APPENDIX

A ACTIVE EXPLORATION WITH INTRINSIC MOTIVATION

Intuitively, our active exploration process in ALP needs to explore the environment to gather informative and diverse observations for both representation learning and downstream tasks. In our experiments, we adopt two existing methods that measure novelty of state visitations as intrinsic rewards and use reinforcement learning (RL) algorithm to train our agent.

Novelty-based Reward. To efficiently compute intrinsic reward from high dimensional pixel space, we measure novelty in lower dimensional feature space from visual representation as an empirical estimate. We use a visual encoder f to extract features from RGB observations and use them to compute reward.

Random Network Distillation (RND) Burda et al. (2018) contains a randomly initialized fixed target network $\overline{g}$ and optimizes a trainable prediction network g trained on data collected by the agent to minimize MSE regression error. Intuitively, function approximation error is expected to be higher on novel unseen states and thus encourages the agent to gather diverse observations. Given a visual encoder f, its novelty-based reward on observation at timestep t is computed by $\mathbf{r}(o_t) = \|g(f(o_t)) - \overline{g}(f(o_t))\|^2$.

Curious Representation Learning (CRL) Du et al. (2021) designs a reward function specifically beneficial for contrastive loss on visual inputs. Given a family of data augmentations $\mathcal{T}$, its novel-based reward is computed by $\mathbf{r}(o_t) = 1 - \text{sim}\left(g(f(\tilde{o}_t^1)), g(f(\tilde{o}_t^2))\right)$, where $\left(\tilde{o}_t^1, \tilde{o}_t^2\right)$ are two transformed pairs of o_t sampled from transformations $\mathcal{T}$ and $\text{sim}(u, v)$ is the dot product similarity metric. Visual encoder f and projection network g is trained on observations visited by the agent to minimize popular InfoNCE loss Chen et al. (2020a) with data augmentations. Following previous works Du et al. (2021); Chen et al. (2020a), we define $\mathcal{T}$ to consist of horizontal flips, random resized crops, and color saturation using their default hyperparameters.

Learning Policy. In practice, we use DD-PPO Wijmans et al. (2019), a distributed version of PPO Schulman et al. (2017) to better scale training in high dimensional complex visual environments. Following previous works Burda et al. (2019), we normalize rewards by the standard deviation of past observed rewards to ensure that reward magnitudes are relatively stable. We train agents with 20 parallel threads, where each environment thread is randomly sampled from training split scenes as specified in Table B.2. The agent is initialized from a randomly sampled location with a random rotation at the beginning of each training episode.

B EXPERIMENTAL DETAILS

B.1 ALGORITHM DETAILS

We present pseudocode describing the process of our ALP framework in Algorithm 1.

B.2 ENVIRONMENT DETAILS

We provide the list of scenes in train split and test split from Gibson Xia et al. (2018) dataset as follows:

Train Split					Test Split
Allensville	Forkland	Leonardo	Newfields	Shelbyville	Collierville
Beechwood	Hanson	Lindenwood	Onagac	Stockmanc	Corozal
Benevolence	Hiteman	Marstons	Pinesdale	Tolstoy	Darden
Coffeen	Klickitat	Merom	Pomaria	Wainscott	Markleeville
Cosmos	Lakeville	Mifflinburg	Ranchester	Woodbine	Wiconisco

Algorithm 1 ALP

Input: Environment E, Online rollout buffer $\mathcal{B}$, Data buffer $\mathcal{D}$, Intrinsic reward $\mathbf{r}(o)$, Policy π_θ with its visual encoder $M_{\texttt{repr}}$ as representation learning model
Output: Representation learning model $M_{\texttt{repr}}$, Downstream training samples $\mathcal{D}$
Initialize $\mathcal{B}, \mathcal{D} \leftarrow \emptyset, \emptyset$
while not converged **do**
  // Collect observations from environments using policy π_θ
  for each timestep t **do**
    $o_t = \text{get\_obs}(E), a_t = \pi_\theta(o_t), o_{t+1} = \text{step}(E, o_t, a_t)$
    // Relabel transitions with intrinsic rewards $\mathbf{r}(o_t)$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{\tau_t = (o_t, a_t, o_{t+1}, \mathbf{r}(o_t))\}$
  end for
  // Update representation learning model M_{repr} *and exploration policy* π_θ
  Update $M_{\texttt{repr}}$ with $\mathcal{L}_{\text{IDM}}(M_{\texttt{repr}}, \mathcal{B})$
  Update π_θ with $\mathcal{L}_{\text{PPO}}(\pi_\theta, \mathcal{B})$
  Update reward network $\mathbf{r}(o)$ with $\mathcal{L}_{\text{reward}}(\mathcal{B})$ (we optimize either RND loss or CRL loss in our experiments)
  // Collect training samples for downstream tasks
  if record labeled data **then**
    Randomly sample a small subset from current rollout buffer $\{(x_{\text{image}}, y_{\text{label}})\} \sim \mathcal{B}$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_{\text{image}}, y_{\text{label}})\}$
  end if
  Empty online rollout buffer $\mathcal{B} \leftarrow \emptyset$
end while

B.3 IMPLEMENTATION DETAILS

To train exploration policy, we use the open-source implementation of DD-PPO baseline Wijmans et al. (2019) from the Habitat simulator `https://github.com/facebookresearch/habitat-lab`. To train downstream perception models, we use the open-source implementation from Detectron2 library `https://github.com/facebookresearch/detectron2` with its default architectures and supervised learning objectives. We provide implementation details of each method and baseline as follows and a full list of hyperparameters can be found in Table 14:

Implementation details for ALP:

- **Model architectures.** In exploration policy, following previous works Wijmans et al. (2019), we use a simplified version of agent architecture without the navigation goal encoder. For RGB observations we use a first layer of 2×2-AvgPool to reduce resolution, essentially performing low-pass filtering and down-sampling, before passing into the visual encoder $M_{\texttt{repr}}$ with output dimension of 2048. This visual feature is concatenated with an embedding of the previous action taken and is then passed into LSTM. The output of LSTM is used as input to a fully connected layer, resulting in a soft-max distribution of the action space as policy and an estimate of the value function.

  In IDM, our projection head $h_{\texttt{proj}}$ is a 2-layer MLP network with an input dimension of 2048, a hidden dimension of 512, and an output dimension of 512; our prediction network $h_{\texttt{IDM}}$ is a MLP with an input dimension of $512 \times (\text{num-steps} + 1)$, 2 hidden layers with 512 units, and an output dimension of $3 \times$ num-steps. Then the output layer is chunked by every 3 units, same as dimension of our action space, as predicted logits of IDM.

  We adopt commonly used architectures for each perception task: for object detection and instance segmentation, we use a Mask-RCNN He et al. (2017) using Feature Pyramid Networks Lin et al. (2017) with a ResNet-50 He et al. (2016) as $M_{\texttt{repr}}$; for depth estimation, we use a Vit-B/16 vision transformer encoder-decoder architecture Dosovitskiy et al. (2020) with its encoder as $M_{\texttt{repr}}$.

- **Pre-training details.** To train exploration policy, we use Generalized Advantage Estimation (GAE) Schulman et al. (2015), a discount factor of $\gamma = 0.99$, and a GAE parameter of $\lambda = 0.95$. Each individual episode has a maximum length of 512 steps. Each parallel worker collects 64 frames of rollouts and then performs 4 epochs of PPO with 2 mini-batches per epoch. We use Adam optimizer Kingma & Ba (2014) with a learning rate of 2.5×10^{-4}. Following previous work Wijmans et al. (2019), we do not normalize advantages as in baseline implementations.

  To train IDM, we use Adam optimizer Kingma & Ba (2014) with a learning rate of 2.5×10^{-4}. Given 64 frames of rollouts, we perform 4 epochs of gradient updates by using all frames to compute the average prediction loss at each iteration.

- **Fine-tuning details.** To sample a small subset of annotated images from all explored trajectories, we randomly sample 100 annotated images per scene from online batches of rollouts 100 times equally spreaded throughout policy training. We then save them as a fixed static dataset to further train downstream perception models.

  For Mask-RCNN, we use batch size of 32, learning rate of 0.02, resolution of 256 for supervised learning until convergence, roughly around $120k$ training steps and pick the model checkpoint with better performance. For ViT, we use a transformer decoder with 8 attention heads and 6 decoder layers, and train for 10 epochs using the AdamW optimizer with a learning rate of 0.0001 and cosine annealing.

Implementation details for visual representation learning baselines:

- **CRL.** Our CRL Du et al. (2021) baseline as visual representation learning method are based on open source implementation available at `https://github.com/yilundu/crl`. For Matterport3D CRL baseline, we use its publicly released pretrained weights in ResNet-50 architecture. For Gibson CRL baseline, we use the same set of hyperparameters as reported in its original setups and pretrain in our environments for 8M frames. We found that given the fixed downstream dataset, pretrained representation from our reproduced experiments in Gibson generally achieve better performance, possibly due to more in-distribution

training data. We thus report all experimental results using our pretrained weights from Gibson environments.

- **ImageNet SimCLR.** For Mask-RCNN results, we tried both SimCLRv1 Chen et al. (2020a) and SimCLRv2 Chen et al. (2020b) checkpoints from https://github.com/google-research/simclr with ResNet-50 architecture same as ours and found that SimCLRv2 achieves better performance. We thus report all experiment results from SimCLRv2 pretrained weights. For transformers, we pretrain on ImageNet ILSVRC-2012 Russakovsky et al. (2015) with a batch size of 256 for 30 epochs. We use the Adam optimizer Kingma & Ba (2014) with a learning rate of 0.0003 and cosine annealing Loshchilov & Hutter (2016).
- **ImageNet Supervised.** We use ResNet-50 and ViT-B/16 weights pretrained for ImageNet classification task available at https://dl.fbaipublicfiles.com/detectron2/ImageNetPretrained/MSRA/R-50.pkl and https://download.pytorch.org/models/vit_b_16-c867db91.pth, respectively.
- **Architectures in self-supervised contrastive learning baselines.** For self-supervised contrastive learning baselines compared to ALP, we train a separate network with same architecture as visual backbone in ALP for baseline comparison. For SimCLR, we use a 2-layer MLP projection head with hidden dimensions of 256 and output dimensions of 128 and temperature hyperparameter $\tau = 0.07$ to compute contrastive loss. For CPC, we use a MLP projection head and a forward prediction MLP both with hidden dimension of 256 and output dimension of 128 to compute contrastive loss. We use same learning rate as ALP framework 2.5×10^{-4} to optimize contrastive loss.

Implementation details for downstream data collection baselines:

- **RND.** Our RND Burda et al. (2018) baseline is based on open sourced implementation available at https://github.com/rll-research/url_benchmark and extends to pixel input. We train separate networks for exploration policy and reward model, both with ResNet-50 architecture. For both RND-ALP and RND, we use a 2-layer MLP projection head with hidden dimensions of 512 and output dimensions of 64 to compute reward and to minimize MSE loss. We use the same sampling scheme as in ALP to randomly sample a small subset of annotated images per scene from its explored trajectories as downstream task dataset.
- **CRL.** CRL Du et al. (2021) could also be interpreted as an exploration strategy in addition to an active visual representation learning approach. As proposed in its original setups, we train separate networks for exploration policy and representation learning model, both with ResNet-50 architecture. For both CRL-ALP and CRL, we use a 2-layer MLP projection head with hidden dimensions of 128 and output dimensions of 128 and temperature hyperparameter $\tau = 0.07$ to compute reward and to minimize contrastive loss. We use the same sampling scheme as in ALP to randomly sample a small subset of annotated images per scene from its explored trajectories as downstream task dataset.
- **ANS.** ANS Chaplot et al. (2020a) is a hierarchical modular policy for coverage-based exploration. It builds a spatial top-down map and learns a higher-level global policy to select waypoints in the top-down map space to maximize area coverage. Note that ANS performs depth-based occupancy mapping and assumes sensor pose readings, instead of using only RGB as in our case. Our ANS baseline is based on open sourced implementation available at https://github.com/devendrachaplot/Neural-SLAM. For better adaptations, we finetune its pretrained ANS policy in our train split scenes as specified in Table B.2 using its original hyperparameters. We randomly sample a small subset of annotated images from its explored trajectories as downstream task dataset. We generally found that finetuning improves downstream performance of self-supervised visual representations (RND-ALP and ImagNet SimCLR), except that for ImageNet supervised pretrained weights, using pretrained policy achieves better performance ($+5.91$ in Test Split).

C Additional Experimental Results

C.1 Quantitative Results

Compare to self-supervised contrastive learning methods. We provide additional experimental results comparing CRL-ALP with self-supervised contrastive learning methods on the same unlabeled pretraining and labeled downstream dataset. Note that CRL exploration policy is trained to maximize reward inverse proportional to contrastive loss and thus able to find images that are particularly useful for SimCLR. Table 7 however shows that CRL-ALP improves performance from baseline by only learning from action information.

	Train Split		Test Split	
Method	ObjDet	InstSeg	ObjDet	InstSeg
SimCLR (FrScr)	77.90	75.06	40.65	36.44
CRL-ALP (ours)	**83.14**	**79.24**	**42.42**	**36.89**

Table 7: **Results of comparing to self-supervised contrastive learning methods.** CRL-ALP outperforms self-supervised learning baselines under same pretraining and downstream dataset, without including *any* form of contrastive loss.

Compare visual representation learning qualities. We provide additional experimental results comparing our method to visual representation learning baselines. In Table 8 , perception models initialized from all visual representations are trained on the same downstream dataset collected from random policy or from Active Neural SLAM policy. We also include performance of CRL-ALP in addition to RND-ALP.

While ImageNet pretrained model generally performs the best compared to simulator pretraining, ALP still performs better than other simulator-pretrained visual representations. This is significant since we do not use any form of contrastive loss to learn visual representations but only consider different forms of supervisions from active movements.

		Train Split		Test Split	
Downstream Data	Pretrained Representation	ObjDet	InstSeg	ObjDet	InstSeg
ANS	From Scratch	73.65	70.87	28.10	22.99
	CRL (MP3D)	77.30	73.81	40.78	36.12
	CRL (Gibson)	76.60	73.10	50.72	43.79
	ImageNet SimCLR	**77.47**	**74.86**	47.92	41.42
	RND-ALP (ours)	75.59	71.37	47.95	41.09
	CRL-ALP (ours)	76.98	73.05	**51.56**	**46.43**
	ImageNet Supervised	80.08	76.14	43.79	36.31
Random Policy	From Scratch	56.22	53.31	29.67	27.28
	CRL (MP3D)	58.30	53.52	33.82	30.90
	CRL (Gibson)	60.52	56.39	37.45	32.12
	ImageNet SimCLR	**64.94**	**61.26**	**43.29**	**37.84**
	RND-ALP (ours)	60.36	55.74	38.46	34.16
	CRL-ALP (ours)	62.16	57.20	40.16	33.14
	ImageNet Supervised	65.58	62.66	43.59	39.80
RND-ALP (ours)	RND-ALP (ours)	87.95	84.56	50.34	46.74
	CRL-ALP (ours)	88.73	84.78	49.32	43.06

Table 8: **Results of comparing visual representation baselines.** Performance of finetuning RND-ALP, CRL-ALP, and all pretrained visual representation baselines on *the same downstream dataset*.

Compare downstream data collection qualities. We provide additional experimental results comparing our method with various downstream data collection baselines. In particular, we finetune ImageNet SimCLR or supervised pretrained models and on different downstream data collection baselines. Table 9 shows that RND-ALP collects better downstream data for perception models initialized with different pretrained weights.

Since ANS uses a hierarchical modular architecture and trains a global and a local policy, this is different from training a single policy architecture end-to-end in learning-based exploration methods. We generally found that there are fewer object masks corresponding to samples collected from ANS ($257k$) compared to RND ($284k$) or CRL ($302k$), among $250k$ image-label pairs collected from each exploration policy.

		Train Split		Test Split	
Pretrained Representation	Downstream Data	ObjDet	InstSeg	ObjDet	InstSeg
ImageNet SimCLR	RND	85.55	82.17	42.89	40.86
	CRL	82.51	79.05	42.49	38.93
	ANS	77.47	74.86	47.92	41.42
	RND-ALP (ours)	**87.22**	**83.50**	**50.66**	**46.55**
ImageNet supervised	RND	87.15	83.34	48.13	42.20
	CRL	84.28	80.79	46.95	41.75
	ANS	80.08	76.14	43.79	36.31
	RND-ALP (ours)	**88.75**	**85.51**	**52.78**	**46.94**

Table 9: **Results of comparing downstream data collection baselines.** Performance of finetuning *the same pretrained representations* from ImageNet SimCLR (top) and ImageNet supervised (bottom) on each data collection method.

Number of timesteps in inverse dynamics prediction. We provide additional experimental results by varying number of timesteps in IDM when training with CRL-ALP and observe its effects on downstream performance in Table 10.

	Train Split			Test Split		
Number of Steps	ObjDet	InstSeg	DepEst	ObjDet	InstSeg	DepEst
4	80.63	77.30	**0.259**	**43.97**	**37.09**	0.352
8	**83.14**	**79.24**	0.261	42.42	36.89	**0.350**

Table 10: **Number of timesteps in inverse dynamics model (IDM).** We vary lengths of input observation sequence and predicted action sequence when training inverse dynamics model to observe its effect on downstream performance of CRL-ALP.

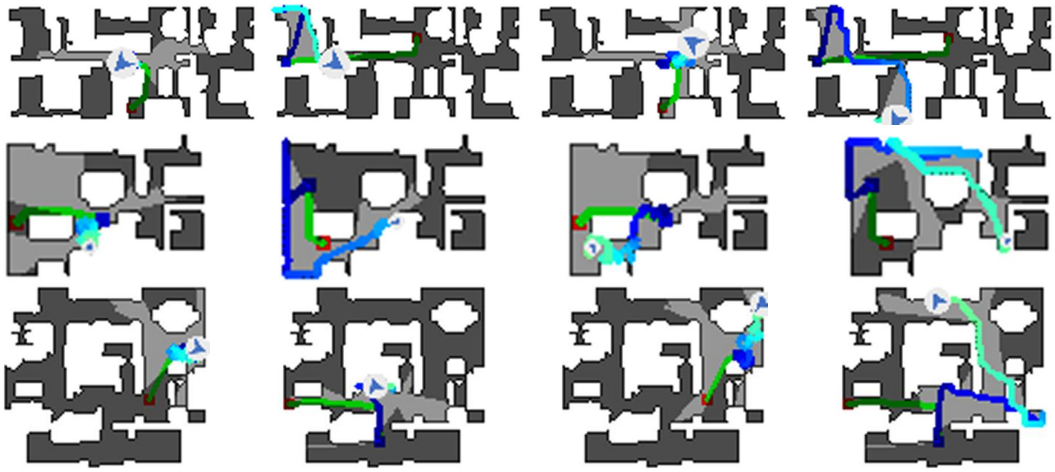
Variance across multiple runs. We try to understand variance across multiple runs given *the same* pretrained representation and *the same* downstream data. Specifically, we initialize Mask-RCNN with the pretrained representation and finetune on the labeled dataset both from RND-ALP. Results over 3 runs in Table 11 shows consistent evaluation performance.

C.2 Qualitative Examples

In Figure 3, we provide additional visualizations of episode trajectories; each line represents policy rollout trajectories in the same house of Habitat environment. We observe that learned exploration policy learned from RND and CRL baseline shows wider coverage of maps and longer movements when combined with ALP, visually demonstrating better exploration policy from ALP framework. We include corresponding videos in the supplementary material.

Train Split		Test Split	
ObjDet	InstSeg	ObjDet	InstSeg
87.95	84.56	50.34	46.74
87.30	83.05	49.35	47.65
88.19	84.06	50.79	46.36
87.81 ± 0.46	83.89 ± 0.77	50.16 ± 0.74	46.92 ± 0.67

Table 11: **Variance across multiple runs.** We report downstream model performance with their mean and standard deviation over 3 runs, given the *same* pretrained representation and the *same* downstream dataset from RND-ALP.



(a) CRL (b) RND (c) CRL-ALP (d) RND-ALP

Figure 3: Episode trajectories of policies trained with exploration baselines and combined with our method on a Habitat environment. Both RND and CRL shows wider coverage of maps and longer movements when combined with ALP.

D Contrastive Learning as part of Training Objective

We also investigated including contrastive loss as self-supervised visual representation learning objective and provide more details below.

D.1 Method

In principle, we could use any representation learning objective suitable for learning visual representations in a self-supervised manner. In particular, we utilize CPC Oord et al. (2018) when combining with RND exploration method, one of popular contrastive learning approaches Chen et al. (2020a); Stooke et al. (2021), since observations from temporally closer frames should be closer in visual representation space than further frames or observations from different environments. We utilize SimCLR Chen et al. (2020a) with data augmentations when combined with CRL, since this is easier to integrate with its original framework that already introduces a self-supervised loss.

Given our representation learning model $M_{\texttt{repr}}$ and projection head $h_{\texttt{con}}$, we encode each observation frame o into its latent representation $z = h_{\texttt{con}}(M_{\texttt{repr}}(o))$, and minimize InfoNCE loss as training objective:

$$\mathcal{L}_{\text{contrast}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp\left(\text{sim}(z_i, z_{i+})\right)}{\sum_{j=1}^{N}\exp\left(\text{sim}(z_i, z_{j+})\right)}$$

where $\text{sim}(u, v) = v^T u$ is dot product similarity metric between feature vectors. For CPC, positive samples come from temporally close frames within 4 steps and negative samples come from observations in other paralleled training environments. For SimCLR, positive samples come from two augmented pairs of the same image and negative samples come from augmented views from different images.

We modify a few design choices when including contrastive loss in order to improve compute and memory efficiency. To train inverse dynamics model, we optimize cross-entropy loss based on predicted action between two consecutive frames, i.e. $h_{\texttt{IDM}}(a_t|o_t, o_{t+1})$, instead of training on sequence of observations and actions. The visual encoder f to compute intrinsic rewards is a momentum encoder He et al. (2020) that parameterizes a slow moving average of weights from our representation learning model $M_{\texttt{repr}}$. We leave more complicated architectures as future investigations.

	Train Split			Test Split		
Method	ObjDet	InstSeg	DepEst	ObjDet	InstSeg	DepEst
CRL	79.61	76.42	0.265	40.81	34.24	**0.352**
CRL-ALP (ours)	**80.84**	**77.44**	**0.262**	**44.23**	**37.60**	0.354
RND	83.65	81.01	0.251	37.13	34.10	0.366
RND-ALP (ours)	**87.78**	**84.12**	**0.239**	**48.63**	**45.80**	**0.308**

Table 12: Results of combining ALP (including contrastive loss) with different exploration methods. We combine our method with two exploration strategies, RND and CRL, and compare performance of downstream perception model with baselines.

D.2 Experimental Results

We perform similar sets of experiments on ALP framework including contrastive loss. We show that including contrastive loss as self-supervised training objective achieves similarly good performance. In Table 12, we report results when combining our framework with two learning-based exploration methods, RND and CRL. We observe much improvements in downstream performance compared to baselines, and are consistent under different exploration strategies. In Table 13, we report full results comparing our framework RND-ALP with visual representation baselines and data collection baselines respectively. RND-ALP outperforms other baselines in downstream perception tasks; it performs similar or even better than ImageNet supervised pretraining without access to any supervisions from semantic labels. Thus this indicates significant benefits of learning signals from active environment interactions in learning better visual representations.

Pretrained	Downstream	Train Split			Test Split		
Representation	Data	ObjDet	InstSeg	DepEst	ObjDet	InstSeg	DepEst
From Scratch	RND-ALP	83.50	79.65	0.327	36.05	30.57	0.524
CRL		85.06	81.54	0.263	45.95	42.53	0.351
ImgNet SimCLR		86.38	83.40	0.257	45.26	37.07	0.349
ImgNet Sup		86.99	**84.28**	0.244	48.53	43.40	0.340
RND-ALP		**87.78**	84.12	**0.239**	**48.63**	**45.80**	**0.308**
RND-ALP	RND	86.36	82.03	0.245	42.37	36.10	0.342
	CRL	83.12	79.43	0.274	40.76	34.43	0.365
	ANS	75.35	71.73	-	48.43	42.62	-
	RND-ALP	**87.78**	**84.12**	**0.239**	**48.63**	**45.80**	**0.308**

Table 13: Results of comparing pretrained visual representations and downstream data collection baselines. We fix either visual representation or labeled dataset from RND-ALP (ours including contrastive loss) and compare downstream performance to all baselines respectively.

Hyperparameter	Value
Observation	(256, 256), RGB
Downsample layer	AveragePooling (2, 2)
Hidden size (LSTM)	512
Optimizer	Adam
Learning rate of π_θ	2.5×10^{-4}
Learning rate annealing	Linear
Rollout buffer length	64
PPO epochs	4
PPO mini-batches	2
Discount γ	0.99
GAE λ	0.95
Normalize advantage	False
Entropy coefficient	0.01
Value loss term coefficient	0.5
Maximum norm of gradient	0.5
Clipping ϵ	0.1 with linear annealing
Learning rate of reward network	1×10^{-4}
Optimizer	Adam
Learning rate of IDM	2.5×10^{-4}
Optimizer	Adam
Number of timesteps	8
IDM epochs	4
Projection network	[512]
Prediction network	[512, 512]

Table 14: Hyperparameters for training exploration policy and representation learning model. We follow previous works Wijmans et al. (2019); Du et al. (2021) as close as possible.