Appendix of CLUE

Anonymous Author(s)
Affiliation
Address
email

A Additional Results

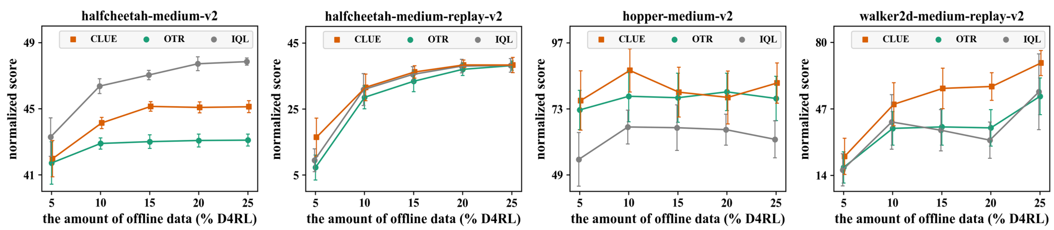

Figure 1: Ablating the number of unlabeled trajectories. We investigate the effect of unlabeled trajectories on the performance. CLUE's performance generally outperforms OTR. Further, we can see that CLUE approximates the vanilla IQL method (with D4RL rewards) more closely and can even outperform IQL given such a lack of offline data ($\leq 25\%$).

Varying the amount of unlabeled offline data. Here we vary the amount of unlabeled offline data available for sparse-reward settings. Figure 1 shows that adding more unlabeled data improves the performance of both CLUE and OTR. However, across a range of offline imitation tasks, CLUE shows better performance compared to OTR. We also plot the performance curve of naive IQL with (reward-labeled) offline data in Figure 1. We can see that with extremely limited offline data ($\leq 25\%$), CLUE approaches IQL's performance more closely on the halfcheetah-medium task, and can even outperform IQL on the remaining three tasks.

Table 1: Using 10% of D4RL data, normalized scores (mean and standard deviation) of CLUE and baselines on antmaze tasks using one (K=1) and ten (K=10) expert demonstrations. The expert trajectories are picked from the chosen 10% dataset. The highest score in each setting is highlighted.

Dataset	IQL	OTR (K=1)	**CLUE** (K=1)	OTR (K=10)	**CLUE** (K=10)
umaze	73.7 ± 7.6	71.4 ± 8.5	75.4 ± 6.1	75.1 ± 8.3	82.5 ± 5.1
umaze-diverse	21.6 ± 9.8	33.0 ± 8.5	45.4 ± 10.4	30.8 ± 13.5*	58.6 ± 9.5*
medium-play	23.0 ± 8.9	38.7 ± 11.1	30.5 ± 13.9	37.3 ± 10.0	36.6 ± 12.7
medium-diverse	54.9 ± 7.8	60.9 ± 8.7	64.4 ± 8.9	59.2 ± 9.2	57.8 ± 8.6
large-play	5.8 ± 3.8	15.0 ± 8.4	12.0 ± 6.5	13.9 ± 5.8	29.4 ± 8.4
large-diverse	7.0 ± 3.6	3.3 ± 3.6	0.9 ± 1.5	9.0 ± 5.9	9.7 ± 4.5
antmaze-v2 total	186.0	222.3	**228.6**	225.3	**274.6**

* Only two successful trajectories are in the chosen sub-dataset and the results belong to K=2.

Varying the number of expert trajectories. Using 10% of D4RL data, we vary the number of expert trajectories for sparse-reward offline RL settings in Table 1. We compare our method with baseline methods (IQL and OTR) when only one expert trajectory is selected. For comparison, we train IQL over the naive sparse-reward D4RL data and train OTR over the relabeled D4RL dataset (using optimal transport to compute intrinsic rewards and employing IQL to learn offline RL policy).

We can find that in 7 out of 12 AntMaze tasks across, our CLUE outperforms the baseline OTR. Meanwhile, compared to naive IQL (with sparse rewards), our CLUE implementation generally outperforms better than IQL. This means that with only a single expert trajectory, we can completely replace the *sparse rewards* with our intrinsic reward in offline RL tasks, which can even achieve higher performance in such a data-scarce scenario (10% of D4RL data).

Table 2: Normalized scores (mean) when varying the temperature factor c with a single expert trajectory (K=1).

	$c=1$	$c=2$	$c=3$	$c=4$	$c=5$	$c=6$	$c=7$	$c=8$	$c=9$	$c=10$
umaze	89.4	89.96	91.84	90.88	91.96	92.12	91.68	90.72	90.92	91.2
umaze-diverse	43.08	46.76	43.16	43.76	42.36	56.72	52.6	59.04	66.48	68
medium-play	60.4	63.2	65.2	68.92	68.04	75.32	71.76	74.12	72.2	73.64
medium-diverse	57.8	63.28	63.24	62.04	66.04	70.12	73	74.56	69.4	72.92
large-play	34.16	44.84	46.88	50.68	52.72	53.08	53.64	55.2	53.52	55.8
large-diverse	27.04	33.96	43.16	46.8	44.88	47.44	47.44	49.92	47.28	47.11

Table 3: Normalized scores (mean) when varying the temperature factor c with 10 expert trajectories (K=10).

	$c=1$	$c=2$	$c=3$	$c=4$	$c=5$	$c=6$	$c=7$	$c=8$	$c=9$	$c=10$
umaze	87.88	90	91.08	90.96	91.16	91	89.92	89.44	90.72	91.92
umaze-diverse	45.64	40.32	41.04	38.8	39.52	51.64	51.2	57.11	69.92	71.68
medium-play	58.72	64.2	68.24	71.44	69.92	75.56	74.12	76.2	75.8	76.48
medium-diverse	60.36	57.04	62.12	64.24	63.56	61.44	62.36	64.64	65.47	69.2
large-play	48.24	45.8	51.56	48.2	48.4	52.36	49.91	50.58	52.28	51.87
large-diverse	36.32	46.08	48.64	50.84	51.16	52.44	53.6	50.92	51.4	53.68

Varying the value of the temperature factor in intrinsic rewards. In Tables 2 and 3, we present the results on AntMaze tasks when we vary the value of the temperature factor c in intrinsic rewards. We can find that CLUE can generally achieve a robust performance across a range of temperature factors. In Figure 2, we further analyze our intrinsic reward distribution following OTR. We can find that CLUE's reward prediction shows a stronger correlation with the ground-truth rewards from the dataset, which can be served as a good reward proxy for downstream offline RL algorithms.

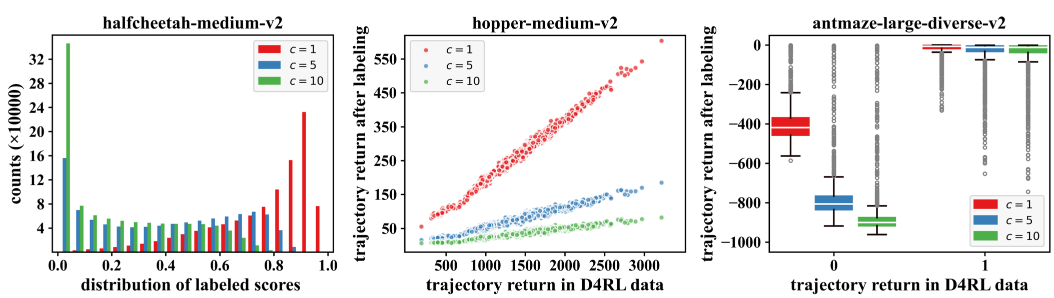

Figure 2: Qualitative comparison of the learned intrinsic rewards with different temperature factors.

B Experimental Details

B.1 Hyperparameters for CVAE Implementation

We list the hyperparameters used for training CVAE models in MuJoCO locomotion, AntMaze, and Adroit tasks. The other CVAE hyperparameters are kept the same as those used in Wu et al. [1].

Table 4: Hyperparameters for training CVAE.

	MuJoCo Locomotion		Antmaze		Adroit
	full-data	partial-data	full-data	partial-data	full-data
Hidden dim	128	128	512	512	128
Batch size	128	128	256	256	128
Numbers of iterations	10^4	10^4	10^5	10^5	10^5
Learning rate	10^{-4}	10^{-4}	10^{-3}	10^{-3}	10^{-4}
Weight for $\mathcal{L}_{\text{calibr}}$	0.1	0.1	0.8	0.8	0.1
Spare-reward setting:					
Number of expert trajectories	3	3	5	5	3

B.2 Hyperparameters for our IQL Implementation

The IQL hyperparameters employed in this paper are consistent with those utilized by Kostrikov et al. [2] in their offline implementation. It is important to note that IQL incorporates a procedure for rescaling rewards within the dataset, which allows for the use of the same hyperparameters across datasets that differ in quality. As CLUE generates rewards offline, we similarly apply reward scaling following the IQL methodology. For the locomotion, adroit, and ant tasks, we rescale rewards with $\frac{1000}{\text{max\_return}-\text{min\_return}}$. To regularize the policy network for the chosen sub-dataset, we similarly introduce Dropout with a rate of 0.2.

MuJoCo locomotion and Adroit tasks. We set the learning rate 10^{-3} for *hopper-medium-expert* dataset (K=10) and 3×10^{-4} for the rest of tasks. We run IQL for 1M gradient steps and average mean returns over 10 random seeds and 10 evaluation trajectories for each seed.

Antmaze tasks. We set the learning rate 5×10^{-4} for *umaze-diverse* dataset (K=1 and K=10) and 3×10^{-4} for the rest of tasks. For *medium-play* dataset (K=1 and K=10), *medium-diverse* dataset (K=1), and *large-play* dataset (K=10), we set the dropout rate 0.2 to gain a better performance. We run IQL for 1M gradient steps for the full dataset and 0.3M for the partial dataset, respectively.

B.3 Hyperparameters in K-means

We use CLUE to learn diversity skills on *Ant-v2*, *HalfCheetah-v2*, and *Walker2d-v2*. The K-means, an unsupervised learning method, is employed to cluster the offline transitions $\{(s, a, s')\}$ from each dataset into 100 classes and take each class as a separate "expert". Specifically, we use *KMEANS* method exacted from *sklearn.cluster* API. The hyperparameters are set as follows: $\text{n\_clusters} = 100, \text{random\_state} = 1, \text{n\_init} = 1, \text{max\_iter} = 300$.

B.4 Offline IL Baselines

SQIL proposes to learn a soft Q-function where the reward labels for the expert transitions are one and the reward labels for the non-expert transitions are zero. The offline implementation of SQIL is adapted from the online SAC agent provided by Garg et al. [3], and we combine it with TD3+BC.

IQ-Learn advocates for directly learning a Q-function by contrasting the expert data with the data collected in the replay buffer, thus avoiding the intermediate step of reward learning. In our experiments, we used the official PyTorch implementation[1] with the recommended configuration by Garg et al. [3].

ORIL assumes the offline dataset is a mixture of both optimal and suboptimal data and learns a discriminator to distinguish between them. Then, the output of the discriminator is used as the reward label to optimize the offline policy toward expert behaviors. We borrowed the TD3+BC implementation reproduced by Ma et al. [4] in our experiments.

[1]https://github.com/Div99/IQ-Learn

ValueDICE is the earliest DICE-based IL algorithm that minimizes the divergence of the state-action distribution between the learning policy and the expert data. The code used in the experiments is the official TensorFlow implementation[2] released by Kostrikov et al. [5].

DemoDICE proposes to optimize the policy via a state-action distribution matching objective with an extra offline regularization term. We report the performance of DemoDICE using the TensorFlow implementation[3] by Kim et al. [6], while the hyperparameters are set as same as the ones in the paper.

SMODICE aims to solve the problem of learning from observation and thus proposes to minimize the divergence of state distribution. Besides, Ma et al. [4] extends the choice of divergence so that the agent is more generalized. The code and configuration used in our experiments are from the official repository[4].

C Learned Diverse Skills

To encourage diverse skills from reward-free offline data, we cluster the offline transitions into 100 classes using K-means and take each class as a separate "expert". Then, we use these expert data from different classes to label the original reward-free data and train IQL policy to learn the corresponding skills. In this section, we illustrate all the learned skills by CLUE.

[2]https://github.com/google-research/google-research/tree/master/value_dice
[3]https://github.com/KAIST-AILab/imitation-dice
[4]https://github.com/JasonMa2016/SMODICE

C.1 Learned Diverse Skills from Ant-Medium Dataset

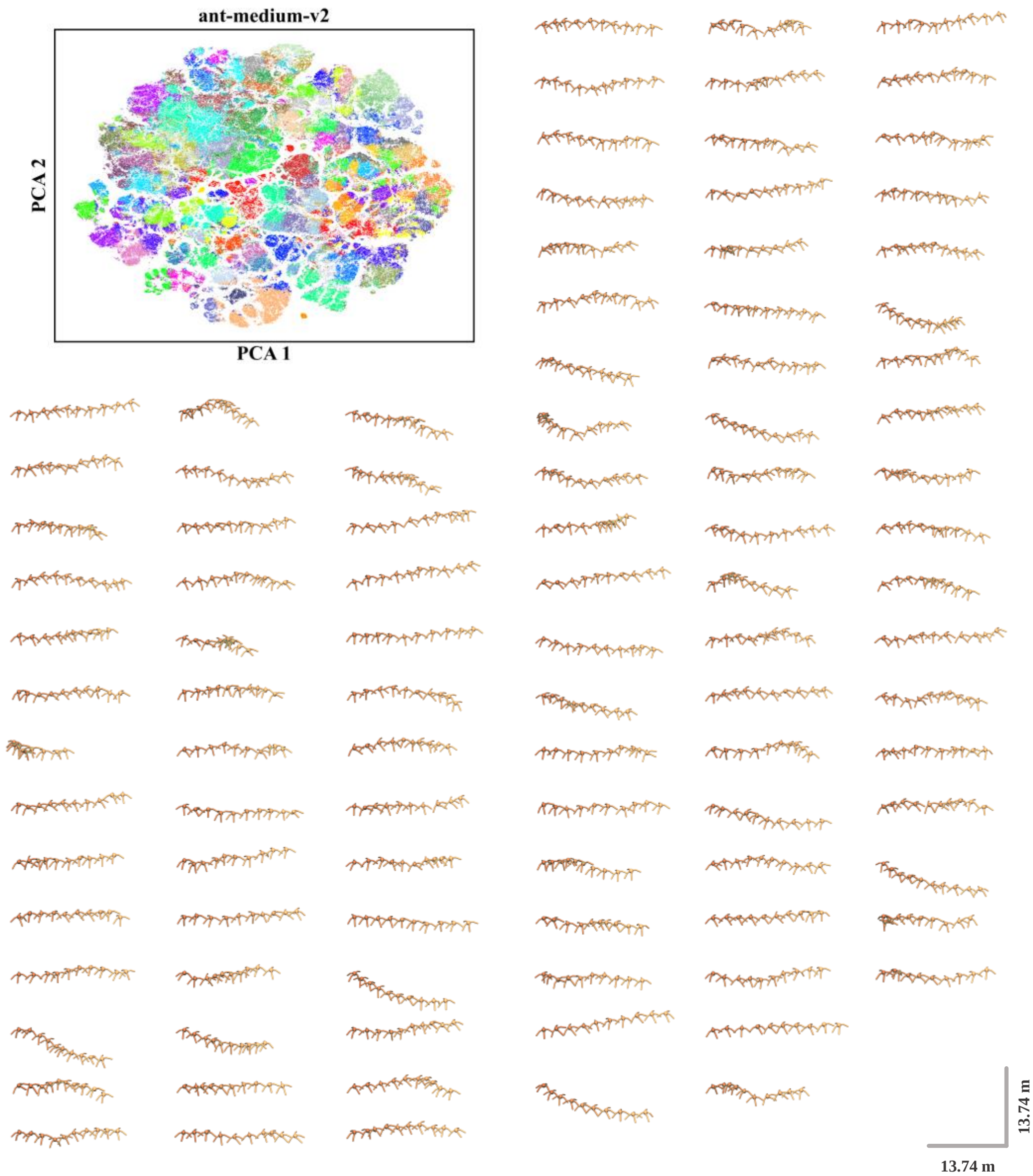

Figure 3: Visualization of unsupervised skills learned from the ant-medium dataset.

78 ### C.2 Learned Diverse Skills from Ant-Random Dataset

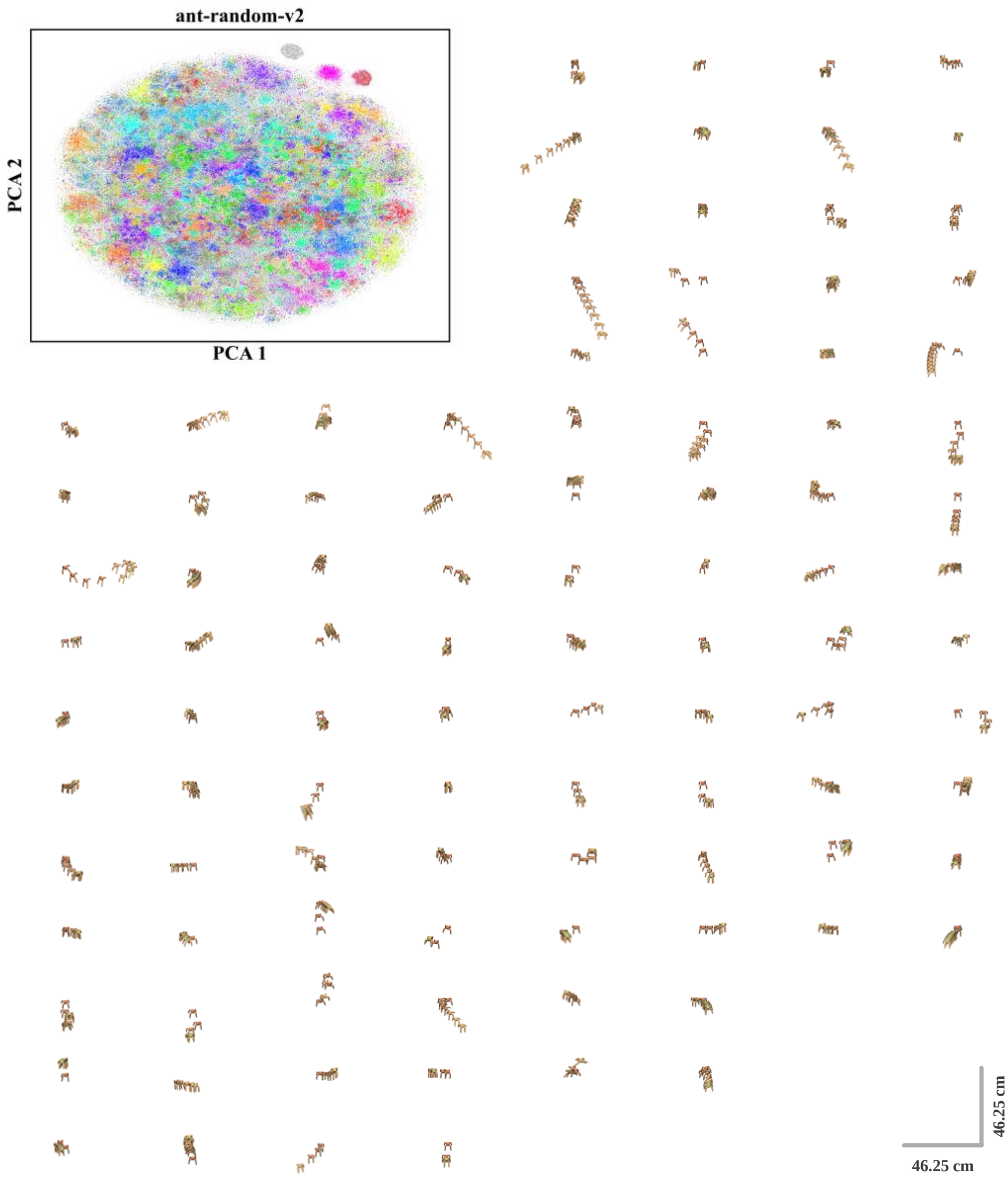

Figure 4: Visualization of unsupervised skills learned from the ant-random dataset.

C.3 Learned Diverse Skills from Halfcheetah-Medium Dataset

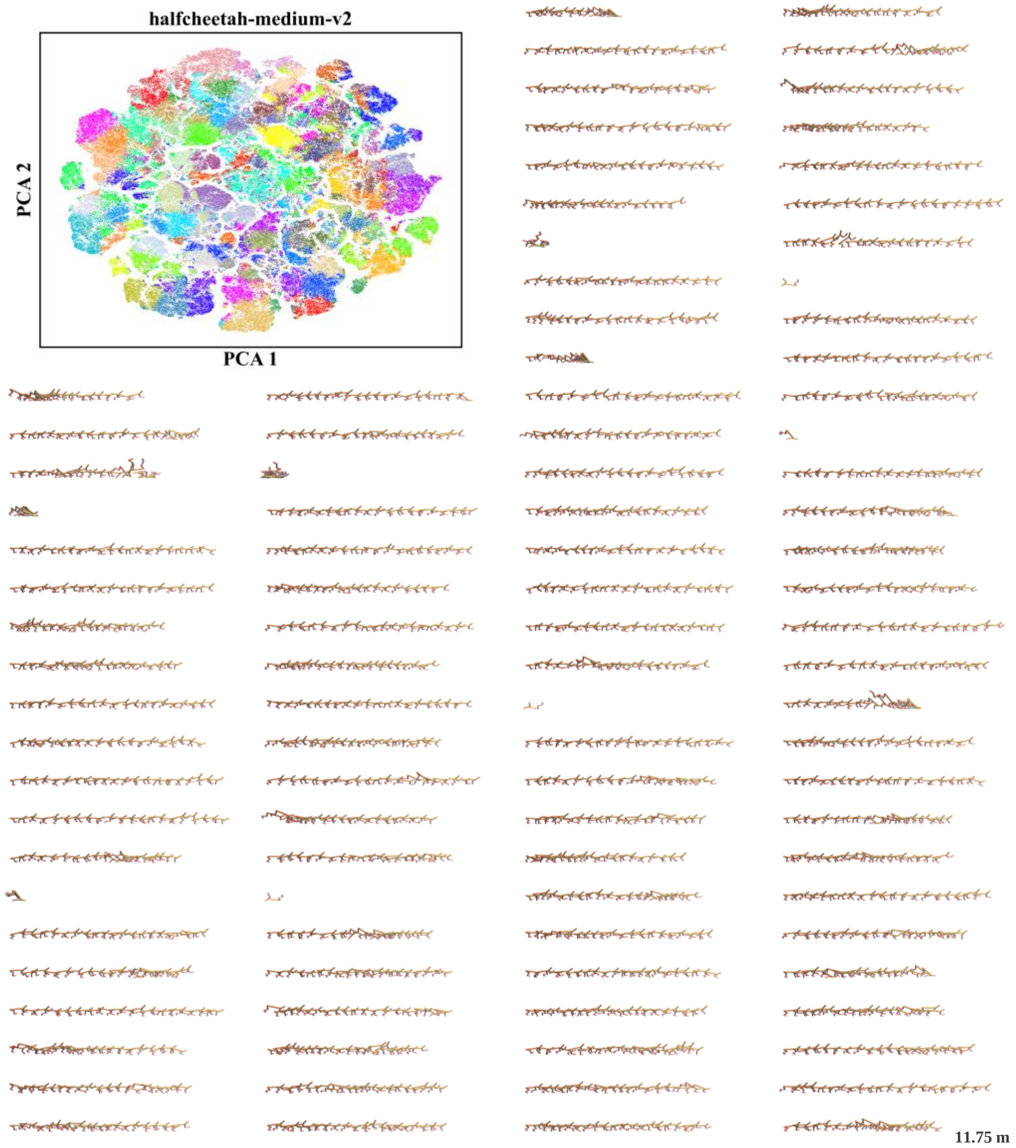

Figure 5: Visualization of unsupervised skills learned from the halfcheetah-medium dataset.

C.4 Learned Diverse Skills from Halfcheetah-Random Dataset

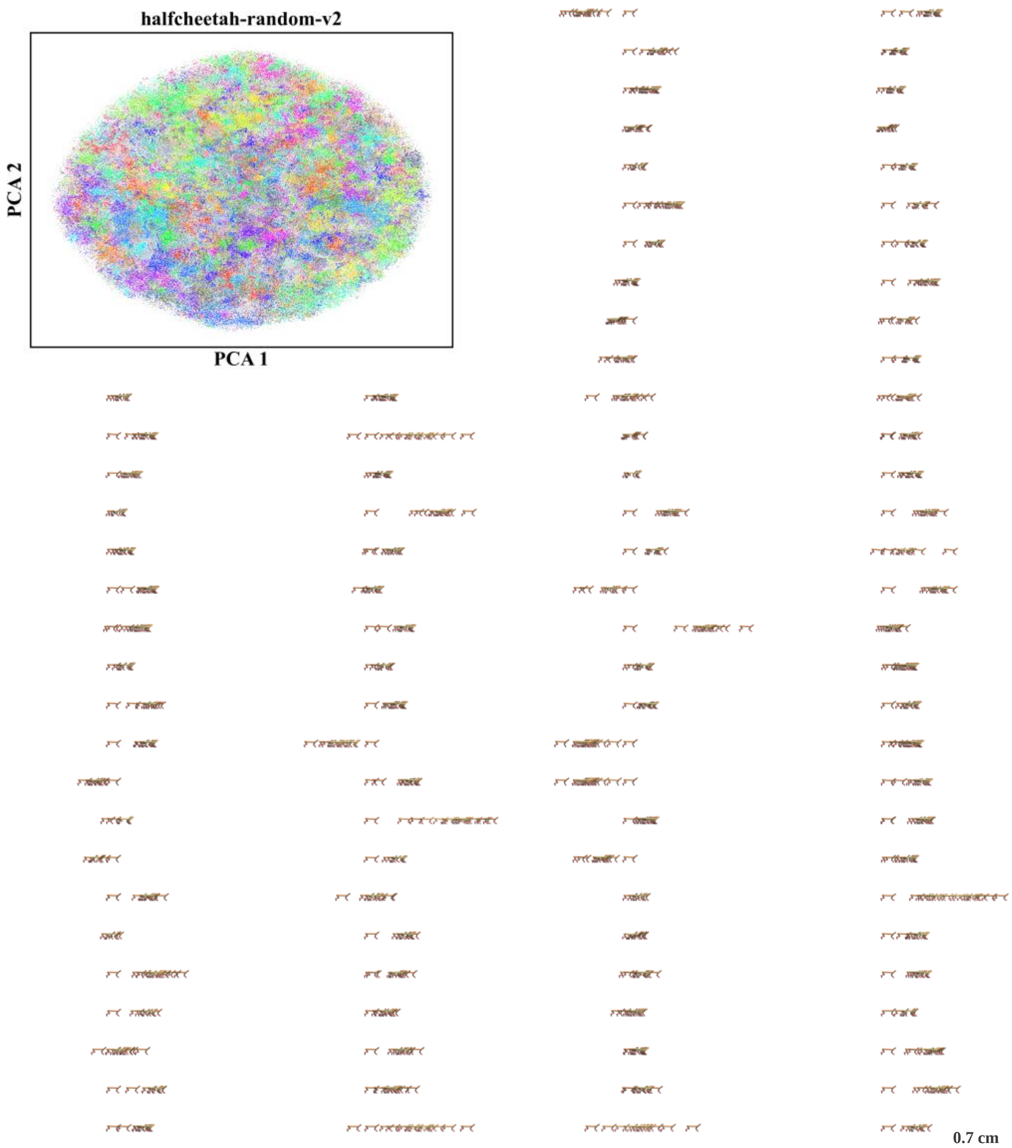

Figure 6: Visualization of unsupervised skills learned from the halfcheetah-random dataset.

81 ### C.5 Learned Diverse Skills from Walker2d-Medium Dataset

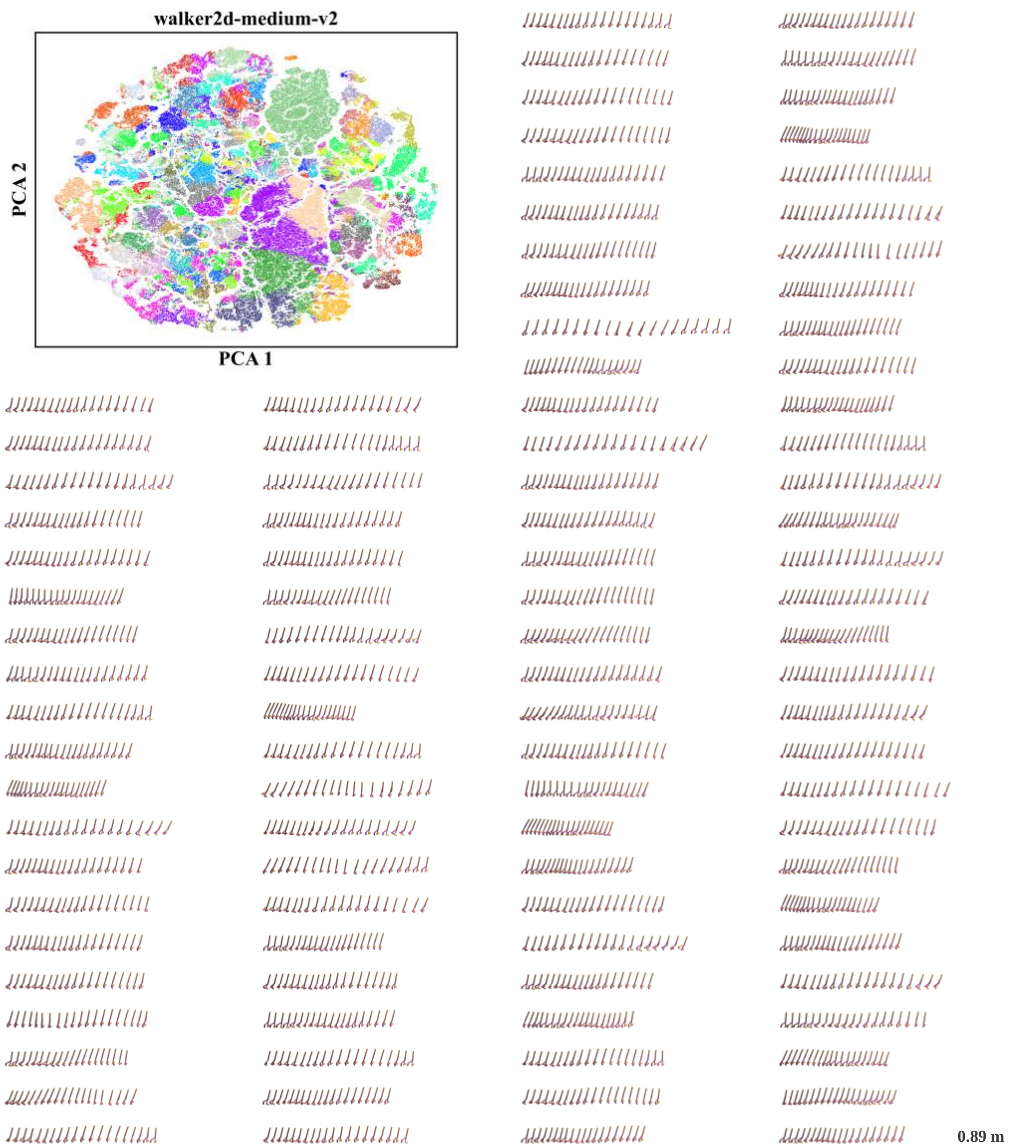

Figure 7: Visualization of unsupervised skills learned from the walker2d-medium dataset.

C.6 Learned Diverse Skills from Walker2d-Random Dataset

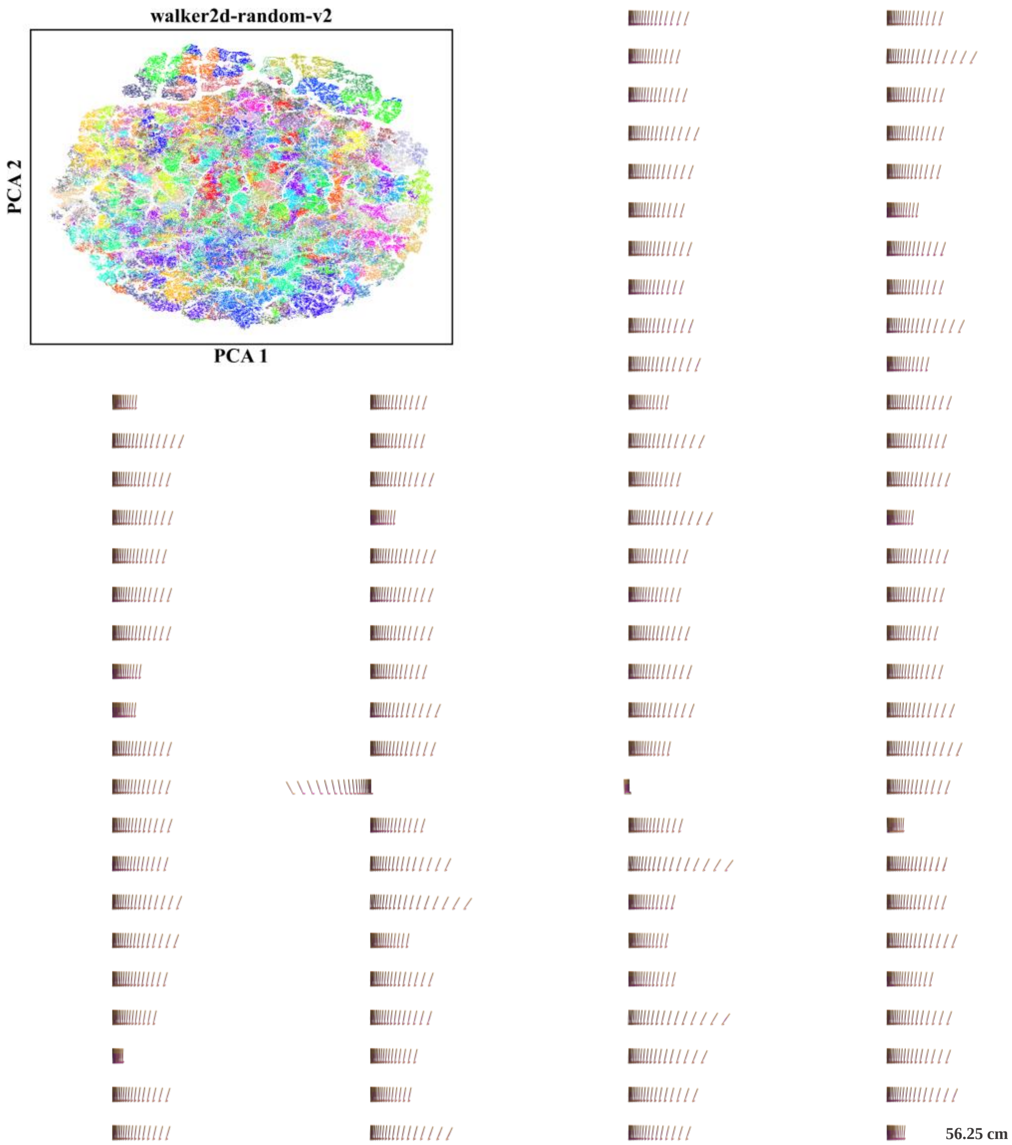

Figure 8: Visualization of unsupervised skills learned from the walker2d-random dataset.

References

[1] J. Wu, H. Wu, Z. Qiu, J. Wang, and M. Long. Supported policy optimization for offline reinforcement learning, 2022.

[2] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning, 2021.

[3] D. Garg, S. Chakraborty, C. Cundy, J. Song, and S. Ermon. Iq-learn: Inverse soft-q learning for imitation. *Advances in Neural Information Processing Systems*, 34:4028–4039, 2021.

[4] Y. J. Ma, A. Shen, D. Jayaraman, and O. Bastani. Smodice: Versatile offline imitation learning via state occupancy matching. *arXiv e-prints*, pages arXiv–2202, 2022.

[5] I. Kostrikov, O. Nachum, and J. Tompson. Imitation learning via off-policy distribution matching. *arXiv preprint arXiv:1912.05032*, 2019.

[6] G.-H. Kim, S. Seo, J. Lee, W. Jeon, H. Hwang, H. Yang, and K.-E. Kim. Demodice: Offline imitation learning with supplementary imperfect demonstrations. In *International Conference on Learning Representations*, 2022.