Appendix

A Full pseudo-code

A.1 ComSD

We provide the ComSD's full pseudo-code. It also serves as an example to demonstrate the process of general unsupervised skill discovery.

Algorithm 1 Pseudo-code of ComSD.

```
###Unsupervised skill discovery by ComSD
Require: Reward-free environment $E_f$, the uniform skill distribution $p(z)$, unsupervised pre-training environment steps $I_p$, and the RL batch size $I_b$.
Initialize: The state encoder $f_{\theta_1}(\cdot)$, the skill encoder $f_{\theta_2}(\cdot)$, the skill-conditioned policy (actor) $\pi(a|s,z)$, the critic $Q(a, concat(s,z))$, and the replay buffer $D$.
 1: for $t = 1, ..., I_p$ do
 2:     Sample a skill vector from uniform distribution $z_t \sim p(z)$.
 3:     Obtain current action $a_t \sim \pi(\cdot|s_t, z_t)$ based on current observation $s_t$.
 4:     Interact with reward-free environment $E_f$ with $a_t$ to get next observation $s_{t+1}$.
 5:     Add the transition $(s_t, z_t, a_t, s_{t+1})$ into replay buffer $D$.
 6:     Sample $I_b$ transitions from $D$ (after enough data collection).
 7:     Compute contrastive learning loss $\mathcal{L}_{NCE}$ shown in Eq.(7) with $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$.
 8:     Use backpropogation to update $f_{\theta_1}(\cdot)$ and $f_{\theta_2}(\cdot)$.
 9:     Compute exploitation reward $r^{intr}_{exploitation}$ with Eq.(8).
10:     Compute exploration reward $r^{intr}_{exploration}$ with Eq.(10).
11:     Compute the final intrinsic reward $r^{intr}_{ComSD}$ with Eq.(12).
12:     Augment sampled transition batch by the $r^{intr}_{ComSD}$.
13:     Use DDPG to update $\pi(a|s,z)$ and $Q(a, concat(s,z))$ over $I_b$ intrinsic-reward transitions.
14: end for
Output: the discovered skills (trained skill-conditioned policy) $\pi(a|s,z)$.
```

A.2 Skill Combination Evaluation

Algorithm 2 Pseudo-code of skill combination evaluation.

```
###ADAPTION EVALUATION OF PRE-TRAINED SKILLS BY SKILL COMBINATION
Require: Reward-specific environment $E_s$, the pre-trained skill-conditioned policy $\pi(a|s,z)$, environment adaption steps $I_a$, and the RL batch size $I_b$.
Initialize: The meta-controller (actor) $\pi'(z|s)$, the critic $Q(z,s)$, and the replay buffer $D$.
1: Freeze the learned skills $\pi(a|s,z)$.
2: for $t = 1, ..., I_a$ do
3:     Obtain current skill vector $z_t \sim \pi(\cdot|s_t)$ based on current observation $s_t$.
4:     Obtain current action $a_t \sim \pi(\cdot|s_t, z_t)$ based on $s_t$ and $z_t$.
5:     Interact with reward-specific environment $E_s$ with $a_t$ to get next observation $s_{t+1}$ and the extrinsic reward $r_{extr}$.
6:     Add the transition $(s_t, z_t, r_{extr}, s_{t+1})$ into replay buffer $D$.
7:     Sample $I_b$ transition batch from $D$.
8:     Use DDPG to update $\pi'(z|s)$ and $Q(z,s)$ over $I_b$ transitions.
9: end for
Output: The performance of $\pi'(z|s)$ serves as the skill combination evaluation result.
```

A.3 Skill Finetuning Evaluation

Algorithm 3 Pseudo-code of skill finetuning evaluation.

```
###ADAPTION EVALUATION OF PRE-TRAINED SKILLS BY SKILL FINETUNING
Require: Reward-specific environment $E_s$, the pre-trained skill-conditioned policy $\pi(a|s,z)$, environment adaption steps $I_a$, skill choice steps $I_c$, and the RL batch size $I_b$.
Initialize: The critic $Q(a, concat(s,z))$, and the replay buffer $D$.
1: Choose a skill vector $z_i$ in $I_c$ steps by your algorithm (e.g., a fixed choice in CIC and ComSD) and save the corresponding $I_c$ extrinsic-reward transitions into $D$.
2: Freeze the chosen skill vector $z_i$.
3: for $t = 1, ..., I_a - I_c$ do
4:     Obtain current action $a_t \sim \pi(\cdot|s_t, z_i)$ based on $s_t$ and $z_i$.
5:     Interact with reward-specific environment $E_s$ with $a_t$ to get next observation $s_{t+1}$ and the extrinsic reward $r_{extr}$.
6:     Add the transition $(s_t, z_i, a_t, r_{extr}, s_{t+1})$ into replay buffer $D$.
7:     Sample $I_b$ transition batch from $D$.
8:     Use DDPG to update $\pi(a|s_t, z_i)$ and $Q(a, concat(s,z))$ over $I_b$ transitions.
9: end for
Output: The performance of $\pi(a|s, z_i)$ serves as the skill finetuning evaluation result.
```

B Baseline Details

DIAYN (Eysenbach et al., 2018) is one of the most classical and original unsupervised skill discovery algorithms, trying to maximize the MI between skills and states. It employs the first MI decomposition, Eq.(1). It uses a discrete uniform prior distribution to guarantee the maximization of skill entropy $H(z)$. The negative state-conditioned entropy $-H(z|s)$ is estimated by a trainable discriminator $\log p(z|s)$ which computes the intrinsic reward. As a foundational work, it provides several reasonable evaluations of skill adaptation, of which skill finetuning and skill combination are employed in our experiments.

SMM (Lee et al., 2019) aims to learn a policy for which the state marginal distribution matches a given target state distribution. It optimizes the objective by reducing it to a two-player, zero-sum game between a state density model and a parametric policy. Like DIAYN, it is also based on the first decomposition (Eq.(1)) of MI and employs discriminator training. The difference is that SMM explicitly maximizes the state entropy with intrinsic reward, which inspires lots of recent advanced works and our ComSD.

APS (Liu & Abbeel, 2021a) first employs the second MI decomposition Eq.(2) for a better MI estimation. For state entropy estimation, it employs a popular particle-based entropy estimation proposed by APT (Liu & Abbeel, 2021b), which is proven effective and supports many advanced works (Laskin et al., 2022b; Yarats et al., 2021b) and our ComSD. For skill-conditioned entropy, it chooses the successor feature (Hansen et al., 2019), introducing it into the final intrinsic reward for an explicit maximization. The weight between different entropy estimations is fixed in APS. APS can't guarantee the behavioral quality (exploration) well. Different from APS, our ComSD employs contrastive learning for better conditioned entropy estimation and designs a novel dynamic weighting algorithm (SMW) to overcome the exploration drop brought by explicit conditioned entropy maximization.

CIC (Laskin et al., 2022b) is a state-of-the-art robot behavior discovery method. It first introduces contrastive learning (Chen et al., 2020) into unsupervised skill discovery. It chooses the second MI decomposition, Eq.(2) with APT particle-based estimation for state entropy, like APS. The contrastive learning between state transitions and skill vectors is conducted for implicit skill-conditioned entropy maximization. The encoder learned by contrastive learning is further used for APT reward improvement. The behaviors produced by CIC are of high activity but not distinguishable. Different from CIC, we employ the contrastive results as diversity intrinsic rewards for explicit conditioned entropy maximization to improve behavioral diversity, with SMW to balance two entropy estimations for exploratory ability maintenance.

BeCL (Yang et al., 2023) is another state-of-the-art method on URLB (Laskin et al., 2021) and 2D exploration (Campos et al., 2020). It tries to mitigate the exploitation problem in CIC by a novel MI objective, $I(s^1, s^2)$, where s^1 and s^2 denote different states generated by the same skill. It provides theoretical proof to show that their novel MI objective serves as the upper bound of the previous MI objective. However, BeCL can't generate enough dynamic robot behaviors, and their intrinsic reward computational consumption is also much larger than other approaches.

C DETAILED EXPERIMENTAL SETTINGS

C.1 COMSD HYPER-PARAMETERS

Table 2: Hyper-parameter settings of ComSD in unsupervised skill discovery.

Hyper-parameter	Setting
Skill vector dimensions	64
Skill vector space	$[0,1]$ continuous
Skill update frequency	50
State embedding MLP in $f_{\theta_1}(\cdot)$	$\dim(s) \rightarrow 1024 \rightarrow 1024 \rightarrow 64$
Predictor (MLP) in $f_{\theta_1}(\cdot)$	$64 \times 2 \rightarrow 1024 \rightarrow 1024 \rightarrow 64$
State encoder activation	ReLU
Skill encoder (MLP) $f_{\theta_2}(\cdot)$	$64 \rightarrow 1024 \rightarrow 1024 \rightarrow 64$
Skill encoder activation	ReLU
β upper bound w_{high}	2
β lower bound w_{low}	0
f_{high} for walker & quadruped	1
f_{low} for walker & quadruped	0
Fixed coefficient α for walker	0.25
Fixed coefficient α for quadruped	$1e-3$
f_{high} for hopper & cheetah	$2/3$
f_{low} for hopper & cheetah	$1/3$
Fixed coefficient α for hopper	1.25
Fixed coefficient α for cheetah	1
RL backbone algorithm	DDPG
Number of pre-training frames	2000000
RL replay buffer size	1000000
Action repeat	1
Seed (random) frames	4000
Return discount	0.99
Number of discounted steps for return	3
Batch size	1024
Optimizer	Adam
Learning rate	$1e-4$
Actor network (MLP)	$\dim(s) + 64 \rightarrow 1024 \rightarrow 1024 \rightarrow \dim(a)$
Actor activation	$\mathrm{layernorm(Tanh)} \rightarrow \mathrm{ReLU} \rightarrow \mathrm{Tanh}$
Critic network (MLP)	$\dim(s) + 64 + \dim(a) \rightarrow 1024 \rightarrow 1024 \rightarrow 1$
Actor activation	$\mathrm{layernorm(Tanh)} \rightarrow \mathrm{ReLU}$
Agent update frequency	2
Target critic network EMA	0.01
Exploration stddev clip	0.3
Exploration stddev value	0.2

C.2 Skill Combination Experimental Settings

Table 3: Hyper-parameter settings of skill combination adaptation task.

Hyper-parameter	Setting
RL backbone algorithm	DDPG
Meta-controller training frames	2000000
RL replay buffer size	1000000
Action repeat	1
Seed (random) frames	4000
Return discount	0.99
Number of discounted steps for return	3
Batch size	1024
Optimizer	Adam
Learning rate	$1e-4$
Actor network (MLP)	$\dim(s) \rightarrow 1024 \rightarrow 1024 \rightarrow 64$
Actor activation	$\text{layernorm(Tanh)} \rightarrow \text{ReLU} \rightarrow \text{Tanh}$
Critic network (MLP)	$\dim(s) + 64 \rightarrow 1024 \rightarrow 1024 \rightarrow 1$
Actor activation	$\text{layernorm(Tanh)} \rightarrow \text{ReLU}$
Agent update frequency	2
Target critic network EMA	0.01
Training stddev clip for meta-controller	0.3
Training stddev value for meta-controller	0.2
Eval frequency	10000
Number of Eval episodes	10
Eval stddev value for meta-controller	0.2
Eval stddev value for pre-trained agent (cheetah)	0.2
Eval stddev value for pre-trained agent (others)	0

C.3 Skill Finetuning Experimental Settings

Table 4: Hyper-parameter settings of skill finetuning adaptation task.

Hyper-parameter	Setting
Fixed target skill for ComSD	$(0, 0.5, 0.5, ..., 0.5)$
RL backbone algorithm	DDPG
Number of finetuning frames	100000
RL replay buffer size	1000000
Action repeat	1
Seed (random) frames	4000
Return discount	0.99
Number of discounted steps for return	3
Batch size	1024
Optimizer	Adam
Learning rate for walker&quadruped	$1e-4$
Learning rate for hopper&cheetah	$2e-5$
Actor network (MLP)	$\dim(s) + 64 \rightarrow 1024 \rightarrow 1024 \rightarrow \dim(a)$
Actor activation	$\text{layernorm(Tanh)} \rightarrow \text{ReLU} \rightarrow \text{Tanh}$
Critic network (MLP)	$\dim(s) + 64 + \dim(a) \rightarrow 1024 \rightarrow 1024 \rightarrow 1$
Actor activation	$\text{layernorm(Tanh)} \rightarrow \text{ReLU}$
Agent update frequency	2
Target critic network EMA	0.01
Training stddev clip	0.3
Training stddev value	0.2
Eval frequency	10000
Number of Eval episodes	10
Eval stddev value	0

D Additional Analysis

D.1 What Skills Do ComSD and Competitive Baselines Discover? & Why does ComSD exhibit better adaptation performance than others?

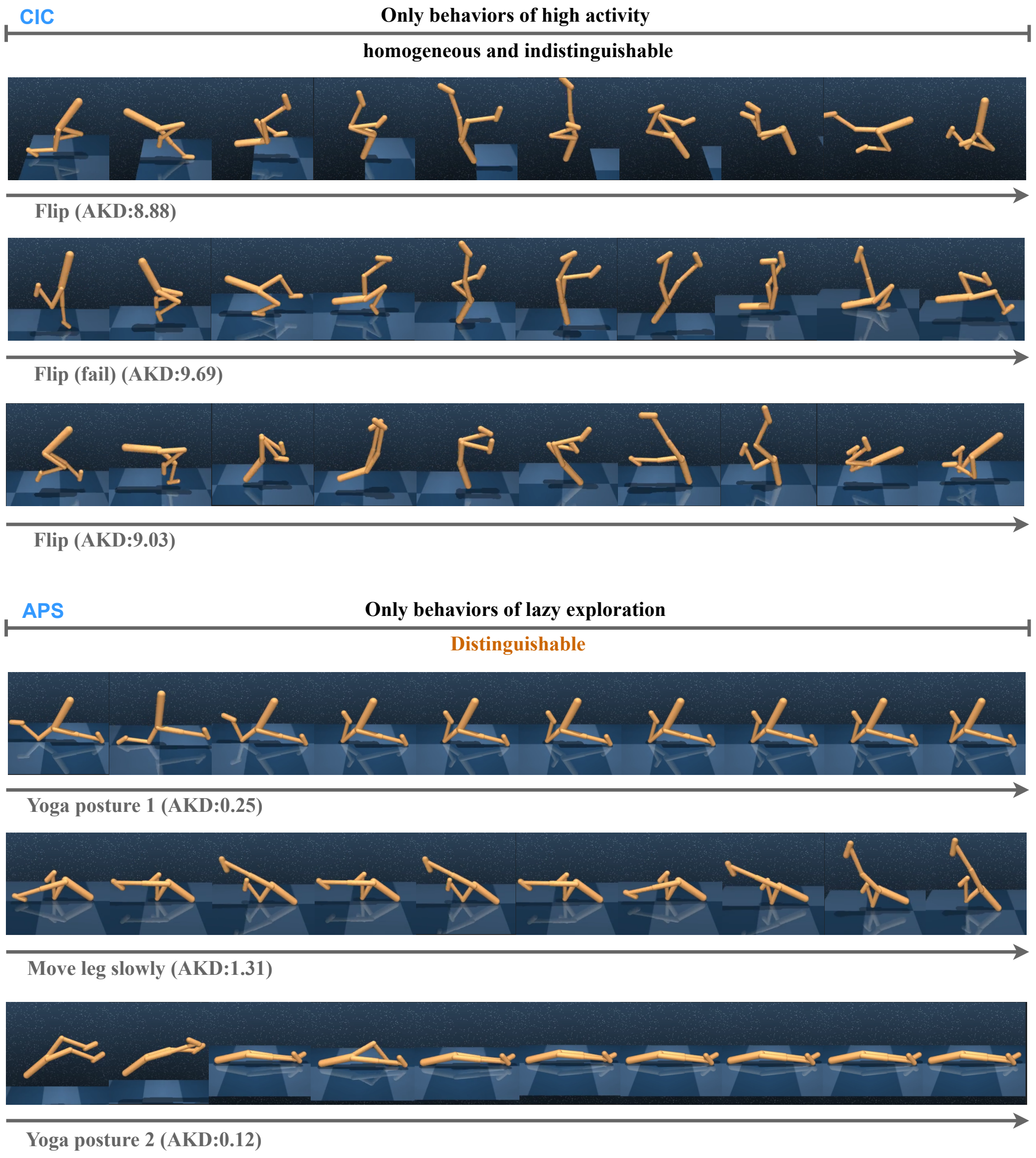
CIC
Only behaviors of high activity
homogeneous and indistinguishable
Flip (AKD:8.88)
Flip (fail) (AKD:9.69)
Flip (AKD:9.03)
APS
Only behaviors of lazy exploration
Distinguishable
Yoga posture 1 (AKD:0.25)
Move leg slowly (AKD:1.31)
Yoga posture 2 (AKD:0.12)

Figure 6: Visualization for representative behaviors discovered by CIC and APS. AKD is a particle-based state entropy estimator used to evaluate skill activity, which we define in Section 4.4. Skill AKD ranges from 7 to 10 for CIC and 0 to 2 for APS, which means they can't discover qualified behaviors at different activity levels.

We provide the skill visualization and corresponding skill AKD of the two most competitive baselines, APS (Liu & Abbeel, 2021a) and CIC (Laskin et al., 2022b), in Figure 6. The visualization and skill AKD of our ComSD are shown in Figure 7. AKD is a particle-based state entropy estimator used to evaluate skill activity, which we define in Section 4.4.

CIC is able to produce continuous movements of high activity, but it can't generate behaviors at other activity levels (AKD range of CIC's skills is 7-10). In addition, CIC suffers from insufficient

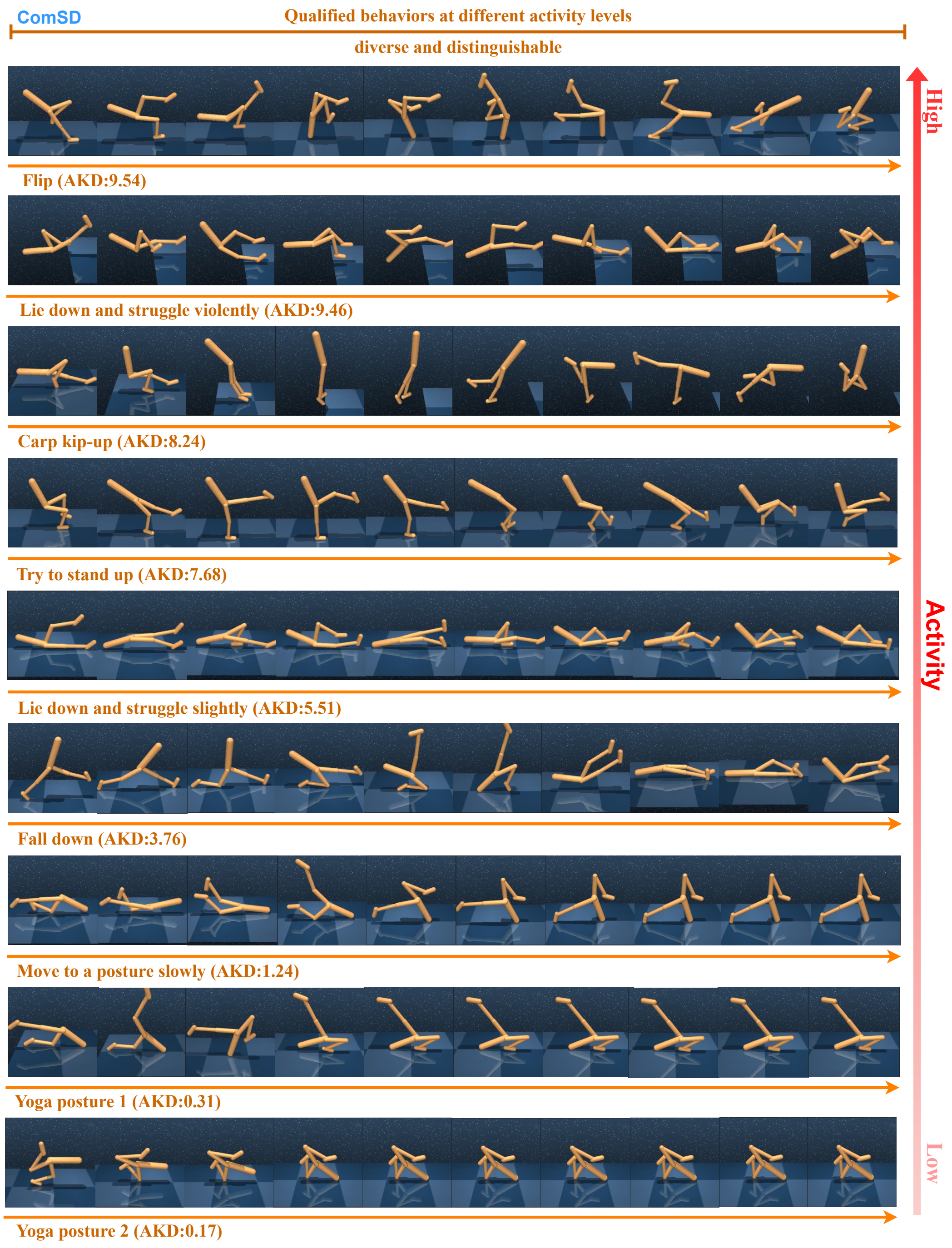
ComSD
Qualified behaviors at different activity levels
diverse and distinguishable
Flip (AKD:9.54)
Lie down and struggle violently (AKD:9.46)
Carp kip-up (AKD:8.24)
Try to stand up (AKD:7.68)
Lie down and struggle slightly (AKD:5.51)
Fall down (AKD:3.76)
Move to a posture slowly (AKD:1.24)
Yoga posture 1 (AKD:0.31)
Yoga posture 2 (AKD:0.17)
High
Activity
Low

Figure 7: Visualization for representative behaviors discovered by ComSD. AKD is a particle-based state entropy estimator used to evaluate skill activity, which we define in Section 4.4. Skill AKD ranges from 0 to 10 for ComSD. The results demonstrate that our ComSD can produce a qualified skill set consisting of diverse behaviors at different activity levels, which recent advanced methods cannot.

exploitation, i.e., the generated skills are indistinguishable and homogeneous. CIC's skills all tend to achieve dynamic flipping, which is consistent with the good initial score on walker flip in skill combination (see Section 4.2). However, the behavioral indistinguishability makes it difficult for meta-controllers to learn ideal combinations for a competitive final score. APS can generate diverse behaviors, but it suffers from lazy exploration and also can't generate behaviors at different activity levels (AKD range of APS's skills is 0-2). In general, poor exploration causes poor performance in skill finetuning evaluation. In skill combination, the diversity allows the meta-controller to complete downstream tasks through the combination of unqualified learned skills, which coincides with the upward trend of APS training curves. In summary, previous advanced methods can't provide a good balance between behavioral quality and diversity, thus failing to exhibit competitive results across different downstream adaptation evaluations.

By contrast, our ComSD can produce diverse behaviors at different activity levels (AKD range of ComSD's skills is 0-10), including flipping, lying down, struggling at different speeds, various postures, and so on. This explains why our ComSD can achieve state-of-the-art adaptation performance across both kinds of downstream tasks while other methods cannot.

D.2 Behavioral Diversity Analysis

In this section, we try to analyze the behavioral diversity of each method. For a skill, we use it to sample 1k states and calculate the Mean State (MS) over all sampled states. MS can partially represent the skill to some extent. For each method, we uniformly sample 41 different walker skills. Over 41 skills, we compute the K-Th-nearest-neighbor MS Distance (KTMSD) and the All-K-nearset-neighbor MS Distance (AKMSD) for skill entropy estimation. KTMSD computes the k-th-nearest-neighbor MS Euclidean distance of each skill and takes the mean, while AKMSD considers all the k-nearest neighbors of each skill and takes the mean distance of MS. (Note that in Section 4.4&Appendix D.1, KTD and AKD are employed for state entropy estimation of one skill, while KTMSD and AKMSD are used for skill entropy estimation of one method in this section.) KTMSD and AKMSD can partially evaluate the skill coverage of one method. In addition, we also calculate the MS Range (MSR) for each method. These metrics are all related to behavioral diversity.

The comparison between all 6 methods on behavioral diversity is shown in Figure 8, demonstrating that ComSD has huge advantages on skill entropy (KTMSD and AKMSD). ComSD is also the most competitive method on MSR. In fact, it's hard to represent an exploratory skill by only MS (exploratory behaviors have much more visited states than static postures), which puts highly exploratory methods (ComSD and CIC) at a disadvantage on these 3 metrics. In this case, ComSD still obtains state-of-the-art evaluation results, which demonstrates that the behavior set discovered by ComSD is of much higher diversity and coverage than other baselines.

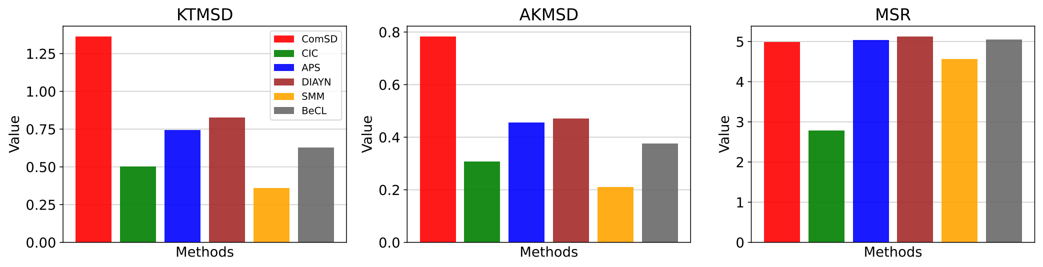

Figure 8: The comparison between all 6 methods on behavioral diversity. ComSD discovers a much more diverse skill set than other baselines.