A Implementation Details

A.1 Distance function

As mentioned in Section 3, we wish to learn an encoding such that our distance metric d is the euclidean distance between the encoded states.

$$d(s, s') = ||h(s) - h(s')||^2 \tag{5}$$

To learn the encoder h, we optimize a contrastive loss on encodings of the current and future states along the same trajectory. We use the InfoNCE Loss [30],

$$\mathcal{L}_q = \log \frac{\exp(q^T W k)}{\exp\left(\sum_{i=0}^{K} \exp(q^T W k_i)\right)} \tag{6}$$

with query $q = h(s_t^i)$ as the encoded starting state, and the keys $k = h(s_{t+H}^i)$ as the encoded future states along the K trajectories in the dataset $\mathcal{D}$.

A.2 Training

The training for both skill extraction and fine-tuning were done on a single NVIDIA 2080Ti GPU. Skill extraction takes approximately 3-4 hours, and fine-tuning requires less than 10 minutes. We build upon the SPiRL codebase at https://github.com/clvrai/spirl. Hyperparameters used are listed in Table 6.

Table 6: Training Hyperparameters

Hyperparameter	**Value**
Contrastive Distance Metric	
Encoder output dim	32
Encoder Hidden Layers	128
Encoder # Hidden Layers	2
Optimizer	Adam($\beta_1 = 0.9$, $\beta_2 = 0.999$, LR=1e-3)
Skill extraction	
Epochs	200
Batch size	128
Optimizer	Adam($\beta_1 = 0.9$, $\beta_2 = 0.999$, LR=1e-3)
H (sub-trajectory length)	10
β	5e-4 (Kitchen), 1e-2 (Maze)
Skill Encoder	
dim-$\mathcal{Z}$ in VAE	128
hidden dim	128
# LSTM Layers	1
Skill Decoder	
hidden dim	128
# hidden layers	5
Inverse Skill Dynamic Model	
hidden dim	128
# hidden layers	5
Fine-tuning	
Epochs	50
Batch size	128
Optimizer	Adam($\beta_1 = 0.9$, $\beta_2 = 0.999$, LR=1e-3)

A.3 Datasets

The PointMaze and Kitchen environment datasets (both skill extraction datasets and few-shot learning datasets) are generated from an expert policy. For the AntMaze environment, the dataset was created from the D4RL dataset [29], licensed under the Creative Commons Attribution 4.0 License (CC BY). Datasets for each blocked section was created by filtering out any trajectories that passed through the blocked regions shown in Figure 3. Code for the dataset generation is included in the supplementary material.