Continual Knowledge Adaptation for Reinforcement Learning

Jinwu Hu[1 2*] **Zihao Lian**[1*] **Zhiquan Wen**[1*] **Chenghao Li**[1 2]
Guohao Chen[1 2] **Xutao Wen**[1] **Bin Xiao**[3†] **Mingkui Tan**[1 4†]
[1]South China University of Technology, [2]Pazhou Laboratory,
[3]Chongqing University of Posts and Telecommunications
[4]Key Laboratory of Big Data and Intelligent Robot, Ministry of Education,

Abstract

Reinforcement Learning enables agents to learn optimal behaviors through interactions with environments. However, real-world environments are typically non-stationary, requiring agents to continuously adapt to new tasks and changing conditions. Although Continual Reinforcement Learning facilitates learning across multiple tasks, existing methods often suffer from catastrophic forgetting and inefficient knowledge utilization. To address these challenges, we propose **C**ontinual **K**nowledge **A**daptation for **R**einforcement **L**earning (**CKA-RL**), which enables the accumulation and effective utilization of historical knowledge. Specifically, we introduce a Continual Knowledge Adaptation strategy, which involves maintaining a task-specific knowledge vector pool and dynamically using historical knowledge to adapt the agent to new tasks. This process mitigates catastrophic forgetting and enables efficient knowledge transfer across tasks by preserving and adapting critical model parameters. Additionally, we propose an Adaptive Knowledge Merging mechanism that combines similar knowledge vectors to address scalability challenges, reducing memory requirements while ensuring the retention of essential knowledge. Experiments on three benchmarks demonstrate that the proposed CKA-RL outperforms state-of-the-art methods, achieving an improvement of 4.20% in overall performance and 8.02% in forward transfer. The source code is available at https://github.com/Fhujinwu/CKA-RL.

1 Introduction

Reinforcement Learning (RL) has emerged as a powerful paradigm in machine learning, enabling agents to learn optimal behaviors through interactions with dynamic environments [25, 11, 45]. RL has achieved significant success in fields such as robotic control [24, 36], embodied intelligence [8, 16], and natural language processing [18, 15, 19]. However, traditional RL usually assumes a static environment where tasks and data distributions remain fixed, intending to solve a single, well-defined problem [33]. In contrast, real-world environments are typically non-stationary, requiring agents to continuously adapt to new tasks and evolving conditions. In light of this, Continual Reinforcement Learning (CRL) [26] has been introduced to enable agents to adapt and maintain performance across multiple tasks, facilitating more robust decision-making in dynamic environments [14, 2, 50].

Unfortunately, CRL still faces several challenges, which are as follows. *1) Cross-task conflict:* In continual learning, tasks may share certain structures or knowledge while also having incompatible goals or constraints. This interplay complicates the direct reuse of previously learned information [28].

*Equal contribution. Email: fhujinwu@gmail.com, lianzihaolzh@gmail.com, sewenzhiquan@gmail.com
†Corresponding author. Email: mingkuitan@scut.edu.cn, xiaobin@cqupt.edu.cn

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

2) Catastrophic forgetting: When learning new tasks, the agent may overwrite or distort previously acquired knowledge, leading to a loss of performance on earlier tasks [33, 50].

Recently, several attempts have been proposed to enable agents to continuously learn across multiple tasks while maintaining or enhancing their performance on previously acquired tasks [60, 20]. The existing methods can be broadly categorized into four main groups [50]. The *regularization-based methods* [21, 38] introduce regularization terms into the learning objective that penalize large updates to important parameters for previously learned tasks. The *rehearsal-based methods* [10, 58, 57] mitigate forgetting by storing previous experiences in memory and periodically replaying them during the learning of new tasks. By leveraging past experiences, these methods help reduce short-term biases and improve task performance across the sequence. The *architecture-based methods* [49, 33, 5, 40] focus on learning a shared structure, such as modularity or composition, to facilitate continual learning. They reuse parts of previous solutions by forming abstract concepts or skills. The *meta-learning based methods* [41, 12, 31, 9] improve the learning efficiency of agents by utilizing past successes and failures to refine their optimization processes. This creates an inductive bias that enhances sample efficiency and adaptability in acquiring new behaviors.

Although there has been significant progress in existing methods, they still face several limitations as follows. **Firstly**, existing methods, such as PackNet [34], often fail to effectively address task dependencies and conflicts, which makes it challenging to transfer knowledge across tasks without interference or degradation in performance. **Secondly**, most methods still exhibit substantial performance degradation on previously learned tasks upon acquisition of new ones. This issue is particularly pronounced in methods that rely on regularization, as they may struggle to efficiently retain knowledge from earlier tasks as the task sequence lengthens. **Lastly**, many methods face scalability issues, especially as the number of tasks grows significantly, leading to increased memory and computational costs, such as CompoNet [33].

To address these limitations, we propose **C**ontinual **K**nowledge **A**daptation for **R**einforcement **L**earning (**CKA-RL**), which enables the accumulation and effective utilization of historical knowledge, thereby accelerating learning in new tasks and explicitly reducing performance degradation on previous tasks. Specifically, we assume that the agent acquires a unique knowledge vector for each task during continual learning. Based on this, we propose a Continual Knowledge Adaptation strategy, which involves maintaining a task-specific knowledge vector pool and dynamically using historical knowledge to adapt the agent to a new task. This method mitigates catastrophic forgetting and facilitates the efficient transfer of knowledge across tasks by preserving and adapting crucial model parameters. Furthermore, we introduce an Adaptive Knowledge Merging mechanism that clusters and consolidates similar knowledge vectors to address scalability issues, reducing memory requirements while ensuring the retention of essential information.

Main novelty and contributions. 1) We propose a novel continual reinforcement learning method, called CKA-RL. It enables agents to reuse knowledge from previously learned tasks, mitigating catastrophic forgetting, and leveraging this historical knowledge to enhance the learning efficiency. **2)** We propose Continual Knowledge Adaptation strategy, which dynamically adapts historical knowledge to a new task. To reduce storage requirements, we introduce an Adaptive Knowledge Merging mechanism that combines similar knowledge vectors, addressing scalability issues. **3)** Experiments demonstrate that the proposed method outperforms state-of-the-art methods on three benchmarks, achieving an improvement of 4.20% in performance and 8.02% in forward transfer.

2 Related Work

Continual Reinforcement Learning (CRL) aims to enable agents to continuously learn and optimize strategies in dynamic environments, improving their ability to adapt to environmental changes and achieve goals more efficiently. The existing methods can be broadly categorized into four main types [50]: regularization-based methods, rehearsal-based methods, architecture-based methods, and meta-learning-based methods, each of which is detailed as:

Regularization-based Methods. These methods [21, 38] add regularization terms to the training process to balance learning new and old tasks without storing models of old tasks. ITER [21] periodically distills the current policy and value function into a newly initialized network during training and imitates the teacher network using a linear combination of loss terms to enhance model generalization. TRAC [38] adaptively adjusts the regularization strength based on online convex

optimization theory to prevent excessive weight drift, thereby reducing plasticity loss and enabling rapid adaptation to new distribution changes. Anand et al. [3] propose a method that decomposes the value function into permanent and transient components to address the stability-plasticity dilemma in continual reinforcement learning, enabling efficient adaptation and control in dynamic environments.

Rehearsal-based Methods. These methods [10, 58, 57, 13] utilize experience replay, generation replay, parameter replay, *etc.*, to store past experiences and prevent catastrophic forgetting in agents. RECALL [57] focuses on improving plasticity and stability in continuous reinforcement learning through multi-head neural network training, coupled with adaptive normalization and policy distillation techniques. IQ [58] employs a context partitioning strategy based on online clustering, combined with multi-head networks and knowledge distillation technology, to reduce interference between different state distributions. DRAGO [13] leverages synthetic experience replay and exploration-based memory recovery to retain knowledge across tasks, mitigating catastrophic forgetting without requiring the storage of past task data.

Architecture-based Methods. These methods [49, 33, 5, 40, 35] concentrate on learning a policy with a set of shared parameters to handle all incremental tasks, including parameter allocation, model reorganization, and modular networks. For example, MaskNet [5] employs a learnable modulation mask to isolate the parameters of different tasks on a fixed neural network and accelerates learning new tasks by linearly combining masks from previous tasks. CompoNet [33] uses a modular architecture with self-composing policies to enable efficient knowledge transfer and scalable learning in continual reinforcement learning. COMP [35] decomposes complex tasks into multiple subtasks corresponding to different neural modules. These modules can be combined to form a complete policy. REWIRE [46] redefines the connection methods of the neural network and reorders the neurons in each layer to achieve additional plasticity in unstable environments.

Meta-Learning based Methods. These methods [41, 12, 31, 9] assist agents in rapidly adapting to new tasks by simulating the reasoning phase during training. MB-MPO [9] is a model-based method that meta-learns policies robust to ensemble dynamics, reducing reliance on accurate models. MAML [12] is a model-agnostic meta-learning algorithm that explicitly trains models to achieve strong generalization from a small number of gradient updates. ESCP [31] enhances the robustness and responsiveness of context encoding, significantly accelerating adaptation in reinforcement learning involving sudden environmental changes.

3 Problem Formulation

The **standard Reinforcement Learning (RL)** problem [47] is modeled as a **Markov Decision Process (MDP)**, with tuple $M = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, where $\mathcal{S}$ is state space, $\mathcal{A}$ is action space, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. At each time step $t \in \mathbb{N}$, the agent observes the current state $s_t \in \mathcal{S}$ and takes an action $a_t \sim \pi(\cdot|s_t)$, where $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the policy. The environment transitions to a new state $s_{t+1} \sim p(\cdot|s_t, a_t)$, and the agent receives a reward $r(s_t, a_t)$. The goal of RL is to find an optimal policy π^* that maximizes the expected discounted return as follows:

$$\max_{\pi} \ \mathbb{E}_{\pi,p}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)], \tag{1}$$

where the expectation is over trajectories generated by policy π and state transitions p. The discount factor γ controls the trade-off between immediate and future rewards.

Given N tasks, the agent is expected to maximize the RL objective for each task in $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_N\}$. Each task τ_i is associated with a distinct MDP $M_i = \langle \mathcal{S}_i, \mathcal{A}_i, p_i, r_i, \gamma_i \rangle$. An RL problem is considered an instance of **Continual Reinforcement Learning (CRL)** if agents are required to continuously learn. Following the definition presented by *Abel et al.* [2], we have the following concept: an RL problem defined by a tuple (e, ν, Λ), where e is the environment, ν is the performance function, and Λ is the set of all agents, the problem is a CRL problem if $\forall_{\lambda^* \in \Lambda^*} \lambda^* \overset{\mathcal{G}}{\not\rightsquigarrow}$ (never reaches) Λ_B, where $\Lambda_B \subset \Lambda$ is a basis such that $\Lambda_B \vdash_e$ (generates) Λ (in e) and $\Lambda^* = \arg\max_{\lambda \in \Lambda} \nu(\lambda, e)$. In other words, the best agents continue to search indefinitely over the basis Λ_B and do not converge to a fixed policy. According to our CKA-RL, whenever the number of tasks $|\mathcal{T}|$ increases, particularly when τ_{N+1} is added, a new knowledge vector $\boldsymbol{v}_{N+1}$ will be introduced. Therefore, the new task will ensure that $\lambda \overset{\mathcal{G}}{\not\rightsquigarrow} \Lambda_B$, and thus continues to learn.

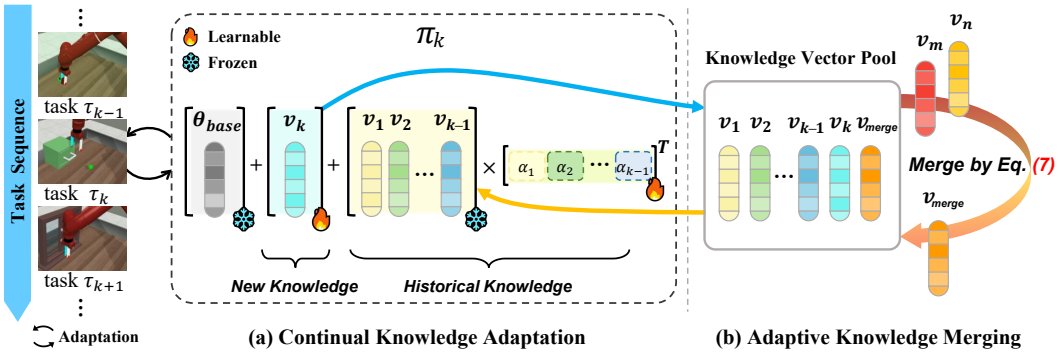


Figure 1: An illustration of the CKA-RL. When learning a new task τ_k, the agent π_k adapts by using historical knowledge vectors $\mathcal{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k-1}\}$ with $|\mathcal{V}| \leq K_{\max}$, and learning a new task-specific knowledge vector $\boldsymbol{v}_k$, while the base parameter $\boldsymbol{\theta}_{\text{base}}$ remains fixed. After training, the new knowledge vector $\boldsymbol{v}_k$ is added to a knowledge vector pool $\mathcal{V}$. To maintain memory efficiency, we merge the most similar pairs of knowledge vectors $(\boldsymbol{v}_m, \boldsymbol{v}_n)$ in $\mathcal{V}$ into $\boldsymbol{v}_{\text{merge}}$ when $|\mathcal{V}| > K_{\max}$, thus ensuring essential knowledge is retained while supporting scalable continual learning across sequential tasks.

4 Continual Knowledge Adaptation for Reinforcement Learning

In this paper, we propose **C**ontinual **K**nowledge **A**daptation for **R**einforcement **L**earning (**CKA-RL**), which enables the accumulation and effective utilization of historical knowledge, thereby accelerating learning in new tasks while mitigating catastrophic forgetting. As shown in Figure 1, our proposed CKA-RL is composed of three key components: **1)** *Knowledge Vectors*: These vectors capture task-specific knowledge that is used to adapt the model to new tasks, as detailed in Sec. 4.1. **2)** *Continual Knowledge Adaptation*: This component dynamically uses historical knowledge to adapt the agent to a new task, enabling efficient transfer and retention of knowledge (see in Sec. 4.2). **3)** *Adaptive Knowledge Merging*: This component merges similar knowledge vectors to address scalability issues, reducing memory requirements while ensuring the retention of essential knowledge (see in Sec. 4.3). The pseudo-code of CKA-RL is summarized in Algorithm 1.

4.1 Knowledge Vectors

In continual reinforcement learning, leveraging knowledge from previous tasks in dynamic environments is crucial to enhance agent performance in subsequent tasks and mitigate catastrophic forgetting. Therefore, we introduce the concept of knowledge vectors to facilitate continual learning, inspired by model editing techniques [22]. Specifically, the knowledge vector, denoted as $\boldsymbol{v}_k \in \mathbb{R}^d$, represents the learned parameters that adapt the pre-trained model to a specific task τ_k, where d denotes the dimensionality of the model parameter. These vectors can be combined linearly with historical knowledge vectors to enable cross-task knowledge transfer.

Formally, given the pre-trained model weights $\boldsymbol{\theta}_{\text{base}} \in \mathbb{R}^d$, the knowledge vector $\boldsymbol{v}_k$ is optimized during task-specific training to capture incremental adaptations. The final task-specific parameter $\boldsymbol{\theta}_k$ is generated through a combination of the base parameter $\boldsymbol{\theta}_{\text{base}}$, historical knowledge matrix $\boldsymbol{V}_{k-1} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{k-1}]$, and the current task vector $\boldsymbol{v}_k$, which is as follows:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{\text{base}} + \sum_{i=1}^{k-1} \boldsymbol{v_i} + \boldsymbol{v}_k, \tag{2}$$

A linear combination of multiple knowledge vectors enables the model to reuse knowledge from previous tasks [27, 6]. This highlights the potential of knowledge vectors to facilitate continual knowledge adaptation, effectively tackling key challenges in continual learning.

4.2 Continual Knowledge Adaptation

The knowledge vector stores task-specific knowledge, and the knowledge from historical tasks facilitates the transfer of policies to new tasks. Under the Continual RL setting, assuming that the

policy π shares a consistent observation space $\mathcal{O}$ and action space $\mathcal{A}$, the policy can be represented using a unified architecture parameterized by $\boldsymbol{\theta}$. To achieve cross-task knowledge transfer, we propose **Continual Knowledge Adaptation** strategy. Specifically, it is first necessary to construct shared base model parameter $\boldsymbol{\theta}_{\text{base}}$. The policy parameter is initialized as $\boldsymbol{\theta}_1$ and trained on the initial task τ_1, with the optimization objective being to maximize the expected discounted return. The optimized parameter $\boldsymbol{\theta}_1$ is then used as the base parameter $\boldsymbol{\theta}_{\text{base}}$, which encompasses general feature representations and serves as the foundation for subsequent knowledge adaptation.

The knowledge from previous tasks is stored as knowledge vectors in a knowledge vector pool $\mathcal{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{k-1}\}$, where each $\boldsymbol{v_i}$ represents the knowledge vector for task τ_i. For flexibility, we define the null knowledge as $\boldsymbol{v}_1 = \boldsymbol{\theta}_1 - \boldsymbol{\theta}_{\text{base}} = \mathbf{0}$ and include it in $\mathcal{V}$. When learning a new task τ_k, the model parameter is generated by adapting the historical knowledge vectors in $\mathcal{V}$ with the new task-specific knowledge $\boldsymbol{v}_k$, which is as follows:

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{\text{base}} + \sum_{j=1}^{k-1} \alpha_j^k \boldsymbol{v}_j + \boldsymbol{v}_k, \quad \text{where } \sum_{j=1}^{k-1} \alpha_j^k = 1. \tag{3}$$

The $\boldsymbol{\alpha}_k = [\alpha_1^k, \ldots, \alpha_{k-1}^k]$ represents the normalized adaptation factors for task τ_k, where $\sum_{j=1}^{k-1} \alpha_j^k = 1$. These factors are derived from learnable parameter $\boldsymbol{\beta}_k = [\beta_1^k, \ldots, \beta_{k-1}^k]$ through the softmax function:

$$\alpha_j^k = \frac{\exp \beta_j^k}{\sum_{i=1}^{k-1} \exp(\beta_i^k)}. \tag{4}$$

Here, $\boldsymbol{\beta}_k$ controls the contribution of each historical knowledge vector $\boldsymbol{v}_j$ to the new task τ_k, and $\boldsymbol{v}_k$ is the optimizable knowledge vector for the current task. Notably, we initialize $\boldsymbol{v}_k$ to $\mathbf{0}$ to allow the model to gradually learn the task-specific knowledge vector. This prevents large initial adjustments, ensuring stable learning without disrupting previously learned tasks. By setting $\alpha_1 = 1$, the model can disregard previously learned knowledge when it is not beneficial for learning new tasks. This design enables continual adaptation of historical knowledge while preserving task-specific knowledge. During the task training, the base parameter $\boldsymbol{\theta}_{\text{base}}$ is fixed to maintain the foundation for knowledge adaptation, and only the adaptation factors $\boldsymbol{\alpha}_k$ (derived from $\boldsymbol{\beta}_k$) and the current task knowledge vector $\boldsymbol{v}_k$ are optimized. Upon the completion of the training process, the obtained knowledge vector $\boldsymbol{v}_k$ is added to the knowledge vector pool $\mathcal{V}$, thereby facilitating efficient knowledge transfer within dynamic environments.

4.3 Adaptive Knowledge Merging

Although the proposed Continual Knowledge Adaptation strategy can accelerate the agent learning process and mitigate catastrophic forgetting by utilizing knowledge vectors from previous tasks, maintaining all vectors becomes impractical as the number of stored knowledge vectors increases. Therefore, we propose **Adaptive Knowledge Merging** to address scalability issues while preserving essential knowledge. Specifically, we merge similar knowledge vectors into compact representations, ensuring that the memory footprint remains manageable. The pairwise similarity between knowledge vectors is measured using normalized cosine similarity as follows:

$$S_{ij} = \frac{\boldsymbol{v}_i \cdot \boldsymbol{v}_j}{\|\boldsymbol{v}_i\| \|\boldsymbol{v}_j\|}, \tag{5}$$

where $S_{ij} \in [-1, 1]$ measures the directional alignment between vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. A value of $S_{ij} = -1$ indicates that the two knowledge vectors are in direct conflict. Conversely, a value closer to 1 signifies strong alignment, suggesting consistency in the knowledge they encode. This similarity metric effectively captures the functional similarities in how vectors modify the base policy. When the number of knowledge vectors exceeds the maximum capacity $K_{\max}$, which serves as a hyper-parameter for controlling memory usage, we compute the pairwise similarity matrix S across all stored knowledge vectors. Then, we identify the most similar pair $(\boldsymbol{v}_m, \boldsymbol{v}_n)$ by finding the pair with the highest similarity score, which is as follows:

$$(\boldsymbol{v}_m, \boldsymbol{v}_n) = \arg\max_{i,j} S_{i,j}. \tag{6}$$

These vectors $\boldsymbol{v}_m$ and $\boldsymbol{v}_n$ are merged by averaging:

$$\boldsymbol{v}_{\text{merge}} = \frac{1}{2}(\boldsymbol{v}_m + \boldsymbol{v}_n), \tag{7}$$

Algorithm 1 The Pipeline of the Proposed CKA-RL

Input: Task sequence $\mathcal{T}=\{\tau_1,\ldots,\tau_N\}$, Initial model parameters $\boldsymbol{\theta}_1$, Knowledge pool $\mathcal{V}=\{\mathbf{0}\}$

Base Policy Learning: $\boldsymbol{\theta}_{\text{base}} \leftarrow \arg\max_{\theta} \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r_1(s_t,a_t)\right]$

for $k \leftarrow 2,\ldots,N$ **do**

  Task Initialization: $\boldsymbol{v}_k \leftarrow \mathbf{0}$, $\boldsymbol{\beta}_k \sim \mathcal{N}(0,1)$, $\boldsymbol{\alpha}_k \leftarrow \text{softmax}(\boldsymbol{\beta}_k)$

  Policy Construction: $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}_{\text{base}} + \sum_{j=1}^{k-1}\alpha_j^k \boldsymbol{v}_j + \boldsymbol{v}_k$

  Policy Optimization: $\arg\max_{\boldsymbol{v}_k,\boldsymbol{\beta}_k} \mathbb{E}_{\pi_{\theta'}}\left[\sum_{t=0}^{\infty}\gamma^t r_k(s_t,a_t)\right]$

  Knowledge Preservation: $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}'$, $\mathcal{V} \leftarrow \mathcal{V}\cup\{\boldsymbol{v}_k\}$

  if $|\mathcal{V}| > K_{\max}$ **then**

    Compute similarity matrix S using Eq.(5)

    $(\boldsymbol{v}_m,\boldsymbol{v}_n) \leftarrow \arg\max_{i,j} S_{i,j}$

    Merge $\boldsymbol{v}_m$ and $\boldsymbol{v}_n$ into $\boldsymbol{v}_{\text{merge}}$ using Eq.(7)

    Update $\mathcal{V}$: $\mathcal{V} \leftarrow (\mathcal{V}\setminus\{\boldsymbol{v}_m,\boldsymbol{v}_n\})\cup\{\boldsymbol{v}_{\text{merge}}\}$

Output: Learned policies $\{\pi_{\theta_1},\ldots,\pi_{\theta_N}\}$

where $\boldsymbol{v}_{\text{merge}}$ is added to the knowledge vector pool $\mathcal{V}$. The merged vector $\boldsymbol{v}_{\text{merge}}$ replaces $\boldsymbol{v}_m$ and $\boldsymbol{v}_n$, reducing memory requirement while maintaining the knowledge from both vectors. It ensures that the knowledge vector pool remains compact and efficient while preserving knowledge from previous tasks. Through iterative merging, this strategy dynamically maintains a balance between memory efficiency and knowledge retention, addressing scalability issues in dynamic environments.

Remark: For a detailed mathematical analysis supporting the performance and stability of the proposed method, please refer to Appendix A.

5 Experiments

5.1 Experimental Settings

Benchmarks. We follow the experimental settings established in prior work [33] and compare CKA-RL with SOTA CRL methods across three distinct dynamic task sequences, including 1) Meta-World [56], 2) Freeway [32], and 3) SpaceInvaders [32]. These sequences are designed to evaluate the robustness and generalization capabilities of different methods under varying levels of task complexity and action space characteristics. More details can be seen in Appendix C.

Metrics. Following standard evaluation metrics in CRL [55], we report key metrics, including average performance and forward transfer. The **average performance** at step t, denoted as $P(t)$:

$$P(t) = \frac{1}{N}\sum_{i=1}^{N} p_i(t). \tag{8}$$

where $p_i(t) \in [0,1]$ is the success rate on task i at step t, and each of the N tasks is trained for Δ steps, where N is the number of tasks, so the total number of steps is $T = N\cdot\Delta$. Its final value $P(T)$, serving as a conventional evaluation metric in CRL [33, 53], effectively captures the model's stable performance across dynamic environments. The **forward transfer** measures the extent to which a CRL method is able to transfer knowledge across tasks. It is computed as the normalized area between the training curve of the measured run and the training curve of a reference model trained from scratch. Let $p_i^b \in [0,1]$ denote the reference performance. The forward transfer on task i, denoted as FT_i, is defined as follows:

$$FT_i = \frac{AUC_i - AUC_i^b}{1 - AUC_i^b}, AUC_i = \frac{1}{\Delta}\int_{(i-1)\cdot\Delta}^{i\cdot\Delta} p_i(t)\,dt, AUC_i^b = \frac{1}{\Delta}\int_0^{\Delta} p_i^b(t)\,dt. \tag{9}$$

The average forward transfer for all tasks FT is defined as:

$$FT = \frac{1}{N}\sum_{i=1}^{N} FT_i. \tag{10}$$

Table 1: Experimental results comparing the proposed method with nine SOTA methods across Meta-World, SpaceInvaders, and Freeway environments. We report the results for average performance (PERF.) and forward transfer (FWT.), with the best results highlighted in **bold**. The proposed CKA-RL achieves superior performance and forward transfer in all three sequences.

Method	Meta-World		SpaceInvaders		Freeway		Average	
	PERF.	FWT.	PERF.	FWT.	PERF.	FWT.	PERF.	FWT.
Baseline	$0.4191_{\pm 0.49}$	$0.0000_{\pm 0.00}$	$0.6314_{\pm 0.27}$	$0.0000_{\pm 0.00}$	$0.1247_{\pm 0.24}$	$0.0000_{\pm 0.00}$	$0.3917_{\pm 0.35}$	$0.0000_{\pm 0.00}$
FT-1	$0.0313_{\pm 0.09}$	$-0.2142_{\pm 0.38}$	$0.4412_{\pm 0.50}$	$0.6864_{\pm 0.25}$	$0.1512_{\pm 0.36}$	$0.6935_{\pm 0.09}$	$0.2079_{\pm 0.36}$	$0.3886_{\pm 0.27}$
FT-N	$0.3774_{\pm 0.48}$	$-0.2142_{\pm 0.38}$	$0.9785_{\pm 0.04}$	$0.6864_{\pm 0.32}$	$0.7532_{\pm 0.16}$	$0.6935_{\pm 0.15}$	$0.7030_{\pm 0.29}$	$0.3886_{\pm 0.30}$
ProgNet	$0.4157_{\pm 0.49}$	$-0.0379_{\pm 0.13}$	$0.3757_{\pm 0.27}$	$-0.0075_{\pm 0.22}$	$0.3125_{\pm 0.27}$	$0.1938_{\pm 0.31}$	$0.3680_{\pm 0.36}$	$0.0495_{\pm 0.23}$
PackNet	$0.2523_{\pm 0.40}$	$-0.6721_{\pm 1.40}$	$0.2299_{\pm 0.30}$	$-0.0750_{\pm 0.13}$	$0.2767_{\pm 0.36}$	$0.1970_{\pm 0.32}$	$0.2530_{\pm 0.36}$	$-0.1834_{\pm 0.83}$
MaskNet	$0.3263_{\pm 0.47}$	$-0.3695_{\pm 0.45}$	$0.0000_{\pm 0.00}$	$-0.3866_{\pm 0.53}$	$0.0644_{\pm 0.17}$	$-0.0503_{\pm 0.12}$	$0.1302_{\pm 0.29}$	$-0.2688_{\pm 0.41}$
CReLUs	$0.3789_{\pm 0.47}$	$-0.0089_{\pm 0.24}$	$0.8873_{\pm 0.10}$	$0.5308_{\pm 0.29}$	$0.7835_{\pm 0.13}$	$0.7303_{\pm 0.12}$	$0.6832_{\pm 0.29}$	$0.4174_{\pm 0.23}$
CompoNet	$0.4131_{\pm 0.50}$	$-0.0055_{\pm 0.20}$	$0.9828_{\pm 0.02}$	$0.6963_{\pm 0.32}$	$0.7629_{\pm 0.12}$	$0.7115_{\pm 0.10}$	$0.7196_{\pm 0.30}$	$0.4674_{\pm 0.23}$
CbpNet	$0.4368_{\pm 0.50}$	$-0.0826_{\pm 0.22}$	$0.8392_{\pm 0.11}$	$0.4844_{\pm 0.28}$	$0.7678_{\pm 0.10}$	$0.7201_{\pm 0.07}$	$0.6813_{\pm 0.30}$	$0.3740_{\pm 0.21}$
CKA-RL	$\mathbf{0.4642}_{\pm 0.50}$	$\mathbf{-0.0032}_{\pm 0.21}$	$\mathbf{0.9928}_{\pm 0.01}$	$\mathbf{0.7749}_{\pm 0.20}$	$\mathbf{0.7923}_{\pm 0.10}$	$\mathbf{0.7429}_{\pm 0.07}$	$\mathbf{0.7498}_{\pm 0.29}$	$\mathbf{0.5049}_{\pm 0.17}$

Methods. We compare the CKA-RL with nine SOTA methods. 1) Baseline. 2) FT-1 (Fine-Tuning Single Model) [53]. 3) FT-N (Fine-Tuning with Model Preservation) [53]. 4) ProgNet [40]. 5) PackNet [34]. 6) MaskNet [5]. 7) CReLUs [1]. 8) CompoNet [33]. 9) CbpNet [11].

Implementation Details. We follow the prior work [33], employing SAC [17] for Meta-World and PPO [43] for Freeway and SpaceInvaders. For high-dimensional Atari inputs (210×160 RGB), a CNN encoder maps images to compact latent features. All tasks are trained for $\Delta = 1M$ steps. We use Adam (momentum 0.9, second moment 0.999), with batch sizes $1024/128$ and learning rates $2.5 \times 10^{-4}/1 \times 10^{-3}$ for PPO/SAC. The discount factor is $\gamma = 0.99$. For SAC, the action standard deviation is constrained to $[e^{-20}, e^{2}]$, with target smoothing coefficient 5×10^{-3}, auto-tuned entropy coefficient 0.2, and action noise clipped to 0.5. Learning starts after 5×10^{3} steps using 10^{4} random actions for exploration. Policy and target networks are updated every 2 and 1 steps, respectively, using 3-layer MLPs with 256 hidden units. For PPO, we apply GAE with $\lambda = 0.95$ across 8 parallel environments, gradient clipping at 0.5, PPO clip of 0.2, entropy coefficient 0.01, and 128 rollout steps. The agent uses a 2-layer MLP with 512 units, and advantage normalization is employed. Following [54, 33, 55], CRL is applied only to the actor, while the critic is reinitialized at each task. We conduct experiments using 10 different random seeds to ensure the robustness and reliability of the results.

5.2 Comparison Experiments

We compare the proposed CKA-RL with SOTA methods to demonstrate its superior performance and knowledge transfer capability. Experiments are conducted on three distinct task sequences, including Meta-World, Freeway, and SpaceInvaders, as summarized in Table 1 and Figure 2.

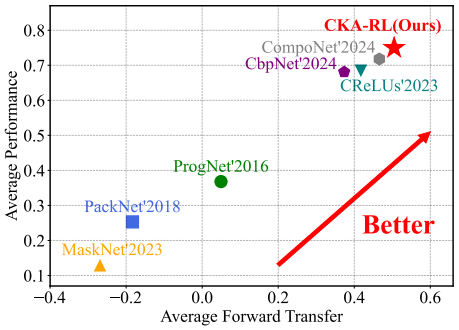


Figure 2: Comparison of SOTA methods with the proposed CKA-RL in terms of average forward transfer and average performance.

Consistent superiority of CKA-RL across diverse environments. The comparative results across three environments, as shown in Table 1, demonstrate the exceptional performance of CKA-RL. Notably, the CKA-RL consistently surpasses SOTA CRL methods in all environments. Specifically, when averaged across three environments, the proposed CKA-RL achieves an improvement of at least 4.20% in performance ($0.7196 \rightarrow 0.7498$) and 8.02% in forward transfer ($0.4674 \rightarrow 0.5049$) compared to CompoNet, demonstrating its superior capability in handling diverse task sequences.

Robust performance in dynamic environments. As shown in Table 1, our CKA-RL maintains consistent superiority over existing CRL methods across various dynamic environments. In Meta-

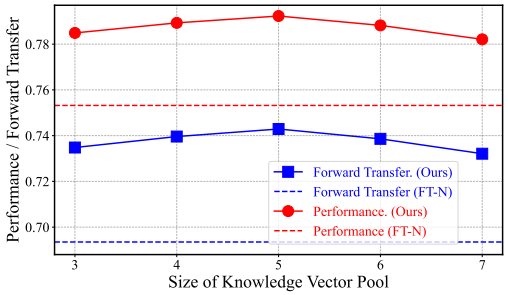


Figure 3: Effects of different pool size $K_{\max}$ in Adaptive Knowledge Merging in the Freeway sequence. Our proposed CKA-RL achieves excellent performance when $K_{\max} = 5$.

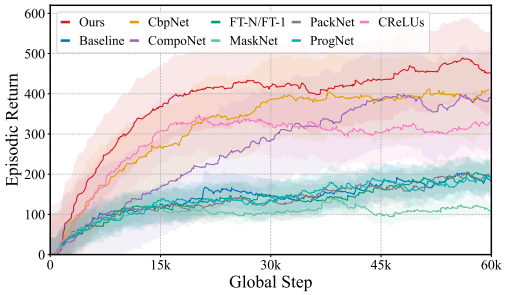


Figure 4: Reward curve comparison in the SpaceInvaders sequence. CKA-RL attains a higher initial reward and converges faster, underscoring superior use of historical vectors.

World sequence, CKA-RL achieves a significant performance improvement of 6.27% compared to CbpNet (0.4368 $\rightarrow$ 0.4642). Similarly, it achieves a performance improvement of 1.02% over CompoNet (0.9828 $\rightarrow$ 0.9928) in SpaceInvaders sequence and a performance improvement of 1.12% over CReLUs (0.7835 $\rightarrow$ 0.7923) in Freeway sequence. These consistent improvements across different environments highlight the robustness and stability of our method in dynamic environments.

Superior knowledge transfer capability. From Table 1, our CKA-RL consistently achieves superior forward transfer performance across all three environments, demonstrating its exceptional knowledge transfer ability. In Meta-World sequence, CKA-RL shows an improvement of 41.82% compared to CompoNet. Similarly, it outperforms CompoNet (0.6963 $\rightarrow$ 0.7749) by 11.29% in SpaceInvaders sequence and surpasses CReLUs by 1.73% in Freeway sequence. These results substantiate that our method enables more efficient knowledge transfer compared to existing CRL methods.

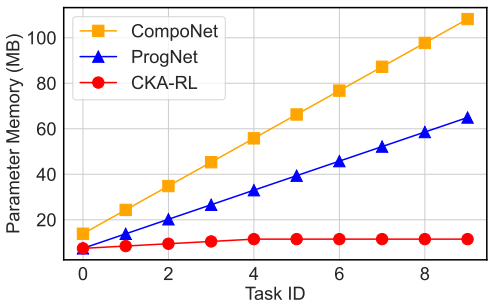
Superior model plasticity. As shown in Table 1, our method outperforms SOTA plasticity enhancement techniques [11, 1], across all evaluation metrics, including CbpNet and CReLUs. Notably, in SpaceInvaders sequence, our method achieves a significant improvement of 18.30% in average performance (0.8392 $\rightarrow$ 0.9928) and an 59.97% increase in forward transfer (0.4844 $\rightarrow$ 0.7749) compared to CbpNet. These comprehensive experimental results demonstrate that our approach surpasses existing plasticity enhancement methods, by effectively leveraging historical knowledge for adaptation. This highlights the potential of adaptive processing to improve model plasticity.

5.3 Ablation Studies

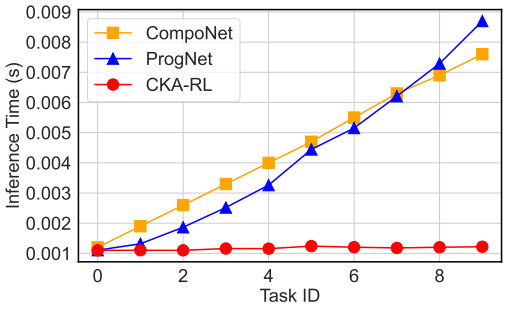
Effectiveness of Components in CKA-RL. Our CKA-RL enhances the learning process by effectively using historical knowledge vectors and merging redundant vectors. We ablate both components in Table 2. Compared with the base method (FT-N), simply averaging knowledge vectors without adaptive weighting actually performs worse than the baseline (0.7437), indicating that naive merging without adaptation harms performance. In contrast, introducing the Continual Knowledge Adaptation strategy achieves higher performance and forward transfer (*e.g.*, 0.7532 (0.6935) *vs.* 0.7821 (0.7321) on Freeway). This confirms the effectiveness of utilizing historical knowledge vectors for enhancing knowledge transfer across dynamic environments. When further merging redundant knowledge vectors that are similar, our method achieves comparable performance (*e.g.,* 0.7821 $\rightarrow$ 0.7923 on Freeway), improves the forward transfer (*e.g.,* 0.7321 $\rightarrow$ 0.7429), and reduces memory requirements (*e.g.,* $N \rightarrow 5$), demonstrating the effectiveness of Adaptive Knowledge Merging strategy in addressing scalability issues and maintaining the stable performance on dynamic environments.

Table 2: Effectiveness of components in CKA-RL in Freeway task sequence. 'Avg' is averaging knowledge vectors, 'Adapt' is the Continual Knowledge Adaptation, and 'Merge' is the Adaptive Knowledge Merging.

Method	Extra Memory	PERF.	FWT.
Base	0	0.7532	0.6935
Base + Avg	N	0.7437	0.7127
Base + Adapt	N	0.7821	0.7321
Base + Adapt + Merge	5	**0.7923**	**0.7429**



(a) Total parameter memory consumption across an increasing number of sequential tasks.



(b) Inference latency across an increasing number of sequential tasks in SpaceInvaders environment.

Figure 5: Analysis of Total Parameter Memory Consumption and Inference Latency: (a) demonstrates that CKA-RL maintains nearly constant parameter and activation size through vector merging, while others grow linearly; (b) shows that CKA-RL maintains nearly constant inference latency, while others suffer from increasing computational overhead. More details can be found in Appendix E.

Effects of Pool Size in Adaptive Knowledge Merging. The size of the knowledge vector pool plays a crucial role in controlling the preservation of historical knowledge. We conduct experiments with $K_{\max}$ values set to $\{3, 4, 5, 6, 7\}$. From Figure 3, our CKA-RL achieves excellent performance when $K_{\max}$ equals 5. Either a smaller or larger $K_{\max}$ hampers the performance. The reasons are as follows. When $K_{\max}$ is small, CKA-RL removes too many knowledge vectors during adaptive knowledge merging, thus being unable to utilize enough knowledge. When $K_{\max}$ is too large, redundant or even conflicting knowledge may slow down the knowledge adaptation process, resulting in performance degradation. Since a larger $K_{\max}$ leads to higher memory requirements, we set $K_{\max}$ to 5 for Freeway and SpaceInvaders experiments, and to 8 for Meta-World, which has a larger number of tasks.

5.4 More Discussions

Superior Initial Performance and Faster Convergence. One of the key advantages of our proposed CKA-RL is its superior initial performance and rapid convergence, as shown in Figure 4. The reward curve of our CKA-RL demonstrates a higher starting point compared to existing CRL methods, indicating that our method effectively utilizes the historical knowledge vectors. Furthermore, the curve exhibits a steeper ascent, reaching convergence much faster than the competing methods. This not only highlights the efficiency of our CKA-RL but also underscores its effectiveness in utilizing historical knowledge vectors.

Cross-task Performance of CKA-RL. To assess how well the final policy consolidates information acquired throughout the learning process, we freeze the model after the last task and evaluate its performance on all previously seen tasks, reporting the average performance across three environments in Table 3. CKA-RL achieves the highest average performance (0.3966), slightly surpassing the closest competitor CbpNet (0.3933) and clearly outperforming FT-N (0.3733) and CReLUs (0.3666). These results indicate that CKA-RL, which combines knowledge adaptation with adaptive merging, results in a more effective final policy that performs better across multiple tasks.

Table 3: Cross-task performance comparison evaluated with the final policy π_{θ_N}.

Method	Pub.'Year	Average Performance
Baseline	–	0.2833
FT-N [53]	–	0.3733
ProgNet [40]	–	0.2966
PackNet [34]	CVPR'2018	0.1900
MaskNet [5]	TMLR'2023	0.2266
CReLUs [1]	CoLLAs'2023	0.3666
CompoNet [33]	ICML'2024	0.3900
CbpNet [11]	Nature'2024	0.3933
CKA-RL (Ours)	–	**0.3966**

Memory Efficiency of CKA-RL. We evaluate the memory efficiency of CKA-RL by comparing the growth of model parameters across tasks from SpaceInvaders. As shown in Figure 5a, the total

parameter memory of CKA-RL remains nearly constant beyond the fifth task due to our use of the adaptive knowledge merging, which maintains a bounded knowledge vector pool. This ensures that model complexity does not scale linearly with the number of tasks.

Inference Efficiency of CKA-RL. From Figure 5b, CKA-RL achieves the highest performance and forward transfer with an average inference time of only **0.0012s** per input. Notably, CKA-RL latency remains essentially constant regardless of the number of tasks, since it consolidates historical knowledge into a fixed-size policy parameter during the parameter construction phase, eliminating the need for complex runtime composition. In contrast, existing architectures such as CompoNet and ProgNet exhibit significant increases in inference cost as the number of tasks grows.

Effectiveness of CKA-RL in Mitigating Catastrophic Forgetting. We use the standard forgetting metric [57], which evaluates performance degradation after training on new tasks. As shown in Table 4, CKA-RL achieves a relatively low forgetting rate (0.45), demonstrating its effective retention of previously learned knowledge while continuing to adapt to new tasks. In contrast, PackNet, which shows the lowest forgetting rate (0.04), sacrifices significant overall performance (0.2767). In contrast, CKA-RL achieves the highest performance (0.7923) and forward transfer (0.7429) across all methods, with a balanced forgetting rate, highlighting its robustness in preventing catastrophic forgetting while maintaining strong performance.

Table 4: Forgetting analysis on Freeway. We report overall performance (PERF.), forward transfer (FWT.), and forgetting, computed as pre–post performance on each prior task after learning a new task (lower is better).

Method	Pub.'Year	PERF.	FWT.	Forgetting
ProgNet [40]	–	0.3125	01938	0.36
PackNet [34]	CVPR'2018	0.2767	0.1970	**0.04**
MaskNet [5]	TMLR'2023	0.0644	-0.0503	**0.04**
CReLUs [1]	CoLLAs'2023	0.7835	0.7303	0.52
CompoNet [33]	ICML'2024	0.7629	0.7115	0.49
CbpNet [11]	Nature'2024	0.7678	0.7201	0.46
CKA-RL (Ours)	–	**0.7923**	**0.7429**	0.45

6 Conclusion

In this paper, we propose **C**ontinual **K**nowledge **A**daptation for **R**einforcement **L**earning (**CKA-RL**), which enables the accumulation and effective utilization of historical knowledge, thereby accelerating learning in new tasks and explicitly reducing performance degradation on previous tasks. Specifically, we assume that the agent acquires a unique knowledge vector for each task during continual learning. Based on this, we develop a Continual Knowledge Adaptation strategy that enhances knowledge transfer from previously learned tasks. Furthermore, we introduce an Adaptive Knowledge Merging mechanism that combines similar knowledge vectors to address scalability challenges. The CKA-RL outperforms SOTA methods, with a 4.20% overall gain and an 8.02% boost in forward transfer.

Acknowledgements

This work was partially supported by the Joint Funds of the National Natural Science Foundation of China (Grant No.U24A20327, No.U23B2013).

References

[1] Z. Abbas, R. Zhao, J. Modayil, A. White, and M. C. Machado. Loss of plasticity in continual deep reinforcement learning. In *Conference on lifelong learning agents*, pages 620–636. PMLR, 2023.

[2] D. Abel, A. Barreto, B. Van Roy, D. Precup, H. P. van Hasselt, and S. Singh. A definition of continual reinforcement learning. *Advances in Neural Information Processing Systems*, 36:50377–50407, 2023.

[3] N. Anand and D. Precup. Prediction and control in continual reinforcement learning. *Advances in Neural Information Processing Systems*, 36:63779–63817, 2023.

[4] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47:253–279, 2013.

[5] E. Ben-Iwhiwhu, S. Nath, P. K. Pilly, S. Kolouri, and A. Soltoggio. Lifelong reinforcement learning with modulating masks. *Transactions on Machine Learning Research*, 2023.

[6] G. Chen, S. Niu, D. Chen, S. Zhang, C. Li, Y. Li, and M. Tan. Cross-device collaborative test-time adaptation. *Advances in Neural Information Processing Systems*, 37:122917–122951, 2024.

[7] L. Chen, H. Zhao, and Y. Sun. Decoupled clip and dynamic sampling policy optimization for large-scale language models. *arXiv preprint arXiv:2401.12345*, 2024.

[8] P. Chen, D. Ji, K. Lin, W. Hu, W. Huang, T. Li, M. Tan, and C. Gan. Learning active camera for multi-object navigation. *Advances in Neural Information Processing Systems*, 35:28670–28682, 2022.

[9] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel. Model-based reinforcement learning via meta-policy optimization. In *Conference on Robot Learning*, pages 617–629. PMLR, 2018.

[10] Z. A. Daniels, A. Raghavan, J. Hostetler, A. Rahman, I. Sur, M. Piacentino, A. Divakaran, R. Corizzo, K. Faber, N. Japkowicz, et al. Model-free generative replay for lifelong reinforcement learning: Application to starcraft-2. In *Conference on Lifelong Learning Agents*, pages 1120–1145. PMLR, 2022.

[11] S. Dohare, J. F. Hernandez-Garcia, Q. Lan, P. Rahman, A. R. Mahmood, and R. S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774, 2024.

[12] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[13] H. Fu, Y. Sun, M. Littman, and G. Konidaris. Knowledge retention in continual model-based reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

[14] J.-B. Gaya, T. Doan, L. Caccia, L. Soulier, L. Denoyer, and R. Raileanu. Building a subspace of policies for scalable continual learning. In *International Conference of Learning Representations*, 2023.

[15] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[16] A. Gupta, S. Savarese, S. Ganguli, and L. Fei-Fei. Embodied intelligence via learning and evolution. *Nature communications*, 12(1):5721, 2021.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[18] J. Hu, Y. Wang, S. Zhang, K. Zhou, G. Chen, Y. Hu, B. Xiao, and M. Tan. Efficient dynamic ensembling for multiple LLM experts. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22,2025*, pages 8095–8103. ijcai.org, 2025.

[19] J. Hu, W. Zhang, Y. Wang, Y. Hu, B. Xiao, M. Tan, and Q. Du. Dynamic compressing prompts for efficient inference of large language models. *arXiv preprint arXiv:2504.11004*, 2025.

[20] S. Hu, Y. Zhou, Z. Fan, J. Hu, L. Shen, Y. Zhang, and D. Tao. Continual task learning through adaptive policy self-composition. *arXiv preprint arXiv:2411.11364*, 2024.

[21] M. Igl, G. Farquhar, J. Luketina, W. Boehmer, and S. Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

[22] G. Ilharco, M. T. Ribeiro, M. Wortsman, L. Schmidt, H. Hajishirzi, and A. Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023.

[23] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[24] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[25] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[26] K. Khetarpal, M. Riemer, I. Rish, and D. Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

[27] B. Kim, C. Ahn, W. J. Baddar, K. Kim, H. Lee, S. Ahn, S. Han, S. Suh, and E. Yang. Test-time ensemble via linear mode connectivity: A path to better adaptation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[28] D. Kumaran, D. Hassabis, and J. L. McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.

[29] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[30] K. Lin, Y. Wang, P. Chen, R. Zeng, Y. Lei, S. Zhou, Q. Du, M. Tan, and C. Gan. When to align: Dynamic behavior consistency for multiagent systems via intrinsic rewards. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2025.

[31] F.-M. Luo, S. Jiang, Y. Yu, Z. Zhang, and Y.-F. Zhang. Adapt to environment sudden changes by learning a context sensitive policy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7637–7646, 2022.

[32] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, and M. Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[33] M. Malagon, J. Ceberio, and J. A. Lozano. Self-composing policies for scalable continual reinforcement learning. In *Forty-first International Conference on Machine Learning*, 2024.

[34] A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[35] J. A. Mendez, H. van Seijen, and E. EATON. Modular lifelong reinforcement learning via neural composition. In *International Conference on Learning Representations*, 2022.

[36] Y. Meng, Z. Bing, X. Yao, K. Chen, K. Huang, Y. Gao, F. Sun, and A. Knoll. Preserving and combining knowledge in robotic lifelong reinforcement learning. *Nature Machine Intelligence*, pages 1–14, 2025.

[37] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937. PMLR, 2016.

[38] A. Muppidi, Z. Zhang, and H. Yang. Fast TRAC: A parameter-free optimizer for lifelong reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[39] B. Peng, X. Li, J. Gao, J. Liu, and K.-F. Wong. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2182–2192, 2018.

[40] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[41] J. Schmidhuber, J. Zhao, and N. N. Schraudolph. Reinforcement learning with self-modifying policies. In *Learning to learn*, pages 293–309. Springer, 1998.

[42] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[44] J. Shen, H. Lai, M. Liu, H. Zhao, Y. Yu, and W. Zhang. Adaptation augmented model-based policy optimization. *Journal of Machine Learning Research*, 24(218):1–35, 2023.

[45] H. Sun, J. Hu, Z. Zhang, H. Tian, X. Xie, Y. Wang, Z. Yu, X. Xie, and M. Tan. Open-world drone active tracking with goal-centered rewards. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[46] Z. Sun and Y. Mu. Rewiring neurons in non-stationary environments. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 28173–28186. Curran Associates, Inc., 2023.

[47] R. S. Sutton, A. G. Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[48] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.

[49] E. Ullah, T. Mai, A. Rao, R. A. Rossi, and R. Arora. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*, pages 4126–4142. PMLR, 2021.

[50] Z. Wang, E. Yang, L. Shen, and H. Huang. A comprehensive survey of forgetting in deep learning beyond continual learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[51] C. J. C. H. Watkins. Learning from delayed rewards. 1989.

[52] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[53] M. Wołczyk, M. Zając, R. Pascanu, Ł. Kuciński, and P. Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

[54] M. Wolczyk, M. Zając, R. Pascanu, Ł. Kuciński, and P. Miłoś. Disentangling transfer in continual reinforcement learning. *Advances in Neural Information Processing Systems*, 35:6304–6317, 2022.

[55] Y. Yang, T. Zhou, J. Jiang, G. Long, and Y. Shi. Continual task allocation in meta-policy network via sparse prompting. In *International Conference on Machine Learning*, pages 39623–39638. PMLR, 2023.

[56] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[57] T. Zhang, K. Z. Shen, Z. Lin, B. Yuan, X. Wang, X. Li, and D. Ye. Replay-enhanced continual reinforcement learning. *Transactions on Machine Learning Research*, 2023.

[58] T. Zhang, X. Wang, B. Liang, and B. Yuan. Catastrophic interference in reinforcement learning: A solution based on context division and knowledge distillation. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):9925–9939, 2023.

[59] X. Zhang, Y. Li, and Q. Wang. Group relative policy optimization. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[60] C. Zheng, H. Yin, J. Chen, T. Ng, Y.-S. Ong, and I. Tsang. Mastering continual reinforcement learning through fine-grained sparse network allocation and dormant neuron exploration. *arXiv preprint arXiv:2503.05246*, 2025.

Supplementary Materials for "Continual Knowledge Adaptation for Reinforcement Learning"

Contents

A Mathematical Analysis

We provide some theoretical analysis to support the improved performance of our proposed method. The analysis includes several key lemmas and corollaries, which demonstrate the stability and performance of the knowledge merging mechanism in the context of continual reinforcement learning.

Preliminaries. We parameterize the policy for task τ_k as:

$$\theta_k = \theta_{\text{base}} + \sum_{j=1}^{k-1} \alpha_j^k v_j + v_k, \quad \text{with } \sum_{j=1}^{k-1} \alpha_j^k = 1, \tag{11}$$

where the normalized adaptation factors α^k are produced by a softmax over learnable β^k (Eq. (3)–(4)). When the pool size exceeds $K_{\max}$, we merge the most similar knowledge vectors by averaging (Eq. (5)–(7)).

Lemma 1 (Drift bound under convex reuse). Let $\Delta_k := \theta_k - \theta_{\text{base}} = \sum_{j<k} \alpha_j^k v_j + v_k$. Since α^k are nonnegative and sum to 1, we have:

$$\|\Delta_k\| \;\le\; \Big\| \sum_{j<k} \alpha_j^k v_j \Big\| + \|v_k\| \;\le\; \sum_{j<k} \alpha_j^k \|v_j\| + \|v_k\| \;\le\; \max_{j<k} \|v_j\| + \|v_k\|. \tag{12}$$

Thus, the deviation from the base is controlled by the magnitudes of (a small subset of) vectors actually reused and the current task vector. This uses the normalization of α^k given by Eq. (3)–(4).

Corollary 1 (Lipschitz Performance Stability). If the task-k return $J_k(\theta)$ is L-Lipschitz in parameters, then:

$$|J_k(\theta_k) - J_k(\theta_{\text{base}} + v_k)| \;\le\; L \Big\| \sum_{j<k} \alpha_j^k v_j \Big\| \;\le\; L \sum_{j<k} \alpha_j^k \|v_j\|. \tag{13}$$

Thus, reusing historical vectors cannot hurt beyond a tunable, data-dependent bound, and the bound tightens as α^k concentrates on small-norm or well-aligned vectors (see Eq. (3)–(4)).

Lemma 2 (Interference Reduces with Near-Orthogonality). Let $S_{ij} = \frac{v_i^\top v_j}{\|v_i\| \|v_j\|}$ be the cosine similarity. If $|S_{ij}| \le \varepsilon \ll 1$ for $i \neq j$, then:

$$\Big\| \sum_{j<k} \alpha_j^k v_j \Big\|^2 = \sum_{j<k} (\alpha_j^k)^2 \|v_j\|^2 + \sum_{i \neq j} \alpha_i^k \alpha_j^k \|v_i\| \, \|v_j\| S_{ij} \;\le\; \sum_{j<k} (\alpha_j^k)^2 \|v_j\|^2 + \varepsilon \sum_{i \neq j} \alpha_i^k \alpha_j^k \|v_i\| \, \|v_j\|. \tag{14}$$

Hence, the "cross-task" term is $O(\varepsilon)$. Our empirical cosine analysis shows knowledge vectors are nearly orthogonal, with off-diagonal values in $[-0.24, 0.12]$, while full fine-tuned parameters have strong correlations ($0.93 \sim 0.95$). This matches the design goal of reducing interference.

Corollary 2 (Combining Lemma 1 and Lemma 2). Combining Lemma 1 and Lemma 2, both the drift and the cross-terms that cause interference are controlled—explaining the improved retention seen in the final-policy evaluation (Table 3).

Lemma 3 (Bounded Error of Adaptive Merging). Suppose we must replace (v_m, v_n) by $v_{\text{merge}} = \frac{1}{2}(v_m + v_n)$ when $|V| > K_{\max}$. For any convex coefficients $\lambda, \mu \ge 0, \lambda + \mu = 1$, we have:

$$\big\| \lambda v_m + \mu v_n - v_{\text{merge}} \big\| \;\le\; \tfrac{1}{2} \|v_m - v_n\|. \tag{15}$$

If their cosine similarity $S_{mn} \ge 1 - \delta$ with $\delta \in [0, 2]$, then:

$$\|v_m - v_n\| \le \sqrt{2(1 - S_{mn})} \, \max(\|v_m\|, \|v_n\|) \le \sqrt{2\delta} \, \max(\|v_m\|, \|v_n\|). \tag{16}$$

Thus, the parameter perturbation induced by merging is $O(\sqrt{\delta})$.

Corollary 3 (Performance Stability under Merging). With L-Lipschitz J_k, we have:

$$|J_k(\theta^{\text{after-merge}}) - J_k(\theta^{\text{before-merge}})| \;\le\; \tfrac{L}{2} \|v_m - v_n\| \;\le\; \tfrac{L}{2} \sqrt{2(1 - S_{mn})} \, \max(\|v_m\|, \|v_n\|). \tag{17}$$

Hence, merging similar vectors (large S_{mn}) has a small, explicitly bounded effect, justifying our "merge-the-most-similar" rule in Eq. (6)–(7).

B More Related Work

Reinforcement Learning (RL) [25, 30] constitutes a paradigm within machine learning wherein an agent learns to optimize its decision-making process through interaction with an environment. This interaction involves performing actions and receiving consequent feedback, typically in the form of rewards or penalties. The principal learning objective in RL is the maximization of a cumulative reward signal. In contrast to supervised learning, which relies on datasets comprising pre-defined input-output pairs for model training, RL entails an agent acquiring knowledge from the repercussions of its actions, mediated by this reward-penalty mechanism. This iterative, trial-and-error learning process, coupled with its emphasis on sequential decision-making under uncertainty, distinguishes RL from supervised learning methodologies that depend on labeled datasets. Existing reinforcement learning algorithms can be broadly categorized based on whether an explicit model of the environment is learned or utilized, leading to two principal classes: Model-free RL and Model-based RL.

Model-free RL. Model-free RL algorithms enable the agent to learn optimal policies directly from trajectory samples accrued through interaction with the environment, without explicitly constructing an environmental model. Within model-free RL, algorithms are further distinguished by the components they learn, leading to three primary sub-categories: actor-only, critic-only, and actor-critic algorithms. Actor-only algorithms directly learn a policy network, denoted as $\pi_\theta(a|s)$, which maps states to actions. This network takes the current state s_t as input and outputs the action a_t. Prominent examples of such algorithms include Reinforce [52] and various policy gradient methods [48]. Critic-only algorithms, in contrast, focus solely on learning a value function (e.g., state-value or action-value function). Given a state s_t, the learned value model is used to evaluate all possible actions $a' \in A$, and the action a_t yielding the maximum estimated value is selected. This category encompasses methods such as Q-learning [51]. Actor-critic algorithms combine these two approaches by concurrently maintaining and learning both a policy network (the actor) for action selection and a value function model (the critic) for evaluating actions or states. This category includes algorithms such as Deep Deterministic Policy Gradient (DDPG) [29], Trust Region Policy Optimization (TRPO) [42], Proximal Policy Optimization (PPO) [43], and Asynchronous Advantage Actor-Critic (A3C) [37]. Notably, PPO has gained considerable traction for training large language models. Recent advancements in this area include GRPO [59], which employs group-based advantage estimates within a KL-regularized loss function to reduce computational overhead and enhance update stability, and DAPO [7], which utilizes distinct clipping mechanisms and adaptive sampling techniques to improve efficiency and reproducibility during the fine-tuning of large-scale models.

Model-based RL. Model-based RL algorithms endeavor to learn an explicit model of the environment, thereby addressing challenges related to sample efficiency. This is because the agent can leverage the learned model for planning and decision-making, reducing the necessity for extensive direct environmental interaction. The learned representation of the environment is commonly termed a 'world model'. This world model typically predicts the subsequent state s_{t+1} and the immediate reward r_t based on the current state s_t and the action a_t taken. Exemplary model-based RL algorithms include Dyna-Q [39], Model-Based Policy Optimization (MBPO) [23], and Adaptation Augmented Model-based Policy Optimization (AMPO) [44].

C Environments and Tasks

Our continual learning experiments evaluate agents across three complementary task domains designed to test different capabilities: robotic manipulation with continuous control (Meta-World) [56], and two vision-based Atari environments [32, 4] emphasizing dynamic decision-making (SpaceInvaders) and sparse-reward navigation (Freeway). This combination spans key RL challenges including high-dimensional state spaces, delayed rewards, procedural variations, and partial observability.

The Meta-World tasks evaluate precise motor control and tool manipulation, while the Atari environments provide contrasting challenges. SpaceInvaders tests rapid visual processing and threat response under varying enemy behaviors, and Freeway examines strategic planning in sparse-reward conditions with evolving obstacle patterns. Collectively, these environments form a comprehensive benchmark for evaluating continual learning.

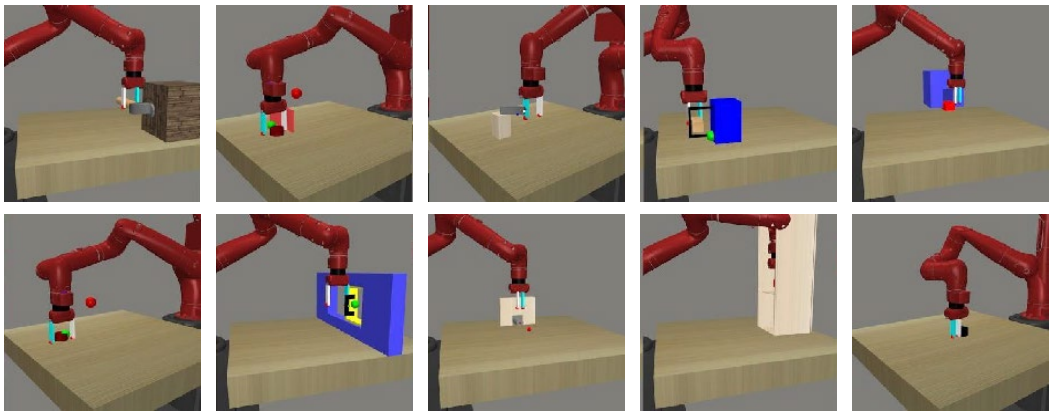


Figure 6: Example frames from each of the 10 Meta-World tasks used in the continual learning benchmark. Tasks are repeated twice to evaluate sequential skill acquisition.

C.1 Meta-World

The robotic manipulation sequence utilizes the Meta-World benchmark [56], featuring 39-dimensional state observations (encoding arm/object positions) and 4-dimensional continuous actions (arm displacement and gripper torque in $[-1, 1]$). To evaluate continual learning capabilities, we adopt a 20-task sequence consisting of 10 distinct manipulation tasks repeated twice, following the Continual World (CW20) benchmark [53]. This selection covers diverse manipulation skills while maintaining consistent evaluation protocols with prior work. The following lines describe the selected tasks:

hammer-v2. Hammer a screw into a wall with randomized initial positions.

push-wall-v2. Navigate a puck around obstacles to reach a target location.

faucet-close-v2. Rotate a faucet handle clockwise from variable starting positions.

push-back-v2. Position a mug beneath a coffee machine with spatial randomization.

stick-pull-v2. Retrieve a box using a stick as a tool.

handle-press-side-v2. Apply lateral force to press down a handle.

push-v2. Basic puck pushing to variable target locations.

shelf-place-v2. Precisely place a puck onto a shelf.

window-close-v2. Slide a window closed from randomized openings.

peg-unplug-side-v2. Remove a laterally mounted peg.

Each episode features randomized object and goal positions, testing robustness to environmental variations. The task sequence progresses from basic manipulations (pushing) to complex tool-use scenarios (hammering, stick-pulling), providing a comprehensive benchmark for evaluating generalization across skills.

C.2 SpaceInvaders

This arcade-style challenge utilizes the *ALE/SpaceInvaders-v5* [3] environment from the Arcade Learning Environment. In this classic game, the agent controls a laser cannon to defend Earth against descending alien invaders. Observations are provided as RGB frames (210×160×3), with the action space comprising six discrete actions: NOOP (no operation), FIRE, RIGHT, LEFT, RIGHTFIRE (combined movement and firing), and LEFTFIRE. The agent has three lives, and the game terminates when either all lives are lost or invaders reach the ground. Rewards are granted for destroying invaders, with higher-value targets located in back rows.

[3] Additional details are available at `https://ale.farama.org/environments/space_invaders/`.

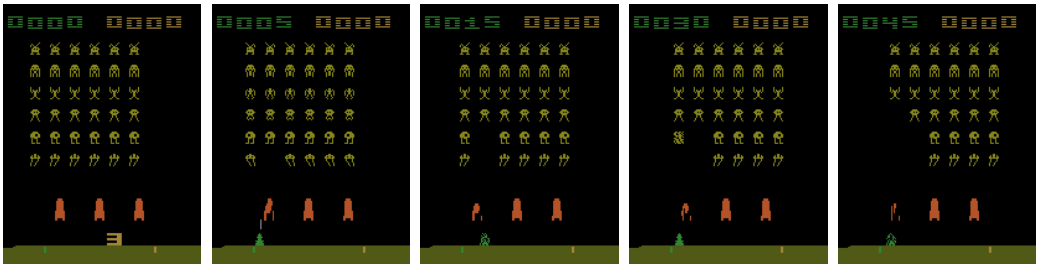


Figure 7: Example gameplay frames from the SpaceInvaders environment.

To systematically evaluate the agent's capability in dynamic threat scenarios, we examine ten strategically selected game modes that modify enemy behavior and environmental dynamics:

Mode 0 (Baseline). Standard configuration with static shields and predictable bomb trajectories.

Mode 1 (Mobile Shields). Shields oscillate horizontally, eliminating reliable cover positions.

Mode 2 (Zigzag Bombs). Invader bombs follow non-linear trajectories, increasing evasion difficulty.

Mode 3 (Composite Challenge). Combines mobile shields (Mode 1) with zigzag bombs (Mode 2).

Mode 4 (High-Speed Bombs). Baseline configuration with accelerated bomb descent rates.

Mode 5 (Mobile Shields + Fast Bombs). Integrates Mode 1's dynamic shields with Mode 4's bomb velocity.

Mode 6 (Zigzag + Fast Bombs). Combines Mode 2's erratic bomb paths with increased speed.

Mode 7 (Full Complexity). Merges all modifiers: mobile shields, zigzag bombs, and high velocity.

Mode 8 (Intermittent Visibility). Invaders periodically become invisible, testing memory and prediction.

Mode 9 (Dynamic Visibility). Mobile shields (Mode 1) coupled with intermittent invader visibility.

C.3 Freeway

The Freeway experiments are conducted using the *ALE/Freeway-v5* [4] environment from the Arcade Learning Environment. In this environment, the agent controls a chicken attempting to cross a multi-lane highway with moving vehicles. Observations are provided as RGB frames (210×160×3), with action space consists of three discrete actions: NOOP (no operation), UP (move forward), and DOWN (move backward). Rewards are exceptionally sparse, the agent only receives +1 upon successfully reaching the top of the screen after crossing all traffic lanes.

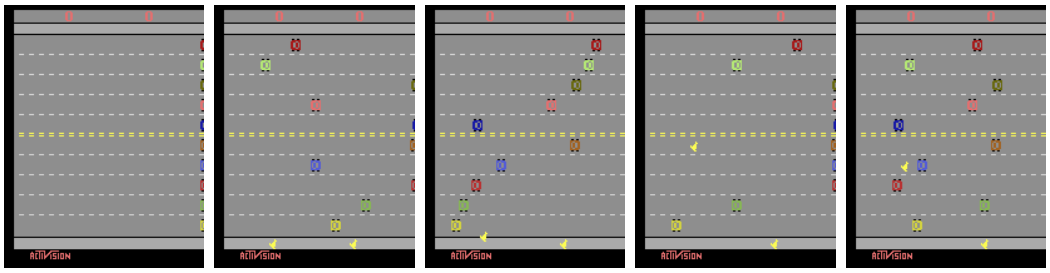


Figure 8: Example gameplay frames from the Freeway environment.

To evaluate the agent's adaptability across varying difficulty levels, we select eight distinct game modes from the available configurations, each modifying traffic patterns and vehicle behaviors:

Mode 0 (Default). Default configuration with standard traffic density and vehicle speeds.

[4] Additional details are available at `https://ale.farama.org/environments/freeway/`.

Mode 1 (Increased Traffic & Trucks). Increased traffic density with faster vehicles. Introduces trucks in the upper lane closest to the center - these longer vehicles require more strategic avoidance.

Mode 2 (High-Speed Trucks). Enhanced difficulty from Mode 1 with trucks moving at higher speeds and further increased traffic density.

Mode 3 (All-Lane Trucks). Maximum truck presence with trucks appearing in all lanes, maintaining the high speeds established in Mode 2.

Mode 4 (Randomized Speeds). Dynamic speed variation where vehicle velocities change randomly during episodes, while maintaining similar traffic density to previous modes without trucks.

Mode 5 (Clustered Vehicles & Speed Variability). Combines characteristics of Mode 1 with additional stochastic elements: vehicle speeds vary dynamically and some vehicles appear in tightly-spaced clusters (2-3 vehicles).

Mode 6 (Heaviest Traffic with Clusters). Builds upon Mode 5 with the most dense traffic configuration, creating the most challenging navigation scenario.

Mode 7 (All-Lane Trucks with Random Speeds). All lanes are filled with trucks, and their speeds vary randomly during the episode.

D Implementation Details

To ensure fairness and reproducibility, we build upon the official implementations released by CompoNet [33], which provide well-tested baselines for standard reinforcement learning algorithms. Our modifications to the original SAC and PPO codebases are kept minimal: we only substitute the agent definition with our proposed architecture and introduce the corresponding learning mechanisms.

Both Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO) follow the standard actor-critic paradigm, where the actor samples actions from a parameterized policy distribution and the critic estimates state values. In all continual reinforcement learning (CRL) methods, the continual adaptation mechanisms are applied only to the actor network, while the critic is reinitialized at the beginning of each task, following common practice in the literature.

Table 5: Hyperparameters shared by all methods in the Meta-World task sequence under the SAC algorithm.

	Description	**Value**
Common	Optimizer	*Adam*
	Adam's β_1 and β_2	(0.9, 0.999)
	Discount rate (γ)	0.99
	Max Std.	e^{2}
	Min Std.	e^{-20}
	Activation Function	*ReLU*
	Hidden Dimension (d_{model})	256
SAC Specific	Batch size	128
	Buffer size	10^{6}
	Target Smoothing Coef. (τ)	0.005
	Entropy Regularization Coef. (α)	0.2
	Auto. Tuning of α	Yes
	Policy Update Freq.	2
	Target Net. Update Freq.	1
	Noise Clip	0.5
	Number of Random Actions	10^{4}
	Timestep to Start Learning	5×10^{3}
Networks	Target Net. Layers	3
	Critic Net. Layers	3
	Actor's Learning Rate	10^{-3}
	Q Networks' Learning Rate	10^{-3}

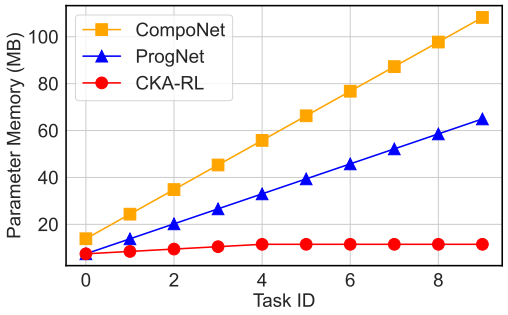
For Meta-World tasks, we adopt SAC as the underlying optimization algorithm. Both the actor and critic networks are implemented as two-layer multilayer perceptrons (MLPs), each followed by separate linear output heads predicting the mean and log standard deviation of the Gaussian policy.

For Atari-based tasks including SpaceInvaders and Freeway, PPO is used for training. All methods employ a shared encoder network to extract compact feature representations from image observations. Two single-layer output heads are used to produce the categorical policy logits (actor) and the scalar value estimates (critic). Unless otherwise noted, all hyperparameters are kept identical across different methods and are consistent with those in the reference implementations.

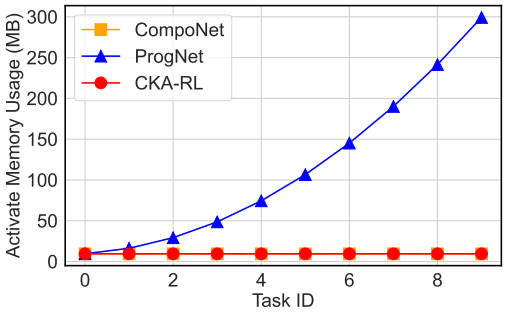
Table 6: Hyperparameters shared by all methods in the SpaceInvaders and Freeway task sequences under the PPO algorithm.

	Description	**Value**
Common	Optimizer	Adam
	AdamW's β_1 and β_2	(0.9, 0.999)
	Max. Gradient Norm	0.5
	Discount Rate (γ)	0.99
	Activation Function	ReLU
	Hidden Dimension (d_{model})	512
	Learning Rate	$2.5 \cdot 10^{-4}$
PPO Specific	PPO Value Function Coef.	0.5
	GAE λ	0.95
	Num. Parallel Environments	8
	Batch Size	1024
	Mini - Batch Size	256
	Num. Mini - Batches	4
	Update Epochs	4
	PPO Clipping Coefficient	0.2
	PPO Entropy Coefficient	0.01
	Learn. Rate Annealing	Yes
	Clip Value Loss	Yes
	Normalize Advantage	Yes
	Num. Steps Per Rollout	128

Methods. We compare the CKA-RL with nine SOTA methods. **1) Baseline** involves training a randomly initialized neural network for each task, providing a fundamental and essential reference point for comparison. **2) FT-1 (Fine-Tuning Single Model)** [53] continuously fine-tunes a single neural network model across all relevant tasks. **3) FT-N (Fine-Tuning with Model Preservation)** [53] follows a similar fine-tuning approach but maintains separate model instances for each task to mitigate forgetting. **4) ProgNet** [40] instantiates a new neural network whenever the task changes, freezing the parameters of the previous modules and adding lateral connections between their hidden layers. **5) PackNet** [34] stores the parameters to solve every task of the sequence in the same network by building masks to avoid overwriting the ones used to solve previous tasks. **6) MaskNet** [5] leverages previous score parameters across tasks to learn task-specific score parameters, which are then used to generate masks for the neural network, dynamically adapting the model to new tasks. **7) CReLUs** [1] enhances the agent capability in CRL by concatenating ReLUs, which reduces the incidence of zero activations and addresses the plasticity loss issue in neural networks. **8) CompoNet** [33] introduces a scalable neural network architecture that dynamically composes action outputs from previously learned policy modules, rather than relying on shared hidden layer representations across tasks. **9) CbpNet** [11] uses a variation of back-propagation, continually and randomly reinitializing a small fraction of underutilized units to maintain the plasticity of neural networks.



(a) Total parameter memory across tasks. CKA-RL plateaus after 5 tasks, while others grow linearly.



(b) Activation related memory usage during training. CKA-RL remains flat, whereas ProgNet shows quadratic growth.

Figure 9: **Memory cost comparison.** CKA-RL maintains nearly constant parameter and activation size through vector merging, while baselines show linear or even quadratic memory growth.

E Efficiency of CKA-RL

E.1 Performance *vs.* Memory Cost Analysis

We evaluate the memory efficiency of CKA-RL by comparing the growth of model parameters and activation-related memory across tasks from SpaceInvaders. As shown in Figure 9a, the total parameter memory of CKA-RL remains nearly constant beyond the fifth task due to our use of the adaptive knowledge merging, which maintains a bounded knowledge vector pool. This ensures that model complexity does not scale linearly with the number of tasks. In contrast, both ProgNet and CompoNet show linearly increasing memory usage, reaching 64.95 MB and 108.21 MB at the tenth task, respectively.

As shown in Figure 9b, the activation-related memory usage in CKA-RL stays nearly constant throughout training. After merging, the knowledge vectors and base parameter construct a fixed-size policy parameter, which drastically reduces the activation memory overhead. Notably, the memory usage in ProgNet exhibits a quadratic growth trend, whereas CKA-RL remains constant. This suggests severe scalability limitations in ProgNet. This memory overhead in ProgNet primarily stems from its architectural design: during training and inference, it relies on the hidden representations of all previously learned policies. As more tasks are added, the number of such dependencies increases, leading to substantial growth in activation memory consumption.

Table 7: **Performance and Forward Transfer Comparison.** CKA-RL outperforms existing methods across three benchmarks while using significantly less memory.

Method	Pub.'Year	Meta-World		SpaceInvaders		Freeway		Average	
		PERF.	FWT.	PERF.	FWT.	PERF.	FWT.	PERF.	FWT.
ProgNet [40]	–	0.4157	-0.0379	0.3757	-0.0075	0.3125	0.1938	0.3680	0.0495
CompoNet [33]	ICML'2024	0.4131	-0.0055	0.9828	0.6963	0.7629	0.7115	0.7196	0.4674
CKA-RL (Ours)	–	**0.4642**	**-0.0032**	**0.9928**	**0.7749**	**0.7923**	**0.7429**	**0.7498**	**0.5049**

CKA-RL achieves a forward transfer of **0.7749** with only **11.49MB** of memory overhead, highlighting its superior efficiency. Unlike methods that accumulate task-specific modules, our approach leverages compact knowledge vectors to consolidate transferable knowledge. This design allows CKA-RL to outperform CompoNet and ProgNet with a significantly smaller memory footprint.

As shown in Table 7, CKA-RL not only achieves the best average performance and forward transfer across diverse tasks (Meta-World, SpaceInvaders, Freeway), but also maintains high efficiency in historical knowledge utilization. These results underscore the practicality of CKA-RL in continual learning scenarios with constrained memory budgets.

E.2 Performance *vs*. Inference Cost Analysis

We further compare the inference efficiency of different methods on the SpaceInvaders benchmark. As shown in Figure 10, CKA-RL achieves the highest performance and forward transfer with an average inference time of only **0.0012s** per input. Importantly, its inference latency remains almost constant regardless of the number of tasks. This is because CKA-RL consolidates historical knowledge into a fixed-size policy parameter during the parameter construction phase, eliminating the need for complex runtime composition.

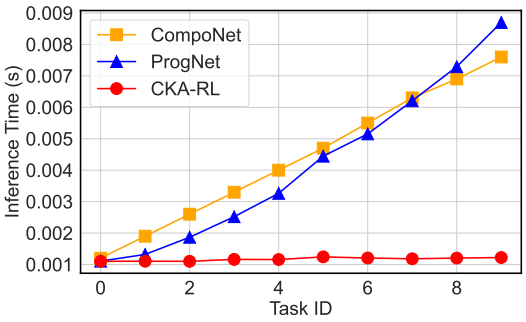
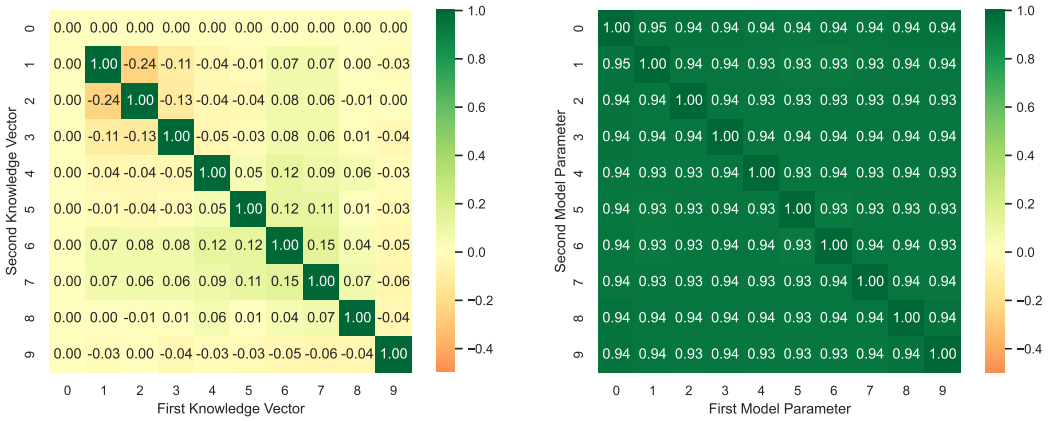


Figure 10: **Inference time *vs.* number of tasks on SpaceInvaders.** CKA-RL maintains nearly constant inference latency due to adaptive knowledge merging, while CompoNet and ProgNet suffer from increasing computational overhead.

In contrast, existing architectures such as CompoNet and ProgNet exhibit significant increases in inference cost as the number of tasks grows. CompoNet shows a linear growth trend in inference time, since its action selection relies on aggregating outputs from all previously learned policies. ProgNet, on the other hand, demonstrates a quadratic growth pattern due to its dependence on all intermediate hidden layers for knowledge integration during inference.

These results underscore the scalability advantage of CKA-RL. By offloading the knowledge fusion process to the model-building stage, it minimizes computational overhead at test time, making it well-suited for continual learning in resource-constrained or real-time environments.

F Further Experimental Results



(a) Knowledge vector similarity (from CKA-RL)

(b) Full parameter similarity (from FN)

Figure 11: Cosine similarity of task representations: (a) Knowledge vectors from CKA-RL exhibit near-orthogonal structure across tasks, with off-diagonal values ranging from -0.24 to 0.12; the zero vector for the first task reflects the base model initialization. (b) In contrast, full parameters from standard fine-tuning show strong off-diagonal correlations ($0.93 \sim 0.95$), indicating significant parameter overlap and interference across tasks.

Our CKA-RL decomposes policy parameters into stable base weights θ_{base} and task-specific knowledge vectors $\{v_i\}$. To validate this decomposition, we compute pairwise cosine similarities between all parameters of models fine-tuned on different tasks (Figure 11b). The strong off-diagonal correlations ($0.93 \sim 0.95$) reveal that standard sequential fine-tuning causes substantial parameter overlap. Then we analyze the cosine similarities between our learned knowledge vectors v_i (Figure 11a). The near-zero off-diagonal values ($-0.24 \sim 0.12$) demonstrate that knowledge vectors occupy nearly

orthogonal directions in parameter space. The first task's zero-valued vector reflects the initial base model training phase.

This decomposition brings two key advantages: (1) stability, by preserving the base parameters θ_{base} across tasks, and (2) plasticity, through task-specific knowledge vectors that remain nearly orthogonal. The orthogonality ensures minimal interference between tasks while enabling effective adaptation. As a result, CKA-RL achieves a forward transfer of 0.7749 and an average performance of 0.9928 across tasks in continual learning settings, outperforming standard fine-tuning (0.6864 and 0.9785, respectively).

G Discussions and Future Works

G.1 Limitations and Future Works

In this paper, we propose **C**ontinual **K**nowledge **A**daptation for **R**einforcement **L**earning (**CKA-RL**), which enables the accumulation and effective utilization of historical knowledge. However, we believe that there are potential studies worth exploring in the future to further capitalize on the advantages of CKA-RL:

- **Complex-Environment Evaluation:** Our current experiments focus on standard continual reinforcement learning benchmarks, and the scalability and robustness of CKA-RL in complex, real-world settings (e.g., high-dimensional visual perception or long-horizon robotic control) remain unverified. In future work, we will extend our evaluation to domains with richer dynamics and observation modalities (e.g., outdoor vision-language navigation tasks) to rigorously assess the generality and practical utility of CKA-RL.
- **Large-Scale Architecture Generalization:** CKA-RL has been validated only on small-scale neural networks, and its applicability to deeper or novel architectures remains untested. In future work, we will evaluate CKA-RL within large-scale models, such as during the RLHF phase of LLMs training, to assess its scalability and effectiveness in deeper networks.

G.2 Broader Impacts

Positive Societal Impacts. The continual knowledge adaptation mechanism introduced by CKA-RL can substantially improve the data- and compute-efficiency of autonomous systems that must operate in non-stationary environments. By enabling robots, intelligent assistants, and other agents to rapidly integrate new skills without repeatedly retraining from scratch, our method can reduce energy consumption and carbon footprint associated with large-scale model updates.

Negative Societal Impacts. As with any advanced continual learning technique, there is a risk that CKA-RL could be exploited to develop adaptive adversarial agents that continuously learn to evade detection or defenses.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We propose **C**ontinual **K**nowledge **A**daptation for **R**einforcement **L**earning (**CKA-RL**), which enables the accumulation and effective utilization of historical knowledge.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The details can be seen in the Appendix G.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The details can be seen in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The source code is available at https://github.com/Fhujinwu/CKA-RL.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The details can be seen in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: See Table 1

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The details can be seen in Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper meets the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The details can be seen in Appenidx G.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We strictly follow the license of the assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.