# A Environment Settings

## A.1 Meta world

Here, we evaluate our model and other methods using the MT10 benchmark in Meta-world [1]. We use relative rewards based on the difference of achieved rewards between successive steps, where each step yields a reward as originally defined in Meta-world. The multi-task set in MT10 includes 10 different environments such as reach, push, pick and place, open door, open drawer, close drawer, press button top-down, insert peg side, open window, and open box; each episode is set to have an individual configuration from 50 different positions of an object and goal.

To generate offline RL datasets, we use the RL agents specifically trained on each task using the Soft actor-critic algorithm [2]. For each task, we build 3 different datasets in terms of their behavior quality including Medium-Replay (MR), Replay (RP), and Medium-Expert (ME). For MR, we use the partially trained agent (that succeeds in completing a given task first) as the medium policy. We collect all the data in an MR dataset until the medium policy is obtained in training. For ME, we use both expert data and suboptimal data; the former is generated by a well trained expert policy and the latter is generated by some policies whose quality is between the medium policy and the expert policy. For RP, we mix the datasets of MR and ME; so, RP corresponds to the datasets generated during a whole training procedure for each task.

## A.2 Drone navigation

To evaluate the applicability of our model in complex problems, we conduct a case study with autonomous quad-copter drones in the Airsim simulator [3]. An agent has observations including lidar data, position, speed, and angle of rotation for drones. In addition, the agent conducts an action at every 500 milliseconds (timestep) and receives the observation data. Specifically, the drone agent manipulates the 3-dimensional acceleration for each timestep $t$, and receives rewards based on the goal distance, i.e.,

$$\textbf{reward} = \begin{cases} \|\text{Pos}_{t-1} - \text{Goal}\|_2 - \|\text{Pos}_t - \text{Goal}\|_2 \\ \min(\|\text{Pos}_{t-1} - \text{Goal}\|_2 - \|\text{Pos}_t - \text{Goal}\|_2 - 0.1,\ 0) \ \text{ if collision.} \end{cases} \tag{1}$$

When the goal is achieved, a new goal location is given. During given fixed timesteps in an episode, the agent learns to achieve as many goals as possible. If the agent collides with obstacles, the reward is subtracted by a given value to learn to avoid the collision.

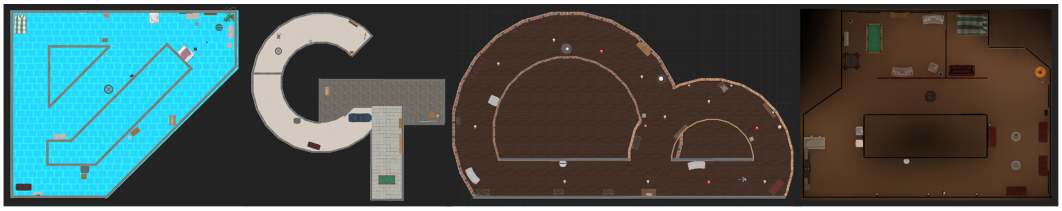


Figure 1: Drone navigation map settings

To build multi-task environments for drone navigation tasks, we use various maps [4] and dynamic wind patterns. Figure 1 shows the example rendering images of various maps. We also configure wind patterns with different velocity on the locations in maps.

$$\textbf{wind} = \left[ \frac{y}{\sqrt{(x^2+y^2)}}, \frac{-x}{\sqrt{(x^2+y^2)}}, 0 \right] \tag{2}$$

where $(x, y)$ is the location. We use 6 different tasks for drone navigation including *indoor-complex*, *indoor-complex-wind*, *indoor-gt*, *indoor-gt-wind*, *indoor-cloud*, and *indoor-pyramid-wind*. For example, *indoor-complex* denotes offline data sampled in the indoor-complex map with no wind, and *indoor-complex-wind* denotes offline data sampled in the indoor-complex map with wind. Notice that we do not include any wind data in the observation to simulate the multi-task environment with 6 different tasks.

# B Implementation

In this section, we provide the implementation details of our model and each comparison method. Our model is implemented using Python v3.8, Jax v0.3.4, and Tensorflow v1.15, and is trained on a system of an Intel(R) Core(TM) i9-10940X processor and an NVIDIA RTX A6000 GPU.

## B.1 TD3+BC

TD3+BC [5] is a state-of-the art offline RL algorithm, which incorporates a behavior cloning regularization term into the update steps of TD3. It induces the learned policy to mimic the actions found in the dataset. This algorithm is used as a baseline to compare our subtask embeddings with conventional multi-task RL methods that use the one-hot task encoding. For implementation, we adopt the open source (https://github.com/sfujim/TD3_BC). Our settings for hyperparameters are summarized in Table 1.

| Hyperparameters | Value |
|---|---|
| Actor network | 3 full connected layers with 256 units |
| Critic network | 3 full connected layers with 256 units |
| The number of critic networks | 2 |
| Activation function | ReLU |
| Learning rate $\eta$ | $1 * 10^{-4}$ |
| Batch size $m$ | 1280 |
| Iterations | 1,000,000 |
| Policy initialization | He initialize |
| Optimizer | Adam |
| Behavior regularizer coefficient $\alpha$ | 2.5 |
| Discounted factor $\gamma$ | 0.99 |
| Target policy noise | 0.2 |
| Target policy clip | 0.5 |
| Policy delay | 2 |
| Polyak update coefficient | 0.005 |

Table 1: Hyperparameter settings for TD3+BC

## B.2 PCGrad

PCGrad [6] is a gradient surgery-based multi-task RL algorithm. It uses a projection function that removes the directional conflicts between gradients for different tasks. For implementation, we use the open source (https://github.com/tianheyu927/PCGrad). The actor and critic models are implemented with 6-layer fully-connected feedforward neural networks. The number of hidden units of each layer is 160. The hyperparameter settings are same as those for TD3+BC in Table 1 except for the actor and critic network structure.

## B.3 Soft modularization

Soft modularization [7] is a modular deep neural network architecture that is tailored with the base and routing networks particularly for multi-task RL. For implementation, we use the open source (https://github.com/RchalYang/Soft-Module). We use this method by adopting offline RL algorithms CQL [8]. The hyperparameter settings are summarized in Table 2, where other settings are same as in Table 1.

| Hyperparameter | Value |
|---|---|
| Number of modular layers | 3 |
| Number of modules for each layer | [4, 4, 4] |
| Number of module hidden units | 256 |
| Representation size | 256 |

Table 2: Hyperparameter settings for Soft modularization

### B.4 SRTD and SRTD+ID

The entire procedure of our SRTD+ID consists of 3-phases: (1) training subtask embeddings via skill-regularized task decomposition, (2) data augmentation by imaginary demonstrations, and (3) training an offline RL agent. We use the TD3+BC algorithm for training an offline RL agent on the subtask embedding space, and we use only given offline data without additional interaction. Given an offline dataset, we add imaginary demonstrations to increase the dataset in SRTD+ID. Specifically, our default setting for this augmentation is to use the data of imaginary demonstrations of half size to the offline dataset. The hyperparameter settings are summarized in Table 3.

| Hyperparameter | Value |
|---|---|
| Skill encoder network $q_\phi$ | 3 full connected layers with 256 units |
| Skill decoder network $p_\phi$ | 3 full connected layers with 256 units |
| Task encoder network $q_\theta$ | 2 full connected layers with 256 units |
| Task decoder network $p_\theta$ | 2 full connected layers with 256 units |
| Latent dimensions | 4 |
| Activation function (for SRTD) | ReLU |
| Learning rate (for SRTD) | $1 * 10^{-4}$ |
| Batch size (for SRTD) | 2048 |
| Length of sub-trajectory $n$ | 4 |
| Epochs | 300 |
| Network initialization | He initialize |
| Optimizer | Adam |

Table 3: Hyperparameter settings for SRTD

Algorithm 1 implements the learning procedure of our proposed SRTD+ID.

**Algorithm 1** The entire procedure of SRTD+ID

Offline dataset $\mathcal{D}$, subtask embedding parameter $\theta$, skill embedding parameter $\phi$, offline RL agent parameter $\psi$
Regulation hyperparameter $\lambda$, batch size $m$, learning rate $\eta$, policy $\pi_\psi$, critic $Q_\psi$
***// Skill-regularized task decomposition***
**loop**
  Sample $\{d_{t_i}, \tau_{t_i}, s_{t_i-n:t_i+n}, a_{t_i-n:t_i+n}, r_{t_i-n:t_i}\}_{i=1}^m \sim \mathcal{D}$
  $\{b_{t_1}, b_{t_2}, ..., b_{t_m}\} = q_\phi(\{d_{t_1}, d_{t_2}, ..., d_{t_m}\}), \{z_{t_1}, z_{t_2}, ..., z_{t_m}\} = q_\theta(\{\tau_{t_1}, \tau_{t_2}, ..., \tau_{t_m}\})$
  $\{\tilde{b}_0, \tilde{b}_1, ..., \tilde{b}_n\} \sim P_B = \mathcal{N}(0, 1)$
  $\{\tilde{z}_0, \tilde{z}_1, ..., \tilde{z}_n\} \sim P_Z = \mathcal{N}(0, 1)$
  $L_{SE}(\phi) = \frac{1}{m}\sum_{i=1}^m \sum_{j=-n}^{n-1} \|a_{t_i+j} - p_\phi(s_{t_i+j}, b_{t_i})\|_2 + L_{PR}(\{b_{t_i}\}_{i=1}^m, \{\tilde{b}_i\}_{i=1}^m)$
  $L_{TE}(\theta) = \frac{1}{m}\sum_{i=1}^m \sum_{j=-n}^{0} \|(s_{t_i+j+1}, r_{t_i+j}) - p_\theta(s_{t_i+j}, a_{t_i+j}, z_{t_i})\|_2$
  $L_{SR}(\theta) = \frac{1}{m}\sum_{i=1}^m \tilde{R}(s_{t_i}, a_{t_i}) \cdot \|q_\theta(\tau_{t_i}) - q_\phi(d_{t_i})\|_2$
  $L_{SRTE}(\theta) = L_{TE}(\theta) + L_{PR}(\{z_{t_i}\}_{i=1}^m, \{\tilde{z}_i\}_{i=1}^m)) + L_{SR}(\theta)$
  $\phi \leftarrow \phi + \eta \cdot \nabla L_{SE},\ \theta \leftarrow \theta + \eta \cdot \nabla L_{SRTE}$
**end loop**
***// Data augmentation by Imaginary demonstrations***
**loop**
  Sample $\{\tau_{t_i}, s_t\} \sim \mathcal{D}$
  $\tilde{a}_t,\ (\tilde{s}_{t+1},\ \tilde{r_t}) = p_\phi(s_t, z_t),\ p_\theta(s_t, \tilde{a}_t, z_t)$ where $z_t = q_\theta(\tau_t)$
  $\mathcal{D} = \mathcal{D} \cup \{(s_t, \tilde{a}_t,\ \tilde{s}_{t+1},\ \tilde{r_t})\}$
**end loop**
***// Train offline RL agent***
**loop**
  Sample $\{\tau_t, s_t, a_t, r_t, \tau_{t+1}, s_{t+1}\} \sim \mathcal{D}$
  $\nabla\mathcal{L}_Q(\psi) \leftarrow \nabla\textsc{CriticLoss}(\psi, ((s_t, q_\theta(\tau_t)), a_t, r_t, (s_{t+1}, q_\theta(\tau_{t+1}))), \pi_\psi)$
  $\nabla\mathcal{L}_{ACTOR}(\psi) \leftarrow \nabla\textsc{ActorLoss}(\psi, ((s_t, q_\theta(\tau_t)), a_t, r_t, (s_{t+1}, q_\theta(\tau_{t+1}))), Q_\psi)$
  $\psi \leftarrow \psi + \eta \cdot (\nabla\mathcal{L}_Q + \nabla\mathcal{L}_{ACTOR})$
**end loop**

# C Additional Experiments

## C.1 Visualization of subtask embeddings

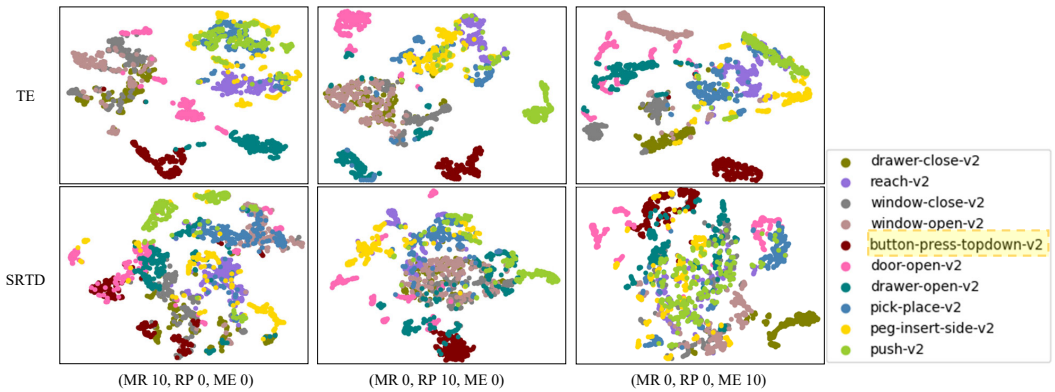


Figure 2: Embedding maps of Skill-regularized task decomposition (SRTD) and task embedding (TE)

Figure 2 shows the embedding maps of the proposed model SRTD and TE in 3 different data configurations such as (MR 10, RP 0, ME 0), (MR 0, RP 10, ME 0), and (MR 0, RP 0, ME 10). Here, TE denotes a model with task embeddings where skill regularization is not used. A configuration (MR 10, RP 0, ME 0) denotes a mixed data setting where all 10 tasks are associated with the MR datasets, and the others follow the same naming convention.

As shown, the embedding maps of SRTD (in the 2nd row) have more common subtask embeddings from different tasks than those of TE (in the 1st row). It is observed that for each color (task), we have more shared regions (subtask embeddings) where the dots of that color are overlapped with the dots of other colors. For example, the *button-press-topdown-v2* task (the dark red-colored dot) has rarely shared regions in the embedding maps of TE, but it has more shared regions in the embedding maps of SRTD.

## C.2 Zero-shot adaptation

To confirm the generalization performance of our skill-regularized task decomposition, we perform a zero-shot evaluation for multi-task settings with new tasks (which are not part of the training multi-task datasets).

Specifically, we use the MT50 benchmark for evaluation using our model that is trained on the MT10 benchmark. The average success rate is shown in Table 4. Our model (SRTD, SRTD+ID) yields higher performance consistently than TE for all the configurations; SRTD and SRTD+ID achieve $1.28 \sim 2.37\%$ and $2.04 \sim 2.91\%$ gains in the success rate compared to TE, respectively. These results clarify that the generalization model performance is improved by our skill-regularized task decomposition. Note that the other comparison methods are not included in this test, since they use the one-hot task encoding which is not relevant for adaptation on new tasks.

| Datasets | TE | SRTD | SRTD+ID |
|---|---|---|---|
| (MR 10, RP 0, ME 0) | $5.79\% \pm 0.61\%$ | $8.16\% \pm 0.91\%$ | $8.70\% \pm 0.84\%$ |
| (MR 0, RP 10, ME 0) | $9.12\% \pm 2.28\%$ | $10.71\% \pm 1.24\%$ | $11.16\% \pm 2.10\%$ |
| (MR 0, RP 0, ME 10) | $10.23\% \pm 0.98\%$ | $11.51\% \pm 0.78\%$ | $12.65\% \pm 1.05\%$ |

Table 4: Zero-shot adaptation for MT50 by offline RL agents trained on MT10

## C.3 Performance comparison about quality estimation methods

While we use episodic returns for quality estimation and sub-trajectory relabeling, our SRTD can be readily extended to other quality-estimation methods. For example, Hindsight credit assignment

(HCA) [9] can be used for quality estimation and sub-trajectory relabeling in SRTD, where HCA exploits the advantage for hindsight relabeling, i.e.,

$$A_\pi(s,a) = \mathbb{E}_{\tau\sim\mathcal{D}}\left[\left(1 - \frac{\pi(a|s)}{\pi_z(a|s,\tilde{R}(s,a))}\right) * \tilde{R}(s,a)\right] \tag{3}$$

where $\tilde{R}(s,a)$ is a return and $\pi_z$ is a return conditioned policy. Compared to the case of using the episodic returns in SRTD, our experiments rarely specify any performance improvement (i.e., as shown in the first 3 rows in Table 5). That was expected to some extent because sampled transitions within an episode (or trajectory) turn out to be relatively either uniformly low-quality or high-quality in our datasets. In the offline RL context, it is common for offline dataset collection that a behavior (sampling) policy remains the same during an episode as it is learned [10].

We also test the other case, the mixed-quality within an episode (MIX-EPI) where the behavior policy's quality are frequently changed even during a single episode. We deliberately set a sequence of sampling policies for each episode of MIX-EPI datasets such that different policies are used for a few timesteps in rotation. This data collection emulates the environment where the quality of sub-trajectories is highly variable within each individual episode. In the MIX-EPI 10 case of Table 5, we observe the performance difference achieved by SRTD and the SRTD variant with HCA (SRTD+HCA) for MT10; this motivates us as our future research to investigate other quality estimation and relabeling strategies for a wide range of mixed configurations of different quality datasets.

| Datasets | SRTD | SRTD+HCA |
|---|---|---|
| (MR 10, RP 0, ME 0) | $21.24 \pm 1.40\%$ | $22.14 \pm 1.09\%$ |
| (MR 0, RP 10, ME 0) | $38.97 \pm 3.38\%$ | $36.50 \pm 2.01\%$ |
| (MR 0, RP 0, ME 10) | $46.60 \pm 3.11\%$ | $47.06 \pm 2.18\%$ |
| (MIX-EPI 10) | $39.6\% \pm 3.24\%$ | $42.4\% \pm 1.95\%$ |

Table 5: Performance comparison of SRTD and SRTD+HCA

### C.4 Performance comparison about the length of sub-trajectory

To estimate the quality of sub-trajectories, our proposed method uses unbiased quality measures such as advantage or episodic return. In the case of having sub-trajectories of variable lengths, the quality measure might vary depending on their length. While it is also interesting to investigate how to stably approximate the quality in variable length settings, we define the length of sub-trajectories as a fixed hyperparameter and perform experiments with various dataset quality conditions, focusing on the quality-aware skill regularization.

We use $n$-length sub-trajectories $(s,a)_{t-n:t}$ in the task embedding procedure, and we use $2n$-length sub-trajectories $(s,a)_{t-n:t+n-1}$ in the skill embedding procedure. In our implementation, task embeddings (generated by the task encoder $q_\theta$ in Figure 2 of the main paper) are used as input for a learned RL policy, so only $n$-length sub-trajectories (without future transitions) are used, similar to the task embedding method in [11]. However, sub-trajectories for skill embeddings are $2n$-length transitions including the past of $n$-length and the future of $n$-length, since skills abstract the action sequence conditioned on a given (current) state, similar to the skill embedding method in [12]. Table 6 shows the performance in multi-task success rates for MT10 achieved by different sub-trajectory length settings $n = 2, 4, 8, 16, 32$. As shown, no significant difference in performance is observed as long as $n$ is not too short or too long.

| Datasets | | | SRTD ($n$: sub-trajectory length) | | | | |
|---|---|---|---|---|---|---|---|
| MR | RP | ME | n=2 | n=4 | n=8 | n=16 | n=32 |
| 10 | 0 | 0 | $19.75 \pm 1.01\%$ | $21.24 \pm 1.40\%$ | $20.28 \pm 1.25\%$ | $21.51 \pm 2.25\%$ | $15.59 \pm 3.81\%$ |
| 0 | 10 | 0 | $34.32 \pm 2.12\%$ | $38.97 \pm 3.38\%$ | $38.50 \pm 3.58\%$ | $40.64 \pm 6.25\%$ | $37.11 \pm 3.19\%$ |
| 0 | 0 | 10 | $38.52 \pm 3.44\%$ | $46.60 \pm 3.31\%$ | $46.43 \pm 2.81\%$ | $44.21 \pm 4.84\%$ | $43.25 \pm 2.57\%$ |

Table 6: Performance with respect to sub-trajectory lengths

### C.5 Performance comparison with conservative data sharing

In conservative data sharing (CDS) [13], the data limitation problem in offline RL was discussed and selective data sharing strategies across different task datasets were presented. Unlike CDS, we don't assume that reward function for each tasks is known, so we compare our model and CDS under different experiment conditions, where CDS exploits known reward functions but our model does not. We observe that CDS achieves good performance when high-quality data is sufficiently given but its performance much degrades when high-quality data is not sufficiently given. We speculate that it is because CDS shares only the transitions with high Q-values learned by CQL algorithm. In Table 7, the dataset configurations (MR 10, RP 0, ME 0), (MR 0, RP 10, ME 0), and (MR 5, RP3, ME 2) represent relatively low-quality conditions, while the dataset configurations (MR 0, RP 0, ME 10) and (MR 4, RP 3, ME 3) represent relatively high-quality conditions. For the former configurations, we observe better performance by SRTD+ID, and for the latter configurations, we observe comparable performance between SRTD+ID and CDS.

| Datasets | SRTD+ID | CDS |
|---|---|---|
| (MR 10, RP 0, ME 0) | $23.87 \pm 2.22\%$ | $17.50 \pm 2.10\%$ |
| (MR 5, RP 3, ME 2) | $32.13 \pm 3.57\%$ | $29.6 \pm 3.30\%$ |
| (MR 0, RP 10, ME 0) | $41.91 \pm 5.88\%$ | $35.88 \pm 2.14\%$ |
| (MR 4, RP 3, ME 3) | $43.53 \pm 3.32\%$ | $42.17 \pm 2.57\%$ |
| (MR 0, RP 0, ME 10) | $49.29 \pm 3.35\%$ | $48.12 \pm 1.41\%$ |

Table 7: Performance comparison of SRTD+ID and CDS

## References


[1] Tianhe Yu et al. "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning". In: *Proceedings of the 3rd Conference on Robot Learning (CoRL)*. PMLR. 2019.

[2] Tuomas Haarnoja et al. "Soft actor-critic algorithms and applications". In: *arXiv:1812.05905* (2018).

[3] Shital Shah et al. "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles". In: *arXiv preprint: 1705.05065* (2017).

[4] Aqeel Anwar and Arijit Raychowdhury. "Autonomous Navigation via Deep Reinforcement Learning for Resource Constraint Edge Nodes using Transfer Learning". In: *arXiv preprint: 1910.05547* (2019).

[5] Scott Fujimoto and Shixiang Shane Gu. "A minimalist approach to offline reinforcement learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20132–20145.

[6] Tianhe Yu et al. "Gradient surgery for multi-task learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 5824–5836.

[7] Ruihan Yang et al. "Multi-task reinforcement learning with soft modularization". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4767–4777.

[8] Aviral Kumar et al. "Conservative q-learning for offline reinforcement learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1179–1191.

[9] Anna Harutyunyan et al. "Hindsight credit assignment". In: *Advances in neural information processing systems* 32 (2019).

[10] Justin Fu et al. "D4rl: Datasets for deep data-driven reinforcement learning". In: *arXiv preprint arXiv:2004.07219* (2020).

[11] Kate Rakelly et al. "Efficient off-policy meta-reinforcement learning via probabilistic context variables". In: *Proceedings of 36th International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 5331–5340.

[12] Taewook Nam et al. "Skill-based Meta-Reinforcement Learning". In: *Proceedings of 10th International Conference on Learning Representations (ICLR)*. 2022.

[13] Tianhe Yu et al. "Conservative data sharing for multi-task offline reinforcement learning". In: *Advances in Neural Information Processing Systems* 34 (2021).