A Detailed for Latent Space Constraint (LSC) and Representation Shift Correction(RSC)

Latent Space Constraint (LSC) As we can see in Fig. 3(a), the latent action representations inside the boundary can be well decoded and estimated the values, while the outliers cannot. Therefore, the most critical problem for latent space constraint (LSC) is to find a reasonable latent space boundary. Simply re-scale policy's outputs in a fixed bounded area $[-b, b]$ could lose some important information and make the latent space unstable (Zhou et al., 2020; Notin et al., 2021) . We propose a mechanism to constrain the action representation space of the latent policy inside a reasonable area adaptively. In specific, we re-scale the output of latent policy (i.e., $[-1, 1]^{d_1+d_2}$ by tanh activation) to a bounded range $[b_{\text{lower}}, b_{\text{upper}}]$. At intervals (actually concurrent with the updates of the hybrid action representation models), we first sample M transitions s, k, x_k from buffer, then we obtain the corresponding latent action representations with current representation models. In this way, we will get M different latent variable values in each dimension. We sort the latent variable of each dimension, calculate the c-percentage central range and let the lower bound and upper bound of the range to be $[b_{\text{lower}}]$ and $[b_{\text{upper}}]$ of the current latent variable. Note that n control the c-percentage central range where $c \in [0, 100]$, we called latent select range. With the decrease of c , the constained latent action representation space becomes smaller. The experiment on the value of n and latent select range is in Appendix C.3.

Representation Shift Correction (RSC) Since the hybrid action representation space is continuously optimized along with the RL learning, the representation distribution of original hybrid actions in the latent space can shift after a certain learning interval (Igl et al., 2020). Fig. 3(b) visualizes the shifting (denoted by different shapes). This negatively influences the value function learning since the outdated latent action representation no longer reflects the same transition at present. To handle this, we propose a representation relabeling mechanism. In specific, we feed the batch of stored original hybrid actions to our representation models to obtain the latest latent representations, for each mini-batch training in Eq.7. For latent discrete action $\hat{z}_k$, if it can not be mapped to the corresponding original action k in the latest embedding table, we will relabel $\hat{z}_k$ through looking up the table with stored original discrete action $\hat{k}$, i.e., $\hat{z}_k \leftarrow e_{\zeta,\hat{k}} + \mathcal{N}(0, 0.1)$. The purpose of adding noise $\mathcal{N}(0, 0.1)$ is to ensure the diversity of the relabeled action representations, For latent continuous action $\hat{z}_x$, we first obtain $\tilde{\delta}_{s,s'}$ through the latest decoder $p_\psi(\hat{z}_x, s, e_{\zeta,\hat{k}})$. Then we verify if $\|\tilde{\delta}_{s,s'} - \delta_{s,s'}\|_2^2 > \delta_0$ (threshold value $\delta_0 =$ is set to be $4 * \hat{L}_{\text{Dyn}}$, where $\hat{L}_{\text{Dyn}}$ is the moving empirical loss), i.e., the case that indicates that the historical representations has no longer semantically consistent (with respect to environmental dynamics) under current representation models. Then $\hat{z}_x$ will be relabeled by the latest latent representations $z_k \sim q_\phi(\cdot \mid \hat{x}_k, s, e_{\zeta,k})$. In this way, the policy learning is always performed on latest representations, so that the issue of representation distribution shift can be effectively alleviated. The experiment on relabeling techniques is in Appendix C.3.

B Experimental Details

B.1 Setups

Our codes are implemented with Python 3.7.9 and Torch 1.7.1. All experiments were run on a single NVIDIA GeForce GTX 2080Ti GPU. Each single training trial ranges from 4 hours to 10 hours, depending on the algorithms and environments. For more details of our code can refer to the `HyAR.zip` in the supplementary results.

Benchmark Environments We conduct our experiments on several hybrid action environments and detailed experiment description is below.

- ***Platform*** (Masson et al., 2016): The agent need to reach the final goal while avoiding the enemy or falling into the gap. The agent need to select the discrete action (run, hop, leap) and determine the corresponding continuous action (horizontal displacement) simultaneously to complete the task. The horizon of an episode is 20.
- ***Goal*** (Masson et al., 2016): The agent shoots the ball into the gate to win. Three types of hybrid actions are available to the agent including *kick-to(x,y), shoot-goal-left(h), shoot-goal-right(h)*. The continuous action parameters position (x, y) and position (h) along the

Layer	Actor Network ($\pi(s)$)	Critic Network ($Q(s,a)$ or $V(s)$)
Fully Connected	(state dim, 256)	(state dim + $\mathbb{R}^{K+\sum_k \lvert\mathcal{X}_k\rvert}$, 256) or (state dim + $\mathbb{R}^{\sum_k \lvert\mathcal{X}_k\rvert}$, 256) or (state dim, 256)
Activation	ReLU	ReLU
Fully Connected	(256, 256)	(256, 256)
Activation	ReLU	ReLU
Fully Connected	(256, $\mathbb{R}^K$) and (256, $\mathbb{R}^{\sum_k \lvert\mathcal{X}_k\rvert}$) or (256, $\mathbb{R}^{\sum_k \lvert\mathcal{X}_k\rvert}$)	(256, 1) or (256, $\mathbb{R}^K$)
Activation	tanh	None

Table 3: Network structures for the actor network and the critic network (Q-network or V-network).

goal line are quit different. Furthermore, We built a complex version of the goal environment, called ***Hard Goal***. We redefined the shot-goal action and split it into ten parameterized actions by dividing the goal line equidistantly. The continuous action parameters of each shot action will be mapped to a region in the goal line. The horizon of an episode is 50.

- ***Catch Point*** (Fan et al., 2019): The agent should catch the target point (orange) in limited opportunity (10 chances). There are two hybrid actions *move* and *catch*. Move is parameterized by a continuous action value which is a directional variable and catch is to try to catch the target point. The horizon of an episode is 20.
- ***Hard Move (designed by us)***: The agent needs to control n equally spaced actuators to reach target area (orange). Agent can choose whether each actuator should be on or off. Thus, the size of the action set is exponential in the number of actuators that is $\mathbf{2^n}$. Each actuator controls the moving distance in its own direction. n controls the scale of the action space. As n increases, the dimension of the action will increase. The horizon of an episode is 25.

B.2 Network Structure

Our PATD3 is implemented with reference to `github.com/sfujim/TD3` (TD3 source-code). PADDPG and PDQN are implemented with reference to `https://github.com/cycraig/MP-DQN`. For a fair comparison, all the baseline methods have the same network structure (except for the specific components to each algorithm) as our HyAR-TD3 implementation. For PDQN, PADDPG, we introduce a Passthrough Layer (Masson et al., 2016) to the actor networks to initialise their action-parameter policies to the same linear combination of state variables. HPPO paper does not provide open source-code and thus we implemented it by ourselves according to the guidance provided in their paper. For HPPO, the discrete actor and continuous actor do not share parameters (better than share parameters in our experiments).

As shown in Tab.3, we use a two-layer feed-forward neural network of 256 and 256 hidden units with ReLU activation (except for the output layer) for the actor network for all algorithms. For PADDPG, PDQN and HHQN, the critic denotes the Q-network. For HPPO, the critic denotes the V-network. Some algorithms (PATD3, PADDPG, HHQN) output two heads at the last layer of the actor network, one for discrete action and another for continuous action parameters.

The structure of HyAR is shown in Tab.4. We introduced element-wise product operation (Tang et al., 2021) and cascaded head structure (Azabou et al., 2021) to our HyAR model. More details about their effects are in Appendix C.3.

B.3 Hyperparameter

For all our experiments, we use the raw state and reward from the environment and no normalization or scaling are used. No regularization is used for the actor and the critic in all algorithms. An exploration noise sampled from $N(0, 0.1)$ (Fujimoto et al., 2018) is added to all baseline methods when select action. The discounted factor is 0.99 and we use Adam Optimizer (Kingma & Ba,

Model Component	Layer (Name)	Structure
Discrete Action Embedding Table E_ζ	Parameterized Table	($\mathbb{R}^{d_1}$, $\mathbb{R}^K$)
Conditional Encoder Network $q_\phi(z \mid x_k, s, e_{\zeta,k})$	Fully Connected (encoding)	($\mathbb{R}^{\mathcal{X}_k}$, 256)
	Fully Connected (condition)	(state dim + $\mathbb{R}^{d_1}$, 256)
	Element-wise Product	ReLU (encoding) · ReLU(condition)
	Fully Connected	(256, 256)
	Activation	ReLU
	Fully Connected (mean)	(256, $\mathbb{R}^{d_2}$)
	Activation	None
	Fully Connected (log_std)	(256, $\mathbb{R}^{d_2}$)
	Activation	None
Conditional Decoder Network $p_\psi(x_k \mid z, s, e_{\zeta,k})$	Fully Connected (latent)	($\mathbb{R}^{d_2}$, 256)
	Fully Connected (condition)	(state dim + $\mathbb{R}^{d_1}$, 256)
	Element-wise Product	ReLU(decoding) · ReLU(condition)
	Fully Connected	(256, 256)
	Activation	ReLU
	Fully Connected (reconstruction)	(256, $\mathbb{R}^{\mathcal{X}_k}$)
	Activation	None
	Fully Connected	(256, 256)
	Activation	ReLU
	Fully Connected (prediction)	(256, state dim)
	Activation	None

Table 4: Network structures for the hybrid action representation (HyAR) including, the discrete action embedding table and the conditional VAE.

2015) for all algorithms. Tab. 5 shows the common hyperparamters of algorithms used in all our experiments.

Hyperparameter	HPPO	PADDPG	PDQN	HHQN	PATD3	PDQN-TD3	HHQN-TD3	HyAR-DDPG	HyAR-TD3
Actor Learning Rate	$1{\cdot}10^{-4}$	$1{\cdot}10^{-4}$	$1{\cdot}10^{-4}$	$1{\cdot}10^{-4}$	$3{\cdot}10^{-4}$	$3{\cdot}10^{-4}$	$3{\cdot}10^{-4}$	$1{\cdot}10^{-4}$	$3{\cdot}10^{-4}$
Critic Learning Rate	$1{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$3{\cdot}10^{-4}$	$3{\cdot}10^{-4}$	$3{\cdot}10^{-4}$	$1{\cdot}10^{-3}$	$3{\cdot}10^{-4}$
Representation Model Learning Rate	-	-	-	-	-	-	-	$1{\cdot}10^{-4}$	$1{\cdot}10^{-4}$
Discount Factor	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Target Update Rate	-	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}
Tau Actor	-	$1{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$5{\cdot}10^{-3}$	$5{\cdot}10^{-3}$	$5{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$5{\cdot}10^{-3}$
Tau Critic	-	$1{\cdot}10^{-2}$	$1{\cdot}10^{-2}$	$1{\cdot}10^{-2}$	$5{\cdot}10^{-3}$	$5{\cdot}10^{-3}$	$5{\cdot}10^{-3}$	$5{\cdot}10^{-3}$	$5{\cdot}10^{-3}$
Exploration Policy	$\mathcal{N}(0, 0.1)$	$\mathcal{N}(0, 0.1)$	$\mathcal{N}(0, 0.1)$	$\mathcal{N}(0, 0.1)$	$\mathcal{N}(0, 0.1)$	$\mathcal{N}(0, 0.1)$	$\mathcal{N}(0, 0.1)$	$\mathcal{N}(0, 0.1)$	$\mathcal{N}(0, 0.1)$
Batch Size	128	128	128	128	128	128	128	128	128
Buffer Size	10^5	10^5	10^5	10^5	10^5	10^5	10^5	10^5	10^5
Actor Epoch	2	-	-	-	-	-	-	-	-
Critic Epoch	10	-	-	-	-	-	-	-	-

Table 5: A comparison of common hyperparameter choices of algorithms.We use '-' to denote the 'not applicable' situation.

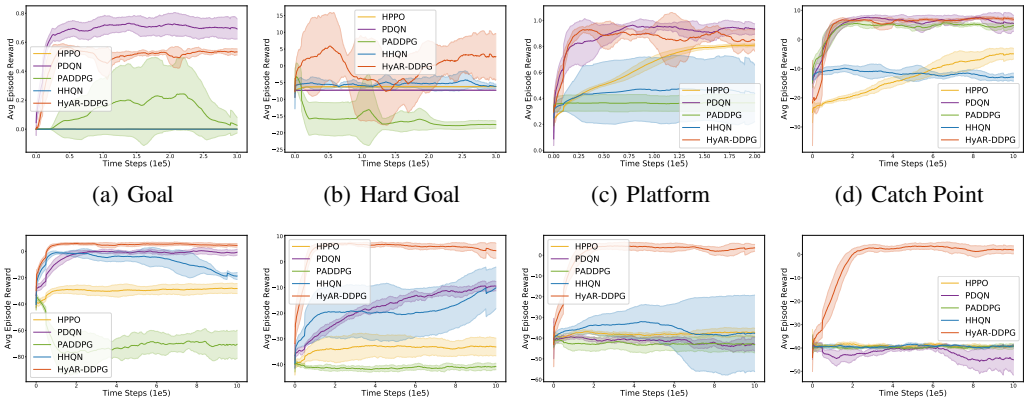
B.4 Additional Implementation Details

Training setup: For PPO, the actor network and the critic network are updated every 2 and 10 episodes respectively for all environment. The clip range of PPO algorithm is set to 0.2 and we use GAE (Schulman et al., 2016) for stable policy gradient. For DDPG-based, the actor network and the critic network is updated every 1 time step. For TD3-based, the critic network is updated every 1 time step and the actor network is updated every 2 time step.

For th the warm-up stage, we run 5000 episodes (empirically about 50k time steps depending on different environments) for experience collection and then pre-train the representation model (discrete action embedding table and conditional VAE) for 5000 iterations with batch size 64, after which we start the training of the latent policy. Note that the discrete action embedding table is initialized randomly before representation pre-training. The representation models (the embedding table and conditional VAE) are trained every 10 episodes for the rest of RL training. The latent action dim (discrete or continuous latent action) default value is 6. We set the KL weight in representation loss L_{VAE} as 0.5 and dynamics predictive representation loss weight β as 10 (default). More details about dynamics predictive representation loss weight are in C.2.

Algorithm 2: HyAR-DDPG

Initialize actor π_ω and critic networks Q_θ with random parameters ω, θ, and the corresponding target network parameters $\bar{\omega}, \bar{\theta}$
Initialize discrete action embedding table E_ζ and conditional VAE q_ϕ, p_ψ with random parameters ζ, ϕ, ψ
Prepare replay buffer $\mathcal{D}$ **repeat** Stage ❶
  Update ζ and ϕ, ψ using samples in $\mathcal{D}$ ▷ see Eq. 6
until *reaching maximum warm-up steps*;
repeat Stage ❷
  for $t \leftarrow 1$ *to* T **do**
    // select latent actions in representation space
    $\hat{z}_k, \hat{z}_x = \pi_\omega(s) + \epsilon_e$, with $\epsilon_e \sim \mathcal{N}(0, \sigma)$
    // decode into original hybrid actions
    $\hat{k} = g_E(\hat{z}_k), \hat{x}_k = p_\psi(\hat{z}_x, s, e_{\zeta,\hat{k}})$ ▷ see Eq. 3
    Execute $(\hat{k}, \hat{x}_k)$, observe r_t and new state s'
    Store $\{s, \hat{k}, \hat{x}_k, \hat{z}_k, \hat{z}_x, r, s'\}$ in $\mathcal{D}$
    Sample a mini-batch B of N experience from $\mathcal{D}$
    Update critic by minimizing empirical loss $\hat{L}_Q(\theta) = N^{-1} \sum_B \left(y - Q_\theta(s, \hat{z}_k, \hat{z}_x)\right)^2$, where $y = r + \gamma Q_{\bar{\theta}}\left(s', \pi_{\bar{\omega}}(s')\right)$
    Update actor by the deterministic policy gradient $\nabla_\omega J(\omega) = N^{-1} \sum_{s \in B} \left[\nabla_{\pi_\omega(s)} Q_\theta(s, \pi_\omega(s)) \nabla_\omega \pi_\omega(s)\right]$.
  repeat
    Update ζ and ϕ, ψ using samples in $\mathcal{D}$ ▷ see Eq. 6
  until *reaching maximum representation training steps*;
until *reaching maximum training steps*;



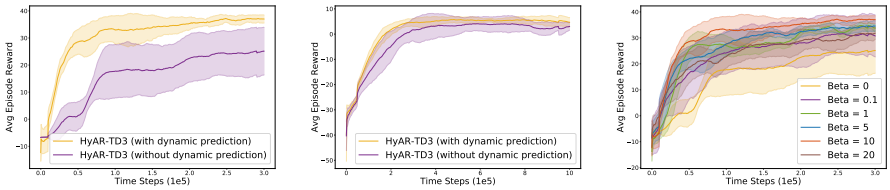
(a) Goal (b) Hard Goal (c) Platform (d) Catch Point

(e) Hard Move (n = 4) (f) Hard Move (n = 6) (g) Hard Move (n = 8) (h) Hard Move (n = 10)

Figure 7: DDPG-based comparisons of related baselines on different environments. The x- and y-axis denote the learning steps ($\times 10^5$) and averaged reward over the recent 100 episodes. The results are averaged using 5 runs, while the solid line and shaded represent the mean value and a standard deviation, respectively.

B.5 DDPG-BASED HYAR ALGORITHM

Additionally, we implemented HyAR with DDPG (Lillicrap et al., 2015), called HyAR-DDPG. The pseudo-code of complete algorithm is shown in Algorithm 2. Results of DDPG-based experimental comparisons can be found in Appendix C.1.



(a) Goal (b) Hard Move (n = 8) (c) Goal

Figure 8: Learning curves of dynamics predictive representation for HyAR. The results are averaged using 5 runs, while the solid line and shaded represent the mean value and a standard deviation, respectively.

C Complete Learning Curves and Additional Experiments

C.1 Learning Curves for DDPG-based Comparisons

Fig. 7 visualizes the learning curves of DDPG-based comparisons, where HyAR-DDPG outperforms other baselines in both the final performance and learning speed in most environments. Besides the learning speed, HyAR-DDPG also achieves the best generalization as HyAR-TD3 across different environments. When the environments become complex (shown in Fig. 7(e-h)), HyAR-DDPG still achieves steady and better performance than the others, particularly demonstrating the effectiveness and generalization of HyAR in high-dimensional hybrid action spaces.

C.2 Learning Curves for the Dynamics Predictive Representation

Fig. 8 shows the learning curves of HyAR-TD3 with dynamics predictive representation loss (Fig. 8(a-b)) and the influence of dynamics predictive representation loss weight β on algorithm performance (Fig. 8(c)). We can easily find that the representation learned by dynamics predictive representation loss is better than without dynamics predictive representation loss. For the weight β of dynamics predictive representation loss, the performance of the algorithm will gradually improve with the increase of weight β, until a certain threshold is reached. We can conclude that the dynamics predictive representation loss is helpful for deriving an environment-awareness representation for further improving the learning performance, efficacy, and stability. More experiments on representation visualization are in Appendix C.4.

C.3 Learning Curves and Table for the Results in Ablation Study

As briefly discussed in Sec. 5.3, we conduct detailed ablation experiments on the key components of the algorithm, including:

- element-wise product (Tang et al., 2021) (v.s. concat) operation;
- cascaded head (Azabou et al., 2021) (v.s. parallel head) structure;
- latent select range(from 80% to 100%), corresponding to latent space constraint (LSC);
- action representation relabeling, corresponding to representation shift correction (RSC);
- latent action dim (from 3 to 12);

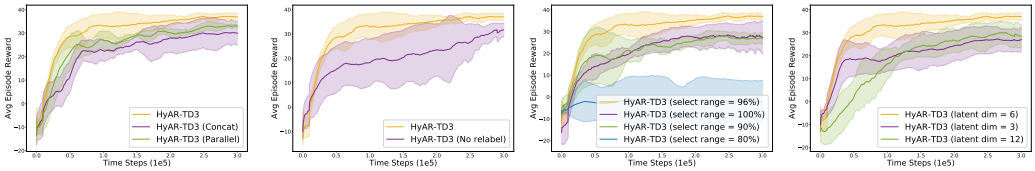
Fig. 9 shows the learning curves of HyAR-TD3 and its variants for ablation studies, corresponding to the results in Tab. 6.

First, we can observe that element-wise product achieves better performance than concatenation (Fig. 9(a,e)). As similarly discovered in (Tang et al., 2021), we hypothesize that the explicit relation between the condition and representation imposed by element wise product forces the conditional VAE to learn more effective hidden features. Second, the significance of cascaded head is demonstrated by its superior performance over parallel head (Fig. 9(a,e)) which means cascaded head can better output two different features. Third, representation relabeling shows an apparent improvement (Fig. 9(b,f)) which show that representation shift leads to data invalidation in the experience

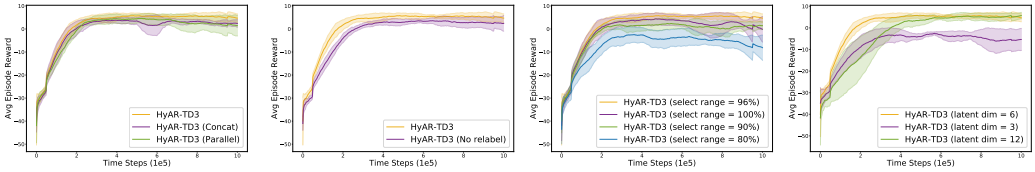
Operation		Structure						Result	
Elem.-Wise Prod.	Concat.	Cascaded	Parallel	Latent Select Range c	Latent Action Dim	Relabeling	Dynamics Predictive	Results (Goal)	Results (Hard Move)
✓		✓		96% (n = 2)	6	✓	✓	**0.78** ± **0.03**	0.89 ± 0.03
	✓	✓		96% (n = 2)	6	✓	✓	0.66± 0.10	0.83± 0.04
✓			✓	96% (n = 2)	6	✓	✓	0.71 ± 0.04	0.80± 0.13
✓		✓		96% (n = 2)	6		✓	0.66 ± 0.07	0.83± 0.08
✓		✓		100% (n = 0)	6	✓	✓	0.62 ± 0.11	0.78± 0.13
✓		✓		90% (n = 5)	6	✓	✓	0.61 ± 0.04	0.78± 0.08
✓		✓		80% (n = 10)	6	✓	✓	0.08 ± 0.17	0.56± 0.12
✓		✓		96% (n = 2)	3	✓	✓	0.59 ± 0.09	0.58± 0.16
✓		✓		96% (n = 2)	12	✓	✓	0.65 ± 0.09	**0.90** ± **0.04**
✓		✓		96% (n = 2)	6	✓		0.55 ± 0.15	0.84 ± 0.05

Table 6: Ablation of our method across each contribution in Goal and Hard Move (n = 8) environment.Results are max Average Episode performance over 5 trials. ± corresponds to a standard deviation.

buffer which will affect RL training. Fourth, a reasonable latent select range plays an important role in algorithm learning (Fig. 9(c,g)). Only constrain the action representation space of the latent policy inside a reasonable area (both large and small will fail), can the algorithm learn effectively and reliably. These experimental results supports our analysis above. We also analyse the influence of latent action dim for RL (Fig. 9(d,h)). In the low-dimensional hybrid action environment, we should choose a moderate value (e.g., 6). While for high-dimensional environment, larger value may be better (e.g., 12).



(a) Operator and Structure (b) Representation relabeling (c) Latent select range (d) Latent action dim



(e) Operator and Structure (f) Representation relabeling (g) Latent select range (h) Latent action dim

Figure 9: Learning curves of ablation studies for HyAR (i.e., element-wise + cascaded head + representation relabeling + latent select range = 96% + latent action dim = 6). From top to bottom is Goal and Hard move (n = 8) environment. The shaded region denotes standard deviation of average evaluation over 5 trials.

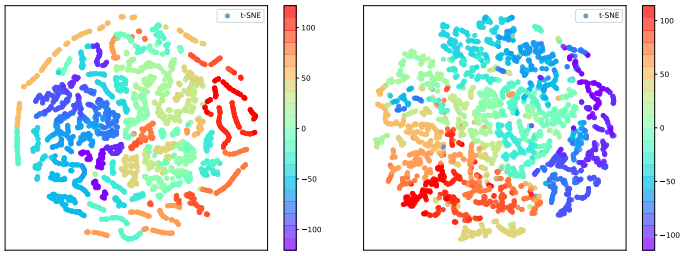
C.4 Representation Visual Analysis

In order to further analyze the hybrid action representation, we visualize the learned hybrid action representations. Fig. 10 and Fig. 11 shows the t-SNE visualization for HyAR in Goal and Hard Move (n = 8) environment.

As we can see from Fig. 10, we adopt t-SNE to cluster the latent continuous actions, i.e., (z_x), outputted by the latent policy, and color each action based on latent discrete actions i.e., (z_k). We can conclude that latent continuous actions can be clustered by latent discrete actions, but there are multiple modes in the global range. Our dependence-aware representation model makes good use of this relationship that the choice of continuous action parameters is depend on discrete actions.

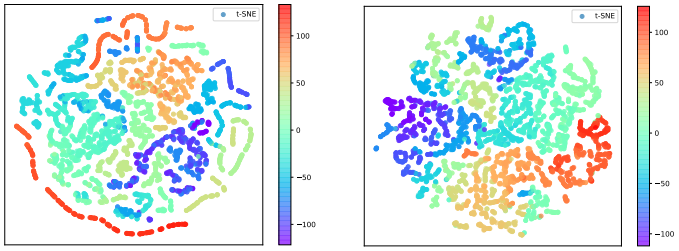
For the dynamics predictive representation loss, we adopt t-SNE to cluster the latent actions, i.e., (z_k, z_x), outputted by the latent policy, and color each action based on its impact on the environment (i.e., $\delta_{s,s'}$). As shown in Fig. 11, we observe that actions with a similar impact on the environment

are relatively closer in the latent space. This demonstrates the dynamics predictive representation loss is helpful for deriving an environment-awareness representation for further improving the learning performance, efficacy, and stability.



(a) Goal (b) Hard Move (n = 8)

Figure 10: t-SNE visualization diagram of continuous action embedding z_x, color coded by discrete action embedding z_k. The continuous actions related to the same discrete actions are mapped to the similar regions of the representation space.



(a) Goal (b) Hard Move (n = 8)

Figure 11: t-SNE visualization diagram of hybrid action embedding pair (z_k, z_x), color coded by $\delta_{s,s'}$. The hybrid actions with a similar impact on the environment are relatively closer in the latent space.