

## A IMPLEMENTATION DETAILS

The edge and the node model in GNN-based transition model, reward model, value model and actor model are MLP’s which consists of two hidden layers of 512 units each, LayerNorm and ReLU activations.

Target entropy parameter is set to -3 for Object Reaching task. For tasks with discrete action space we use the scale the entropy of a uniform random policy with coefficient 0.6, which means 1.66 for Navigation 5x5 and PushingNoAgent 5x5 tasks and 2 for Navigation 10x10 task.

Learning	Temp. Cooldown	1
	Temp. Cooldown Steps	30000
	LR for DVAE	0.0003
	LR for CNN Encoder	0.0001
	LR for Transformer Decoder	0.0003
	LR Warm Up Steps	30000
	LR Half Time	250000
	Dropout	0.1
	Clip	0.05
	Batch Size	24
	Epochs	150
DVAE	vocabulary size	4096
CNN Encoder	Hidden Size	64
Slot Attention	Iterations	3
	Slot Heads	1
	Slot Dim.	192
	MLP Hidden Dim.	192
	Pos Channels	4
Transformer Decoder	Layers	4
	Heads	4
	Hidden Dim	192

Table 1: Hyperparameters for SLATE

Gamma	0.99
Buffer size	1000000
Batch size	128
$\tau_{polyak}$	0.005
Buffer prefill size	5000
Number of parallel environments	16

Table 2: Hyperparameters for ROCA

## B SOFT ACTOR CRITIC

### B.1 SOFT ACTOR-CRITIC FOR CONTINUOUS ACTION SPACES

Soft Actor-Critic (SAC) (Haarnoja et al., 2018; 2019) is a state-of-the-art off-policy reinforcement learning algorithm for continuous action settings. The goal of the algorithm is to find a policy that maximizes the maximum entropy objective:

$$\pi^* = \operatorname{argmax}_{\pi} \sum_{i=0}^{\tau} \mathbb{E}_{(s_t, a_t) \sim d_{\pi}} [\gamma^t (R(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)))]$$

where  $\alpha$  is the temperature parameter,  $\mathcal{H}(\pi(\cdot|s_t)) = -\log \pi(\cdot|s_t)$  is the entropy of the policy  $\pi$  at state  $s_t$ ,  $d_\pi$  is the distribution of trajectories induced by policy  $\pi$ . The relationship between the soft state-value function and the soft action-value function is determined as

$$V(s_t) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t)}[Q(s_t, a_t) - \alpha \log(\pi(a_t|s_t))] \quad (1)$$

The soft action-value function  $Q_\theta(s_t, a_t)$  parameterized using a neural network with parameters  $\theta$  is trained by minimizing the soft Bellman residual:

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t) \sim D} [(Q_\theta(s_t, a_t) - R(s_t, a_t) - \gamma \mathbb{E}_{s_{t+1} \sim T(s_t, a_t)} V_{\bar{\theta}}(s_{t+1}))^2] \quad (2)$$

where  $D$  is a replay buffer of past experience and  $V_{\bar{\theta}}(s_{t+1})$  is estimated using a target network for  $Q$  and a Monte Carlo estimate of (1) after sampling experiences from the  $D$ .

The policy  $\pi$  is restricted to a tractable parameterized family of distributions. A Gaussian policy is often parameterized using a neural network with parameters  $\phi$  that outputs a mean and covariance. The parameters are learned by minimizing the expected KL-divergence between the policy and the exponential of the  $Q$ -function:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D} [\mathbb{E}_{a_t \sim \pi_\phi(\cdot|s_t)} [\alpha \log(\pi_\phi(a_t|s_t)) - Q_\theta(s_t, a_t)]] \quad (3)$$

After reparameterization of the policy with the standard normal distribution, the (3) becomes feasible for backpropagation:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D, \epsilon_t \sim \mathcal{N}(0,1)} [\alpha \log(\pi_\phi(f_\phi(\epsilon_t; s_t)|s_t)) - Q_\theta(s_t, f_\phi(\epsilon_t; s_t))] \quad (4)$$

where action are parameterized as  $a_t = f_\phi(\epsilon_t; s_t)$ .

The objective for the temperature parameter is given by:

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t)} [-\alpha(\log \pi(a_t|s_t) + \bar{H})] \quad (5)$$

where  $\bar{H}$  is a hyperparameter representing the target entropy. In practice, two separately trained soft Q-networks are maintained, and then the minimum of their two outputs are used to be the soft Q-network output.

## B.2 SOFT ACTOR-CRITIC FOR DISCRETE ACTION SPACES

While SAC solves problems with continuous action space, it cannot be straightforwardly applied to discrete domains since it relies on the reparameterization of Gaussian policies to sample action. A direct discretization of the continuous action output and Q value (SACD) was suggested by (Christodoulou, 2019). In the case of discrete action space,  $\pi_\phi(a_t|s_t)$  outputs a probability for all actions instead of a density. Thus, the expectation (1) can be calculated directly and used in the Q-function objective (2):

$$V(s_t) = \pi(s_t)^T [Q(s_t) - \alpha \log \pi(s_t)] \quad (6)$$

The temperature objective (5) changes to:

$$J(\alpha) = \pi(s_t)^T [-\alpha(\log \pi(s_t) + \bar{H})] \quad (7)$$

The expectation over actions in (3) can be calculated directly, which leads to the policy objective:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D} [\pi(s_t)^T [\alpha \log(\pi_\phi(s_t)) - Q_\theta(s_t, \cdot)]] \quad (8)$$

## C SLATE

The SLATE (Singh et al., 2022) model is used as an object-centric representations extractor from image-based observations  $s_t$ . It consists of a slot-attention module (Locatello et al., 2020), dVAE, and GPT-like transformer (Ramesh et al., 2021).

The purpose of dVAE is to reduce an input image of size  $H \times W$  into lower dimension representation by a factor of  $K$ . First, the observation  $s_t$  is fed into the encoder network  $f_\phi$ , resulting in log probabilities  $o_t$  for a categorical distribution with  $C$  classes. Then, these log probabilities are used

to sample relaxed one-hot vectors  $j_t^{\text{soft}}$  from the relaxed categorical distribution with temperature  $\tau$ . Each token from  $j_t^{\text{soft}}$  represents information about  $K \times K$  size patch of overall  $P = HW/K^2$  patches on the image. After that,  $j_t^{\text{soft}}$  the vector is being used to reconstruct observation  $\tilde{s}_t$  by these patches with the decoder network  $g_\theta$ .

$$\begin{cases} o_t = f_\phi(s_t) \\ j_t^{\text{soft}} \sim \text{RelaxedCategorical}(o_t; \tau); \\ \tilde{s}_t = g_\theta(j_t^{\text{soft}}) . \end{cases}$$

The training objective of dVAE is to minimize MSE between observation  $s_t$  and reconstruction  $\tilde{s}_t$ :

$$L_{dVAE} = \text{MSE}(s_t, \tilde{s}_t), \quad (9)$$

Discrete tokens  $j_t$ , obtained from categorical distribution, are mapped to embedding from learnable dictionaries. Those embeddings are summed with learned position embedding  $p_\phi$  to fuse information about patches on the image. Then, the resulting embeddings  $u_t^{1:P}$  are fed into the slot attention module. The slot attention returns  $N$  object slots  $z_t^{1:N}$ , which are vectors of the fixed dimension **Slot Dim**, along with  $N$  attention maps  $A_t^{1:N}$ .

$$\begin{cases} o_t = f_\phi(s_t); \\ j_t \sim \text{Categorical}(o_t); \\ u_t^{1:P} = \text{Dictionary}_\phi(j_t) + p_\phi; \\ z_t^{1:N}, A_t^{1:N} = \text{SlotAttention}_\phi(u_t^{1:P}) . \end{cases}$$

The transformer predicts log-probabilities autoregressively  $\hat{o}_t^i$  for path  $i$  from vectors  $\hat{u}_t^{<i}$  generated for previous patches, combined with object centric representations  $z_t^{1:N}$ . The vector  $\hat{u}_t^l$ ,  $l < i \in [1 : P]$  is formed dictionary embedding from previously generated token  $\hat{j}_t^l$  for path  $l$  with added position embedding  $p_\phi^l$ . The token  $\hat{j}_t^i$  is mapped to the class  $c \in C$  with the highest log-probability  $\hat{o}_{t,c}^i$ . The resulting token can be used to reconstruct observation  $\hat{s}_t$  by combining reconstructed patches  $\hat{s}_t^i$ .

$$\begin{cases} \hat{u}_t^{<i} = \text{Dictionary}_\phi(\hat{j}_t^{<i}) + p_\phi^i; \\ \hat{o}_t^i = \text{Transformer}_\theta(\hat{u}_t^{<i}; z_t^{1:N}); \\ \hat{j}_t^i = \arg \max_{c \in [1, C]} \hat{o}_{t,c}^i; \\ \hat{s}_t^i = g_\theta(\hat{j}_t^i) . \end{cases}$$

The training objective of the transformer is to minimize cross entropy between the distribution of tokens  $\hat{j}_t$  generated by the transformer and tokens  $j_t$  extracted by dVAE.

$$L_T = \sum_{i=1}^P \text{CrossEntropy}(z_t^i, \hat{z}_t^i) \quad (10)$$

Combining 9 and 10 we receive loss for the SLATE model:

$$L_{SLATE} = L_{dVAE} + L_T$$

Figure 1 illustrates the examples of original observations and slot-attention masks learned by the SLATE model in the Object Reaching and Shapes2D tasks.

## D OCRL FINE-TUNING DETAILS

We conducted additional experiments to tune the OCRL baseline in the tasks where it was outperformed by ROCA. In Navigation10x10 and PushingNoAgent5x5 tasks we went through combinations of hyperparameters, but did not observe significant improvements:

- Entropy coefficient: [0, 0.001, 0.01, 0.025, 0.05, 0.075, 0.1]
- Clip range: [0.1, 0.2, 0.4]
- Epochs: [10, 20, 30]
- Batch size: [64, 128]



Figure 1: Examples of observations and slots extracted by the SLATE model in the Object Reaching task (top), Navigation 10x10 task (middle), and Navigation 5x5 task (bottom).

## E ADDITIONAL EXPERIMENTS WITH DREAMERV3

In order to ensure a fair comparison with DreamerV3, we conducted experiments with a pretrained encoder obtained from the DreamerV3 model that solves the task. For all the tasks, we conducted experiments using two different modes: one with the encoder frozen and another with the encoder unfrozen. However, we did not observe any improvement in the convergence rate. The results are shown in the Figure 2.

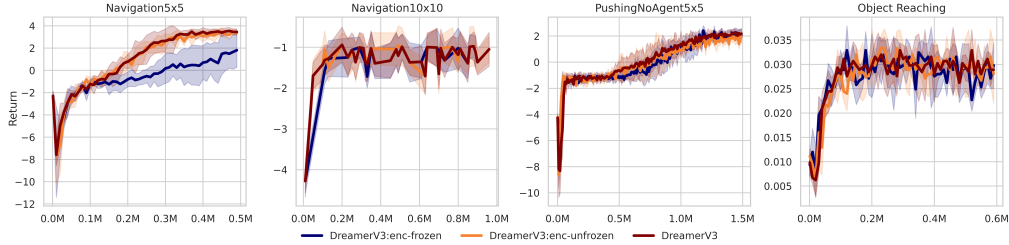


Figure 2: The plots illustrate the impact of encoder pretraining for DreamerV3 algorithm. *DreamerV3* is a default version that trains its encoder from scratch. *DreamerV3:enc-frozen* is a version with a pretrained frozen encoder. *DreamerV3:enc-unfrozen* is a version with a pretrained unfrozen encoder. Return and success rate averaged over 30 episodes and three seeds for different.

## F EVALUATION OF OUT-OF-DISTRIBUTION GENERALIZATION TO UNSEEN COLORS

We evaluated the generalization of the ROCA to unseen colors of distractor objects in the Object Reaching task. When we tested the model with the same colors it was trained on, it achieved a success rate of  $0.975 \pm 0.005$ . However, when we used new colors for the distractor objects, the success rate dropped to  $0.850 \pm 0.005$ . The results were averaged over three instances of the ROCA models, and each model was evaluated on 100 episodes.

## REFERENCES

- Petros Christodoulou. Soft actor-critic for discrete action settings, 2019.
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.
- Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019.

---

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention, 2020.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021.

Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate dall-e learns to compose. In *ICLR*, 2022.