SKILL EXPANSION AND COMPOSITION IN PARAMETER SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Humans excel at reusing prior knowledge to address new challenges and developing skills while solving problems. This paradigm becomes increasingly popular in the development of autonomous agents, as it develops systems that can self-evolve in response to new challenges like human beings. However, previous methods suffer from limited training efficiency when expanding new skills and fail to fully leverage prior knowledge to facilitate new task learning. We propose Parametric Skill Expansion and Composition (PSEC), a new framework designed to iteratively evolve the agents' capabilities and efficiently address new challenges by maintaining a manageable skill library. This library can progressively integrate skill primitives as plug-and-play Low-Rank Adaptation (LoRA) modules in parameter-efficient finetuning, facilitating efficient and flexible skill expansion. This structure also enables the direct skill compositions in parameter space by merging LoRA modules that encode different skills, leveraging shared information across skills to effectively program new skills. Based on this, we propose a context-aware modular to dynamically activate different skills to collaboratively handle new tasks. Empowering diverse applications including multi-objective composition, dynamics shift, and continual policy shift, the results on D4RL, DSRL benchmarks, and the DeepMind Control Suite show that PSEC exhibits superior capacity to leverage prior knowledge to efficiently tackle new challenges, as well as expand its skill libraries to evolve the capabilities.

1 INTRODUCTION

Humans excel at leveraging existing skills and knowledge to efficiently tackle new tasks while continually evolving their capabilities to rapidly adapt to new tasks. (Driscoll et al., 2024; Courellis et al., 2024; Eppe et al., 2022; Eichenbaum, 2017). For instance, a child can rapidly learn to recognizing a tiger by integrating prior experiences of recognizing a cat, and subsequently, combines these knowledge to adapt to recognizing a lion. This fundamental approach to problem-solving highlights a key aspect of human intelligence that is equally crucial for autonomous agents. However, most current decision-making algorithms adhere to a *tabula rasa* paradigm, where they are trained from scratch without utilizing any prior knowledge or resources (Akkaya et al., 2019; Berner et al., 2019; Silver et al., 2016), leading to severe sample inefficiency and elevated cost when the agent encounters new tasks (Agarwal et al., 2022; Peng et al., 2019; Du & Kaelbling, 2024). Therefore, in this paper, we aim to explore the capability of autonomous agents to leverage and expand upon their existing knowledge base in novel situations to enhance learning efficiency and adaptability.

While some existing studies, such as continual learning (Liu et al., 2024; Gai & Wang, 2024), compositional policies (Peng et al., 2019; Janner et al., 2022; Ajay et al., 2023), or finetuning-based methods (Agarwal et al., 2022), aim to replicate this process, they jointly failed to tackle several key limitations. *1) Catastrophic forgetting*: these approaches typically lack a fundamental mechanism to guarantee continuous improvement when acquiring new skills, making the autonomous agents very susceptible to overfitting on new tasks while forgetting previously learned skills without proper regularization (Liu et al., 2023c; 2024; Gai & Wang, 2024); *2) Limited efficiency in learning new tasks*: Some methods avoid the catastrophic forgetting problem by adopting a parameter-isolation approach via encoding new skills in independent new parameters. However, they typically do not fully utilize prior knowledge from old skills to enhance training in current tasks, lacking an efficient way to learn new skills in terms of both parameters and training samples (Peng et al., 2019; Zhang et al., 2023a), leading to tremendous costs as the number of skills progressively grows.

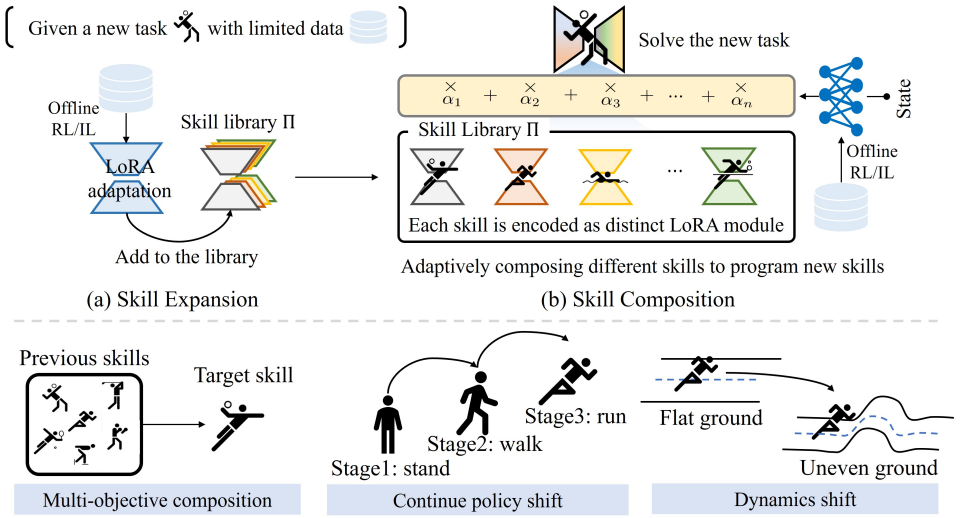


Figure 1: PSEC framework and its application in diverse scenarios. (a) We maintain a skill library that contains many skills primitives and can progressively expand by adding new LoRA modules. (b) Then we train a context-aware compositional network to adaptively compose different elements in the skill library to solve new tasks. (c) PSEC framework is versatile to diverse applications where reusing prior knowledge is crucial.

We propose Parametric Skill Expansion and Composition (PSEC), a framework that facilitates efficient self-evolution of autonomous agents by maintaining a skill library that progressively integrates new skills, facilitating rapid adaptation to evolving demands. The key insight is we can efficiently manage the primitives in this library to tackle new challenges by exploiting the shared information across different skills within the parameter space. As shown in Figure 1 (a), we adopt the Low-Rank Adaptation (LoRA) (Hu et al., 2021) approach, which encodes skills as trainable parameters injected into existing frozen layers. This parameter-isolation approach naturally resolves the catastrophic forgetting problem, and significantly reduces computational burden thanks to the low-rank decomposition structure. This efficient modular design allows for managing skills as plug-and-play modules, and thus can directly blend different abilities within the parameter space to interpolate new skills (Clark et al., 2024), as shown in Figure 1 (b). This approach can leverage more shared or complementary structures across skills for optimal compositions. Based on this insight, we propose a context-aware modular to adaptively compose skills and each primitive is modeled by diffusion models to ensure both flexibility and expressiveness in composition. Through endless expansion and composition, PSEC forms an iterative framework that can continually evolve and efficiently tackle new challenges, offering one promising pathway for developing human-level autonomous agents.

Empowering diverse settings including multi-objective composition, continue policy shift and dynamics shift, PSEC demonstrates its capacity to evolve and effectively solve new tasks by leveraging prior knowledge, evaluated on the D4RL (Fu et al., 2020), DSRL (Liu et al., 2023a) and DeepMind Control Suite (Tassa et al., 2018), showcasing significant potential for real-world applications.

2 Related Works

Tabula Rasa. Tabula rasa learning is one popular paradigm for diverse existing decision-making applications, such as robotics and games (Silver et al., 2017; Andrychowicz et al., 2020; Berner et al., 2019; Vinyals et al., 2019). It directly learns policies from scratch without the assistance of any prior knowledge. However, it suffers from notable drawbacks related to poor sample efficiency and constraints on the complexity of skills an agent can acquire (Agarwal et al., 2022).

Compositional Policies. Some previous methods try to leverage prior knowledge relying on pre-trained primitive policies. More specifically, they propose a compositional network in a hierarchical structure to adaptively compose primitives to form complex behaviors (Peng et al., 2019; Qureshi et al., 2020; Pertsch et al., 2021; Merel et al., 2019; 2020). However, their expressiveness is limited by the expressiveness of simple Gaussian primitives. Recently, due to the strong expressiveness of the diffusion model and its inherent connection with Energy-Based Models (LeCun et al., 2006), many compositional policies have been approached by diffusion model. Diffusion models learn the

gradient fields of an implicit energy function, which can be combined at inference time to generalize to new complex distribution readily (Janner et al., 2022; Wang et al., 2024a; Du & Kaelbling, 2024; Liu et al., 2022; Luo et al., 2024b). However, these approaches rely on independently trained policies with fixed combination weights, which lack the flexibility to adapt to complex scenarios. Moreover, most previous methods can only combine skills after the policy distribution generation of each skill. Therefore, they fail to fully utilize the shared features of different skills to achieve optimal compositions. We systematically investigate the advantages of skill composition within the parameter space, and compose skills in a context-aware manner with each skill modeled as a diffusion model. This ensures both flexibility and expressiveness in composing complex behaviors.

Continual Learning for Decision Making. Current continual learning methods for decision making, including continual reinforcement learning (RL) and imitation learning (IL), primarily focus on mitigating catastrophic forgetting of prior knowledge when learning new tasks. They can be roughly classified into three categories: structure-based (Smith et al., 2023; Wang et al., 2024c), regularization-based (Kessler et al., 2020), and rehearsal-based methods (Liu et al., 2024; Peng et al., 2023). Our study capitalizes on leveraging existing skills to facilitate efficient new task learning and enables the extension of skill sets. In addition, it naturally solves the catastrophic forgetting challenge thanks to the parameter isolation induced by the LoRA module (Liu et al., 2023c), directly bypassing the key challenges of existing continual learning methods.

Finetune-based Methods. Some finetune-based methods aim to accelerate policy learning by leveraging prior knowledge. This knowledge may come from pretrained policy or offline data, such as Offline-to-online RL (Nair et al., 2020; Lee et al., 2022b; Agarwal et al., 2022) and transfer RL (Barreto et al., 2018; Li et al., 2019). Some methods maintain a policy library that contains pretrained policies and adaptively selects one policy from this set to assist policy training (Kim et al., 2024; Wang et al., 2024b; Barreto et al., 2018). However, they are generally restricted to single-task scenarios where all policies serve the same task (Zhang et al., 2023a), or only sequentially activate one policy in the pretrained sets, which greatly limits the expressiveness of the pretrained primitives (Li et al., 2019). Our method, on the contrary, can both leverage multi-task knowledge to fulfill the new task, and can simultaneously activate all skills to compose more complex behaviors.

3 METHODS

We propose PSEC, a generic framework that can efficiently reuse prior knowledge and self-evolve to address emerging new tasks. Next, we will elaborate on our problem setup and technical details.

3.1 PRELIMINARY

Diffusion Model for Policy Modeling. Recently, diffusion model has become popular for policy modeling because of its superior expressiveness to model complex distributions (Wang et al., 2023; Chen et al., 2022; Lu et al., 2023; Zheng et al., 2024). Considering a policy distribution $\pi(a|s)$ and a sample (s, a) drawn from an empirical dataset $\mathcal{D}$ of $\pi(a|s)$, the diffusion process (Ho et al., 2020) progressively introduces Gaussian noise to the sample over T steps, producing a sequence of noisy samples $a_0, a_1, ..., a_T$ with $a_0 = a$ following the forward Gaussian kernel:

$$q(a_t|a_{t-1}) = \mathcal{N}(a_t; \sqrt{1-\beta_t}a_{t-1}, \beta_t I), \quad q(a_t|a_0) = \mathcal{N}(a_t; \sqrt{\bar{\rho}_t}a_0, (1-\bar{\rho}_t)I), \tag{1}$$

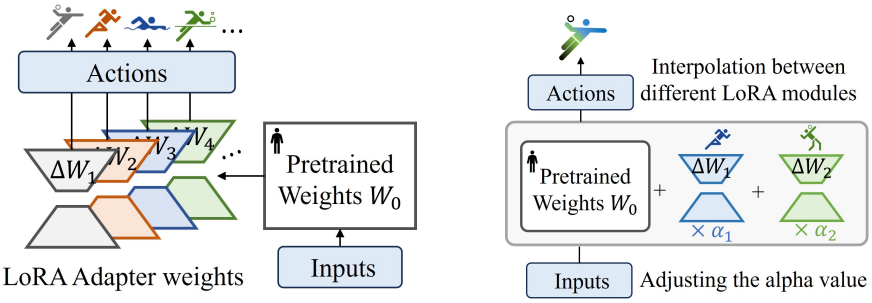
where $\rho_t := 1 - \beta_t, \bar{\rho}_t = \prod_{t=1}^{t} \rho_t$, and the noise is controlled by a variance schedule $\beta_1, ..., \beta_t$ to ensure $p(a_T) = \mathcal{N}(0, I)$. The denoise process aims to recover the sample from $p(a_T)$ by learning a conditional distribution $p_\theta(a_{t-1}|a_t, s)$. The policy $\pi_\theta(a|s)$ is typically modeled as:

$$\pi_\theta(a|s) = p(a_t) \prod_{t=1}^{T} p_\theta(a_{t-1}|a_t, s); p_\theta(a_{t-1}|a_t, s) = \mathcal{N}(a_{t-1}; \mu_\theta(a_t, t, s), \Sigma_\theta(a_t, t, s)), \tag{2}$$

where $\Sigma_\theta = \beta_t I$ is set as untrained time-dependent constants and $\mu_\theta(a_t, t, s) = \frac{1}{\sqrt{\rho_t}}(a_t - \frac{\beta_t}{\sqrt{1-\bar{\rho}_t}}\epsilon_\theta(a_t, t, s))$ is reparameterized by ϵ_θ. The trainable parameter θ, modeled by deep networks, can be optimized via minimizing the following objective by predicting the noise:

$$\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{t\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}}\left[w(s,a)\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\rho}_t}a + \sqrt{1-\bar{\rho}_t}\epsilon, t, s\right)\right\|^2\right]. \tag{3}$$

where $\mathcal{U}$ is uniform distribution over the discrete set $\{1, ..., T\}$. $w(s, a)$ is a flexible weight function that encodes human preference (Zheng et al., 2024). For example, $w(s, a) \propto f(A(s, a)), f \geq 0$



(a) Learning new skills using LoRA modules. (b) Interpolation in LoRA modules.

Figure 2: (a) Each skill is encoded in separate LoRA modules respectively. (b) By adjusting the composing weights α_i, different LoRA modules can merge together to interpolate new skills.

with $A(s, a)$ as the advantage function leads to weighted behavior cloning (BC) in offline reinforcement learning (RL) (Zheng et al., 2024; Kostrikov et al., 2022; Xu et al., 2023), and $w(s, a) := 1$ degenerates to traditional BC (Chen et al., 2023). After obtaining the approximated μ_θ and Σ_θ, we can substitute them into Eq. (2) to iteratively denoise and obtain actions conditioned on the state.

Problem Setups. We consider a Markov Decision Process with $s \in \mathcal{S}$ and $a \in \mathcal{A}$ are state and action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is transition dynamics, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is reward function. We assume the state space $\mathcal{S}$ and action space $\mathcal{A}$ remain unchanged during training, which is a mild assumption in many relevant works (Peng et al., 2019; Ajay et al., 2023; Nair et al., 2020; Luo et al., 2024a). We consider an agent with π_0 as its initial policy and then is progressively tasked with new tasks $\mathcal{T}_i, i = 1, 2, ...$, with differences in the rewards r or dynamics $\mathcal{P}$, to mirror real-world scenarios with non-stationary dynamics or new challenges continually emerge (Luo et al., 2024a). Each task is provided with several expert demonstrations $\mathcal{D}_e^{\mathcal{T}_i} := \{(s, a)\}$ or a mixed-quality dataset with reward labels $\mathcal{D}_o^{\mathcal{T}_i} := \{(s, a, r_i, s')\}$. So, we can use either offline RL or imitation learning (IL) to adapt to the new challenges. Inspired by previous works (Peng et al., 2019; Barreto et al., 2018; Zhang et al., 2023a), we maintain a policy library Π to store the policies associated with different tasks and aim to utilize the prior knowledge to enable efficient policy learning and gradually expand it to incorporate new abilities across training.

$$\Pi = \{\pi_0, \pi_1, \pi_2, \pi_3, ...\}. \tag{4}$$

We aim to explore *1) Efficient Expansion*: How to manage the skill library Π to learn new skills in an efficient and manageable way, and *2) Efficient Composition*: How to fully utilize the prior knowledge from primitives in the skill set Π to tackle the emerging challenges.

3.2 Efficient Policy Expansion via Low Rank Adaptation

For the first objective, previous methods typically train each primitive from scratch in a tabula rasa paradigm (Peng et al., 2019; Janner et al., 2022; Lu et al., 2023), failed to leverage the prior knowledge in Π to efficiently obtain a good skill primitive. This presents significant issues in terms of computational efficiency when the number of skills grows. To mitigate these challenges, we turn to Parameter-Efficient Fine-Tuning (PEFT) (Ding et al., 2023), which has proven highly effective in various natural language processing and computer vision applications. One of the most popular PEFT implementations is LoRA (Hu et al., 2022). It injects trainable low-rank decomposed matrices into the pretrained layer to avoid overfitting with limited adaptation data and significantly reduces computational and memory burden. Inspired by this, we try to employ LoRA to efficiently learn new skills given solely limited data for the target skill.

Policy Expansion via Low-Rank Adaptation. We consider a pretrained policy π_0 and denote $W_0 \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ as its associated weight matrix. Directly finetuning W_0 to adapt to new skills might be extremely inefficient (Liu et al., 2023c), instead, we introduce a tune-able LoRA module ΔW upon W_0, *i.e.*, $W_0 + \Delta W = W_0 + BA$ to do the adaptation and keep W_0 frozen, where $B \in \mathbb{R}^{d_{\text{in}} \times n}$, $A \in \mathbb{R}^{n \times d_{\text{out}}}$ and $n \ll \min(d_{\text{in}}, d_{\text{out}})$. Specifically, the input feature of the linear layer is denoted as $h_{\text{in}} \in \mathbb{R}^{d_{\text{in}}}$, and the output feature of the linear layer is $h_{\text{out}} \in \mathbb{R}^{d_{\text{out}}}$, the final output of a LoRA augmented layer can be calculated through the following forward process:

$$h_{\text{out}} = (W_0 + \alpha \Delta W) h_{\text{in}} = (W_0 + \alpha BA) h_{\text{in}} = W_0 h_{\text{in}} + \alpha BA h_{\text{in}}, \tag{5}$$

where α is a weight to balance the pre-trained model and LoRA modules. This operation naturally prevents catastrophic forgetting in a parameter isolation approach, and the low-rank decomposition structure of A and B significantly reduces the computational burden. Benefiting from this lightweight characteristic, we can manage numerous LoRA modules $\{\Delta W_i = B_i A_i | i \in 1, 2, ..., k\}$ to encode different skill primitives π_i, respectively, as shown in Figure 2a. This flexible approach allows us to easily integrate new skills based on existing knowledge, while also facilitating library management by removing suboptimal primitives and retaining the effective ones. More importantly, by adjusting the value of α, it holds the potential to interpolate the pretrained skill in W_0 and other primitives in ΔW_i (Clark et al., 2024) to generate novel skills, as shown in Eq. (6) and Figure 2b.

$$W = W_0 + \sum_{i=1}^{k} \alpha_i \Delta W_i = W_0 + \sum_{i=1}^{k} \alpha_i B_i A_i, \tag{6}$$

where α_i is the weight to interpolate pre-trained weights and LoRA modules. This interpolation property has been explored in fields like text-to-image generation (Clark et al., 2024) and language modeling (Zhang et al., 2023b), but its application in decision-making scenarios remains highly underexplored, despite LoRA has proven efficacy in skill acquisition (Liu et al., 2023c). Next, we will elaborate on how to effectively combine LoRA modules to adapt to decision-making applications.

3.3 Context-aware composition in parameter space

Effectively combining skills encoded as different LoRA modules to solve new tasks is crucial. Previous methods (Du & Kaelbling, 2024; Ajay et al., 2023; Janner et al., 2022) typically rely on fixed combinations of skills, resulting in limited compositional flexibility. This approach may be acceptable in static domains like language models, but it falls short in decision-making applications where dynamic skill composition is crucial. For example, in autonomous driving, the ability to dynamically prioritize skills of obstacle avoidance in potential collision scenarios, or acceleration when speeds are suboptimal, is essential. Naively adopting a fixed set of α_i like previous approaches (Du & Kaelbling, 2024; Ajay et al., 2023; Janner et al., 2022; Clark et al., 2024), however, cannot adequately support such flexible deployment of skills based on real-time environmental demands.

Context-aware Composition. We propose a simple yet effective context-aware composition method that adaptively leverages pretrained knowledge to optimally address the encountering tasks according to the agent's current context. Specifically, we introduce a context-aware modular $\alpha(s;\theta) \in \mathbb{R}^k$ with α_i as its i-th dimension. The composition method can be expressed by Eq. (7):

$$W(\theta) = W_0 + \sum_{i=1}^{k} \alpha_i(s;\theta) \Delta W_i = W_0 + \sum_{i=1}^{k} \alpha_i(s;\theta) B_i A_i. \tag{7}$$

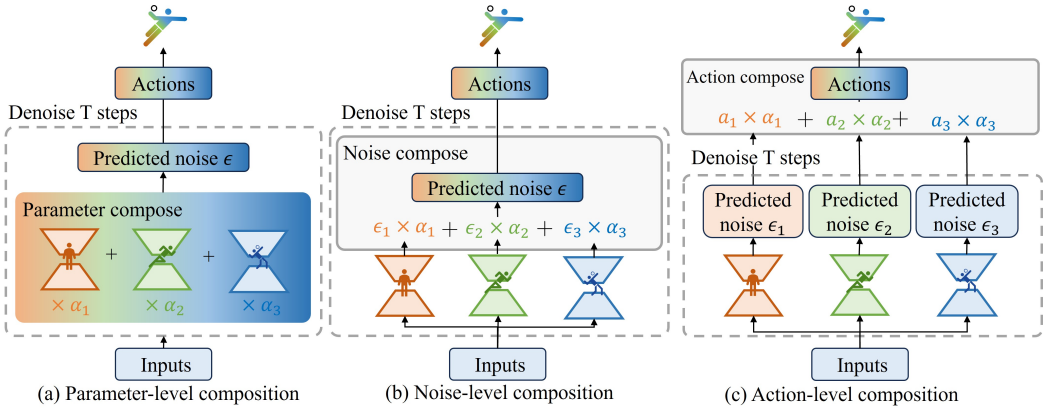
Here, $\alpha(s;\theta)$ adaptively adjusts output weights based on the agent's current situation s with the parameter θ optimized via minimizing the diffusion loss in Eq. (3). Note that the trainable parameter θ lies solely in the composition network α_θ with the pretrained weights W_0 and all LoRA modules ΔW_i being kept frozen, thus θ can be efficiently trained in terms of both samples and parameters.

Parameter-level *v.s.* Action-level Composition. Careful readers may notice that our context-aware composition is similar to previous works that adaptively compose Gaussian primitive skills to create complex behaviors (Peng et al., 2019; Qureshi et al., 2020), such as the one shown in Eq. (8) (Peng et al., 2019):

$$\pi(a|s) = \frac{1}{Z(s)} \prod_{i=1}^{k} \pi_i(a|s)^{\alpha_i(s;\theta)}, \quad \pi_i(a|s) = \mathcal{N}\left(\mu_i(s), \Sigma_i(s)\right), \tag{8}$$

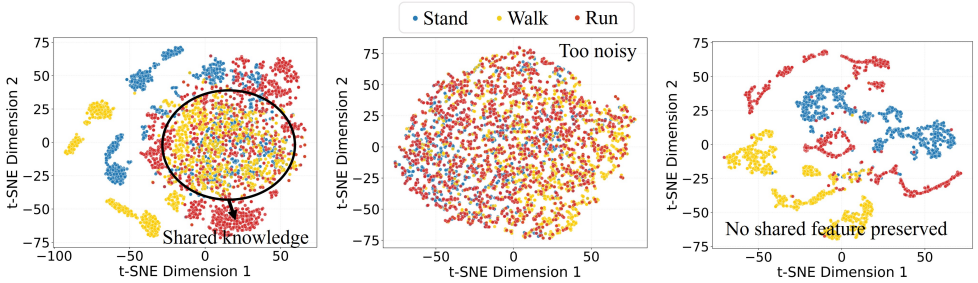
where $\alpha(s;\theta)$ is optimized to combine the policy distributions $\pi_i, i = 0, ..., k$ to collaboratively build a new policy distribution π to solve the new task.

However, these two methods differ fundamentally in their stages of composition, mirroring the advantages of *early fusion* over *late fusion* across various domains (Gadzicki et al., 2020; Wang et al., 2024d). PSEC employs a *parameter-level composition*, where different skills are seamlessly integrated within the parameter space. By contrast, Eq. (8) represents an *action-level composition* that explicitly combines the output distributions of various skills. In comparison, *parameter-level composition* will be more efficient, as it can leverage more shared or complementary information



(a) Parameter-level composition (b) Noise-level composition (c) Action-level composition

Figure 3: Comparison between parameter-, noise-, and action-level composition. Parameter-level composition offers more flexibility to leverage the shared or complementary structure across skills to compose new skills. Noise- and action-level composition, however, is too late to benefit from this information.



(a) Parameter space (b) Noise space (c) Action space

Figure 4: t-SNE projections of samples from different skills in parameter, noise, and action space. The parameter space exhibits a good structure for skill composition, where skills share common knowledge while retaining their unique features to avoid confusion. Noise and action spaces are either too noisy to clearly distinguish between skills or fail to capture the shared structure across them. See Appendix C.4 for details.

between different skills to enhance compositionality and overall performance before generating the final policy distribution (Shazeer et al., 2016; Wang et al., 2024c). Conversely, *action-level composition* only merges skills after the action generation, which is too late to effectively leverage features across skills for optimal composition. Besides, previous *action-level* methods typically employ simple Gaussian primitives to construct their skill library, significantly limiting its expressiveness.

Parameter-level *v.s.* Noise-level Composition. Some approaches use diffusion models for policy modeling and exhibit remarkable compositionality by identifying its connections to Energy-Based Models (Du & Kaelbling, 2024; Wang et al., 2024a; Janner et al., 2022; Ajay et al., 2023; Lu et al., 2023). Specifically, the noise predicted by diffusion models can be regarded as the gradient field of some energy functions. It thus can be directly merged to form new skills during sampling in a *noise-level composition*, as shown in Eq. (9). This is equivalent to doing a logical operation on the energy functions to form complex behaviors (Du et al., 2023; Liu et al., 2022; LeCun et al., 2006).

$$\epsilon(a_t, t, s) = \sum_{i=0}^{k} \alpha_i \epsilon_i(a_t, t, s). \tag{9}$$

Here, ϵ_i represents the predicted noise derived from various skills, while ϵ is the aggregated noise resulting from their composition. Utilizing ϵ for denoising in Eq. (2) allows for the generation of a joint distribution of skills, thereby facilitating the effective composition of these diverse capabilities (Ajay et al., 2023; 2024; Janner et al., 2022). However, these methods employ fixed weights α_i for policy composition, limiting their flexibility and adaptability in dynamical scenarios where real-time adjustment on the compositional weights is required. In our paper, PSEC not only employs diffusion models to enhance the expressiveness of primitives, but also adaptively adjusts the context-aware compositional weights to enhance compositional flexibility. Additionally, this *noise-level composition* also tends to be less effective than *parameter-level composition*, as the latter integrates different skills at an earlier stage, leading to improved performance, as shown in Figure 3.

Table 1: Normalized DSRL (Liu et al., 2023a) benchmark results. Costs below 1 indicates safety. ↑: the higher the better. ↓: the lower the better. Results are averaged over 20 evaluation episodes and 4 seeds. **Bold**: Safe agents with costs below 1. **Blue**: Safe agents achieving the highest reward.

Task	BC		CDT		CPQ		COptiDICE		FISOR		ASEC		NSEC		PSEC	
	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
easysparse	0.32	4.73	**0.05**	**0.10**	**-0.06**	**0.24**	0.94	18.21	**0.38**	**0.53**	0.95	5.8	**0.55**	**0.08**	**0.55**	**0.02**
easymean	0.22	2.68	**0.27**	**0.24**	**-0.06**	**0.24**	0.74	14.81	**0.38**	**0.25**	**0.63**	**0.75**	**0.39**	**0.54**	**0.37**	**0.00**
easydense	0.20	1.70	0.43	2.31	**-0.06**	**0.29**	0.60	11.27	**0.36**	**0.25**	0.85	5.28	0.76	1.45	**0.51**	**0.01**
mediumsparse	0.53	1.74	0.26	2.20	**-0.08**	**0.18**	0.64	7.26	**0.42**	**0.22**	0.93	2.52	**0.60**	**0.08**	**0.76**	**0.03**
mediummean	0.66	2.94	0.28	2.13	**-0.08**	**0.28**	0.73	8.35	**0.39**	**0.08**	0.74	1.00	0.82	2.87	**0.61**	**0.01**
mediumdense	0.65	3.79	**0.29**	**0.77**	**-0.08**	**0.20**	0.91	9.52	**0.49**	**0.44**	**0.81**	**0.52**	**0.76**	**0.27**	**0.66**	**0.02**
hardsparse	0.28	1.98	**0.17**	**0.47**	**-0.04**	**0.28**	0.34	7.34	**0.30**	**0.01**	**0.30**	**0.41**	0.34	1.21	**0.34**	**0.04**
hardmean	0.34	3.76	0.28	3.32	**-0.05**	**0.24**	0.36	7.51	**0.26**	**0.09**	0.46	1.05	**0.38**	**0.32**	**0.39**	**0.07**
harddense	0.40	5.57	0.24	1.49	**-0.04**	**0.24**	0.42	8.11	**0.30**	**0.34**	**0.36**	**0.82**	**0.19**	**0.03**	**0.34**	**0.07**
MetaDrive Average	0.40	3.21	0.25	1.45	**-0.06**	**0.24**	0.63	10.26	**0.36**	**0.25**	0.67	2.02	**0.53**	**0.76**	**0.50**	**0.03**
AntRun	0.73	11.73	0.70	1.88	**0.00**	**0.00**	0.62	3.64	**0.45**	**0.03**	0.74	4.97	0.79	6.81	**0.59**	**0.33**
BallRun	0.67	11.38	**0.32**	**0.45**	0.85	13.67	0.55	11.32	**0.18**	**0.00**	0.35	4.35	0.58	7.46	**0.15**	**0.95**
CarRun	0.96	1.88	0.99	1.10	1.06	10.49	**0.92**	**0.00**	**0.73**	**0.14**	**0.93**	**0.39**	**0.93**	**0.66**	**0.83**	**0.00**
DroneRun	0.55	5.21	**0.58**	**0.30**	0.02	7.95	0.72	13.77	**0.30**	**0.55**	0.57	2.29	0.62	7.3	**0.47**	**0.87**
AntCircle	0.65	19.45	0.48	7.44	**0.00**	**0.00**	0.18	13.41	**0.20**	**0.00**	0.46	5.55	0.36	2.08	**0.20**	**0.00**
BallCircle	0.72	10.02	0.68	2.10	0.40	4.37	0.70	9.06	**0.34**	**0.00**	0.54	1.58	0.58	2.08	**0.34**	**0.22**
CarCircle	0.65	11.16	0.71	2.19	0.49	4.48	0.44	7.73	**0.40**	**0.11**	0.41	2.86	0.40	2.62	**0.36**	**0.20**
DroneCircle	0.82	13.78	0.55	1.29	-0.27	1.29	0.24	2.19	**0.48**	**0.00**	0.65	3.60	0.71	4.93	**0.33**	**0.07**
BulletGym Average	0.72	10.58	0.63	2.09	0.32	5.28	0.55	7.64	**0.39**	**0.10**	0.58	3.20	0.62	4.24	**0.41**	**0.33**

Empirical Observations. To support the advantages of parameter-level composition over other levels of composition, we employ t-SNE (Van der Maaten & Hinton, 2008) to project the output features of LoRA modules into a 2D space, alongside the noise and generated actions of various skills. Figure 4 illustrates that in the parameter space, different skills not only share common knowledge, but also retain their unique features to avoid confusion. In contrast, noise and action spaces are either too noisy to clearly distinguish between skills or fail to capture the shared structure across them, making the compositions in noise and action space less effective than the parameter space.

4 Experiments

PSEC enjoys remarkable versatility across various scenarios since many problems can be resolved by reusing pre-trained policies and gradually evolving its capabilities during training. Thus, we present a comprehensive evaluation across diverse scenarios, including multi-objective composition, policy learning under policy shifts and dynamics shifts, to answer the following questions:

- Can the context-aware modular effectively compose different skills?
- Can our parameter-level composition outperform noise- and action-level compositions?
- Can the introduction of LoRA modules enhance training and sample efficiency?
- Can PSEC framework iteratively evolve after incorporating more skills?

4.1 Multi-objective composition

In many real-world applications, a complex task can be decomposed into simpler objectives, where collaboratively combining these atomic skills can tackle the complex task. In this setting, we aim to evaluate the advantages of parameter-level composition over other levels of composition in Figure 3, and the effectiveness of the context-aware modular. We consider one practical multi-objective composition scenario within the safe offline RL domain (Zheng et al., 2024). This setting requires solving a constrained MDP (Altman, 2021) to tackle a complex trilogy objective: avoiding distributional shift, maximizing rewards, and meanwhile minimizing costs. These objectives can conflict, thus requiring a nuanced composition to optimize performance effectively (Zheng et al., 2024).

Setup. We evaluate on a popular safe offline RL benchmark, DSRL (Liu et al., 2023a). We set $w(s,a) = 1$ in Eq. (3) to train our initial policy π_0 as a behavior policy. Then, we set $w(s,a) = \exp(A_r^*(s,a))$ and $w(s,a) = \exp(-A_h^*(s,a))$ with $A_r^*(s,a)$ and $A_h^*(s,a)$ are the optimal reward and feasible value function learned by expectile regression (Zheng et al., 2024) to train π_1 and π_2 that separately consider reward and safety performance respectively. During composition, we adopt

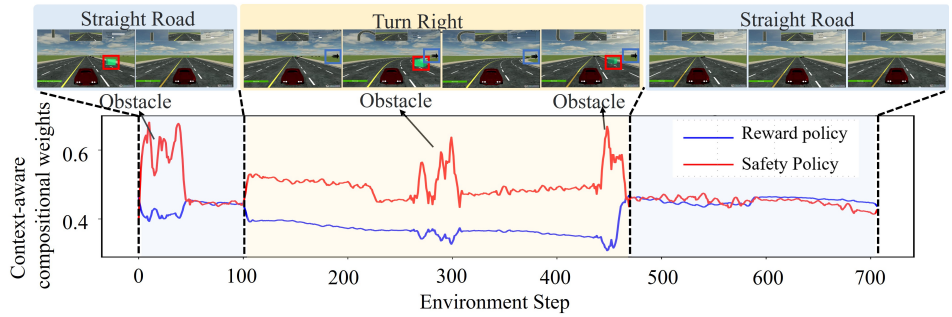


Figure 5: Output weights of context-aware modular evaluated on the MetaDrive-easymean task. The network dynamically adjusts the weights to handle real-time demands: It prioritizes safety policies when the vehicle approaches obstacles or navigates a turn while avoiding boundary lines. When no obstacles are present and the task is simply driving straight, it shifts focus toward maximizing rewards and maintain some safety insurance.

a few filtered near-expert demonstrations that jointly consider the trilogy objective, which is too limited to imitate good policies. However, we can adopt these data to train a context-aware modular $\alpha(s;\theta)$ in Eq. (7) to adaptively compose $\pi_{0,1,2}$ to handle the conflicts in an efficient way.

Baselines. To demonstrate the effectiveness of the composition in parameter space, we compare two other composition methods: *noise-level* and *action-level* composition. We denote them as NESC and ASEC respectively, where we control the only differences to PSEC being the composition stage as shown in Figure 3 to ensure a fair comparison. We also compare recent state-of-the-art (SOTA) safe offline RL methods including FISOR (Zheng et al., 2024), CDT (Liu et al., 2023b), COptiDICE (Lee et al., 2022a), CPQ (Xu et al., 2022) and BC. These traditional safe offline RL methods typically use human-tuned trade-offs to balance the trilogy objective, which is equivalent to using fixed composition weights compared to PSEC. All policies are trained on the full DSRL dataset to ensure a fair comparison (see Appendix C.1 for details).

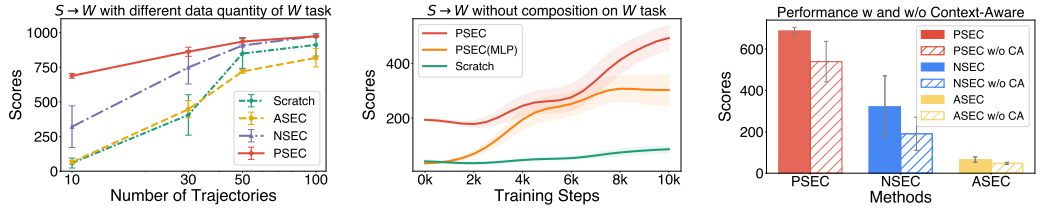
Main Results. Table 1 shows that PSEC achieves a good balance between high returns and satisfactory safety performance, and simultaneously mitigates distributional shift across all tasks, enjoying highly competitive performance. In contrast, NSEC and ASEC exhibit skewed learning behaviors, where both of them fail to discover an effective composition to ensure both good safety performance and high returns, resulting in relatively poor safety outcomes despite high rewards. PSEC also outperforms all traditional safe offline RL baselines, demonstrating the necessity of context-aware composition over fixed composition when the task requires intricate balance between different elements. To further support this, we visualize the outputs of our context-aware modular $\alpha(s;\theta)$ to illustrate its adaptive capabilities. Figure 5 demonstrates that the network dynamically adjusts the weightings to combine different skills, enabling a collaborative response to real-time environmental changes. This adaptive behavior highlights the importance of dynamically adjusting the compositional weights rather than relying on a fixed combination of different skills to jointly solve a new task like previous methods (Ajay et al., 2023; Zheng et al., 2024; Janner et al., 2022).

4.2 Continue Policy Shift Setting

We evaluate another practical scenario where the agent is progressively tasked with new tasks. We aim to continuously expand the skill libraries to test if the capabilities of agents to learn new skills can be gradually enhanced as prior knowledge grows and test the efficiency of LoRA.

Setup. We conduct experiments on the DeepMind Control Suite (DMC) (Tassa et al., 2018) environments, where an agent is progressively required to stand, walk, and run. We investigate whether PSEC can leverage the standing skill to rapidly learn to walk, and then effectively combine standing and walking skills to adapt to running. For this purpose, we pretrain π_0 to learn the basic standing skill by setting $w(s,a) := 1$ in Eq. (3) trained on a expert dataset $\mathcal{D}_e^{\mathcal{T}_0}$. Subsequently, we provide small expert datasets $\mathcal{D}_e^{\mathcal{T}_1}$ for walk and $\mathcal{D}_e^{\mathcal{T}_2}$ for run, while maintaining $w(s,a) := 1$ to adapt to π_1 and π_2. After training π_1, we integrate it into the skill library Π to assist π_2 training alongside π_0. See Appendix C.2 for detailed experimental setups.

Baselines. *1)* We compare NSEC and ASEC to further demonstrate the superiority of parameter-over noise- and action-level composition. *2)* We evaluate training from scratch (denoted as Scratch),



(a) Sample efficiency. (b) Training efficiency. (c) Context-aware efficiency.

Figure 6: Comparisons on sample and training efficiency and the effectiveness of context-aware modular. S, W, R denote stand, walk and run, respectively. Each value is averaged over 10 episodes and 5 seeds.

or replacing LoRA modules with multiplayer perceptions (MLP) to demonstrate the efficiency of compositions and LoRA module. *3)* We evaluate different PSEC variants without context-aware modular (denoted as w/o CA) to further highlight the crucial role of dynamically combining skills.

Continue Evolution. Table 2 shows that PSEC effectively leverages prior knowledge to facilitate efficient policy learning given solely limited data. Notably, S+W→R outperforms S→R, demonstrating that the learning capability of PSEC gradually evolves as the skill library grows. In contrast, training from scratch or replacing the LoRA modules with MLP fails to learn new skills given limited data, highlighting the effectiveness of both utilizing prior knowledge and the introduction of LoRA to efficiently adapt to new skills and self-evolution. Moreover, note that even PSEC (MLP) outperforms NSEC and ASEC, further highlighting the advantages of parameter-level compositions.

Table 2: Results in policy shift setting. S, W, R denote stand, walk and run. 10 trajectories are provided for W and R tasks

	S→ W	S→ R	S+W→ R
Scratch	58.9	25.5	25.5
ASEC	65.7	24.3	30.8
NSEC	320.9	38.5	39.4
PSEC (MLP)	424.1	143.3	194.5
PSEC	**688**	**221**	**247**

Training and sample efficiency. To demonstrate the training and sample efficiency of PSEC, we conduct extensive evaluations across varying numbers of trajectories and different methods. Figure 6(a) shows that PSEC achieves superior sample efficiency across different training sample sizes, particularly when data is scarce (e.g., only 10 trajectories). Figure 6(b) shows that PSEC can quickly attain excellent performance even without composition, highlighting the effectiveness of the LoRA modules. Hence, we train less then 50k gradient steps for almost all tasks, while previous methods typically require millions of gradient steps and data to obtain reasonable results.

Context-aware Composition *v.s.* Fixed Composition. We carefully tune the fixed composition (w/o CA) of different skills during composition. However, Figure 6(c) shows that the context-aware modular can consistently outperform the fixed ones across different levels of compositions. This demonstrates the advantages of context-aware composition network to fully leverage the prior knowledge in the skill library to enable efficient policy adaptations.

4.3 Dynamics Shift Setting

We evaluate PSEC in another practical setting to further validate its versatility, where the dynamics $\mathcal{P}$ shift to encompass diverse scenarios such as cross-embodiment (O'Neill et al., 2024), sim-to-real transfer (Tobin et al., 2017), and policy learning in non-stationary environments (Xue et al., 2024).

Setup. We evaluate on the D4RL environments (Fu et al., 2020) , where we modify the dynamics and morphology of locomotive robots to reflect the dynamics changes. Specifically, we first pretrain π_0 using a dataset $\mathcal{D}_o^{\mathcal{P}_0}$ collected from a modified dynamics $\mathcal{P}_0$ and then equip it with a new small dataset $\mathcal{D}_o^{\mathcal{P}_1}$ collected under the original D4RL dynamics $\mathcal{P}_1$. *Friction*, *Thigh Size* and *Gravity* denote $\mathcal{P}_0$ modifies the friction condition, the thigh size of cheetah/walker, and the gravity respectively. Based on the new small dataset $\mathcal{D}_o^{\mathcal{P}_1}$, we set $w(s,a) = \exp(A_r^*(s,a))$ with $A_r^*(s,a)$ as the advantage function trained by expectile regression on $\mathcal{D}_o^{\mathcal{P}_1}$ (Kostrikov et al., 2022) to obtain a new policy π_1 and then optimize the context-aware composition network $\alpha(s;\theta)$ to combine $\pi_{0,1}$ to collaboratively work under dynamics $\mathcal{P}_1$. See Appendix C.3 for details.

Baselines. One branch of baselines consists in training π_1 from scratch on the small dataset $\mathcal{D}_o^{\mathcal{P}_1}$, which may face data scarcity challenges, including BC, offline RL methods like CQL (Kumar et al.,

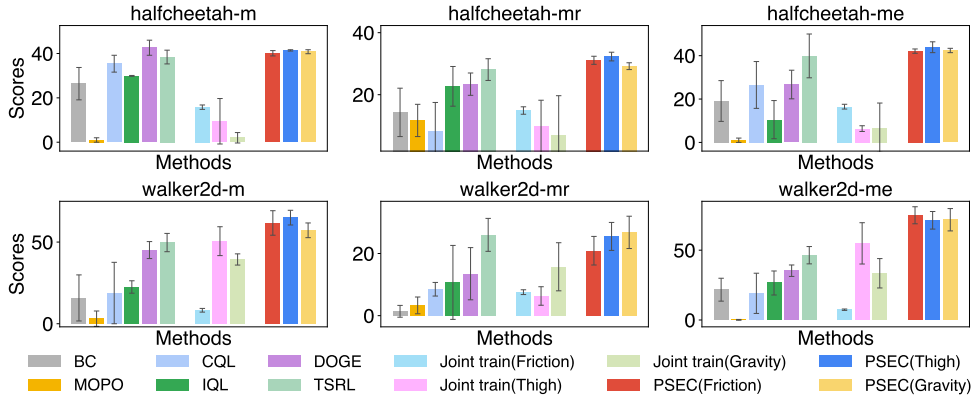


Figure 7: Results in the dynamics shift setting over 10 episodes and 5 seeds. -m, -mr and -me refer to $\mathcal{D}_o^{\mathcal{P}_1}$ sampling from medium, medium-replay and medium-expert V2 data in D4RL (Fu et al., 2020), respectively.

2020), IQL (Kostrikov et al., 2022), MOPO (Yu et al., 2020b). In addition, we evaluate some generalizable offline RL methods including DOGE (Li et al., 2023) and TSRL (Cheng et al., 2023) that are superior in the small sample regimes. Additionally, we evaluate the policy trained on the combination of $\mathcal{D}_o^{\mathcal{P}_0}$ and $\mathcal{D}_o^{\mathcal{P}_1}$, referred to as Joint train, to show the advantages of the PSEC framework over a brute-force method of combining all data to address dynamic gaps.

Main Results. Figure 7 demonstrates that PSEC effectively utilizes transferable knowledge from the pretrained policy π_0 to enhance performance under changed dynamics. In contrast, traditional offline RL methods perform poorly with limited data in new dynamic settings. Moreover, PSEC surpasses specialized sample-efficient offline RL methods like TSRL and DOGE, showcasing its superior ability to leverage prior knowledge for increased training efficiency.

4.4 Ablation Study

We primarily ablate on different LoRA ranks n to assess the robustness of our methods. Figure 8 demonstrates that under varied LoRA n ranks, PSEC consistently outperforms the MLP variant across various LoRA ranks, demonstrating the superior robustness of LoRA modules. Among the different rank settings, we observe that $n = 8$ yields the best results, thus is opted as the default choice for all experiments in our paper. We hypothesize using rank larger than 8 degenerates is due to the training data being quite limited (e.g. only 10 demonstrations).

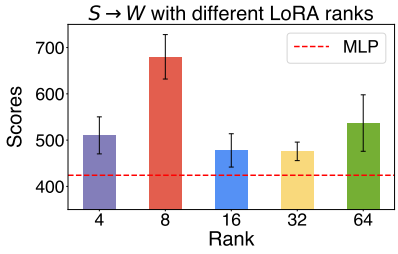


Figure 8: Ablations on LoRA ranks.

5 Conclusion

We propose PSEC, a framework that handles different skills as plug-and-play LoRA modules within an expandable skill library. This flexible approach enables the agents to reuse prior knowledge for efficient new skill acquisition and to progressively evolve in response to new challenges like humans. By exploiting the interpolation property of LoRA, we propose a context-aware compositional network that adaptively activates and blends different skills directly in the parameter space by merging the corresponding LoRA modules. This parameter-level composition enables the exploitation of more shared and complementary information across different skills, allowing for optimal compositions that collaboratively generate complex behaviors in dynamical environments. PSEC demonstrates exceptional effectiveness across diverse practical applications, such as multi-objective composition, continual policy shift and dynamic shift settings, making it highly versatile for real-world scenarios where knowledge reuse and monotonic policy improvements are crucial. One limitation is the pretrained policy π_0 may encompass diverse distributions to ensure good LoRA tuning. However, this can be mitigated by utilizing the broad out-of-domain dataset to enhance distribution coverage. More discussions on limitations and future works can be found in Appendix A.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. *Advances in neural information processing systems*, 35:28955–28971, 2022.

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B Tenenbaum, Tommi S Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023.

Anurag Ajay, Seungwook Han, Yilun Du, Shuang Li, Abhi Gupta, Tommi Jaakkola, Josh Tenenbaum, Leslie Kaelbling, Akash Srivastava, and Pulkit Agrawal. Compositional foundation models for hierarchical planning. *Advances in Neural Information Processing Systems*, 36, 2024.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Ferran Alet, Maria Bauza, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie P Kaelbling. Modular meta-learning in abstract graph networks for combinatorial generalization. *arXiv preprint arXiv:1812.07768*, 2018.

Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 39–48, 2016.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Brandon Araki, Xiao Li, Kiran Vodrahalli, Jonathan DeCastro, Micah Fry, and Daniela Rus. The logical options framework. In *International Conference on Machine Learning*, pp. 307–317. PMLR, 2021.

Gasser Auda and Mohamed Kamel. Modular neural network classifiers: A comparative study. *Journal of Intelligent and robotic Systems*, 21:117–129, 1998.

Gasser Auda and Mohamed Kamel. Modular neural networks: a survey. *International journal of neural systems*, 9(02):129–151, 1999.

Chenjia Bai, Lingxiao Wang, Jianye Hao, Zhuoran Yang, Bin Zhao, Zhen Wang, and Xuelong Li. Pessimistic value iteration for multi-task data sharing in offline reinforcement learning. *Artificial Intelligence*, 326:104048, 2024.

Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pp. 501–510. PMLR, 2018.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Peng Cheng, Xianyuan Zhan, Zhihao Wu, Wenjia Zhang, Shoucheng Song, Han Wang, Youfang Lin, and Li Jiang. Look beneath the surface: Exploiting fundamental symmetry for sample-efficient offline rl. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2024.

Hristos S Courellis, Juri Minxha, Araceli R Cardenas, Daniel L Kimmel, Chrystal M Reed, Taufik A Valiante, C Daniel Salzman, Adam N Mamelak, Stefano Fusi, and Ueli Rutishauser. Abstract representations emerge in human hippocampal neurons during inference. *Nature*, pp. 1–9, 2024.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

Laura N Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience*, pp. 1–15, 2024.

Yilun Du and Leslie Pack Kaelbling. Position: Compositional generative modeling: A single model is not all you need. In *Forty-first International Conference on Machine Learning*, 2024.

Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International conference on machine learning*, pp. 8489–8510. PMLR, 2023.

Howard Eichenbaum. Prefrontal–hippocampal interactions in episodic memory. *Nature Reviews Neuroscience*, 18(9):547–558, 2017.

Manfred Eppe, Christian Gumbsch, Matthias Kerzel, Phuong DH Nguyen, Martin V Butz, and Stefan Wermter. Intelligent problem-solving as integrated hierarchical reinforcement learning. *Nature Machine Intelligence*, 4(1):11–20, 2022.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Konrad Gadzicki, Razieh Khamsehashari, and Christoph Zetzsche. Early vs late fusion in multimodal convolutional neural networks. In *2020 IEEE 23rd international conference on information fusion (FUSION)*, pp. 1–6. IEEE, 2020.

Sibo Gai and Donglin Wang. Single-task continual offline reinforcement learning. *arXiv preprint arXiv:2404.12639*, 2024.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Bart LM Happel and Jacob MJ Murre. Design and evolution of modular neural network architectures. *Neural networks*, 7(6-7):985–1004, 1994.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*, 2023.

Kaixin Huang, Li Shen, Chen Zhao, Chun Yuan, and Dacheng Tao. Solving continual offline reinforcement learning with decision transformer. *arXiv preprint arXiv:2401.08478*, 2024.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022.

Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J Roberts. Unclear: A straightforward method for continual reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Junsu Kim, Seohong Park, and Sergey Levine. Unsupervised-to-online reinforcement learning. *arXiv preprint arXiv:2408.14785*, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Jongmin Lee, Cosmin Paduraru, Daniel J Mankowitz, Nicolas Heess, Doina Precup, Kee-Eung Kim, and Arthur Guez. Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation. In *International Conference on Learning Representations*, 2022a.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pp. 1702–1712. PMLR, 2022b.

Jianxiong Li, Xianyuan Zhan, Haoran Xu, Xiangyu Zhu, Jingjing Liu, and Ya-Qin Zhang. When data geometry meets deep function: Generalizing offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Siyuan Li, Rui Wang, Minxue Tang, and Chongjie Zhang. Hierarchical reinforcement learning with advantage-based auxiliary rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Jinmei Liu, Wenbin Li, Xiangyu Yue, Shilin Zhang, Chunlin Chen, and Zhi Wang. Continual offline reinforcement learning via diffusion-based dual generative replay. *arXiv preprint arXiv:2404.10662*, 2024.

Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, pp. 423–439. Springer, 2022.

Zuxin Liu, Zijian Guo, Haohong Lin, Yihang Yao, Jiacheng Zhu, Zhepeng Cen, Hanjiang Hu, Wenhao Yu, Tingnan Zhang, Jie Tan, et al. Datasets and benchmarks for offline safe reinforcement learning. *arXiv preprint arXiv:2306.09303*, 2023a.

Zuxin Liu, Zijian Guo, Yihang Yao, Zhepeng Cen, Wenhao Yu, Tingnan Zhang, and Ding Zhao. Constrained decision transformer for offline safe reinforcement learning. In *International Conference on Machine Learning*, 2023b.

Zuxin Liu, Jesse Zhang, Kavosh Asadi, Yao Liu, Ding Zhao, Shoham Sabach, and Rasool Fakoor. Tail: Task-specific adapters for imitation learning with large pretrained models. *arXiv preprint arXiv:2310.05905*, 2023c.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2023.

Yu Luo, Tianying Ji, Fuchun Sun, Jianwei Zhang, Huazhe Xu, and Xianyuan Zhan. OMPO: A unified framework for RL under policy and dynamics shifts. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=R83VIZtHXA.

Yunhao Luo, Chen Sun, Joshua B Tenenbaum, and Yilun Du. Potential based diffusion motion planning. In *Forty-first International Conference on Machine Learning*, 2024b.

Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJl6TjRcY7.

Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)*, 39(4):39–1, 2020.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Liangzu Peng, Paris Giampouras, and René Vidal. The ideal continual learner: An agent that never forgets. In *International Conference on Machine Learning*, pp. 27585–27610. PMLR, 2023.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in neural information processing systems*, 32, 2019.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pp. 188–204. PMLR, 2021.

Edoardo Maria Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. Combining parameter-efficient modules for task-level generalisation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 687–702, 2023.

Akshara Prabhakar, Yuanzhi Li, Karthik Narasimhan, Sham Kakade, Eran Malach, and Samy Jelassi. Lora soups: Merging loras for practical skill composition tasks. *arXiv preprint arXiv:2410.13025*, 2024.

Ahmed H Qureshi, Jacob J Johnson, Yuzhe Qin, Taylor Henderson, Byron Boots, and Michael C Yip. Composing task-agnostic policies with deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

Thomas Schmied, Markus Hofmarcher, Fabian Paischer, Razvan Pascanu, and Sepp Hochreiter. Learning to modulate pre-trained models in rl. *Advances in Neural Information Processing Systems*, 36, 2024.

AMANDA J C SHARKEY. On combining artificial neural nets. *Connection science*, 8(3-4):299–314, 1996.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2016.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. Continual diffusion: Continual customization of text-to-image diffusion with c-lora. *arXiv preprint arXiv:2304.06027*, 2023.

Shagun Sodhani, Mojtaba Faramarzi, Sanket Vaibhav Mehta, Pranshu Malviya, Mohamed Abdelsalam, Janarthanan Janarthanan, and Sarath Chandar. An introduction to lifelong supervised learning. *arXiv preprint arXiv:2207.04354*, 2022.

Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. Paco: Parameter-compositional multi-task reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 21495–21507, 2022.

Lingfeng Sun, Haichao Zhang, Wei Xu, and Masayoshi Tomizuka. Efficient multi-task and transfer reinforcement learning with parameter-compositional framework. *IEEE Robotics and Automation Letters*, 8(8):4569–4576, 2023.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Lirui Wang, Jialiang Zhao, Yilun Du, Edward H Adelson, and Russ Tedrake. Poco: Policy composition from and for heterogeneous robot learning. *arXiv preprint arXiv:2402.02511*, 2024a.

Shenzhi Wang, Qisen Yang, Jiawei Gao, Matthieu Lin, Hao Chen, Liwei Wu, Ning Jia, Shiji Song, and Gao Huang. Train once, get a family: State-adaptive balances for offline-to-online reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024b.

Yixiao Wang, Yifei Zhang, Mingxiao Huo, Thomas Tian, Xiang Zhang, Yichen Xie, Chenfeng Xu, Pengliang Ji, Wei Zhan, Mingyu Ding, and Masayoshi Tomizuka. Sparse diffusion policy: A sparse, reusable, and flexible policy for robot learning. In *8th Annual Conference on Robot Learning*, 2024c. URL https://openreview.net/forum?id=zeYaLS2tw5.

Zhe Wang, Siqi Fan, Xiaoliang Huo, Tongda Xu, Yan Wang, Jingjing Liu, Yilun Chen, and Ya-Qin Zhang. Emiff: Enhanced multi-scale image feature fusion for vehicle-infrastructure cooperative 3d object detection. In *ICRA*, 2024d.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Maciej Wolczyk, Michal Zajac, Razvan Pascanu, Lukasz Kucinski, and Piotr Milos. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized q-learning for safe offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8753–8760, 2022.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline RL with no OOD actions: In-sample learning via implicit value regularization. In *The Eleventh International Conference on Learning Representations*, 2023.

Zhenghai Xue, Qingpeng Cai, Shuchang Liu, Dong Zheng, Peng Jiang, Kun Gai, and Bo An. State regularized policy optimization on data with dynamics shift. *Advances in neural information processing systems*, 36, 2024.

Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020a.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020b.

Haichao Zhang, Wei Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023a.

Jinghan Zhang, Junteng Liu, Junxian He, et al. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, 36:12589–12610, 2023b.

Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. *arXiv preprint arXiv:2401.10700*, 2024.

Ming Zhong, Yelong Shen, Shuohang Wang, Yadong Lu, Yizhu Jiao, Siru Ouyang, Donghan Yu, Jiawei Han, and Weizhu Chen. Multi-lora composition for image generation. *arXiv preprint arXiv:2402.16843*, 2024.

A Limitations and Future works

In this section, we provide detailed discussions about the limitations and their potential solutions.

- **Assumption on the expressiveness of the pretrain policy**. The main limitation of PSEC is the assumption that the pre-trained π_0 covers a diverse distribution, which allows for efficient fine-tuning using small add-on LoRA modules. If this assumption does not hold, learning new skills through parameter-efficient fine-tuning may prove challenging, as significantly more parameters might be required to acquire new skills.

  Potential solutions: Note that this assumption is mild in relevant papers that utilize LoRA to learn new skills (Hu et al., 2021; Liu et al., 2023c). To tackle this problem, one straightforward solution is to increase the value of LoRA ranks to increase the learning capabilities of the newly introduced modules. Another simple solution is to leverage the cheap and abundant out-of-domain data to enhance the distribution coverage of the pretrained π_0 to enable efficient LoRA adaptations.

- **Redundant skill expansion**. In this paper, PSEC includes policies for all tasks in the skill library across its lifelong time. Although we adopt LoRA to reduce computational burden and memory usage, maintaining an extensive library of skill primitives may still lead to substantial computational costs.

  Potential solutions: Note that not all skills should be incorporated into the skill library, particularly those that are redundant and can be synthesized from other primitives. An interesting direction for future research is to develop an evaluation metric to assess the interconnections between different skills, such as the skill diversity (Pertsch et al., 2021; Eysenbach et al.), to only include essential, non-composable atomic primitives. Such a strategy could significantly reduce the management costs associated with maintaining the skill library.

- **Hyperparameter-tuning**: Another limitation is PSEC introduces another LoRA modules to learn new skills, which can introduce additional hyperparameters required to be tuned.

  Potential solutions: This limitation is widely existed in relevant works that try to reuse prior knowledge to learn new skills (Liu et al., 2023c; Clark et al., 2024; Wang et al., 2024c; Peng et al., 2019; Barreto et al., 2018), since almost all papers require additional parameters or regularization to adapt to the new skills. In this paper, we have ablated the robustness of PSEC against varied LoRA ranks, and demonstrate consistent superiority over the naive MLP modules in Figure 8, highlighting the robustness of PSEC for hyperparameter tuning.

- **Simple context-aware compositional modular**: We employ a simple context-aware modular $\alpha(s;\theta)$ to dynamically combine different primitives. This operation is simple and may not fully leverage the shared structure across skills for the target task.

  Potential Solutions: However, in our paper, we have demonstrated the superior advantages of this simple context-aware modular, as shown in Figure 6c. One interesting future direction is to adopt a more advanced model architecture, training objective, or more flexible gating approach to optimize the modular.

B Discussions on more related works

Multi-task RL and IL. Some multi-task RL and IL methods also attempt to leverage the shared features across different tasks to enhance the overall performance (Wang et al., 2024c; Yang et al., 2020; Sun et al., 2022; 2023). However, they primarily focus on avoiding conflicts between tasks and often overlook the benefits of reusing prior knowledge to tackle new tasks.

MoE in decision making. The recent SDP (Wang et al., 2024c) is particularly relevant to our work. Specifically, SDP employs Mixture of Experts (MoE) (Shazeer et al., 2016) to encode skills as flexible combinations of forward path gated by distinct routers, allowing for efficient adaptation to new tasks by fine-tuning newly introduced expert tokens and task-specific routers. However, SDP necessitates that the pretrained policy π_0 be modeled with MoE layers, which imposes additional requirements on the model architecture. In contrast, our approach does not impose any constraints on the structure of the pretrained network and allows for the direct incorporation of new skills as plug-and-play LoRA modules. Moreover, when we identify a skill that is underperforming, we can easily

modify the skill library by simply removing its plug-and-play LoRA modules. In contrast, using MoE limits this flexibility in managing different skills, making it challenging to mitigate the side effects caused by suboptimal skills. Therefore, PSEC offers a more flexible approach to managing the skill library, making it more feasible to scale up and incorporate a larger number of skills.

LoRA in decision making. Other relevant works such as TAIL (Liu et al., 2023c), LoRA-DT (Huang et al., 2024) and L2M (Schmied et al., 2024) also employ LoRA to encode skills. However, they solely investigate the parameter-isolation property of LoRA to prevent catastrophic forgetting, while overlooking the potential to merge different LoRA modules to interpolate new skills. Moreover, TAIL only studies the IL domain, L2M and LoRA-DT only study the RL domain, while PSEC both explore the effectiveness in RL and IL settings.

LoRA for composition in other domains. (Ponti et al., 2023; Clark et al., 2024; Huang et al., 2023; Zhong et al., 2024; Prabhakar et al., 2024) use LoRA for multi-task learning but using a fixed combination of LoRA modules, focusing on static settings like language model or image generation, thus limiting its expressiveness of the pretrained LoRA modules and flexibility of composition. In contrast, PSEC combines different LoRA via a context-aware modular, maximizing the expressiveness of pretrained skills to flexibly compose new skills, which is crucial for decision making since the real-time adjustment is required to handle the dynamical problems as shown in Figure 5.

Other relevant works. The concept of employing modularized networks to encode different skills has been proposed in previous studies (Happel & Murre, 1994; SHARKEY, 1996; Auda & Kamel, 1998; 1999; Sodhani et al., 2022; Andreas et al., 2016; Alet et al., 2018; Ponti et al., 2023; Clark et al., 2024; Huang et al., 2023; Zhong et al., 2024; Prabhakar et al., 2024). This modularized design enables efficient adaptation to new tasks by exploiting the shareable knowledge stored in different modules. In addition, it naturally facilitates continual evolvement by absorbing new skills in new modules in a parameter-isolation manner (Sodhani et al., 2022). However, all these previous works fail to answer one crucial question in decision making: *How to efficiently compose different skills to tackle dynamical demands?* Specifically, all previous works typically resort to fixed combinations of different modules, thus limiting the flexibility to handle decision-making scenarios where real-time composition adjustment is required to satisfy the dynamical demands. Some recent works take the dynamical demands into consideration by adaptively composing different skills (Peng et al., 2019; Qureshi et al., 2020). These methods, however, fail to effectively exploit the shared structure across different skills by naively composing different components in the original action space. PSEC, however, systematically investigates the advantages of skill compositions in parameter space over the noise space and action space, offering clear guidance for future research to expand and compose skills in parameter spaces rather than noise/action spaces.

Some works use logical options for skill composition (Araki et al., 2021) but require significant human effort for skill management, limiting scalability. Additionally, Araki et al. (2021) focuses on efficient pretraining, not on fast adaptation/continual improvement. In contrast, PSEC targets the later setups and minimizes human effort by incorporating new skills as LoRA modules, which are then combined through auto-learned compositional networks.

C Experimental Setups

C.1 Multi-objective Composition

Training details of PSEC. In this setting, we have four networks required to train: the behavior policy π_0, the safety policy π_1 that minimizes the cost, the reward policy π_2 that maximizes the return, and the context-aware modular $\alpha(s;\theta) \in \mathbb{R}^2$. For each task, we first pretrain π_0 parameterized by W_0 as behavior policy by minimizing the following objective on the full DSRL dataset $\mathcal{D}$ (Liu et al., 2023a) to ensure a diverse pretrained distribution coverage:

$$\mathcal{L}_{\pi_0}(W_0) = \mathbb{E}_{t\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}}\left[\left\|\epsilon - \epsilon_{W_0}\left(\sqrt{\bar{\rho}_t}a + \sqrt{1-\bar{\rho}_t}\epsilon, t, s\right)\right\|^2\right]. \tag{10}$$

Then, we equip the agent with the same dataset $\mathcal{D}$ but provide feasible label h and reward labels r, forming the dataset $\mathcal{D}^h = \{(s,a,h,s')\}$ and $\mathcal{D}^r\{(s,a,r,s')\}$. Then we train π_1 and π_2 based on these datasets by optimizing their newly introduced LoRA modules ΔW_1 and ΔW_2 via minimizing the following objectives in Eq. (11-12):

$$\mathcal{L}_{\pi_1}(\Delta W_1) = \mathbb{E}_{t\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}^h}\left[w^h(s,a)\left\|\epsilon - \epsilon_{W_1}\left(\sqrt{\bar{\rho}_t}a + \sqrt{1-\bar{\rho}_t}\epsilon, t, s\right)\right\|^2\right], \quad (11)$$

$$\mathcal{L}_{\pi_2}(\Delta W_2) = \mathbb{E}_{t\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}^r}\left[w^r(s,a)\left\|\epsilon - \epsilon_{W_2}\left(\sqrt{\bar{\rho}_t}a + \sqrt{1-\bar{\rho}_t}\epsilon, t, s\right)\right\|^2\right], \quad (12)$$

where the weights of LoRA augmented layer are $W_1 = W_0 + 16\Delta W_1$ and $W_2 = W_0 + 16\Delta W_2$ as defined in Eq. (5). $w^h(s,a) := \exp(-A_h^*(s,a))$ and $w^r(s,a) := \exp(A_r^*(s,a))$ are the weighting function derived from the optimal feasible value function $A_h^*(s,a) = Q_h^*(s,a) - V_h^*(s)$ and reward value function $A_r^*(s,a) = Q_r^*(s,a) - V_r^*(s,a)$, optimized via expectile regression following (Kostrikov et al., 2022; Zheng et al., 2024), where $Q_h^*(s,a)$ and $V_h^*(s)$ can be obtained via minimizing Eq. (13-14), $Q_r^*(s,a)$ and $V_r^*(s)$ can be obtained via minimizing Eq. (15-16):

$$\mathcal{L}_{V_h} = \mathbb{E}_{(s,a)\sim\mathcal{D}^h}\left[L_{\text{rev}}^{\tau}\left(Q_h(s,a) - V_h(s)\right)\right], \quad (13)$$

$$\mathcal{L}_{Q_h} = \mathbb{E}_{(s,a,s',h)\sim\mathcal{D}^h}\left[\left(((1-\gamma)h(s) + \gamma\max\{h(s), V_h(s')\}) - Q_h(s,a)\right)^2\right], \quad (14)$$

$$\mathcal{L}_{V_r} = \mathbb{E}_{(s,a)\sim\mathcal{D}^r}\left[L^{\tau}\left(Q_r(s,a) - V_r(s)\right)\right], \quad (15)$$

$$\mathcal{L}_{Q_r} = \mathbb{E}_{(s,a,s',r)\sim\mathcal{D}^r}\left[\left(r + \gamma V_r(s') - Q_r(s,a)\right)^2\right]. \quad (16)$$

where $L^{\tau}(u) = |\tau - \mathbb{I}(u<0)|\,u^2$ and $L_{\text{rev}}^{\tau}(u) = |\tau - \mathbb{I}(u>0)|u^2$ with $\tau \in (0.5, 1)$. By doing so, π_1 and π_2 become one safety policy that avoids unsafe outcomes and one reward policy that tries to maximize the cumulative returns, respectively.

Then, we train our context-aware modular network $\alpha(s;\theta)$ to combine $\pi_{0,1,2}$ to collaboratively tackle the safe offline RL problem. We filter the Top-30 trajectories with the highest rewards and costs below 5 from the dataset $\mathcal{D}$ to form a small near-expert dataset $\mathcal{D}^*$ that obtains a good balance among distributional shift, reward maximization and safety constraint. Then, we train $\alpha(s;\theta)$ by minimizing the following imitation learning loss based on the $\mathcal{D}^*$:

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}^*}\left[\left\|\epsilon - \epsilon_W\left(\sqrt{\bar{\rho}_t}a + \sqrt{1-\bar{\rho}_t}\epsilon, t, s\right)\right\|^2\right], \quad (17)$$

where $W = W_0 + \sum_{i=1}^{2}\alpha_i(s;\theta)\Delta W_i$ as defined in Eq. (7).

We train π_0 for 1M gradient steps with a batch size of 2048 to ensure a good performance of π_0. Then, we only train π_1 and π_2 for 50K gradient steps, for the efficiency of LoRA modules. For $\alpha(s;\theta)$, we only train it for 1K gradient steps since all decomposed policies including $\pi_{0,1,2}$ are ready to be composed, which can significantly reduce the computational burden leveraging these pretrained policies. Summarized hyperparameters can be found in Table 4.

Baselines. For FISOR (Zheng et al., 2024), CDT (Liu et al., 2023b), COptiDICE (Lee et al., 2022a), CPQ (Xu et al., 2022) and BC, we adopt the results from FISOR (Zheng et al., 2024). For NSEC and ASEC results, we only change the compositional stages, and meanwhile keep all other training details the same to ensure a fair comparison. Specifically, the context-aware modular for NSEC is trained via the following reparameterization method instead of the one in Eq. (12):

$$\epsilon_{\text{NSEC}} = \epsilon_0 + \sum_{i=1}^{2}\alpha_i(s;\theta)\epsilon_i, \quad (18)$$

where $\epsilon_{0,1,2}$ is generated from networks with layers of W_0, $W_1 = W_0 + 16\Delta W_1$ and $W_2 = W_0 + 16\Delta W_2$, respectively. We can see that the composition in Eq. (18) between skills happens in the noise space, and thus we denote it as NSEC (noise skill expansion and composition).

For ASEC, we directly compose the generated actions of different policies:

$$a_{\text{ASEC}} = a_0 + \sum_{i=1}^{2}\alpha_i(s;\theta)a_i \quad (19)$$

where $a_{0,1,2}$ are the actions generated from the denoising process in Eq. (2) using the predicted noise $\epsilon_{0,1,2}$ generated by networks with layers of W_0, $W_1 = W_0 + 16\Delta W_1$ and $W_2 = W_0 + 16\Delta W_2$, respectively. The composition happens in action space, and thus we denote it as ASEC (action skill expansion and composition).

C.2 Continual Policy Shift

To evaluate PSEC's ability to continually evolving its capabilities when tackling new challenges, we conduct experiments on DeepMind Control Suite (DMC) (Tassa et al., 2018), where a walker agent is progressively required to stand, walk, and run, as shown in Figure 10. We use three expert datasets including walker-stand $\mathcal{D}_e^{\mathcal{T}_0}$, walker-walk $\mathcal{D}_e^{\mathcal{T}_1}$, and walker-run $\mathcal{D}_e^{\mathcal{T}_2}$, released by Bai et al. (2024) for the policy learning. Specifically, $\mathcal{D}_e^{\mathcal{T}_0}$, $\mathcal{D}_e^{\mathcal{T}_1}$ and $\mathcal{D}_e^{\mathcal{T}_2}$ contains 1000, 10 and 10 trajectories, respectively. $\mathcal{D}_e^{\mathcal{T}_1}$ and $\mathcal{D}_e^{\mathcal{T}_2}$ contain only a handful of data because we aim to test if the agent can leverage the knowledge from the standing skill to efficiently adapt to new tasks. We first pretrain π_0 on the large $\mathcal{D}_e^{\mathcal{T}_0}$ to obtain the basic standing policy via minimizing the following behavior cloning loss:

$$\mathcal{L}_{\pi_0}(W_0) = \mathbb{E}_{t\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}_e^{\mathcal{T}_0}}\left[\left\|\epsilon - \epsilon_{W_0}\left(\sqrt{\bar{\rho}_t}a + \sqrt{1-\bar{\rho}_t}\epsilon, t, s\right)\right\|^2\right]. \tag{20}$$

Stand→ Walk (S→ W) task. Then, we can integrate the walking skill π_1 into the skill library Π by optimizing the following objective:

$$\mathcal{L}_{\pi_1}(\Delta W_1) = \mathbb{E}_{t\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}_e^{\mathcal{T}_1}}\left[\left\|\epsilon - \epsilon_{W_1}\left(\sqrt{\bar{\rho}_t}a + \sqrt{1-\bar{\rho}_t}\epsilon, t, s\right)\right\|^2\right], \tag{21}$$

where $W_1 = W_0 + 16\Delta W_1$. Then, we train a context-aware modular $\alpha^{\text{walk}}(s;\theta_1) \in \mathbb{R}$ to combine π_0 and π_1 to jointly tackle the walking task:

$$\mathcal{L}(\theta_1) = \mathbb{E}_{t\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}_e^{\mathcal{T}_1}}\left[\left\|\epsilon - \epsilon_{W_{\text{walk}}}\left(\sqrt{\bar{\rho}_t}a + \sqrt{1-\bar{\rho}_t}\epsilon, t, s\right)\right\|^2\right], \tag{22}$$

where $W_{\text{walk}} = W_0 + \alpha^{\text{walk}}(s;\theta_1)\Delta W_1$. In this setting, we hope the final policy parameterized by W_{walk} can outperform the naive policy that is trained from scratch on the small data $\mathcal{D}_e^{\mathcal{T}_1}$ to demonstrate the significance of utilizing the prior knowledge in π_0 for efficient task adaptation.

Stand→Run (S→R) task. Here, the adaptation for the running policy π_2 is similar. We can replace W_1 in Eq. (21) with $W_2 = W_0 + 16\Delta W_2$ and $\mathcal{D}_e^{\mathcal{T}_1}$ as $\mathcal{D}_e^{\mathcal{T}_2}$ to train π_2 parameterized by ΔW_2. Additionally, we replace W_{walk} in Eq. (22) as $W_{\text{run}} = W_0 + \alpha^{\text{run}}(s;\theta_2)\Delta W_2$ and $\mathcal{D}_e^{\mathcal{T}_1}$ as $\mathcal{D}_e^{\mathcal{T}_2}$ to train $\alpha^{\text{run}}(s;\theta_2)$ to combine π_0 and π_2 to generate the running skill.

Stand+Walk→Run (S+W→R) task. After obtaining π_0, π_1, and π_2, the composition for the running skill becomes very simple. We can replace W_{walk} in Eq. (22) as $W = W_0 + \sum_{i=1}^{2}\alpha_i(s;\theta)\Delta W_i$ to train $\alpha(s;\theta) \in \mathbb{R}^2$ to combine $\pi_{0,1,2}$ to generate the running skill. In this setup, we aim to prove that utilizing the library that contains $\pi_{0,1,2}$ (S+W→R) can outperform $\pi_{0,2}$ (S→R) to show the learning capability of PSEC can gradually grow after incorporating more skill primitives.

We train π_0 for 1M gradient steps with a batch size of 1024 to ensure a good performance of π_0. Then, we only train π_1 and π_2 for 10K gradient steps with 10 trajectories thanks to the efficiency of LoRA. For $\alpha^{\text{walk}}(s;\theta), \alpha^{\text{run}}(s;\theta), \alpha(s;\theta)$, we only train them for 1K gradient steps since the decomposed policies including $\pi_{0,1,2}$ in the skill library are ready to be composed, which can significantly reduce the computational burden leveraging these pretrained policies. The summarized hyperparameters can be found in Table 5.

Baselines. We compare PSEC with other composition methods NSEC and ASEC, the Scratch method, and the variant PSEC (MLP). NSEC and ASEC train the context-aware modular represented by Eq. (18) and Eq. (19), respectively. Scratch method means training a policy from scratch by IDQL (Hansen-Estruch et al., 2023), since we build our model based on the IDQL method. PSEC (MLP) replaces the LoRA matrices with the MLP network in PSEC.

Experimental setups for Figure 6. For Figure 6(a), we evaluate the sample efficiency of PSEC framework. Specifically, we evaluate on the **S**→ **W** task with different data quantities of the **W** dataset $\mathcal{D}_e^{\mathcal{T}_1}$, including 10, 30, 50, and 100 trajectories, trained with 10K, 30K, 50K, and 100K training steps, respectively. We compare PSEC with other baselines to demonstrate the sample efficiency of parameter-level composition over other composition methods.

For Figure 6(b), we visualize the training curves of PSEC, PSEC (MLP) and Scratch for the **S**→**W** task trained solely on Eq. (21) without the composition in Eq. (22) to demonstrate the efficiency of LoRA modules over the naive MLPs and the efficiency to leverage pretrain policies. In this setting, $\mathcal{D}_e^{\mathcal{T}_1}$ contains 10 trajectories and we train each method for 10K training steps.

For Figure 6(c), w/o CA represents the compositional weight α is tuned by humans, rather than auto-generated by our context-aware modular α_θ. We compare PSEC, NSEC, ASEC with their corresponding w/o CA variants to further demonstrate the importance of dynamical compositions.

We conduct similar experiments on the **S**→**R** task and the results are presented in Figure 11. Note that the running skill is more difficult. PSEC shows marked superiority on this challenging setting.

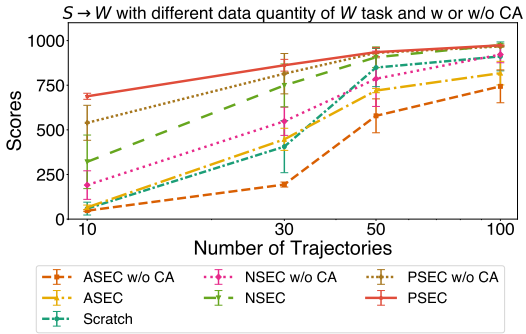
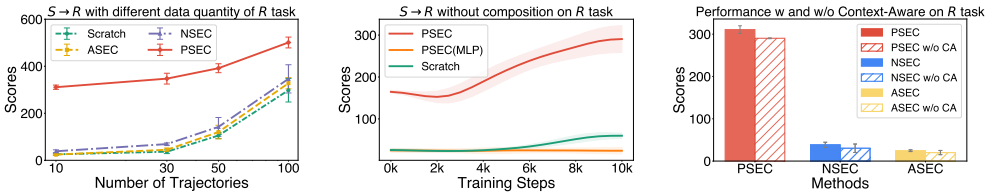


Figure 9: Results in the policy shift setting. Each value is averaged over 10 episodes and 5 seeds.

Figure 10: Continual evolution on DeepMind Control Suite for Continual policy shift.



(a) Sample efficiency. (b) Training efficiency. (c) Context-aware efficiency.

Figure 11: Comparisons on sample and training efficiency and the effectiveness of context-aware modular. S, R denote stand, run, respectively. Each value is averaged over 10 episodes and 5 seeds.

C.3 DYNAMIC SHIFT

To further validate the versatility of PSEC, we conduct experiments in a practical and common setting: dynamic shift. We conduct experiments on the D4RL benchmark, where we modify the dynamics and morphology of locomotive robots to reflect the dynamics changes. Our goal is to leverage the policies based on the source datasets $\mathcal{D}_o^{\mathcal{P}_0}$ and a small amount of the target datasets $\mathcal{D}_o^{\mathcal{P}_1}$ to adapt to the target task quickly. Specifically, the datasets $\mathcal{D}_o^{\mathcal{P}_0}$ contain 20K transitions with 3 types of dynamic modifications on $\mathcal{P}_0$: *1) Friction*: the friction coefficient of the robot is modified; *2) Gravity*: the gravity acceleration in the simulation environment is changed. *3) Thigh*: the thigh is enlarged to double its original size to produce a morphology gap on the embodiment. The target datasets $\mathcal{D}_o^{\mathcal{P}_1}$ are sampled from the D4RL benchmark with un-modified dynamics $\mathcal{P}_1$, including 6 types: halcheetah-medium-v2, halfcheetah-medium-replay-v2, halfcheetah-medium-expert-v2, walker2d-medium-v2, walker2d-medium-replay-v2, walker2d-medium-expert-v2, as shown in Figure 12. Each dataset type of $\mathcal{D}_o^{\mathcal{P}_1}$ contains solely 10K transitions, which are too limited to train good policies directly on the target dynamics $\mathcal{P}_1$ from scratch.

We first pretrain π_0 with dataset $\mathcal{D}_o^{\mathcal{P}_0}$ for 20k training steps by behavior cloning via minimizing the following objectives:

$$\mathcal{L}_{\pi_0}(W_0) = \mathbb{E}_{t\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}_o^{\mathcal{P}_0}}\left[\left\|\epsilon - \epsilon_{W_0}\left(\sqrt{\bar{\rho}_t}a + \sqrt{1-\bar{\rho}_t}\epsilon, t, s\right)\right\|^2\right]. \tag{23}$$

Then, we try to use the limited $\mathcal{P}_1$ to adapt π_0 to the target domain . PSEC uses LoRA to train a new policy π_1 with the pretrained source policy π_0 by minimizing the following objectives:

$$\mathcal{L}_{\pi_1}(\Delta W_1) = \mathbb{E}_{t\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}_o^{\mathcal{P}_1}}\left[w^r(s,a)\left\|\epsilon - \epsilon_{W_1}\left(\sqrt{\bar{\rho}_t}a + \sqrt{1-\bar{\rho}_t}\epsilon, t, s\right)\right\|^2\right], \tag{24}$$

where $W_1 = W_0 + 16\Delta W_1$. Finally, PSEC uses the context-aware modular $\alpha(s;\theta)$ to integrate policy π_0, π_1 using the target dataset $\mathcal{D}_o^{\mathcal{P}_1}$ to transfer to the target dynamics $\mathcal{P}_1$. The context-aware modular $\alpha(s;\theta)$ is trained for only 1k training steps by minimizing the following objectives:

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U},\epsilon\sim\mathcal{N}(0,I),(s,a)\sim\mathcal{D}_o^{\mathcal{P}_1}}\left[\left\|\epsilon - \epsilon_{W}\left(\sqrt{\bar{\rho}_t}a + \sqrt{1-\bar{\rho}_t}\epsilon, t, s\right)\right\|^2\right], \tag{25}$$

where $W = W_0 + \alpha(s;\theta)\Delta W_1$ as defined in Eq. (7).

We train π_0 for 1M gradient steps with a batch size of 1024 to ensure a good performance of π_0. Then, we only train π_1 for 20k gradient steps, for the efficiency of LoRA modules. For $\alpha(s;\theta)$, we only train it for 1K gradients steps since all decomposed policies including $\pi_{0,1}$ are ready to be composed, which can efficiently adapt to the target domain leveraging the pretrained source policies. Summarized hyperparameters can be found in Table 6.

Baselines. We compare PSEC with other methods in dynamic shift settings, including behavioral cloning (BC), offline RL approaches like CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2022), and model-based methods such as MOPO (Yu et al., 2020b). Additionally, we evaluate more generalizable offline RL methods, specifically DOGE (Li et al., 2023) and TSRL (Cheng et al., 2023), which have demonstrated superiority in small sample regimes. The baseline results for comparison are sourced from the TSRL paper (Cheng et al., 2023), which reports state-of-the-art performance in these regimes. Furthermore, we assess policies trained on combinations of the offline datasets $\mathcal{D}_o^{\mathcal{P}_0}$ and $\mathcal{D}_o^{\mathcal{P}_1}$ under various dynamics settings, referred to as Joint train (Gravity), Joint train (Friction), and Joint train (Thigh). These combinations involve training with one source dataset under dynamic shifts (e.g., changes in gravity, friction, or thigh size) and target datasets such as halfcheetah-medium-v2, halfcheetah-medium-replay-v2, halfcheetah-medium-expert-v2, walker2d-medium-v2, walker2d-medium-replay-v2, and walker2d-medium-expert-v2. In order to maintain fairness, the joint train method is trained in the same way as PSEC is trained on the source datasets. The results and training curves of PSEC across these settings are presented in Table 3 and Figure 15, respectively. These comparisons showcase the effectiveness of PSEC under dynamic shifts and small sample conditions.

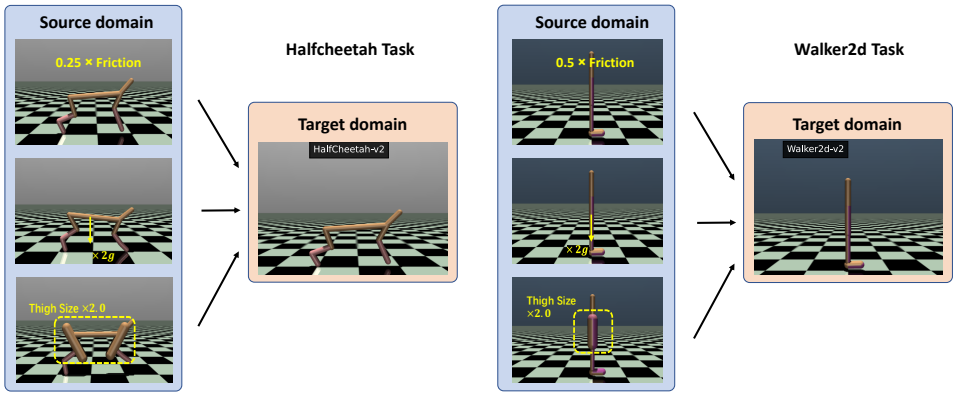
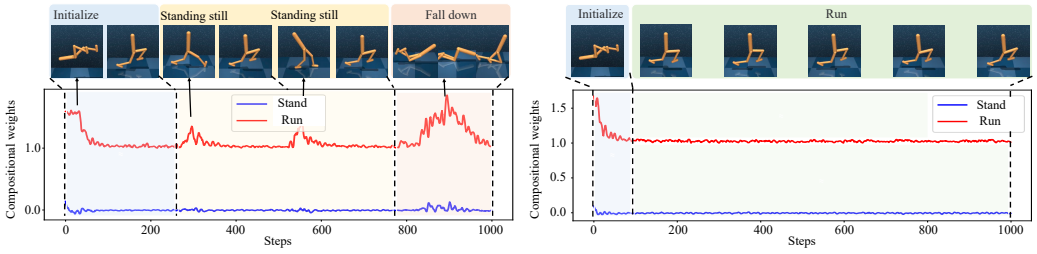


Figure 12: The illustration of the source and target domains for the dynamic shift setting.

C.4 T-SNE EXPERIMENTAL SETUPS FOR FIGURE 4

To provide empirical support of the advantages of parameter-level composition over other levels of composition, we visualize the t-SNE (Van der Maaten & Hinton, 2008) projection of data samples in different spaces. Specifically, for each dataset $\mathcal{D}_e^{\mathcal{T}_0}$, $\mathcal{D}_e^{\mathcal{T}_1}$, $\mathcal{D}_e^{\mathcal{T}_2}$ in the continual policy shift setting in Section C.2, we randomly sample 512 data samples (s, a), which forms three types of data that encode the standing, walking and running skill, respectively. In the action space, we directly utilize t-SNE projection to map these sampled data into a 2-dimentional space in Figure 4 (c). For the noise space, we add 1 step of noise on the sampled actions following the forward diffusion process in Eq. (1) and get the tuple (s, a_1) for different skills. Then, we generate the noise based on this noisy tuples and visualize their t-SNE projections in Figure 4 (b). In parameter-space, we feed the noisy tuples (s, a_1) into the trained networks and get the output features of the middle LoRA augmented layers. Then, we project these features using t-SNE in Figure 4 (a).

D MORE EXPERIMENTAL RESULTS



(a) Fixed composition on **S**→**R** task

(b) PSEC on **S**→**R** task

Figure 13: Outputs weights of the context-aware modular on DeepMind Control

D.1 THE EFFECTIVENESS OF THE CONTEXT-AWARE MODULAR

Context-aware modular for the continual policy shift. To further explore the effectiveness of the context-aware module, we employ it to analyze the trajectories generated by policies composed using fixed compositional weights. Specifically, for the **S**→**R** task in Section C.2, the fixed composition method denote $W_{\text{run}} = W_0 + \alpha \Delta W_2$, which uses a fixed $\alpha = 16$ to compose π_0 and π_2. Figure 13 (a) shows that naively using fixed compositional weights might accidentally stuck in some local suboptimal behavior such as standing still or falling down. We can clearly observe that our context-aware modular provides corresponding responses to correct these undesired behaviors. Therefore, it is necessary to adjust the weights of different strategies to fit the current states. Figure 13 (b) presents the trajectories generated by PSEC. It clearly demonstrates that by utilizing the

context-aware modular, the agent can make subtle adjustments between skills and stably run across the entire episodes.

D.2 THE PARAMETER EFFICIENCY OF PSEC

Parameter efficiency. To evaluate the parameter efficiency of PSEC, we compare its parameter count and performance on various tasks against both the Scratch method and PSEC (MLP). The parameter count for PSEC includes the LoRA parameters and context-aware parameters specific to the walker-walk or walker-run tasks. The Scratch method represents training the policy from scratch with standard MLP. PSEC (MLP), which substitutes the LoRA weights with a standard MLP and retains the context-aware modular, has a higher parameter count than the Scratch method. The parameter counts are illustrated in Figure 14. In terms of performance, the results from the DeepMind Control Suite (DMC) tasks, as shown in Figures 6 (b) and 11 (b), indicate that PSEC achieves significantly better performance despite having only 7.58% of the parameters used in the Scratch method. This performance advantage over both the Scratch method and PSEC (MLP) demonstrates that PSEC possesses strong parameter efficiency, effectively leveraging a smaller number of parameters for superior task performance. In this way, PSEC can leverage and expand upon its existing knowledge base in novel situations to enhance learning efficiency and adaptability.

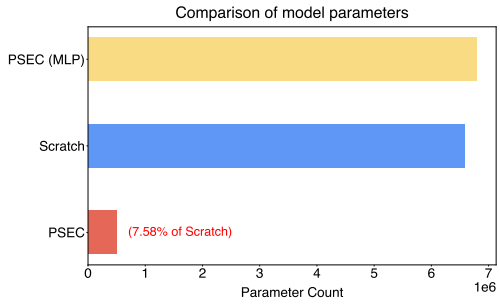


Figure 14: Comparison of Model Parameters: The parameter count for PSEC is approximately 7.58% of Scratch, demonstrating a significantly smaller model size while maintaining strong parameter efficiency, effectively leveraging a smaller number of parameters for superior task performance..

Table 3: Results in the dynamics shift setting over 10 episodes and 5 seeds. -m, -mr and -me refer to $\mathcal{D}_o^{\mathcal{P}_1}$ sampling from medium, medium-replay and medium-expert V2 data in D4RL (Fu et al., 2020), respectively.

Metric	Halfcheetah-m	Halfcheetah-mr	Halfcheetah-me	Walker2d-m	Walker2d-mr	Walker2d-me
BC	26.4 ± 7.3	14.3 ± 7.8	19.1 ± 9.4	15.8 ± 14.1	1.4 ± 1.9	21.7 ± 8.2
MOPO	-1.1 ± 4.1	11.7 ± 5.2	-1.1 ± 1.4	3.1 ± 4.7	3.3 ± 2.7	0.1 ± 0.3
CQL	35.4 ± 3.8	8.1 ± 9.4	26.5 ± 10.8	18.8 ± 18.8	8.5 ± 2.19	19.1 ± 14.4
IQL	29.9 ± 0.2	22.7 ± 6.4	10.5 ± 8.8	22.5 ± 3.8	10.7 ± 11.9	26.5 ± 8.6
DOGE	**42.6 ± 3.4**	23.4 ± 3.6	26.7 ± 6.6	45.1 ± 10.2	13.5 ± 8.4	35.3 ± 4.1
TSRL	38.4 ± 3.1	28.1 ± 3.5	39.9 ± 21.1	49.7 ± 10.6	26.0 ± 11.3	46.4 ± 13.2
Joint train(Gravity)	2.0 ± 1.4	6.8 ± 3.9	6.8 ± 5.4	39.4 ± 3.4	15.7 ± 7.7	33.5 ± 10.5
Joint train(Friction)	15.8 ± 1.0	14.9 ± 1.2	16.5 ± 1.1	8.3 ± 1.1	7.6 ± 0.8	7.4 ± 0.5
Joint train(Thigh)	9.5 ± 5.3	9.8 ± 8.5	6.4 ± 1.3	50.6 ± 8.8	6.3 ± 3.0	54.9 ± 14.8
Dynamic shift						
PSEC(Gravity)	40.8 ± 0.9	29.2 ± 1.1	42.4 ± 1.0	57.2 ± 4.5	**26.8 ± 5.2**	71.8 ± 8.0
PSEC(Friction)	40.1 ± 1.2	31.1 ± 1.3	42.1 ± 1.0	61.7 ± 7.5	20.9 ± 4.6	**75.0 ± 12.1**
Body shift						
PSEC(Thigh)	**41.4 ± 0.3**	**32.3 ± 1.4**	**43.9 ± 2.5**	**64.96 ± 4.5**	25.5 ± 4.5	71.4 ± 14.3

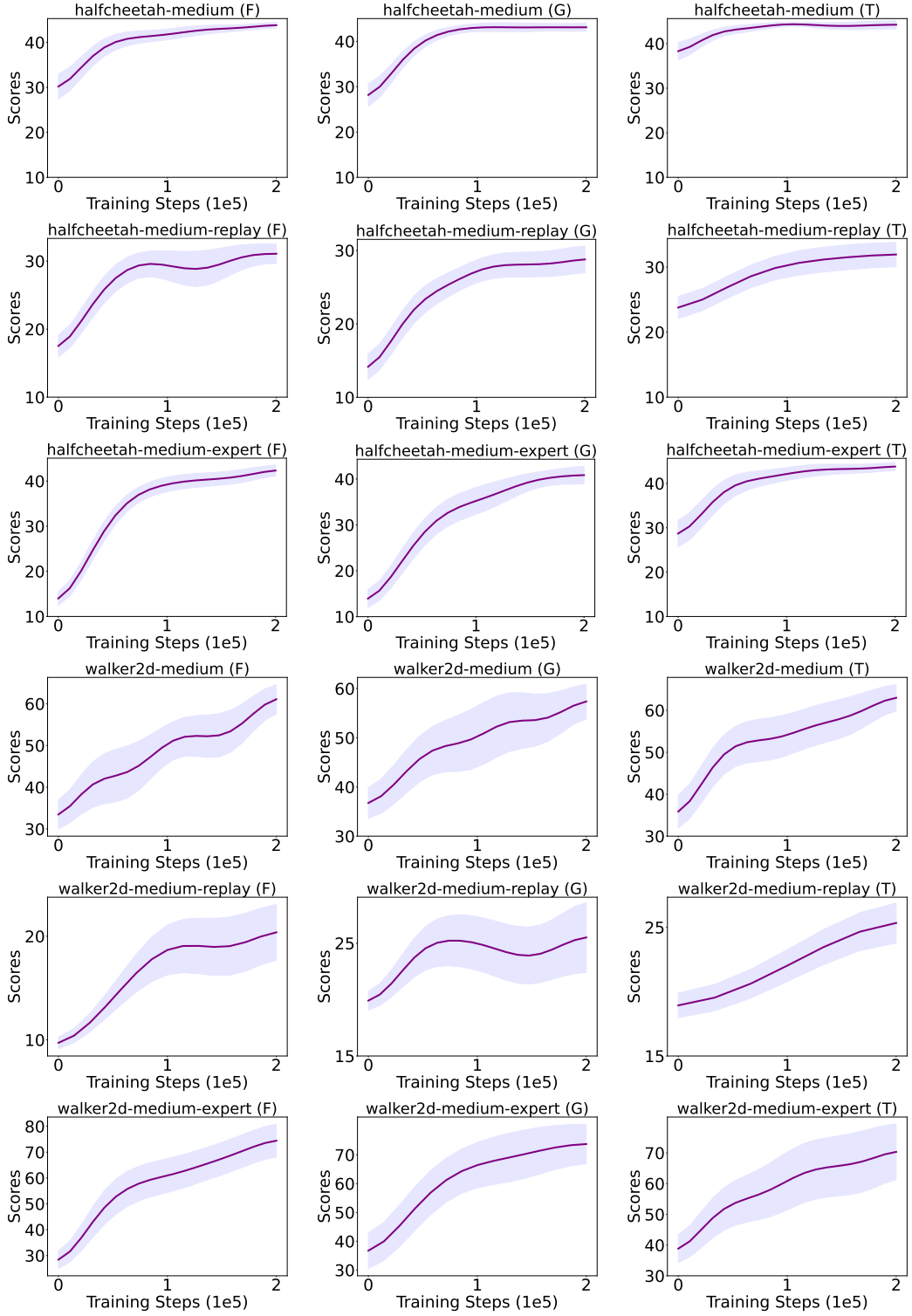


Figure 15: Results of performance conducted on dynamic shift and body shift tasks. The lines and shaded areas indicate the averages and standard deviations calculated over 5 random seeds.

Table 4: Hyperparameters for multi-objective composition tasks.

	Hyper-parameters	**Value**
shared hyperparameters	Normalized state	True
	Target update rate	1e-3
	Expectile τ	0.9
	Discount γ	0.99
	Actor learning rate	3e-4
	Critic learning rate	3e-4
	Number of added Gaussian noise T	5
π_0	hidden dim	256
	hidden layers	2
	activation function	ReLU
	Mini-batch size	2048
	Optimizer	Adam (Kingma & Ba, 2014)
	Training steps	1e6
π_1	$Q_r^*(s,a)$ hidden dim	256
	$Q_r^*(s,a)$ hidden layers	2
	$Q_r^*(s,a)$ activation function	ReLU
	$V_r^*(s)$ hidden dim	256
	$V_r^*(s)$ hidden layers	2
	$V_r^*(s)$ activation function	ReLU
	Actor hidden dim	256
	Actor hidden layers	2
	Actor Activation function	ReLU
	Mini-batch size	2048
	Optimizer	Adam
	Training steps	5e4
π_2	$Q_h^*(s,a)$ hidden dim	256
	$Q_h^*(s,a)$ hidden layers	2
	$Q_h^*(s,a)$ activation function	ReLU
	$V_h^*(s)$ hidden dim	256
	$V_h^*(s)$ hidden layers	2
	$V_h^*(s)$ Activation function	ReLU
	Actor hidden dim	256
	Actor hidden layers	2
	Actor Activation function	ReLU
	Mini-batch size	2048
	Optimizer	Adam
	Training steps	5e4
$\alpha(s;\theta)$	hidden dim	256
	hidden layers	2
	activation function	ReLU
	Mini-batch size	2048
	Optimizer	Adam
	Training steps	1e3
LoRA	rank n	8

Table 5: Hyperparameters for continual policy shift.

	Hyper-parameters	**Value**
shared hyperparameters	Normalized state	True
	Target update rate	1e-3
	Expectile τ	0.9
	Discount γ	0.99
	Actor learning rate	3e-4
	Critic learning rate	3e-4
	Number of added Gaussian noise T	5
π_0	hidden dim	256
	hidden layers	2
	activation function	ReLU
	Mini-batch size	1024
	Optimizer	Adam
	Training steps	1e6
π_1	hidden dim	256
	hidden layers	2
	Activation function	ReLU
	Mini-batch size	1024
	Optimizer	Adam
	Training steps	1e4
π_2	Actor hidden dim	256
	Actor hidden layers	2
	Actor Activation function	ReLU
	Mini-batch size	1024
	Optimizer	Adam
	Training steps	1e4
$\alpha(s;\theta)$	hidden dim	256
	hidden layers	2
	activation function	ReLU
	Mini-batch size	1024
	Optimizer	Adam
	Training steps	1e3
LoRA	rank n	8

Table 6: Hyperparameters for dynamic shift.

	Hyper-parameters	**Value**
shared hyperparameters	Normalized state	True
	Target update rate	1e-3
	Expectile τ	0.9
	Discount γ	0.99
	Actor learning rate	3e-4
	Critic learning rate	3e-4
	Number of added Gaussian noise T	5
π_0	hidden dim	256
	hidden layers	2
	activation function	ReLU
	Mini-batch size	1024
	Optimizer	Adam
	Training steps	1e6
π_1	$Q_r^*(s,a)$ hidden dim	256
	$Q_r^*(s,a)$ hidden layers	2
	$Q_r^*(s,a)$ activation function	ReLU
	$V_r^*(s)$ hidden dim	256
	$V_r^*(s)$ hidden layers	2
	$V_r^*(s)$ activation function	ReLU
	Actor hidden dim	256
	Actor hidden layers	2
	Actor Activation function	ReLU
	Mini-batch size	1024
	Optimizer	Adam
	Training steps	2e4
$\alpha(s;\theta)$	hidden dim	256
	hidden layers	2
	activation function	ReLU
	Mini-batch size	1024
	Optimizer	Adam
	Training steps	1e3
LoRA	rank n	8

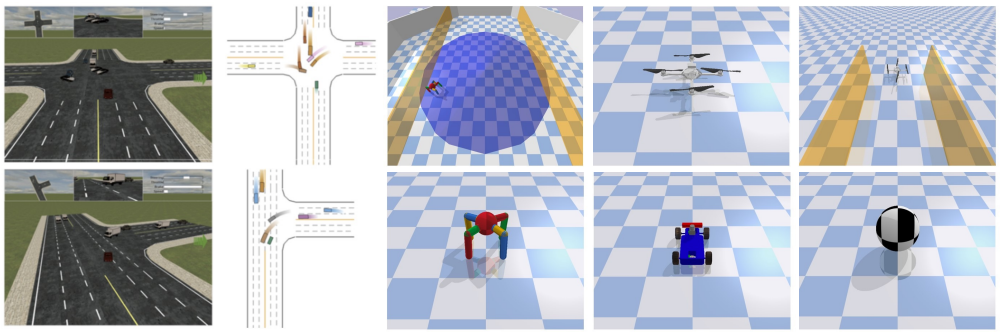
E MORE EXPERIMENTAL DETAILS

E.1 DESCRIPTION OF TASKS

We conduct experiments on 9 MetaDrive tasks and 8 Bullet-Safety-Gym tasks in the DSRL benchmark (Liu et al., 2023a). The visualization of the environments is shown in Figure 16. The tasks aim to learn policy from different level datasets such that the policy satisfies a safety constraint (normalized cost < 1) and achieves higher rewards.

MetaDrive. It leverages the Panda3D game engine to simulate realistic driving scenarios. The tasks are categorized as {`Road`}{`Vehicle`}, where "Road" encompasses three levels of difficulty for self-driving cars: easy, medium, and hard, while "Vehicle" represents four levels of surrounding traffic density: sparse, mean, and dense. In MetaDrive's autonomous driving tasks, costs are incurred from three safety-critical scenarios: (i) collision, (ii) out of road, and (iii) over-speed.

Bullet-Safety-Gym. The environments are built on the PyBullet physics simulator. They feature four types of agents: Ball, Car, Drone, and Ant, alongside two task types: Circle and Run. Tasks are designated as {`Agent`}{`Task`}, combining the agent and the corresponding task type.



(a) MetaDrive (b) Bullet-Safety-Gym

Figure 16: Visualization of the simulation environments and representative tasks of MetaDrive and Bullet-Safety-Gym. The figure is credited to Liu et al. (2023a).

E.2 ILLUSTRATION OF THE RECORDED DATA

To get a more intuitive look at the recorded data, we calculate the total reward and total cost for each trajectory in the datasets. These values are then plotted on a two-dimensional plane, where the x-axis corresponds to the total cost and the y-axis to the total reward. The results are shown in Figure 17 in the Appendix E of the paper. The plot highlights the dataset's diversity, particularly in how it captures a range of trajectory behaviors. The reward frontiers relative to cost illuminate the task's complexity, as the shape of these frontiers can significantly influence the challenges faced by offline learners. Trajectories offering high rewards but incurring high costs pose an alluring yet risky opportunity, often testing the balance between optimizing performance and maintaining safety constraints. This duality underscores the importance of robust algorithms that can navigate the trade-off effectively.

E.3 ADVANTAGE OF THE BENCHMARK

By generating diverse datasets across many environments with systematically varied complexities, the DSRL benchmark creates a rich and representative evaluation suite. This diversity ensures that our method is tested under a wide range of conditions, capturing different task structures, safety constraints, and levels of stochasticity. Meanwhile, the DSRL benchmark includes multiple objectives, making it well-suited for testing the flexibility and efficiency of our method in handling new tasks. Providing diverse datasets across varying difficulty levels and incorporating multiple optimization goals enables a comprehensive evaluation of our method's adaptability and performance across a broad spectrum of scenarios.

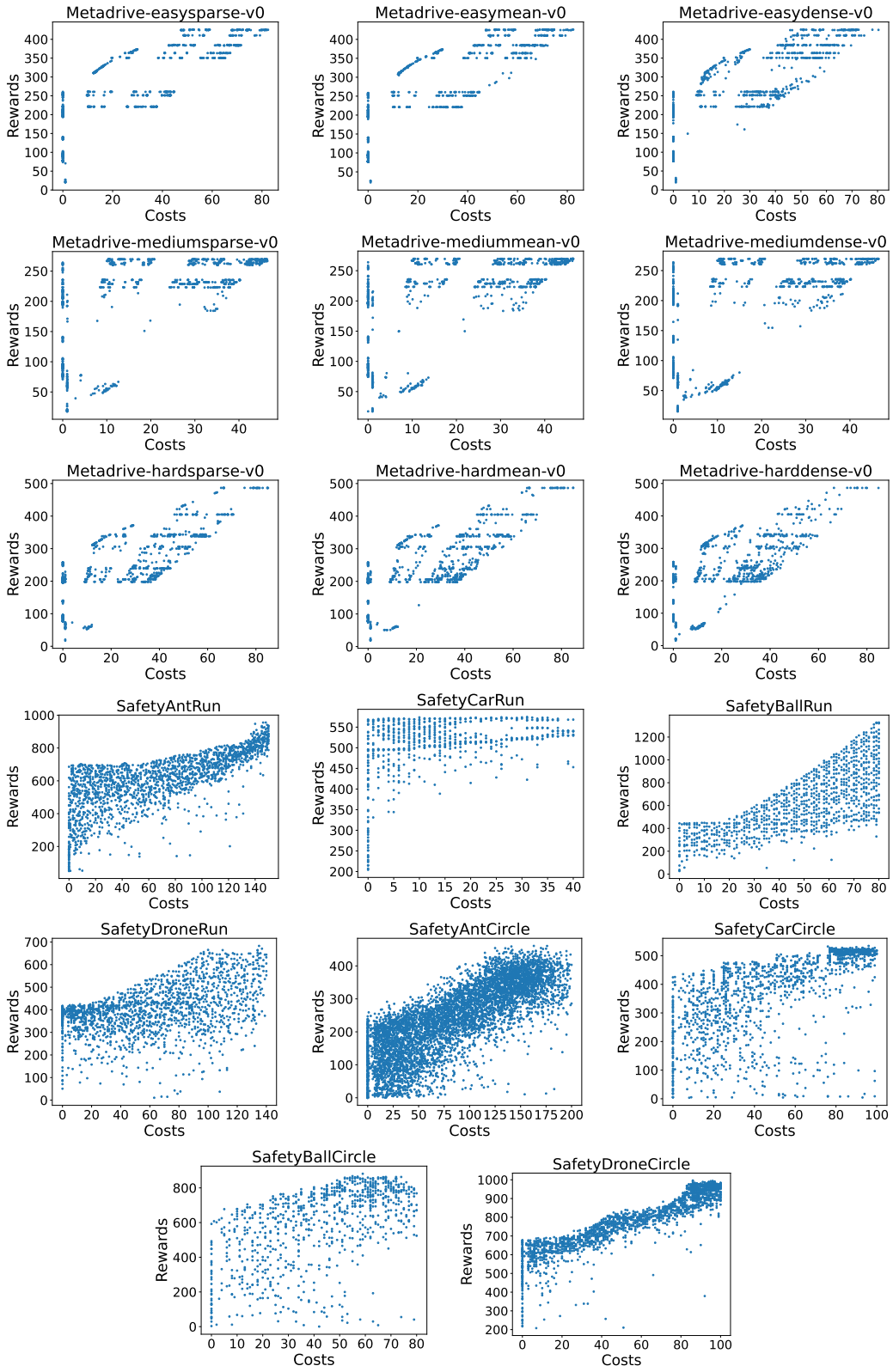


Figure 17: Illustration of the cost-reward plot for datasets from MetaDrive and Bullet-Safety-Gym.

F MORE EXPERIMENTS ON META-WORLD

To evaluate the effectiveness of PSEC on more complex experiments, we conduct experiments on Meta-World benchmark (Yu et al., 2020a), which consists of 50 diverse tasks for robotic manipulation, such as grasping, manipulating objects, opening/closing a window, pushing buttons, locking/unlocking a door, and throwing a basketball. We compare PSEC with the strong baseline L2M (Schmied et al., 2024). Next, we will elaborate on the three experiment settings in our paper.

Figure 18: Visualization of the simulation environments and representative tasks of Meta-World.

F.1 CONTINUAL LEARNING SETTING

Following Continual world (Wolczyk et al., 2021) and L2M (Schmied et al., 2024), we split the 50 tasks into 40 pre-training tasks and 10 fine-tuning unseen tasks (CW10). The training datasets are the same as the datasets collected by L2M. We train 10K steps per task in CW10, which is only 10% training steps of L2M, with a batch size of 1024. After every 10K update steps, we switch to the next task in the sequence. Then we evaluate it on all tasks in the task sequence. The results are shown in Table 7 and Table 8. We compare the performance of PSEC with L2M and other strong baselines. Thanks to the efficiency of skill composition in parameter space, PSEC can substantially outperform all L2M variants in a large margin, demonstrating that PSEC can achieve better performance on complex tasks.

Table 7: Success rates of different methods.

Methods	Success Rate
L2M	0.65
L2M-oracle	0.77
L2P-Pv2	0.40
L2P-PreT	0.34
L2P-PT	0.23
EWC	0.17
L2	0.10
PSEC (Ours)	**0.87**

Table 8: Performance of PSEC on different tasks.

Tasks	PSEC
peg-unplug-side-v2	0.87
window-close-v2	0.88
shelf-place-v2	0.85
push-v2	0.89
handle-press-side-v2	0.95
stick-pull-v2	0.74
push-back-v2	0.85
faucet-close-v2	0.92
push-wall-v2	0.86
hammer-v2	0.91
Mean	**0.87**

Table 9: Performance comparison on 18 pretrained tasks.

Tasks	Scratch	ASEC	NSEC	PSEC
peg-insert-side-v2	0.50	0.87	0.88	**0.90**
peg-unplug-side-v2	0.35	0.61	0.78	**0.86**
button-press-topdown-v2	0.71	0.88	0.88	**0.89**
push-back-v2	0.26	0.61	0.76	**0.88**
window-close-v2	0.65	0.84	0.84	**0.88**
door-open-v2	0.74	0.85	**0.86**	**0.86**
handle-press-v2	0.67	0.96	**0.97**	**0.97**
plate-slide-side-v2	0.27	0.23	0.53	**0.74**
handle-pull-side-v2	0.76	0.94	0.94	**0.95**
window-open-v2	0.87	0.75	0.88	**0.89**
door-close-v2	0.90	0.89	0.89	**0.91**
reach-v2	0.89	**0.95**	**0.95**	**0.95**
push-v2	0.15	0.58	0.81	**0.92**
stick-push-v2	0.44	0.54	0.17	**0.79**
drawer-close-v2	**0.97**	**0.97**	**0.97**	**0.97**
plate-slide-back-v2	0.90	0.94	0.94	**0.95**
coffee-button-v2	0.91	0.94	0.94	**0.95**
hand-insert-v2	0.30	0.68	0.63	**0.89**
Mean	0.62	0.78	0.81	**0.90**

F.2 Unseen tasks setting

To further evaluate the efficiency of PSEC on more challenging tasks, we pretrain on fewer (18) tasks and evaluate it on more (12) unseen tasks than the first setting. Firstly, we pretrain and finetune 18 tasks to obtain 18 LoRA modules. The performance on the 18 pretrained tasks is reported in Table 9. We compare the performance of PSEC with Scratch, ASEC and NSEC methods. The results show that PSEC can achieve enhanced skill learning even when the pretrained model is combined with one LoRA for each task if the skill is composed in parameter space. Then, we evaluate PSEC with the obtained 18 LoRA modules on the unseen tasks. For the unseen tasks, we conduct two types of experiments: few-shot setting and zero-shot setting.

Few-shot. We perform few-shot learning by training the context-aware modular for 1k steps using only 10% of the total available data for unseen tasks. This setup simulates scenarios with limited data on new tasks. The results, summarized in Table 10, demonstrate that PSEC achieves a high success rate on unseen tasks. This indicates that PSEC can effectively adapt to new tasks, showcasing its capability for rapid transfer learning and efficient adaptation in data-scarce environments.

Zero-shot. No data from the unseen tasks is used to train the context-aware modular. Instead, the modular is trained for 2k steps using datasets from 18 pre-trained tasks. It is then evaluated directly on 12 unseen tasks, utilizing 4 seeds and 10 episodes per task. The results are shown in Table 11. Interestingly, even without access to unseen task data during training, PSEC demonstrates strong performance on several tasks. Notably, PSEC substantially outperforms NSEC and ASEC on this zero-shot transfer setting, highlighting the advantages of skill compositions in parameter spaces over noise and action spaces. Overall, the results demonstrate PSEC's ability to effectively utilize knowledge from previously learned skills to achieve strong zero-shot transfer.

G More visualization of advantages of PSEC over NSEC and ASEC

To test whether the newly learned skills effectively utilize the shared knowledge of previous skills, we evaluate the running policy obtained through context-aware modular combined with standing and walking skills on three rewards: stand, walk, and run. If the running skill can still get a relatively high stand or walk reward, this represents the final combined running skill retaining these previous skills. We compare PSEC with other composition methods ASEC and NSEC. For each

Table 10: Few-shot performance comparison on 12 unseen tasks.

Tasks	ASEC	NSEC	PSEC
plate-slide-v2	0.14	0.66	**0.89**
handle-press-side-v2	0.73	0.65	**0.92**
button-press-wall-v2	0.09	0.03	**0.72**
button-press-topdown-wall-v2	0.87	0.88	**0.89**
push-wall-v2	0.57	0.68	**0.88**
reach-wall-v2	0.41	0.36	**0.90**
faucet-close-v2	0.41	0.49	**0.90**
button-press-v2	0.02	0.14	**0.23**
plate-slide-back-side-v2	0.17	0.19	**0.92**
handle-pull-v2	0.15	0.21	**0.93**
faucet-open-v2	0.14	0.16	**0.89**
stick-pull-v2	0.00	0.00	**0.32**

Table 11: Zero-shot performance comparison on 12 unseen tasks.

Tasks	ASEC	NSEC	PSEC
plate-slide-v2	0.03	0.00	**0.15**
handle-press-side-v2	0.50	0.60	**0.62**
button-press-wall-v2	0.00	0.00	**0.40**
button-press-topdown-wall-v2	0.85	0.87	**0.89**
push-wall-v2	0.53	0.53	**0.71**
reach-wall-v2	0.34	0.05	**0.90**
faucet-close-v2	0.00	0.00	**0.16**
button-press-v2	0.00	0.00	**0.15**
plate-slide-back-side-v2	0.00	0.00	0.00
handle-pull-v2	0.00	0.00	0.00
faucet-open-v2	0.00	0.00	**0.77**
stick-pull-v2	0.00	0.00	0.00

method, we rollout 10K steps and record the three rewards. The summarized rewards can be found in Figure 19. The results show that PSEC achieves high rewards across all tasks, whereas NSEC and ASEC cannot, demonstrating that the PSEC's running skill retains behaviors from walking and standing and suggesting superior skill sharing of PSEC compared to NSEC and ASEC.

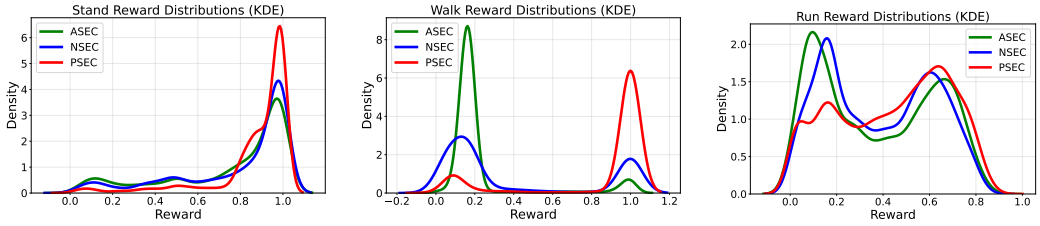


Figure 19: We evaluate the final running policies of PSEC, NSEC and ASEC with the "stand," "walk," and "run" rewards with 10 episodes and 3 random seeds. Then we plot the reward distribution by kernel density estimation (KDE). Each curve represents the probability density of rewards obtained for a specific reward. The results show that PSEC achieves high rewards across all tasks, whereas NSEC and ASEC cannot, demonstrating that the PSEC's running skill retains behaviors from walking and standing and suggesting superior skill sharing of PSEC compared to NSEC and ASEC.