[Appendix]
Unsupervised Skill Discovery with Bottleneck Option Learning

Jaekyeom Kim [*1] **Seohong Park** [*1] **Gunhee Kim** [1]

A. Additional Qualitative Results

As a complement to Section 4.1 and Figure 3 from the main paper, we present videos that demonstrate various skills learned by IBOL at `https://vision.snu.ac.kr/projects/ibol`.

B. Derivation of the Lower Bound

We describe the derivation of Equation (4) from the main paper. Starting with Equation (3) from the main paper,

$$\begin{aligned}&\mathbb{E}_t[I(Z;G_t|S_t) - \beta \cdot I(Z;S_{0:T})]\\ &= \mathbb{E}_{\substack{\tau\sim p_{\theta_s}(\tau),t,\\ z\sim p_\phi(z|s_{0:T})}}\left[\log \frac{p_{\theta_s}(g_t|s_t,z)}{\pi_{\theta_s}(g_t|s_t)} - \beta \log \frac{p_\phi(z|s_{0:T})}{p_\phi(z)}\right].\end{aligned}$$

For the first term, as described in the main paper, we use the skill policy's output distribution $\pi_{\theta_z}(g_t|s_t,z)$ as a variational approximation of $p_{\theta_s}(g_t|s_t,z)$, which derives

$$\begin{aligned}&\mathbb{E}_{\substack{\tau\sim p_{\theta_s}(\tau),t,\\ z\sim p_\phi(z|s_{0:T})}}\left[\log \frac{p_{\theta_s}(g_t|s_t,z)}{\pi_{\theta_s}(g_t|s_t)}\right]\\ &= \mathbb{E}_{\substack{\tau\sim p_{\theta_s}(\tau),t,\\ z\sim p_\phi(z|s_{0:T})}}\left[\log \frac{\pi_{\theta_z}(g_t|s_t,z)}{\pi_{\theta_s}(g_t|s_t)}\right]\\ &\quad + \mathbb{E}_{\substack{s_{0:T}\sim p_{\theta_s}(\cdot),t,\\ z\sim p_\phi(z|s_{0:T})}}\left[D_{\mathrm{KL}}(p_{\theta_s}(G_t|s_t,z)\|\pi_{\theta_z}(G_t|s_t,z))\right]\\ &\geq \mathbb{E}_{\substack{\tau\sim p_{\theta_s}(\tau),t,\\ z\sim p_\phi(z|s_{0:T})}}\left[\log \frac{\pi_{\theta_z}(g_t|s_t,z)}{\pi_{\theta_s}(g_t|s_t)}\right]. \qquad (1)\end{aligned}$$

Also, for the second term, we use $r(z)$ as the variational approximation of the marginal distribution $p_\phi(z)$, and it derives

$$\begin{aligned}&\mathbb{E}_{\substack{\tau\sim p_{\theta_s}(\tau),\\ z\sim p_\phi(z|s_{0:T})}}\left[\log \frac{p_\phi(z|s_{0:T})}{p_\phi(z)}\right]\\ &= \mathbb{E}_{\substack{\tau\sim p_{\theta_s}(\tau),\\ z\sim p_\phi(z|s_{0:T})}}\left[\log \frac{p_\phi(z|s_{0:T})}{r(z)}\right] - D_{\mathrm{KL}}(p_\phi(Z)\|r(Z))\\ &\leq \mathbb{E}_{\substack{\tau\sim p_{\theta_s}(\tau),\\ z\sim p_\phi(z|s_{0:T})}}\left[\log \frac{p_\phi(z|s_{0:T})}{r(z)}\right]\\ &= \mathbb{E}_{\tau\sim p_{\theta_s}(\tau)}\Big[D_{\mathrm{KL}}(p_\phi(Z|s_{0:T})\|r(Z))\Big]. \qquad (2)\end{aligned}$$

Combining the derivation of Equations (1) and (2) obtains Equation (4) from the main paper.

C. Encouraging Disentanglement

Disentanglement learning methods often model the generation process as $X \sim p(\cdot|Z)$, assuming that data X is generated with its underlying latent factors Z (Kim & Mnih, 2018; Do & Tran, 2020). While the aggregated posterior of Z is defined as $q(Z) = \int_x q(Z|x)p(x)dx$ for an encoder $q(Z|X)$ (Makhzani et al., 2015), one of common disentanglement approaches is to penalize the total correlation of $q(Z)$, expressed as $TC(q(Z)) = D_{\mathrm{KL}}(q(Z)\| \prod_i^d q(Z_i))$.

The IB framework has theoretical connections to the disentanglement of representation (Achille & Soatto, 2018b;a; Chen et al., 2018). In Section 3.2, we derived our objective in the form of IB. Especially, if we model the prior of Z as $r(Z) = \prod_i^d r(Z_i)$, the term $D_{\mathrm{KL}}(p_\phi(Z|s_{0:T})\|r(Z))$ in Equation (4) from the main paper is decomposed into three terms revealing the total correlation term $TC(p_\phi(Z)) = D_{\mathrm{KL}}(p_\phi(Z)\| \prod_i^d p_\phi(Z_i))$ (Chen et al., 2018) (refer to Appendix D for the proof), where the aggregated posterior of the trajectory encoder is $p_\phi(Z) = \int_{s_{0:T}} p_\phi(Z|s_{0:T})p(s_{0:T})ds_{0:T}$. By penalizing $TC(p_\phi(Z))$, we encourage each dimension of the skill latent space $\mathcal{Z}$ disentangled from the others with respect to $S_{0:T}$. As a result, the learned skill latent Z can provide improved abstraction, where each dimension focuses more on only its corresponding behavior of the discovered skills.

[*]Equal contribution [1]Department of Computer Science and Engineering, Seoul National University, South Korea. Correspondence to: Gunhee Kim <gunhee@snu.ac.kr>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

D. Decomposition of the KL Divergence Term

When the prior of Z is modeled as a factorized distribution, *i.e.* $r(Z) = \prod_i^d r(Z_i)$, the KL divergence term in Equation (4) from the main paper can be decomposed as follows (Chen et al., 2018):

$$\begin{aligned}
&D_{\mathrm{KL}}(p_\phi(Z|s_{0:T})\|r(Z)) && (3)\\
&= \mathbb{E}_{z\sim p_\phi(Z|s_{0:T})}\left[\log\frac{p_\phi(z|s_{0:T})}{r(z)}\right]\\
&= \mathbb{E}_{z\sim p_\phi(Z|s_{0:T})}\left[\log\frac{p_\phi(z|s_{0:T})}{p_\phi(z)}\right]\\
&\quad + \mathbb{E}_{z\sim p_\phi(Z|s_{0:T})}\left[\log\frac{p_\phi(z)}{\prod_{i=1}^d p_\phi(z_i)}\right]\\
&\quad + \mathbb{E}_{z\sim p_\phi(Z|s_{0:T})}\left[\log\frac{\prod_{i=1}^d p_\phi(z_i)}{\prod_{i=1}^d r(z_i)}\right]\\
&= D_{\mathrm{KL}}(p_\phi(Z|s_{0:T})\|p_\phi(Z))\\
&\quad + D_{\mathrm{KL}}(p_\phi(Z)\|\prod_{i=1}^d p_\phi(Z_i))\\
&\quad + \sum_{i=1}^d D_{\mathrm{KL}}(p_\phi(Z_i)\|r(Z_i)), && (4)
\end{aligned}$$

where $p_\phi(Z) = \int_{s_{0:T}} p_\phi(Z|s_{0:T})p(s_{0:T})ds_{0:T}$ denotes the aggregated posterior.

The second term corresponds to the total correlation of Z, as $TC(p_\phi(Z)) = D_{\mathrm{KL}}(p_\phi(Z)\|\prod_i^d p_\phi(Z_i))$, encouraging Z to have a more disentangled representation. The third term operates as a regularizer, which pushes each dimension of the aggregated posterior $p_\phi(Z)$ to be located in the vicinity of the prior. The expectation of the first term can be represented in the form of mutual information, as $\mathbb{E}_{s_{0:T}}[D_{\mathrm{KL}}(p_\phi(Z|s_{0:T})\|p_\phi(Z))] = I(S_{0:T};Z)$. This corresponds to the original compression term before applying the variational approximation.

E. Information-Theoretic Evaluation Metrics

In this section, we describe the information-theoretic metrics used for the evaluation in Section 4.2. In addition to $I(Z;S_T^{(\mathrm{loc})})$, we use the SEPIN@$k$ and WSEPIN metrics for measuring informativeness and separability (Do & Tran, 2020). For SEPIN@k, $I(S_T^{(\mathrm{loc})};Z_i|Z_{\neq i})$ quantifies the amount of information about $S_T^{(\mathrm{loc})}$ contained by Z_i but not $Z_{\neq i}$, and the metric is defined as

$$\text{SEPIN@}k = \frac{1}{k}\sum_{j=1}^k I(S_T^{(\mathrm{loc})};Z_{r_j}|Z_{\neq r_j}), \tag{5}$$

where r_j is the dimension index with the j-th largest value of $I(S_T^{(\mathrm{loc})};Z_i|Z_{\neq i})$ for $i=1,\dots,d$. That is, SEPIN@k is the average of the top k values of $I(S_T^{(\mathrm{loc})};Z_i|Z_{\neq i})$. They also define WSEPIN as

$$\text{WSEPIN} = \sum_{i=1}^d \rho_i\cdot I(S_T^{(\mathrm{loc})};Z_i|Z_{\neq i}) \tag{6}$$

for $\rho_i = \frac{I(S_T^{(\mathrm{loc})};Z_i)}{\sum_{j=1}^d I(S_T^{(\mathrm{loc})};Z_j)}$. It is the sum of $I(S_T^{(\mathrm{loc})};Z_i|Z_{\neq i})$ weighted based on their informativeness, $I(S_T^{(\mathrm{loc})};Z_i)$.

F. Varying Number of Bins for MI Estimation

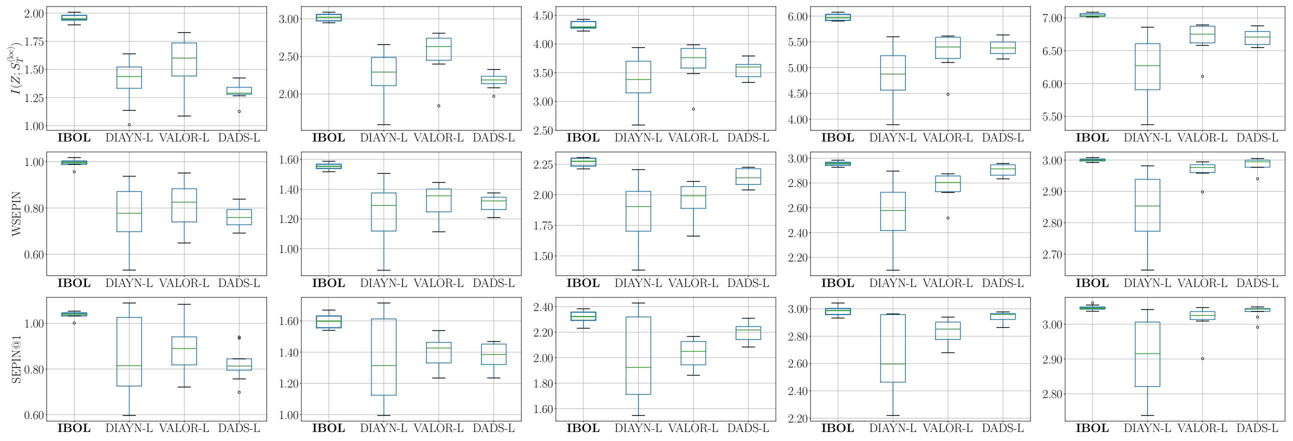
We quantize variables for the estimation of mutual information for measuring the information-theoretic metrics in Section 4.2 from the main paper (see Appendix I.6 for the details). To show that IBOL outperforms the baseline skill discovery methods under different evaluation configurations, we make a more comprehensive comparison between the methods using different numbers of bins for quantization.

Figures 1, 2, 3 and 4 compare the skill discovery methods on Ant, HalfCheetah, Hopper and D'Kitty, varying the number of bins for the mutual information estimation. The results show that on all the four environments, IBOL outperforms DIAYN-L, VALOR-L and DADS-L in the evaluation metrics of $I(Z;S_T^{(\mathrm{loc})})$, WSEPIN and SEPIN@1 regardless of binning, which robustly supports IBOL's improved performance.

G. Diversity of External Returns

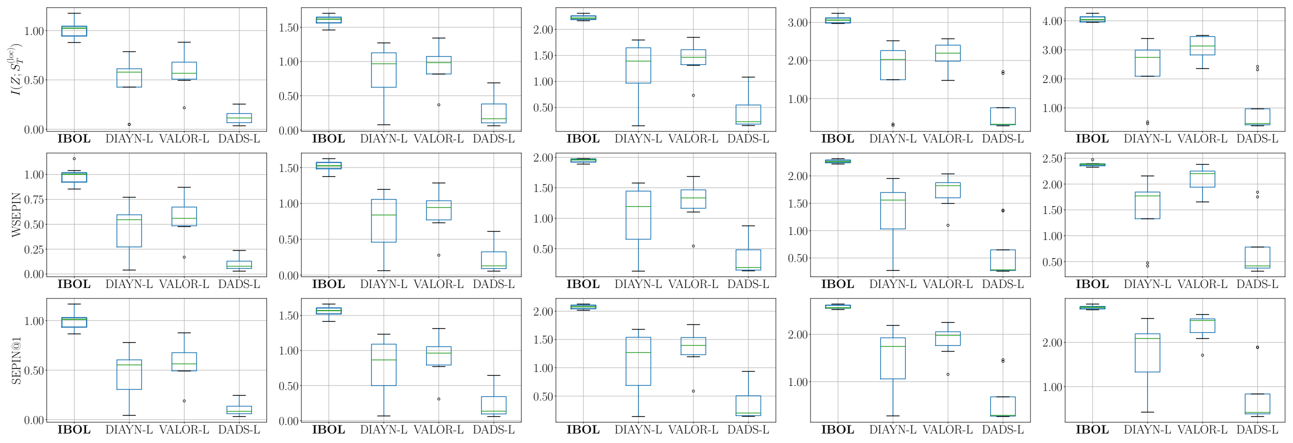
We qualitatively demonstrate the diversity of external returns the methods receive for their skills. As the skill discovery methods learn their skill policies without any external rewards, examining their skills in regard to external returns can be used to evaluate the diversity of the skills as well as their usefulness on the original tasks.

We compare IBOL with DIAYN-L, VALOR-L and DADS-L in Ant, HalfCheetah and Hopper. For every pair of a skill discovery method and an environment, each of the eight skill policies learned by the method with $d=2$ in the environment is used to sample trajectories given 2000 random skill latents from their prior distribution, $p(z)$ *i.e.* the standard normal distribution. Figure 5 visualizes the results, where each vertically stacked histogram denotes the external returns for the 2000 skills with corresponding colors from the color bar for the environment. In the visualizations, the skills learned by IBOL exhibit not only wider but also more diverse ranges of external returns compared to the baseline methods, which suggests that IBOL can acquire a more varied and useful set of skills in the environments.



(a) # bins = 8 (b) # bins = 16 (c) # bins = 32 (d) # bins = 64 (e) # bins = 128

Figure 1. Comparison of IBOL (ours) with the baseline methods, DIAYN-L, VALOR-L and DADS-L, in the evaluation metrics of $I(Z; S_T^{(\mathrm{loc})})$, WSEPIN and SEPIN@1, on Ant, with different bin counts for the range of each variable estimating mutual information. For each method, we use the eight trained skill policies.



(a) # bins = 8 (b) # bins = 16 (c) # bins = 32 (d) # bins = 64 (e) # bins = 128

Figure 2. Comparison of IBOL (ours) with the baseline methods, DIAYN-L, VALOR-L and DADS-L, in the evaluation metrics of $I(Z; S_T^{(\mathrm{loc})})$, WSEPIN and SEPIN@1, on HalfCheetah, with different bin counts for the range of each variable estimating mutual information. For each method, we use the eight trained skill policies.

H. Comparison of Reward Function Choices for the Linearizer

Prior work on hierarchical reinforcement learning. We first review previous works that train low-level policies similarly to ours. SNN4HRL (Florensa et al., 2016) trains a high-level policy on top of a context-conditioned low-level policy, which is pre-trained with a task-related auxiliary reward function that facilitates the desired behaviors as well as exploration. For example, as a reward for its low-level policy in locomotion tasks, it uses the speed of the agent combined with an information-theoretic regularizer that encourages diversity. FuN (Vezhnevets et al., 2017) jointly trains both a high-level policy and a low-level goal-conditioned policy rewarded by the cosine similarity between goals and directions in its latent space. HIRO (Nachum et al., 2018) takes a similar approach to FuN, but its high-level policy generates goals in the raw state space, without having a separate latent goal space. Its low-level policy is guided by the Euclidean distance instead of the cosine similarity. Nachum et al. (2019) train a goal-conditioned low-level policy with the Huber loss, which is a variant of the Euclidean distance, in a learned representation space within the framework of sub-optimality.

In contrast to these approaches, we train the linearizer, which can be viewed as a low-level policy, with the reward in the inner-product form. Also, we reward the linearizer with the state difference between *macro* time steps: $(s_{(i+1)\cdot\ell} - s_{i\cdot\ell})$, where ℓ is the interval of the macro step

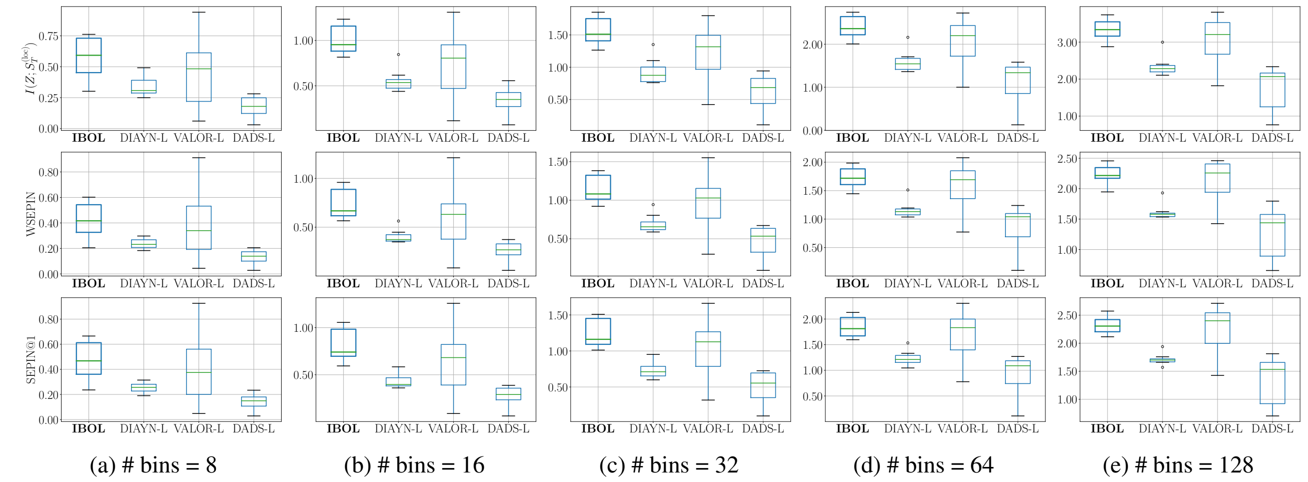


Figure 3. Comparison of IBOL (ours) with the baseline methods, DIAYN-L, VALOR-L and DADS-L, in the evaluation metrics of $I(Z; S_T^{(\text{loc})})$, WSEPIN and SEPIN@1, on Hopper, with different bin counts for the range of each variable estimating mutual information. For each method, we use the eight trained skill policies.

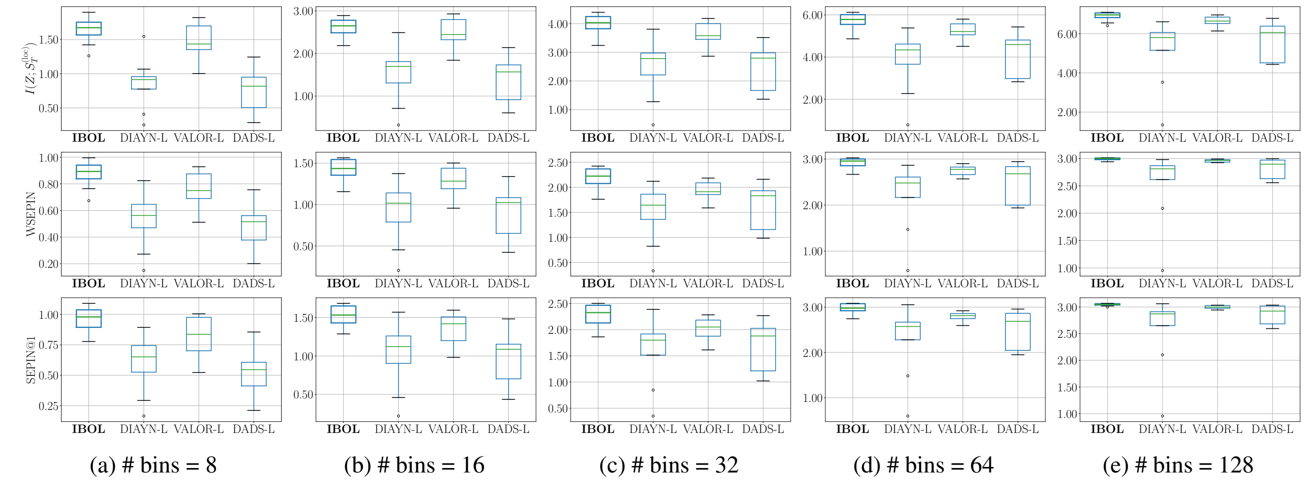


Figure 4. Comparison of IBOL (ours) with the baseline methods, DIAYN-L, VALOR-L and DADS-L, in the evaluation metrics of $I(Z; S_T^{(\text{loc})})$, WSEPIN and SEPIN@1, on D'Kitty, with different bin counts for the range of each variable estimating mutual information. For each method, we use the eight trained skill policies.

(we use $\ell = 10$ in our experiments).

Comparison of different reward function choices. We now compare our reward function for the linearizer with other choices. We experiment on Ant and evaluate them by their state coverage in the x-y plane. We sample 2000 trajectories from each of the linearizers, where we only change the values of x and y dimensions in the goal space and set the other dimensions' value to 0. We measure the state coverage by the number of bins occupied by the trajectories out of 1024 equally divided bins in the x-y plane. For the comparison, we test different values of $\ell = 1, 10, 100$ with our inner-product reward function, as well as one in the form of the Euclidean distance as in HIRO (Nachum et al., 2018) with $\ell = 1, 10$. Since the Euclidean distance reward function requires the specification of the valid goal ranges, we employ the goal range values used by HIRO. As a consequence, we follow the practice of HIRO to exclude the state dimensions for velocities in specifying the goal space for the Euclidean distance reward function. On the contrary, we use the full state dimensions to design the goal space for the inner-product reward function.

Figure 6 compares the performances of the reward function choices. It suggests that using an appropriate size of the macro step (*i.e.* $\ell = 10$) improves the state coverage, especially exhibiting drastic performance improvement over the case of $\ell = 1$. We also observe that our inner-product reward function shows a better state coverage compared to the Euclidean reward function.

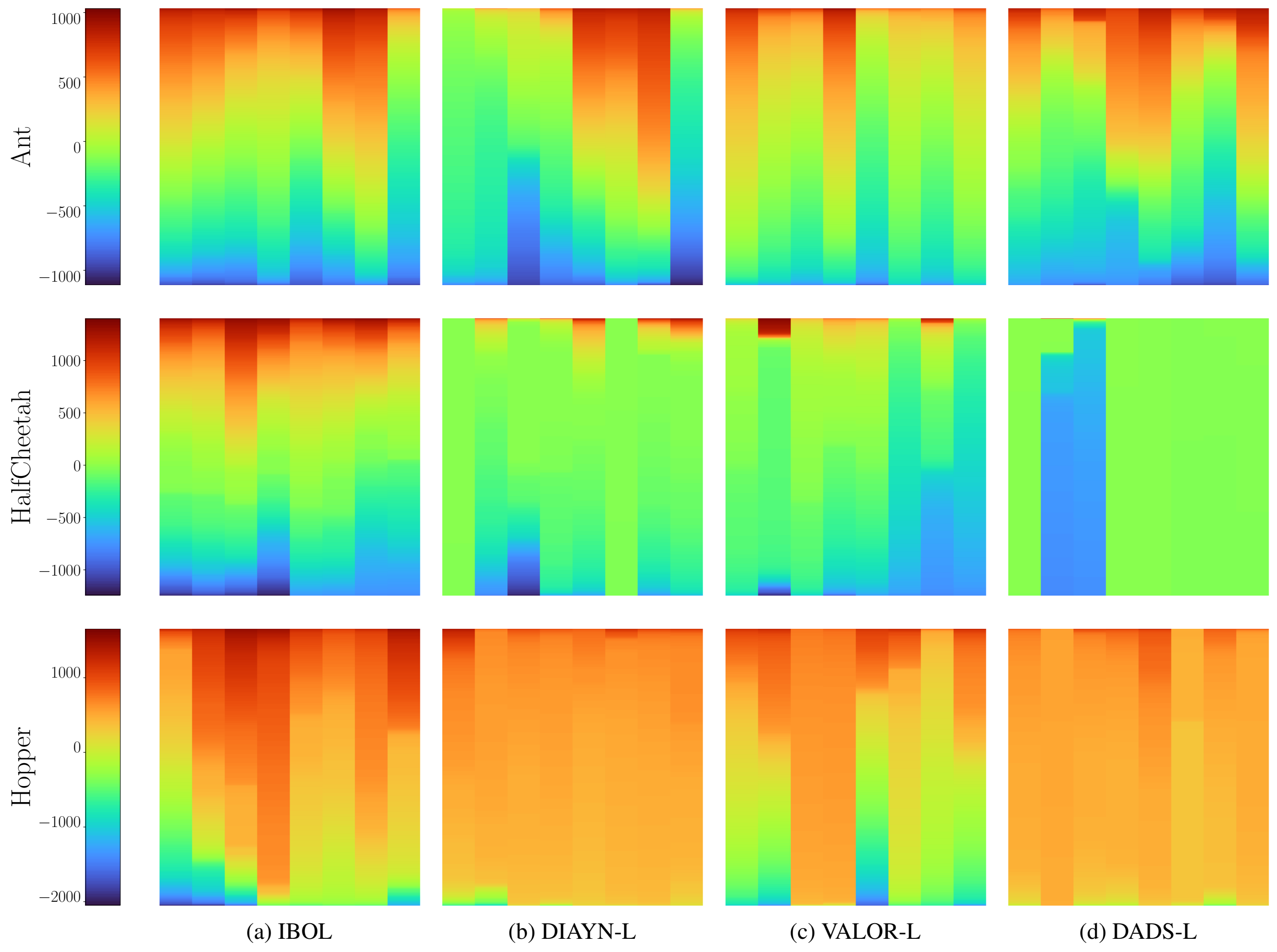


Figure 5. Comparison of IBOL (ours) with the baseline methods, DIAYN-L, VALOR-L and DADS-L, in the diversity of external returns for the skills discovered without any rewards. For every method, each of the eight vertical bars visualizes the external returns for 2000 skills sampled randomly with one trained skill policy of the skill discovery method, as a stacked histogram with corresponding colors from the color bar on the left.

I. Experimental Details

I.1. Implementation

We employ garage (garage contributors, 2019) and PyTorch (Paszke et al., 2019) to implement IBOL, DIAYN (Eysenbach et al., 2019), VALOR (Achiam et al., 2018) and DADS (Sharma et al., 2020b). We use the official implementations for EDL[1] (Campos Camúñez et al., 2020) and SeCTAR[2] (Co-Reyes et al., 2018) with additional tuning of hyperparameters to ensure fair comparisons.

I.2. Environments

We experiment with robot simulation environments in MuJoCo (Todorov et al., 2012): Ant, HalfCheetah, Hopper and Humanoid from OpenAI Gym (Brockman et al., 2016) adopting the configurations by Sharma et al. (2020b) and D'Kitty with random dynamics from ROBEL (Ahn et al., 2020) with the setups provided by Sharma et al. (2020a). We use a maximum episode horizon of 200 environment steps for Ant, HalfCheetah and D'Kitty, 500 for Hopper and 1000 for Humanoid. Note that D'Kitty and Humanoid have variable episode horizons, and we use an alive bonus of $3e-2$ at each step in the training of the linearizers for Humanoid to stabilize the training.

For the linearizer, we omit the locomotion coordinates of the torso (x and y for Ant, Humanoid and D'Kitty, and x for the others) from the input of the policy. Note that the linearizer could be agnostic to the agent's global location since its rewards are computed only with the change of the state. On the other hand, we retain them for skill discovery policies and meta-controller policies since, without those

[1] https://github.com/victorcampos7/edl
[2] https://github.com/wyndwarrior/Sectar

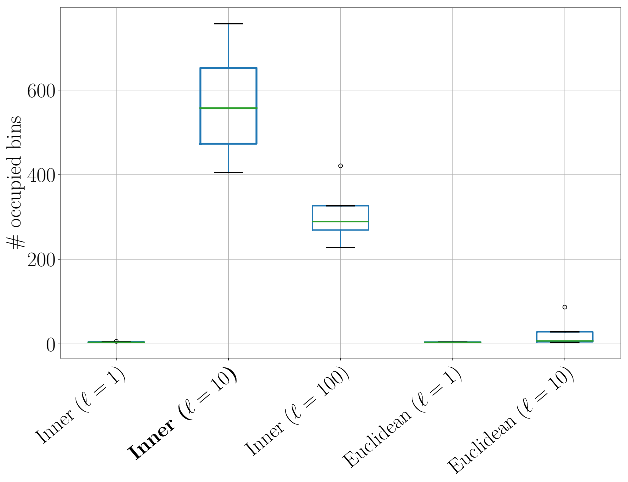


Figure 6. Comparison of various reward function choices for the linearizer. The box plot shows the state coverage of each reward function, measured by the number of bins occupied by the 2000 trajectories in the state space. We use four random seeds for each method.

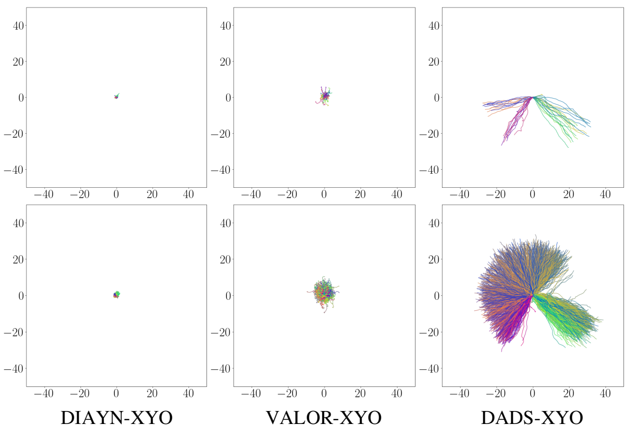


Figure 7. Visualization of the x-y traces of the skills for Ant discovered by each baseline method trained with the omission of the x-y coordinates from the inputs and the x-y prior (Sharma et al., 2020b). The same skill latents are used with Figure 1 of the main paper.

coordinates, the expressiveness of learnable skills may be restricted.

However, as DADS originally omits the x-y coordinates from the inputs in Ant (Sharma et al., 2020b), we also test baseline methods of $d = 2$ with both the omission and the x-y prior (Sharma et al., 2020b), denoted with the suffix '-XYO', in Figure 7.

I.3. Models

In the experiments, we use an MLP with two hidden layers of 512 dimensions for each non-recurrent learnable component except for the linearizer, which uses two hidden layers of 1024 dimensions. We use the $\tanh$ and ReLU nonlinearities for the policies and the others, respectively. We model the outputs of the linearizer and the meta-controller for downstream tasks with the factorized Gaussian distribution followed by a $\tanh$ transformation to fit into the action space of environments. We use the Beta distribution policies for the skill discovery methods. To feed policies with the skill latent variable, we concatenate the skill latent z for each episode with its state s_t at every time step t.

For the trajectory encoder of IBOL and VALOR, we use a bidirectional LSTM with a 512-dimensional hidden layer followed by two 512-dimensional FC layers. When training VALOR without the linearizer, we use a subset of the full state sequence of each trajectory with evenly spaced states to match the effective horizon with VALOR-L, following Achiam et al. (2018). We employ the original implementation choice of DADS to predict $\Delta s = s' - s$ (instead of s') from s and z with its skill dynamics model (Sharma et al., 2020b). Both s and Δs are batch-normalized, with a fixed covariance matrix of I and a Gaussian mixture model with four heads, again following Sharma et al. (2020b).

I.4. Training

We use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of $1e - 4$ for skill discovery methods and $3e-4$ for the others. We normalize each dimension of states, which is important since it helps skill discovery methods equally focus on every dimension of the state space rather than solely on large-scale dimensions. Note that while we observe that the skill discovery methods primarily focus on the locomotion dimensions in the absence of the x-y prior (Sharma et al., 2020b) as in Figure 1 from the main paper, this is not due to the scale of those dimensions, as all the state dimensions are normalized. We hypothesize it is because the locomotion dimensions are those which can have high informativeness with the skill latent variable. When training meta-controllers or skill policies with the linearizer, we use the exponential moving average. For the rest, we use the mean and standard deviation pre-computed from 10000 trajectories with an episode length of 50. Meta-controllers for downstream tasks and skill policies use the mode of each output distribution from their lower-level policies.

At every epoch of the training of the linearizer or meta-controller for downstream tasks, we collect ten trajectories for Ant, HalfCheetah, Hopper and D'Kitty, and five trajectories for Humanoid. For the skill discovery methods with the linearizer, at each epoch 64 trajectories are sampled for

Ant, HalfCheetah and D'Kitty and 32 for Hopper and Humanoid. When training the methods without the linearizer (*e.g.* VALOR-XY), we collect ten trajectories for Ant, since their effective horizon is longer than that with the linearizer.

The linearizer. We train the linearizer using SAC (Haarnoja et al., 2018a) with the automatic entropy adjustment (Haarnoja et al., 2018b) for $8e4$ epochs for D'Kitty, $3e5$ epochs for Humanoid and $1e5$ epochs for the others. We apply 4 gradient steps and consider training with and without a replay buffer, where rewards are normalized with their exponential moving average without a buffer and 2048-sized mini-batches are used with a buffer of $1e6$. We set the initial entropy to 0.1, the target entropy to $-dim(\mathcal{A})/2$, the target smoothing coefficient to 0.005 and the discount rate to 0.99. We choose a prior goal distribution for each environment from $\{\text{Beta}(1,1), \text{Beta}(2,2)\}$. We determine the hyperparameters based on the state coverage of the trained linearizer.

Skill discovery methods. We train IBOL and the '-L' variants of other skill discovery methods for $1e4$ epochs with $\ell = 10$, while the methods without the linearizer are trained with the number of transitions that matches the total number of transitions for the training of both the linearizer and each skill discovery method on top of it. We employ SAC for DADS and DIAYN, and the vanilla policy gradient (VPG) for IBOL and VALOR. We set the entropy regularization coefficient to 1e-3 for VALOR and VALOR-L (searched over $\{1e-1, 1e-2, 1e-3, 0\}$), and use the automatic entropy adjustment for DADS with an initial entropy coefficient of $1e-1$, DADS-L with $1e-3$, DIAYN with $1e-1$ and DIAYN-L with $1e-2$ (searched over $\{1e-1, 1e-2, 1e-3\}$ with and without the automatic regularization). For those using VPG, we apply four gradient steps with the entire batch at each epoch. For SAC, we apply 64 gradient steps (or 32 steps for the skill dynamics model in DADS) with 256-sized mini-batches, since increased gradient steps expedite the training by exploiting the off-policy property of SAC. We use $L = 100$ for DADS and IBOL, and set $\lambda = 2$ (searched over $\{0.1, 1, 2\}$) and $\beta = 1e-2$ (searched over $\{1e-1, 1e-2, 1e-3\}$) for IBOL.

Meta-controllers for downstream policies. SAC is used for training the meta-controllers. We fix the entropy coefficient to 0.01, and apply four gradient steps with the full-sized batches. The meta-controllers select skill latents in a range of $[-2, 2]$, where they are fed into the learned skill policies.

I.5. Downstream Tasks

In *AntGoal*, we sample a goal $w \in [-50, 50]^2$ at the beginning of each roll-out. In *AntMultiGoals*, a goal w is sampled from $[s^{(x)} - 15, s^{(x)} + 15] \times [s^{(y)} - 15, s^{(y)} + 15]$ at every $\eta = 50$ steps, where $[s^{(x)}, s^{(y)}]$ denotes the agent's position when the goal is about to be sampled. In *CheetahGoal*, we sample a goal $w \in [-60, 60]$ when each episode starts.

I.6. Information-Theoretic Evaluations

For each environment, we employ two pre-trained linearizers, and train every method four times for each linearizer, resulting in eight runs in total. To measure the quantities, we sample 2000 trajectories per run and use quantization, where for each variable we divide the range of the values from all the runs into 32 bins.

I.7. Additional Settings

For the rendered scenes of skills, we additionally consider excluding velocity dimensions defining the goal space for the linearizer as in HIRO (Nachum et al., 2018), to get more visually diverse skills. For learning the non-locomotion skills in Section 4.4 from the main paper, we exclude the x and y dimensions from the input of each component. Also, for the experiments with the goal space distortion in Section 4.4, we preserve only the x and y dimensions in the inputs.

J. Ablation Study

In this section, we demonstrate the effect of each hyperparameter of IBOL by showing qualitative results on a synthetic environment named *PointEnv*, which is suitable for clear illustrations. In PointEnv, a state $s \in \mathbb{R}^2$ is defined as the x-y coordinates of the agent (point), and an action $a \in [-0.1, 0.1]^2$ indicates a vector by which the agent moves. The initial state is sampled from $[-0.05, 0.05]^2$ uniformly at random. As PointEnv is already linearized, we do not use the linearizer for IBOL as well as other baseline methods. We also reduce the common dimensionality of the neural networks to 32 in lieu of 512. We train IBOL, DIAYN, VALOR and DADS for $5e3$ epochs with an episode length of 50 and a learning rate of $3e-4$, having two-dimensional skill latents with various hyperparameter settings on this environment. For IBOL, we test $\beta \in \{2.25e-1, 2.25e-3, 0\}$ and $\lambda \in \{1.5, 0.45, 0.15\}$, and we also consider the setting without the auxiliary parameter u for the sampling policy π_{θ_s}, in which we model the sampling policy as an LSTM policy (instead of a non-recurrent policy) to compensate for the reduced expressiveness that comes from the dropping of u. We examine the entropy regularization coefficient $\alpha \in \{1e+1, 1e-1, 1e-3\}$ for VALOR, DADS and DIAYN, and we test the automatic entropy regularization for SAC (Haarnoja et al., 2018b) as well for the latter two.

Figures 8 and 9 illustrate the x-y traces of the skills discovered by each method with various settings. First, we observe that an appropriate value of β (especially $\beta = 2.25e-3$

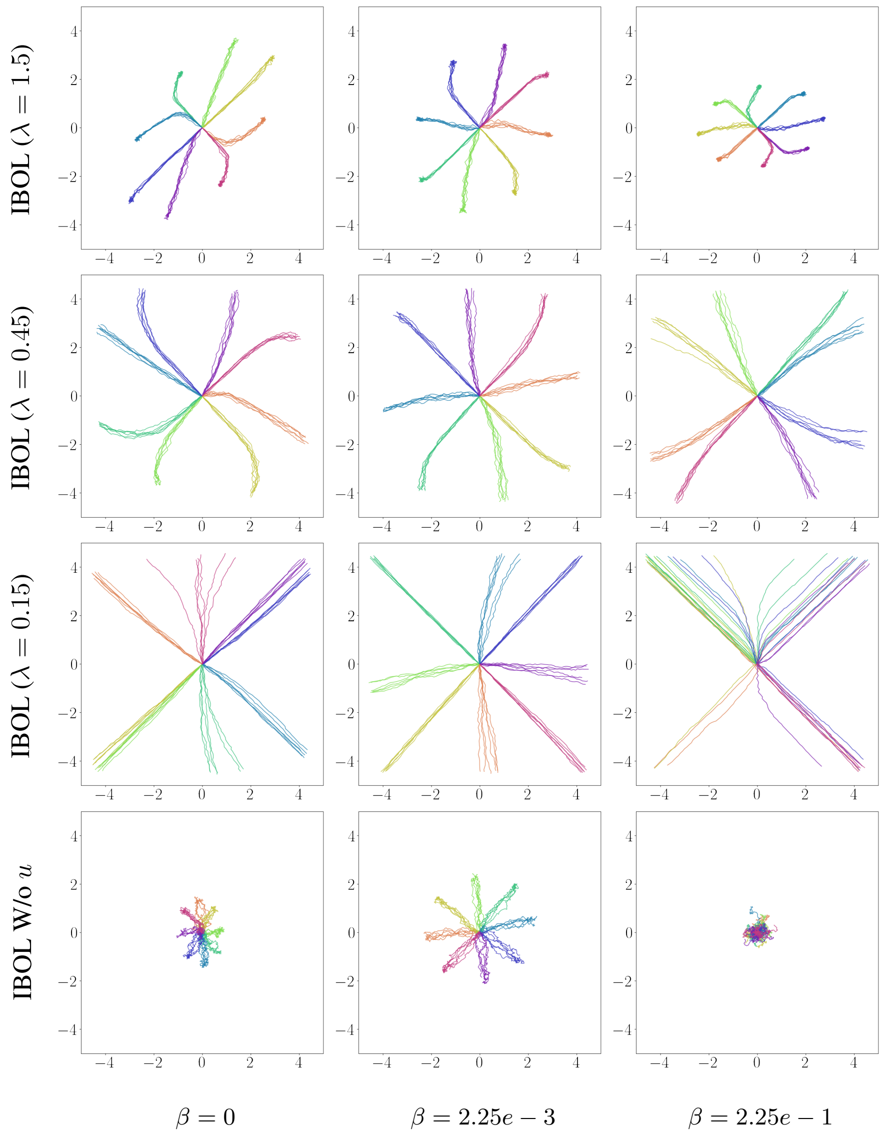


Figure 8. Visualization of the x-y traces of the skills discovered by IBOL in PointEnv with various hyperparameter settings. The fourth row corresponds to IBOL without u and the auxiliary term, modelling π_{θ_s} as a LSTM policy. The same skill latents are used with the top row of Figure 1 of the main paper.

in Figure 8) for IBOL helps discover more disentangled and evenly distributed skills. Also, since the auxiliary term $\mathbb{E}_{u\sim p(u),\tau\sim p_{\theta_s}(\tau|u)}[\lambda \cdot p_\phi(u|s_{0:T})]$ encourages IBOL to discover skills that can be easily reconstructed from the trajectories, increasing λ results in having relatively condensed trajectories. The fourth row of Figure 8 shows that IBOL can still discover visually disentangled (yet slightly noisy) skills in the absence of u and the auxiliary term. Figure 9 presents that for the baseline methods, overly increasing α could result in collapsing while having a moderate value of α improves the quality of discovered skills.

References

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

Achille, A. and Soatto, S. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018a.

Achille, A. and Soatto, S. Information dropout: Learning optimal representations through noisy computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2897–2905, 2018b.

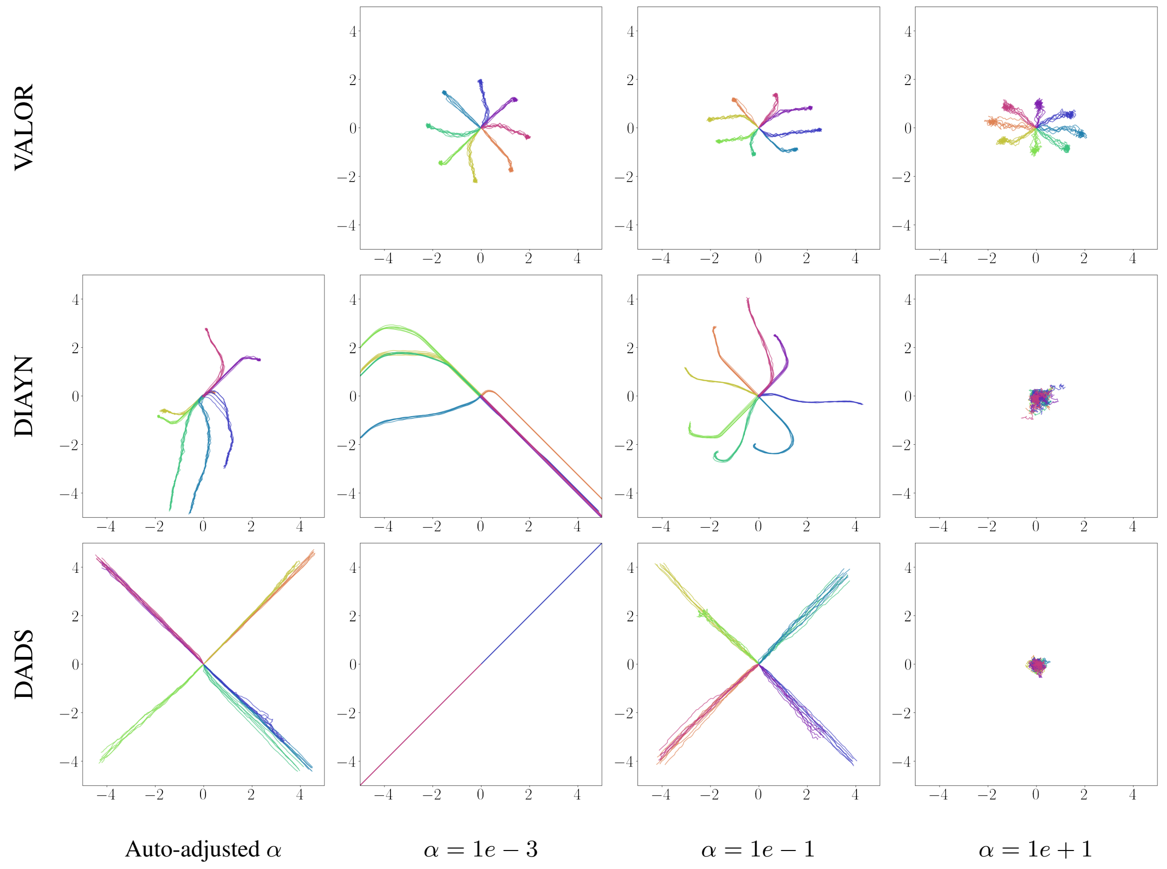


Figure 9. Visualization of the x-y traces of the skills discovered by VALOR, DIAYN and DADS in PointEnv with various hyperparameter settings. The same skill latents are used with the top row of Figure 1 of the main paper.

Ahn, M., Zhu, H., Hartikainen, K., Ponte, H., Gupta, A., Levine, S., and Kumar, V. Robel: Robotics benchmarks for learning with low-cost robots. In *Conference on Robot Learning*, pp. 1300–1313. PMLR, 2020.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *ArXiv*, abs/1606.01540, 2016.

Campos Camúñez, V., Trott, A., Xiong, C., Socher, R., Giró Nieto, X., and Torres Viñals, J. Explore, discover and learn: unsupervised discovery of state-covering skills. In *ICML 2020, Thirty-seventh International Conference on Machine Learning:[posters]*, pp. 1–17, 2020.

Chen, R. T., Li, X., Grosse, R. B., and Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. In *Advances in neural information processing systems*, pp. 2610–2620, 2018.

Co-Reyes, J. D., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. In *ICML*, 2018.

Do, K. and Tran, T. Theory and evaluation metrics for learning disentangled representations. In *International Conference on Learning Representations*, 2020.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. 2019.

Florensa, C., Duan, Y., and Abbeel, P. Stochastic neural networks for hierarchical reinforcement learning. In *ICLR*, 2016.

garage contributors, T. Garage: A toolkit for reproducible reinforcement learning research. https://github.com/rlworkgroup/garage, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018a.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and

Levine, S. Soft actor-critic algorithms and applications. *ArXiv*, abs/1812.05905, 2018b.

Kim, H. and Mnih, A. Disentangling by factorising. In *ICML*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Nachum, O., Gu, S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In *NeurIPS*, 2018.

Nachum, O., Gu, S., Lee, H., and Levine, S. Near-optimal representation learning for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Sharma, A., Ahn, M., Levine, S., Kumar, V., Hausman, K., and Gu, S. Emergent real-world robotic skills via unsupervised off-policy reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2020a.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020b.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In *ICML*, 2017.