shown by the left half of Figure 6, importance  
 233 sampling and value clipping are helpful, but  $r_{\text{dense}}$  makes the most significant difference, allowing  
 234 the robot to complete the long-horizon task reliably much earlier in training.

235 For question 2, we experiment with two variants of LaNE, one without DINOv2 and one without  
 236 fine-tuning. When LaNE trains without DINOv2, it initializes the encoder and decoder from scratch  
 237 and learns to predict the images directly. When LaNE uses the pre-trained model and skips the fine-  
 238 tuning steps, we directly use the Euclidean distance between their respective DINOv2 embedding  
 239 vectors. The results in the right side of Figure 6 demonstrate that performing reward augmentation  
 240 without pre-trained DINOv2 or fine-tuning is useful, but combining both allows the agent to explore  
 241 the most efficiently.



Figure 7: We deploy LaNE on four tasks with the Franka Panda robot: reach a fixed location, lift a block, open a drawer, and insert a pen. The RL agent observes two RGB images, one in front of the robot and one on its wrist. Trained from scratch, the robot achieves a 10/10 evaluation success rate with only a few demonstrations and less than one hour of learning.

### 5.3 Real Robot Experiments

As Figure 7 illustrates, LaNE enables efficient RL in four diverse tasks with the Franka Panda robot. From the easiest to the hardest, the tasks are Reaching a Fixed Goal, Opening a Drawer, Lifting a Block (random location), and Inserting a Pen. Demonstrations are collected via teleoperation from an iPhone app, which uses inside-out tracking to stream the device’s pose.

Results show that our method is extremely data-efficient, leading to task success with under an hour of training. Table 1 shows the number of demonstrations provided and training performance. For the most straightforward goal-reaching task, LaNE uses only one demo and completes the task for the first time in only 13 minutes. The agent finds a consistently successful policy quickly, only 8 minutes later. With five demonstrations, LaNE can learn more challenging tasks with high stochasticity or requiring precision, including lifting a block and inserting a pen. Overall, the results prove that our approach is outstanding in simulation and practical in the real world.

Task		Reach	Drawer	Lift	Insert
Demos	Episodes	1	1	5	5
	Steps	12	18	103	135
	Time	0:20	0:25	2:00	3:00
CoDER	Steps	N/A	N/A	N/A	N/A
First Success	Time	> 1h	> 1h	> 1h	> 1h
LaNE (Ours)	Steps	476	1054	1109	1854
First Success	Time	13:10	31:51	30:07	38:01
LaNE (Ours)	Steps	806	1929	1820	2927
Convergence	Time	20:55	51:40	44:16	57:58

Table 1: We deploy LaNE on a Franka Panda arm to learn manipulation tasks requiring up to 7 degrees of freedom. Human demonstrations are provided via teleoperation, taking only a few minutes. LaNE trains every task to a 10/10 success rate from scratch under one hour of wall clock time. In comparison, our strongest baseline CoDER [31] fails to succeed even once during the first hour of training.

## 6 Conclusion

This paper presents LaNE, a data-efficient RL algorithm to learn sparse reward tasks from image observations by utilizing a few demonstrations. LaNE outperforms state-of-the-art benchmarks and enables under-an-hour RL training with a real robot. Our key innovation is to learn a latent dynamics model, which provides a temporally consistent embedding space. When a transition lands sufficiently close to a demonstration, we assign an extra task-progress-informed reward modulated by the distance to the goal. As such, we convert a sparse reward task to a task with dense proxy rewards, dramatically improving learning efficiency. Our work lends itself to exciting future directions. For example, we can leverage recent advancements in causal RL and counterfactual analysis [39, 40, 41] to determine the state in an expert demonstration that directly caused task success. This might further improve our search for the nearest demonstration and overall learning.

## References

- [1] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018. URL <https://proceedings.mlr.press/v80/haarnoja18b.html>.
- [2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. ISSN 00280836. URL <http://dx.doi.org/10.1038/nature14236>.
- [3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.
- [4] Y. Wang, H. He, X. Tan, and Y. Gan. Trust region-guided proximal policy optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL <https://proceedings.neurips.cc/paper/2019/file/a666587afda6e89aec274a3657558a27-Paper.pdf>.
- [5] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. doi:10.1177/0278364919887447. URL <https://doi.org/10.1177/0278364919887447>.
- [6] D. Hafner, T. P. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=0oabwyZb0u>.
- [7] M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119*, 2020. arXiv:2004.04136.
- [8] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19884–19895. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/file/e615c82aba461681ade82da2da38004a-Paper.pdf>.
- [9] D. Yarats, I. Kostrikov, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=GY6-6sTvGaf>.
- [10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.
- [11] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.
- [12] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function, 2018.
- [13] K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control, 2016. URL <https://arxiv.org/abs/1611.07507>.

- [14] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *CoRL*, abs/1806.10293, 2018. URL <http://arxiv.org/abs/1806.10293>.
- [15] G. Cideron, B. Tabanpour, S. Curi, S. Girgin, L. Hussenot, G. Dulac-Arnold, M. Geist, O. Pietquin, and R. Dadashi. Get back here: Robust imitation by return-to-distribution planning, 05 2023.
- [16] R. Dadashi, L. Hussenot, M. Geist, and O. Pietquin. Primal wasserstein imitation learning. *ArXiv*, abs/2006.04678, 2020. URL <https://api.semanticscholar.org/CorpusID:219531578>.
- [17] Y. Guo, J. Gao, Z. Wu, C. Shi, and J. Chen. Reinforcement learning with demonstrations from mismatched task under sparse reward. In *Conference on Robot Learning*, 2022. URL <https://api.semanticscholar.org/CorpusID:254246549>.
- [18] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, G. Dulac-Arnold, J. Agapiou, J. Z. Leibo, and A. Gruslys. Deep q-learning from demonstrations. *AAAI’18/IAAI’18/EAAI’18*. AAAI Press, 2018. ISBN 978-1-57735-800-8.
- [19] M. Večerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. 07 2017.
- [20] Y. Wu, M. Mozifian, and F. Shkurti. Shaping rewards for reinforcement learning with imperfect demonstrations using generative models. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6628–6634, 2021. doi:10.1109/ICRA48506.2021.9561333.
- [21] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, page 6292–6299. IEEE Press, 2018. doi:10.1109/ICRA.2018.8463162. URL <https://doi.org/10.1109/ICRA.2018.8463162>.
- [22] I.-C. A. Liu, S. Uppal, G. S. Sukhatme, J. J. Lim, P. Englert, and Y. Lee. Distilling motion planner augmented policies into visual control policies for robot manipulation. In *5th Annual Conference on Robot Learning*, 2021. URL <https://openreview.net/forum?id=NZnz3cExrDW>.
- [23] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In H. Kress-Gazit, S. S. Srinivasa, T. Howard, and N. Atanasov, editors, *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, 2018. doi:10.15607/RSS.2018.XIV.049. URL <http://www.roboticsproceedings.org/rss14/p49.html>.
- [24] P. Ladosz, L. Weng, M. Kim, and H. Oh. Exploration in deep reinforcement learning: A survey. *Inf. Fusion*, 85(C):1–22, sep 2022. ISSN 1566-2535. doi:10.1016/j.inffus.2022.03.003. URL <https://doi.org/10.1016/j.inffus.2022.03.003>.
- [25] A. D. Laud. *Theory and Application of Reward Shaping in Reinforcement Learning*. PhD thesis, USA, 2004. AAI3130966.
- [26] S. Cabi, S. Gómez, A. Novikov, K. Konyushova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Bud-den, M. Vecerik, O. Sushkov, D. Barker, J. Scholz, M. Denil, N. Freitas, and Z. Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning. In *Robotics: Science and Systems*, 07 2020. doi:10.15607/RSS.2020.XVI.076.

- [27] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.
- [28] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. Shakkottai. Reinforcement learning with sparse rewards using guidance from offline demonstration. *International Conference on Learning Representations (ICLR)*, 2022. URL <https://par.nsf.gov/biblio/10327543>.
- [29] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Neural Information Processing Systems*, 2016. URL <https://api.semanticscholar.org/CorpusID:16153365>.
- [30] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. *ArXiv*, abs/1710.11248, 2017. URL <https://api.semanticscholar.org/CorpusID:21529792>.
- [31] A. Zhan, R. Zhao, L. Pinto, P. Abbeel, and M. Laskin. Learning visual robotic control efficiently with contrastive pre-training and data augmentation. In *International Conference on Robotics and Automation*, 2022. URL <https://arxiv.org/abs/2012.07975>.
- [32] N. Hansen, Y. Lin, H. Su, X. Wang, V. Kumar, and A. Rajeswaran. Modem: Accelerating visual model-based reinforcement learning with demonstrations. In *International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=JdTnc9gjVfJ>.
- [33] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.
- [34] Z. Yuan, Z. Xue, B. Yuan, X. Wang, Y. Wu, Y. Gao, and H. Xu. Pre-trained image encoder for generalizable visual reinforcement learning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL <https://openreview.net/forum?id=FQtku8rkp3>.
- [35] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.
- [36] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL <https://proceedings.neurips.cc/paper/2015/file/a1afc58c6ca9540d057299ec3016d726-Paper.pdf>.
- [37] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.
- [38] H. v. Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI’16*, page 2094–2100. AAAI Press, 2016.
- [39] E. Bareinboim and J. Pearl. Causal inference and the data-fusion problem. *Proceedings of the National Academy of Sciences*, 113(27):7345–7352, 2016.
- [40] T. Mesnard, T. Weber, F. Viola, S. Thakoor, A. Saade, A. Harutyunyan, W. Dabney, T. Stepleton, N. Heess, A. Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. *arXiv preprint arXiv:2011.09464*, 2020.

- 409 [41] S. Zhu, I. Ng, and Z. Chen. Causal discovery with reinforcement learning. *arXiv preprint*  
410 *arXiv:1906.04477*, 2019.
- 411 [42] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven  
412 reinforcement learning, 2020.
- 413 [43] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press,  
414 second edition, 2018. URL <http://incompleteideas.net/book/the-book-2nd.html>.
- 415 [44] A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger. Cartesian impedance control of redundant  
416 robots: recent results with the dlr-light-weight-arms. In *2003 IEEE International Conference*  
417 *on Robotics and Automation (Cat. No.03CH37422)*, volume 3, pages 3704–3709 vol.3, 2003.  
418 doi:10.1109/ROBOT.2003.1242165.
- 419 [45] S. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped  
420 least squares methods. *IEEE Transactions in Robotics and Automation*, 17, 05 2004.
- 421 [46] J. Linowes and K. Babilinski. *Augmented reality for developers: Build practical augmented*  
422 *reality applications with unity, ARCore, ARKit, and Vuforia*. Packt Publishing Ltd, 2017.

## 7 Appendix

### 7.1 Motivating Example

We motivate the need to learn a dynamics-aware embedding space: finding a good distance measure between states from their respective high-dimensional image observations is non-trivial. We use the PointMaze environment from the D4RL benchmark [42] to provide a clear illustration. In this environment, the controllable point mass is marked in green. As shown in Figure 8, we place the point at three positions in the maze such that state (a) is much closer to state (b) than state (c).

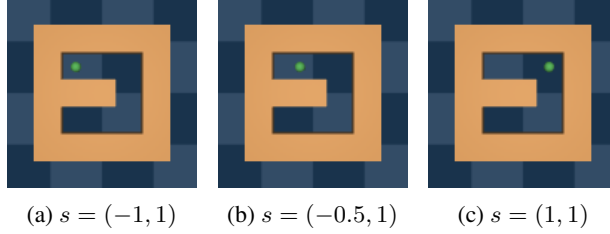


Figure 8: Observations from the PointMaze environment. The point mass in green is the controllable agent, whose location is indicated by  $s$ . As indicated by the agent’s location, state  $b$  is closer to state  $a$  than  $c$ .

Distance Measure $d$	$d(a, b)$	$d(a, c)$	$d(a, c)/d(a, b)$
Pixel $L^2$	2.835	3.303	1.165
DINOv2 $L^2$	0.004	0.003	0.75
Ground Truth $L^2$	0.5	1.5	<b>3</b>
ADM (Ours)	0.046	0.135	<b>2.935</b>

Table 2: Comparison across different distance measures in PointMaze. Distance in the pixel space and DINOv2 embedding space cannot capture the true relations of the underlying states. Whereas our proposed ADM can estimate the relative distance the most accurately.

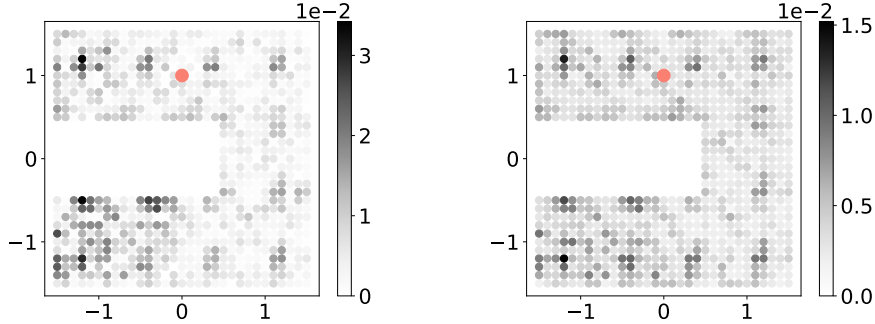


Figure 9: **Left:** ADM (Ours). **Right:** DINOv2. We plot the  $L^2$  distance in ADM and DINOv2 embedding space from each location in the maze to the pink location. In the ADM embedding space, computed distances match the environment dynamics, showing the lower left corner as the farthest due to the turn. In contrast, DINOv2 cannot identify the differences between states.

Table 2 compares two baseline distance measures with our proposed method ADM, trained from 10000 random interactions in the environment. For Pixel  $L^2$ , we directly compute the Euclidean distance. As expected,  $d(a, b)$  and  $d(a, c)$  are not distinguishable from the pixel-wise distance. Similarly, the pre-trained DINOv2 embeddings do not capture the transition information and fail to estimate the distance between states. ADM is the only method to capture the relative proportions of the ground-truth distances.

In addition, we visualize the quality of the learned embeddings in Fig. 9. We show that the learned embeddings respect the dynamics of the maze, understanding that the point must go around the corner instead of through the wall. This simple experiment illustrates the benefits of using our latent space distance measure to quantify the task-relevant similarity between image observations.

## 7.2 Visualization of Embedded Robot Trajectories

This section aims to provide a better intuitive understanding of the learned ADM embedding space. We visualize the learned embeddings in the real robot Reach task by plotting their 2-D projections in Fig. 10. Specifically, we plot a demonstration, a random successful, and a random unsuccessful episode. Unsurprisingly, we find a clear separation between successful and failed episodes: the successful episode closely follows the demonstration. In addition, we observe that the latent states move linearly as the agent progresses in the task.

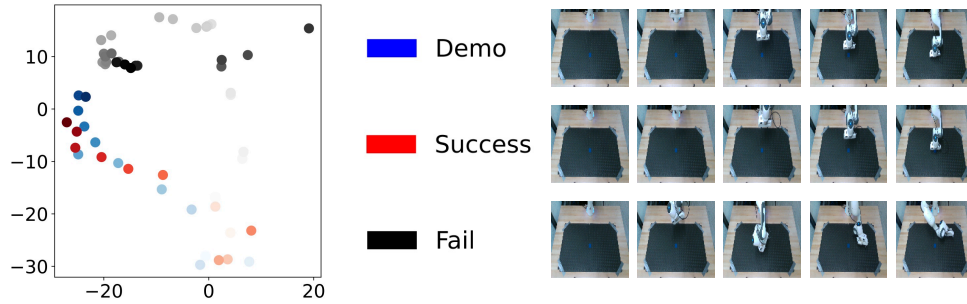


Figure 10: 2-D t-SNE of a demonstration, a successful and an unsuccessful trajectory on the Reach task. Colors change from light to dark as the episodes progress. The successful episode is mapped closer to the demonstration than the failed episode.

## 7.3 Exploration Reward Ablation Studies

To justify our choice of the exploration reward shown in Eqn. 13, we conduct ablation studies on variants of the exploration reward. 1. We set  $\alpha = 1$  to make sure the discounting is necessary. 2. Alternatively, we use a simpler reward based on the L2 distance between the observation and its nearest-neighbor demonstration:  $r_e = \max(1, \frac{\epsilon}{d(o', o_{t^*}^*)})$ . This simplified reward is the inverse L2, scaled to the average distance between demonstration states, and clipped to make the  $q$ -value bound hold. Our method holds a clear advantage in the long-horizon can-moving task.

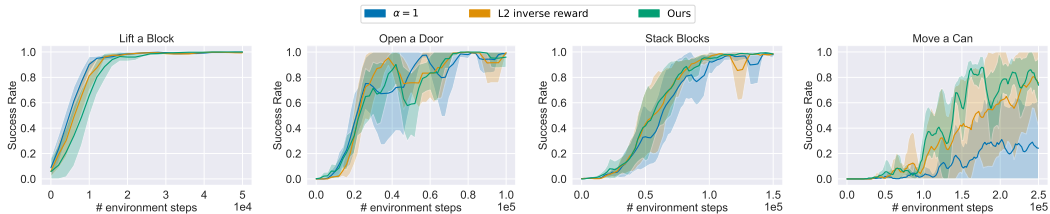


Figure 11: We conduct ablation studies on 2 other exploration rewards. We observe a clear advantage of our proposed reward in the long-horizon task and similar performance in shorter-horizon tasks.

## 7.4 Comparison with DreamerV2

We compare LaNE with DreamerV2, another state-of-the-art model-based RL method. Because DreamerV2 does not explicitly take demonstrations, we pre-fill its replay buffer with demonstration trajectories. In addition, we allow DreamerV2 to run for longer to get a better sense of its performance. As shown in Fig. 12 below, it takes at least 5 times the environment steps for DreamerV2 to learn the Robosuite block-lifting task compared to LaNE. We point out that DreamerV2 does not distinguish the pre-filled demonstrations from the regular episodes and thus fails to sample the sparse reward often enough. Nevertheless, LaNE can also integrate with model-based RL methods to boost their sample efficiency in sparse-reward settings.

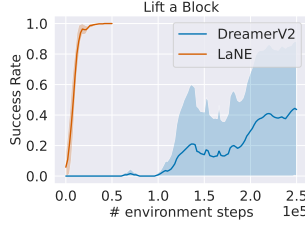


Figure 12: DreamerV2 takes over 5 times samples to learn the Robosuite Lift a Block task compared to LaNE because it does not distinguish between demonstration and regular episodes.

## 7.5 Robustness of LaNE Policies

We evaluate the out-of-distribution robustness of trained LaNE policies. Specifically, we push the block and the pen with a stick during policy execution, as illustrated below in Fig. 13. We find that the policies are robust to small perturbations. For example, the agent can lift the yellow block when pushed, even when the stick and the operator’s arm enter the camera view. Additionally, when the pen is pushed to unseen locations, the robot can still pick it up.

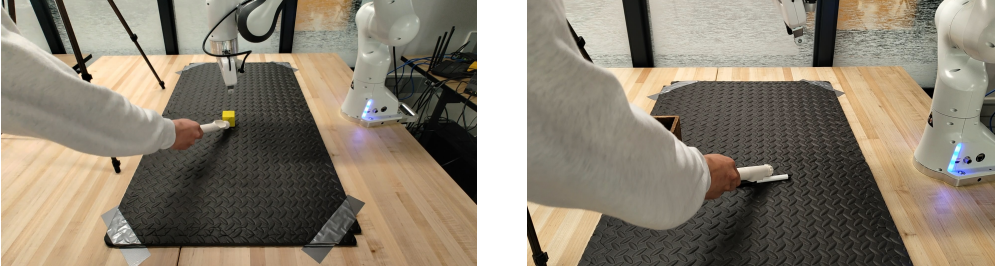


Figure 13: The learned LaNE policies are robust to small perturbations during evaluation, including moving objects, unseen initialization locations, and foreign objects.

## 7.6 Real Robot Setup

### 7.6.1 Highly Compliant Real-time Controller

Because reinforcement learning requires a trial-and-error process [43], we expect the robot to make frequent physical contact with the environment. To eliminate safety hazards, the controller must be compliant with external forces. On the other hand, we want the robot to move swiftly so that each RL step takes less time to execute. We develop a highly compliant real-time controller extended from the Cartesian Impedance Controller [44, 45].

At a high level, the end-effector tracks an equilibrium pose following a mass-spring-damper model. The robot asserts higher torque in the opposite direction as the current pose deviates further from the equilibrium. In the Cartesian space, we limit the maximum force exerted on the end effector by the robot, preventing it from causing damage. In the joint space, we apply a counter torque when a joint gets close to its hardware limit. Combining these control rules builds a safety net around the robot for smoother RL training.

### 7.6.2 System Architecture

The robot uses two Intel Realsense cameras – one mounted on the end-effector and another in front of the robot. Our method only utilizes color images. Both cameras are connected via USB to an Nvidia GPU desktop, which runs inference and training for the RL agent. Specifically, the GPU desktop runs an OpenAI gym interface [10], with which the RL agent interacts. At each timestep, the agent chooses its action based on its policy:  $a = \pi(o)$ . The action  $a$  consists of a displacement of the end effector position, change in roll/pitch/yaw, and open/close of the gripper. Next, the action

selected by the RL agent is sent via ethernet to an Intel NUC, which directly interfaces with the Panda Robot. Specifically, the Intel NUC runs the Robot Operating System (ROS), where our real-time controller communicates with the FRANKA\_ROS library.

### 7.6.3 Demonstration Collection via Tele-Operation

A key aspect of our approach is to learn from a small set of human demonstrations efficiently. We build an application where a user can tele-operate the robot by moving and rotating an iPhone. Our iPhone application utilizes primitives from Apple’s ARKit [46] to stream the position and orientation of the device to the PC controlling the robot.

During demonstration collection, a Python script translates the tracking data into gym actions and executes them on the real robot. The gym environment updates the robot’s equilibrium pose to follow the movement of the iPhone. These demonstration trajectories are stored on the Nvidia GPU desktop and used during training.

### 7.7 Environment Details

*Maximum Episode Length:* For our simulated experiments in Robosuite, we set maximum episode lengths based on the difficulty of each task, as shown in Table 3. Note that these numbers are slightly over the average steps the demonstrator takes to complete each task, leaving the RL agent plenty of time to finish. For our real robot experiments, all three tasks share the same maximum episode length of 30 steps. The episodes are terminated if the maximum length is reached or when the task is completed.

Task	Lift	Door	Stack	Move a Can
# Max Episode Length	40	80	80	120

Table 3: We choose the maximum episode length of each task slightly over the average steps the demonstrator took, giving the RL agent ample time to finish.

*Task-specific Settings:* The block-lifting, door-opening, and block-stacking tasks are all taken directly from the Robosuite simulator. The move-a-can task is a single-object version of the pick-and-place task where only the soda can is included. The robot’s base is placed in the middle of the two bins for the move-a-can task.

### 7.8 Hyper-parameters and Training Scheduling

We use discounting factor  $\gamma = 0.99$  and exploration reward discount  $\alpha = 0.98$  for all our experiments. Actor and critic learning rates are  $10^{-3}$ . The latent dynamics model learning rate is  $4 \cdot 10^{-3}$ . Batch size  $B = 128$ . We choose different fractions of demonstrations  $p_d$  when we perform prioritized replay, as shown in the table below.

Task	Lift	Door	Stack	Move a Can	Real Robot
$p_d$	0.15	0.15	0.15	0.2	0.2

Table 4: We vary the fraction  $p_d$  of demonstration interactions within each batch depending on the task we are training.

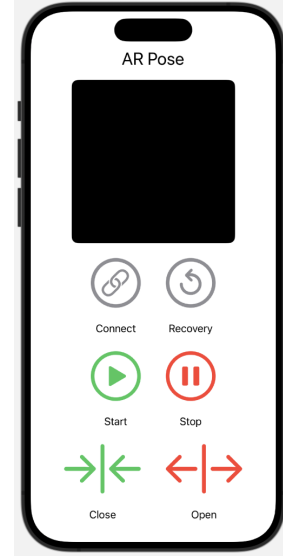


Figure 14: We develop an iOS app to stream the device pose to the PC.

For simulated environments, we perform one Actor-Critic update for each environment step. After loading the demonstrations, we update the latent dynamics model for every 5000 environment steps until convergence. For real robot experiments, we update the latent dynamics model for every 30 steps the robot takes. Since each step takes about 0.5 seconds to execute on the real robot, we keep performing SAC updates while the robot is in motion.

## 7.9 Neural Network Architectures

Our method consists of 6 components parameterized by neural networks, namely: model encoder  $E_\phi$ , model decoder  $D_\theta$ , locally-linear dynamics model  $M_\psi$ , RL encoder  $E_{\text{RL}}$ , Actor  $\pi_{\text{RL}}$  and Critic  $Q_{\text{RL}}$ . The numbers below correspond to our specific setting where the latent space has 16 dimensions, and the action space has 7 dimensions. Our locally linear dynamics model predicts a low-rank approximation of  $16 \times 16$  matrix  $A$  using two 16-dimensional vectors  $u$  and  $v$ , where  $A = I + uv^T$ .

```

535 Model Encoder (with DINOv2):
536 Input: 2 of 3x112x112 randomly cropped images
537 Pretrained DINOv2 variant: dinov2_vits14_reg
538 Concatenate 2 of DINOv2 embeddings
539 ReLU(Linear(out_features=512))
540 ReLU(Linear(out_features=512))
541 ReLU(Linear(out_features=512))
542 Linear(out_features=32)
543 Output: 16-dim mean + 16-dim log-std

544 Model Decoder (with DINOv2):
545 Input: 16-dim latent vectors
546 ReLU(Linear(out_features=512))
547 ReLU(Linear(out_features=512))
548 ReLU(Linear(out_features=512))
549 Linear(out_features=2 * 768)
550 Output: 2 of predicted DINOv2 embeddings

551 Model Encoder (No DINOv2):
552 Input: 6x112x112 randomly cropped images
553 ReLU(LayerNorm(Conv2D(6, 32, kernel=3, stride=2)))
554 ReLU(LayerNorm(Conv2D(32, 32, kernel=3)))
555 ReLU(LayerNorm(Conv2D(32, 32, kernel=3)))
556 Flatten()
557 ReLU(Linear(out_features=128))
558 ReLU(Linear(out_features=128))
559 Linear(out_features=32)
560 Output: 16-dim mean + 16-dim log-std

561 Model Decoder (No DINOv2):
562 Input: 16-dim latent vectors
563 ReLU(Linear(out_features=128))
564 ReLU(Linear(out_features=128))
565 ReLU(Linear(out_features=32768))
566 Reshape into 128x16x16
567 Upsample into 128x32x32
568 ReLU(Conv2D(128, 128, kernel=3, stride=1, pad=1)))
569 Upsample into 128x64x64
570 ReLU(Conv2D(128, 128, kernel=3, stride=1, pad=1)))
571 Upsample into 128x128x128
572 Conv2D(128, 6, kernel=3, stride=1, pad=1))
573 Output: 6x128x128 images

574 Locally-Linear Dynamics model:
575 Input: 16-dim latent vectors
576 ReLU(Linear(out_features=512))
577 ReLU(Linear(out_features=512))
578 ReLU(Linear(out_features=160))
579 Output: 16-dim vector u + 16-dim vector
580         + 16x7 matrix B + 16-dim offset c

581 RL Encoder:
582 Input: 6x112x112 randomly cropped images
583 ReLU(LayerNorm(Conv2D(6, 32, kernel=3, stride=2)))
584 ReLU(LayerNorm(Conv2D(32, 32, kernel=3, stride=2)))
585 ReLU(LayerNorm(Conv2D(32, 32, kernel=3, stride=2)))
586 ReLU(LayerNorm(Conv2D(32, 32, kernel=3, stride=2)))
587 Flatten()
588 LayerNorm(Linear(out_features=32))
589 Output: 32-dim feature

```

```
590 Actor:
591 Input: 32-dim feature
592 ReLU(Linear(out_features=1024))
593 ReLU(Linear(out_features=1024))
594 Linear(out_features=14)
595 Output: 7-dim mean + 7-dim log-std

596 Critic:
597 Input: 32-dim feature + 7-dim action
598 ReLU(Linear(out_features=1024))
599 ReLU(Linear(out_features=1024))
600 Linear(out_features=1)
601 Output: Q-value
```