# ROUTING-AWARE MASKED DECISION TRANSFORMER FOR TASK-AGNOSTIC CONTINUAL OFFLINE REINFORCEMENT LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Continual offline reinforcement learning (CORL) offers an efficient approach to building autonomous decision-making agents from static datasets without additional environment interactions during training. However, substantial differences in environment dynamics across tasks make it challenging to preserve previously acquired knowledge while effectively transferring it to subsequent tasks. Many existing structure-based methods manage knowledge through coarse-grained module composition, while their reliance on explicit task identities at inference limits their applicability when such information is unavailable. To address these challenges, we propose the Routing-Aware Masked Decision Transformer (RMDT), which integrates trajectory-based routing with parameter-level masking for task-agnostic CORL. Specifically, RMDT allocates subnetworks using learnable binary masks to preserve and reuse historical knowledge, while applying capacity regularization to discourage unnecessary parameter expansion. Gated low-rank expansion supplements model capacity on demand to accommodate task-specific learning requirements. To handle scenarios without explicit semantic task labels, RMDT introduces a lightweight task router during training that combines classifier predictions with Gaussian prototype matching. During inference, the router identifies the current task using multi-step probability aggregation and confidence-based locking, thereby activating the corresponding mask configuration to enable hierarchical knowledge selection. Experiments on Offline Continual World show that RMDT outperforms existing baselines, demonstrating better knowledge reuse and transfer capabilities. Code is available here.

## 1 INTRODUCTION

Offline reinforcement learning (Offline RL) Levine et al. (2020) aims to learn reward-maximizing policies from offline data, making it particularly valuable in real-world scenarios where online data collection is expensive, time-consuming, or impractical. However, conventional offline RL settings focus on training and evaluating agents on a single task, which does not reflect real-world scenarios where data arrive sequentially. To address this gap, continual offline reinforcement learning (CORL) Hu et al. (2026a) aims to enable agents to accumulate knowledge without continual access to previous task environments. Compared with settings such as vision and language, where tasks exhibit a high degree of homogeneity, the substantial heterogeneity across RL tasks makes learning new tasks more likely to induce catastrophic forgetting of previously learned ones, further limiting forward knowledge transfer.

Existing CORL methods largely adapt Continual Learning (CL) Wang et al. (2024) and primarily use three strategies to balance stability and transferability. Rehearsal-based methods Zhang et al. (2022b); Fu et al. (2025); Dey et al. (2025) alleviate forgetting by storing and replaying historical data, but introduce additional storage overhead, and their approximation to multitask learning is not fully applicable to settings where past task data cannot be retained. Regularization-based methods Zhang et al. (2022a); Hu et al. (2026b) constrain policy updates to protect weights important to previous tasks but often struggle to balance contributions across tasks. Structure-based methods Qin et al. (2021); Yang et al. (2023) effectively mitigate forgetting through parameter isolation that reduces inter-task interference. However, they typically add and combine parameter modules at a



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

coarse granularity, limiting the flexibility of knowledge transfer. Furthermore, many existing methods rely heavily on explicit task identifiers Hu et al. (2026b;a) or strong pretrained priors Schmied et al. (2023) to select appropriate policies or modules during training and evaluation. These requirements may limit their applicability to practical task-agnostic deployment, where agents must infer the current task from observations collected during interaction and select relevant knowledge for decision-making, placing additional demands on task identification and knowledge selection.

To address these limitations, we propose Routing-Aware Masked Decision Transformer (RMDT), which builds on Decision Transformer (DT) and combines trajectory-aware task routing with weight-level sparse masks to enable fine-grained knowledge management and hierarchical decision-making in task-agnostic settings Remonda et al. (2025). Specifically, during training, RMDT learns a task-specific parameter mask using a straight-through estimator. By selecting and combining parameters according to each task's characteristics, RMDT preserves and reuses existing knowledge, providing a favorable initialization for learning new tasks. A capacity regularization term further discourages unnecessary parameter allocation to promote computational efficiency. When the base network's remaining capacity is insufficient, RMDT adaptively expands the model through gated LoRA to add trainable parameters. To support deployment in task-agnostic settings, we construct multiple Gaussian prototypes for each task from the statistical features of its training trajectories and train an expandable classifier on current-task trajectories and historical pseudo-features for efficient task routing. During inference, RMDT identifies the current task from initial interaction trajectories by combining classifier scores with distances between the current trajectory representation and the Gaussian prototypes, and subsequently retrieves the corresponding mask configuration for decision-making. Multi-step probability aggregation and confidence-based locking further improve prediction stability. Experiments on the Offline Continual World (OCW) Hu et al. (2024) benchmark demonstrate that RMDT achieves competitive performance in both task-aware and task-agnostic settings while effectively balancing knowledge transfer, retention, and model capacity efficiency. The main contributions of this work are as follows:

- We first formulate task-agnostic CORL as a hierarchical knowledge selection problem, in which a trajectory-based routing mechanism learns to identify the task at the higher level and select the corresponding configuration for policy execution at the lower level.
- We design parameter-level binary masks and an on-demand expansion mechanism to flexibly reuse historical parameters according to task requirements and mitigate catastrophic forgetting, with capacity regularization further improving parameter utilization.
- Extensive experiments on the OCW benchmarks demonstrate that RMDT outperforms existing methods in average performance and forward transfer, while its near-zero average forgetting further indicates effective knowledge retention across tasks.

## 2 Related Work

### 2.1 Offline Reinforcement Learning

In online RL, agents interact with the environment to collect data and improve their policies through trial and error. In contrast, offline RL Qu et al. (2023); Deng et al. (2023) optimizes policies solely from a static dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=0}^{N-1}$ of $N$ transitions collected by a behavior policy $\pi_\beta$, without further environmental interaction during training. The dataset may contain suboptimal behavior and provide limited coverage, making out-of-distribution (OOD) Ada et al. (2024) actions a central challenge in offline RL.

Previous works have primarily proposed three types of solutions to mitigate this issue: (1) behavior policy regularization Tiapkin et al. (2024); Gao et al. (2025) constrains the learned policy to remain close to the behavior policy that generated the dataset; (2) conservative value estimation Kostrikov et al. (2022); Chen et al. (2023) operates directly on the value function to produce conservative estimates of future rewards for infrequently observed or unseen state-action pairs; and (3) model-based approaches Yu et al. (2021); Hafner et al. (2025) learn explicit models of the environment dynamics from the offline dataset, which can then generate additional synthetic data for policy improvement or safely evaluate candidate policies. Inspired by the success of Transformers in natural language processing and computer vision, conditional sequence modeling methods (CSM) Chen et al. (2021); Zheng et al. (2024) have recently been introduced to offline RL, learning action sequences from

trajectories through supervised learning. By conditioning on multi-step histories, these methods employ scalable Transformer or diffusion architectures to capture long-range temporal dependencies and reduce reliance on the Markov assumption, while potentially mitigating stability issues associated with bootstrapping Zheng et al. (2026).

### 2.2 CONTINUAL REINFORCEMENT LEARNING

Continual reinforcement learning (CRL) aims to enable agents to learn tasks sequentially without access to data from previous tasks while mitigating catastrophic forgetting. Existing methods can be broadly categorized into three groups: (1) rehearsal-based approaches Zhang et al. (2022b); Fu et al. (2025); Dey et al. (2025) retain a subset of past trajectories and combine them with current task data for training, incurring additional storage costs; (2) regularization-based approaches Zhang et al. (2022a); Hu et al. (2026b) constrain policy updates to preserve previously acquired knowledge but face challenges in determining the appropriate constraint strength; and (3) structure-based approaches Qin et al. (2021); Yang et al. (2023) allocate task-specific parameters through parameter isolation or expansion. However, extending these online methods to offline settings remains relatively underexplored and has motivated the emergence of CORL. CoD Hu et al. (2025) combines diffusion-based trajectory modeling with experience rehearsal, replaying a small subset of past task data to mitigate forgetting while adapting to new tasks; CompoFormer Hu et al. (2024) adaptively composes policies based on task similarity; and VQ-CD Hu et al. (2026a) selectively activates task-specific weights in a shared diffusion model to mitigate catastrophic forgetting. Despite these advances, developing finer-grained, learnable parameter management mechanisms to balance stability and transferability remains an open challenge.

A closely related work is WSN Kang et al. (2022), which isolates parameters in CNNs by selecting subnetworks with a fixed sparsity ratio. In contrast, our method combines threshold-based mask learning with capacity regularization that penalizes only newly allocated weights, flexibly balancing historical parameter reuse and new parameter allocation without prescribing subnetwork sizes. Gated low-rank expansion provides additional capacity when needed. Furthermore, we integrate the aforementioned mechanism into the practically important task-agnostic CORL setting through a routing-aware hierarchical framework that identifies tasks from short-trajectory statistics and invokes the corresponding mask configurations, without requiring explicit task identifiers during inference or relying on multi-task pretraining.

## 3 PRELIMINARY

### 3.1 TASK-AGNOSTIC CORL

We consider the task-agnostic CORL setting, where a sequence of $K$ tasks arrives sequentially with known task boundaries. Each task is defined as a Markov Decision Process (MDP) $\mathcal{M}^k = \langle \mathcal{S}, \mathcal{A}, P^k, R^k, \gamma, d_0^k \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the shared state and action spaces, $P^k : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition dynamics, $R^k : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and $d_0^k$ is the initial state distribution. The key distinction between task-agnostic and task-aware settings is that, during both training and evaluation on task $k$, the policy has no access to explicit semantic task identifiers. The policy is trained solely on a static dataset $\mathcal{D}^k = (s_t^k, a_t^k, r_t^k, s_{t+1}^k)$ collected by a behavior policy $\pi_\beta^k$. Furthermore, we have neither previous task datasets nor environment interaction. Our objective is to learn a policy $\pi^k$ that maximizes the expected cumulative discounted return on the current task, $\mathbb{E}_{\pi^k}\left[\sum_{t=0}^{\infty} \gamma^t r_t^k\right]$, while preserving the performance of previously learned policies $\{\pi^i\}_{i=1}^{k-1}$ to avoid catastrophic forgetting.

### 3.2 DECISION TRANSFORMER

DT is trained on offline trajectories consisting of return-to-go (RTG) $\hat{r}_t$, state $s_t$, and action $a_t$ tokens, and predicts actions conditioned on the trajectory context. The RTG $\hat{r}_t = \sum_{j=t}^{T} r_j$ denotes the cumulative reward from the current timestep to the end of the episode. During training, DT processes a trajectory sequence $\tau_t$ autoregressively using a context window of $L$ timesteps:

$$\tau_t = (\hat{r}_{t-L+1}, s_{t-L+1}, a_{t-L+1}, \ldots, \hat{r}_t, s_t, a_t)\,. \tag{1}$$

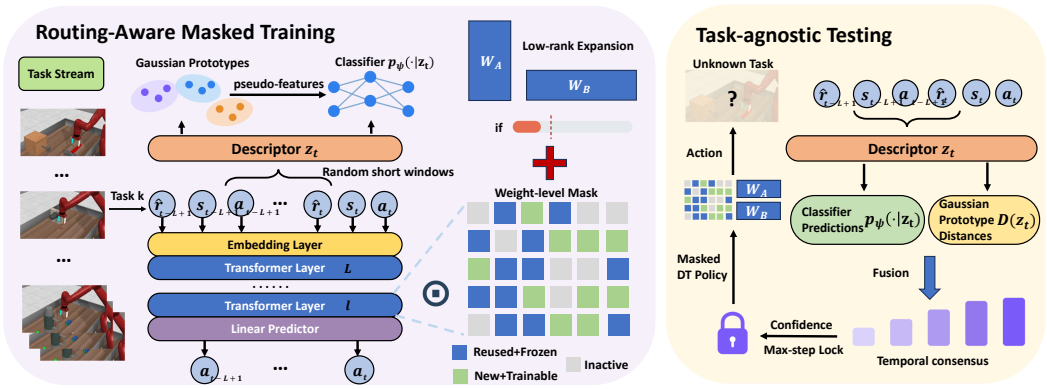



Figure 1: The architecture of RMDT. Left: Routing-Aware Masked Training. RMDT learns from sequential offline datasets through weight-level masking and gated low-rank expansion, while training a trajectory-based router with Gaussian pseudo-replay. Right: Task-Agnostic Testing. The router combines classifier predictions and prototype distances, then uses temporal aggregation and task locking to select the masks and low-rank gates for action generation.

The prediction head associated with each state token $s_i$ predicts the corresponding action $a_i$. Causal masking prevents this prediction from accessing $a_i$ or subsequent tokens. For continuous action spaces, DT is trained on the offline dataset $\mathcal{D}^k$ of task $k$ by minimizing the mean squared error:

$$\mathcal{L}_{\mathrm{DT}}^k = \mathbb{E}_{\tau_t \sim \mathcal{D}^k}\left[\frac{1}{L}\sum_{i=t-L+1}^{t} \|a_i - \pi_\theta(\tau_t)_i\|_2^2\right], \tag{2}$$

where $\pi_\theta(\tau_t)_i$ denotes the action predicted from the hidden representation of the state token $s_i$.

## 4 Method

This section details the proposed RMDT framework, comprising three components: mask allocation, low-rank expansion, and task routing. Binary masks enable fine-grained knowledge reuse and composition through parameter isolation, while gradient freezing and capacity regularization prevent forgetting and reduce model complexity. When the base network's remaining capacity falls below a predefined threshold, the model expands through gated low-rank modules and learns to combine newly added and existing modules. The router uses statistical features from short trajectories to support task-agnostic settings, enabling efficient hierarchical knowledge selection.

### 4.1 Parameter-Level Binary Mask Allocation

For the weight matrix $W_\ell \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$ at layer $\ell$, we introduce a learnable mask score matrix $E_\ell^k$ of the same shape for task $k$ and obtain binary mask $M_\ell^k = \mathbb{I}\left[\sigma(sE_\ell^k) \geq \tau_{\mathrm{mask}}\right]$ through sigmoid-based thresholding, where $s$ controls the sharpness of the continuous relaxation. We then apply $M_\ell^k$ to $W_\ell$ via element-wise multiplication, so only weights with mask values of one participate in the current task computation. To preserve historical knowledge, we maintain a cumulative occupancy mask $O_\ell^{k-1} = \bigvee_{i=1}^{k-1} M_\ell^i$ to mask out gradients of occupied weights while allowing task $k$ to select them freely for forward computation. After training on task $k$ completes, we update the cumulative occupancy mask to $O_\ell^k$. Since the embedding and output layers of DT typically encode knowledge shared across tasks, we freeze them after training on the first task and exclude them from both mask allocation and low-rank expansion. This preserves shared knowledge while reducing parameter-update overhead for subsequent tasks. To prevent the current task from consuming excessive remaining capacity, we augment the standard DT loss with a capacity regularization term:

$$\mathcal{L}_{\mathrm{cap}}^k = \frac{\sum_\ell \left\|(1 - O_\ell^{k-1}) \odot \widehat{M}_\ell^k\right\|_1}{\sum_\ell \left\|1 - O_\ell^{k-1}\right\|_1 + \epsilon}, \tag{3}$$

where $\epsilon > 0$ ensures numerical stability. Normalizing by the remaining capacity increases the penalty for allocating the same number of new weights as free capacity decreases, encouraging more conservative allocation and greater reuse of historical weights.

Since discrete thresholding is non-differentiable, in order to ensure engineering deployability, we adopt a Straight-Through Estimator (STE) Huh et al. (2023) that preserves binary masks in the forward pass and uses gradients from the continuous relaxation to optimize the mask scores during backpropagation: $\widehat{M}_\ell^k = \mathrm{sg}\big(M_\ell^k - \sigma(sE_\ell^k)\big) + \sigma(sE_\ell^k)$, where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator.

### 4.2 Gated Low-Rank Expansion

Although increasing the base network size can provide additional capacity, a fixed architecture may still become insufficient as the number of tasks grows substantially. To address this, at the beginning of each new task, we compute the remaining capacity of the base network as $\rho^{k-1} = \frac{\sum_\ell \|1 - O_\ell^{k-1}\|_1}{\sum_\ell |W_\ell|}$, where $|W_\ell|$ denotes the number of elements in the matrix. When $\rho^{k-1}$ falls at or below a predefined capacity threshold, the model adds low-rank increments to all weight matrices using masks. For $W_\ell$, the increment introduced by the $j$-th expansion is $\Delta W_\ell^j = \frac{\alpha}{r} B_\ell^j A_\ell^j$, where $A_\ell^j \in \mathbb{R}^{r \times d_{\mathrm{in}}}$ and $B_\ell^j \in \mathbb{R}^{d_{\mathrm{out}} \times r}$ are trainable matrices, $r \ll \min(d_{\mathrm{in}}, d_{\mathrm{out}})$ is the rank, and $\alpha$ is a scaling coefficient. Each low-rank increment adds only $r(d_{\mathrm{in}} + d_{\mathrm{out}})$ parameters, providing additional capacity with modest overhead to alleviate potential bottlenecks in the base network as the number of tasks grows substantially.

Since each low-rank module serves as a compact unit of task knowledge, we introduce a trainable scalar gate score $u^{k,j}$ for each module available to task $k$, allowing the model to select and combine low-rank increments according to task requirements adaptively. The effective weights for the current task are therefore given by

$$\widetilde{W}_\ell^k = W_\ell \odot M_\ell^k + \sum_{j \in \mathcal{J}^k} \sigma(u^{k,j}) \Delta W_\ell^j, \tag{4}$$

where $\mathcal{J}^k$ denotes the set of low-rank modules available to task $k$. We jointly optimize the newly introduced low-rank modules, the current task's masks, and module-specific gate coefficients shared across layers, while keeping historical low-rank modules frozen. After training, we preserve the new modules and the task-specific gate and mask configuration, allowing subsequent tasks to reuse these modules through their own gates. Furthermore, we impose a sparsity regularizer on the current task's gate coefficients, $\mathcal{L}_{\mathrm{gate}}^k = \sum_{j \in \mathcal{J}^k} |\sigma(u^{k,j})|$, to discourage unnecessary module contributions and reduce potential interference.

### 4.3 Trajectory-Based Task-Agnostic Routing

Identifying tasks solely from interaction trajectories without semantic information is highly challenging, as tasks with different dynamics may still share similar scenes and actions, while an incorrect policy may further interfere with task identification. During training on task $k$, we randomly sample short windows ranging from 1 to 30 timesteps from offline trajectories to ensure diversity across temporal scales. To further compress window data into a fixed-dimensional representation for router training, we construct a descriptor $z_t^k$ for each window ending at timestep $t$ by concatenating the initial state, the state displacement over the window, the mean state, and the mean action. We then initialize multiple Gaussian prototypes and assign each descriptor to the nearest prototype based on Euclidean distance, updating the corresponding mean, squared-deviation statistic, and sample count. A two-layer MLP classifier $p_\psi$, whose output dimension expands as new tasks arrive, is trained using current-task descriptors and pseudo-features sampled from historical-task prototypes under the average cross-entropy objective:

$$\mathcal{L}_{\mathrm{router}}^k = -\frac{1}{k} \left[ \mathbb{E}_{z \sim \mathcal{Z}^k} \log p_\psi(k \mid z) + \sum_{i=1}^{k-1} \mathbb{E}_{\widetilde{z} \sim \mathcal{G}^i} \log p_\psi(i \mid \widetilde{z}) \right], \tag{5}$$

where $\mathcal{Z}^k$ denotes the set of trajectory descriptors extracted from the current task, and $\mathcal{G}^i$ denotes the Gaussian mixture distribution for historical task $i$. By approximately preserving historical feature

distributions through compact statistics, this procedure enables the classifier to learn the current task while mitigating forgetting of previously learned decision boundaries, without replaying raw historical trajectories.

At test time, we use the same target RTG across all tasks to avoid introducing implicit cues about task identity. We extract descriptors from the initial interaction steps and compute a fused score $q_t$ that combines classifier predictions with Gaussian prototype distances to identify the task:

$$q_t = \mathrm{softmax}(\log p_\psi(\cdot \mid z_t) - \beta \log(1 + D(z_t))), \tag{6}$$

where $p_\psi(\cdot \mid z_t)$ is the vector of task probabilities predicted by the classifier, and $D(z_t)$ contains the distances from the current descriptor to the nearest valid Gaussian prototype of each task. By accounting for both discriminative predictions and distributional compatibility, the router improves task identification accuracy, while logarithmic compression prevents excessively large distances from dominating the predictions. To reduce prediction fluctuations during early interactions, starting from the $S$-th decision step, we average the task probabilities from the five most recent predictions to obtain $\bar{q}_t$. Once the highest averaged probability reaches a confidence threshold, or the number of interaction steps reaches a predefined limit, we lock the task identity to $\widehat{k} = \arg\max_i \bar{q}_t^i$ and use the corresponding binary masks and low-rank gates for all subsequent decisions in the current episode. The overall training objective of RMDT is

$$\mathcal{L}^k_{\mathrm{total}} = \mathcal{L}^k_{\mathrm{DT}} + \lambda_{\mathrm{cap}}\mathcal{L}^k_{\mathrm{cap}} + \lambda_{\mathrm{gate}}\mathcal{L}^k_{\mathrm{gate}} + \lambda_{\mathrm{router}}\mathcal{L}^k_{\mathrm{router}}, \tag{7}$$

where $\lambda_{\mathrm{cap}}$, $\lambda_{\mathrm{gate}}$, and $\lambda_{\mathrm{router}}$ control the contributions of capacity regularization, gate regularization, and the routing loss, respectively.

## 5 Experiment

In this section, we extensively evaluate our proposed method on the OCW benchmark Hu et al. (2024). We first compare RMDT with widely used baselines on average performance, forgetting, and forward transfer. Ablation studies further examine how fine-grained parameter management balances stability and plasticity and how task routing enables hierarchical knowledge selection in task-agnostic settings.

**Benchmarks.** Built upon Meta-World Yu et al. (2020), OCW extends the Continual World framework Wołczyk et al. (2021) to the offline RL setting. The benchmark provides two task sequences: OCW10, comprising 10 representative robotic manipulation tasks, and OCW20, which repeats OCW10 twice to assess policy transferability when tasks are revisited. We provide a detailed description of OCW in Appendix B.

**Baselines.** We compare RMDT with three categories of classical continual learning methods under both task-aware and task-agnostic settings: rehearsal-based methods, including PM Wołczyk et al. (2021), A-GEM Chaudhry et al. (2019), and OER Gai et al. (2023); regularization-based methods, including EWC Kirkpatrick et al. (2017), MAS Aljundi et al. (2018), VCL Nguyen et al. (2018), and UCL Ahn et al. (2019); and structure-based methods, including LoRA Huang et al. (2024), PackNet Mallya & Lazebnik (2018), WSN Kang et al. (2022), CompoFormer Hu et al. (2024), and VQ-CD Hu et al. (2026a). All performance scores are obtained from runs under the same experimental settings to ensure a fair comparison.

**Metrics.** Following standard evaluation protocols in continual learning Wang et al. (2024); Hu et al. (2024); Zhou et al. (2026), we adopt three key metrics: Average Performance (AP) measures the average success rate across all tasks at the end of training; Forgetting (F) measures the average decrease in success rate from immediately after learning each task to the end of the full task sequence; and Forward Transfer (FWT) measures the average improvement in zero-shot performance immediately before learning each new task relative to random initialization.

$$\mathrm{AP} = \frac{1}{K}\sum_{i=1}^{K} p_i(K\delta), \quad \mathrm{F} = \frac{1}{K}\sum_{i=1}^{K}[p_i(i\delta) - p_i(K\delta)], \quad \mathrm{FWT} = \frac{1}{K-1}\sum_{i=2}^{K}[p_i((i-1)\delta) - p_i(0)], \tag{8}$$

where $K$ is the total number of tasks, $p_i(t)$ denotes the success rate on task $i$ at training step $t$, and $\delta$ is the number of training steps allocated to each task.

Table 1: Performance comparison on OCW10 and OCW20 under task-aware and task-agnostic settings. Results are reported as mean ± standard deviation across random seeds.

| Group | Method | Task-aware | | | Task-agnostic | | |
|---|---|---|---|---|---|---|---|
| | | AP ↑ | F ↓ | FWT ↑ | AP ↑ | F ↓ | FWT ↑ |
| | | **OCW10** | | | | | |
| Rehearsal | PM | $0.26_{\pm 0.01}$ | $0.56_{\pm 0.05}$ | $0.04_{\pm 0.03}$ | $0.11_{\pm 0.03}$ | $0.73_{\pm 0.06}$ | $0.01_{\pm 0.02}$ |
| | A-GEM | $0.12_{\pm 0.04}$ | $0.70_{\pm 0.06}$ | $0.02_{\pm 0.01}$ | $0.09_{\pm 0.03}$ | $0.75_{\pm 0.07}$ | $0.00_{\pm 0.01}$ |
| | OER | $0.56_{\pm 0.04}$ | $0.08_{\pm 0.03}$ | $0.04_{\pm 0.02}$ | $0.49_{\pm 0.01}$ | $0.12_{\pm 0.01}$ | $0.00_{\pm 0.01}$ |
| Regularization | EWC | $0.16_{\pm 0.02}$ | $0.67_{\pm 0.02}$ | $0.02_{\pm 0.03}$ | $0.11_{\pm 0.03}$ | $0.72_{\pm 0.05}$ | $0.01_{\pm 0.02}$ |
| | MAS | $0.29_{\pm 0.04}$ | $0.47_{\pm 0.07}$ | $0.01_{\pm 0.01}$ | $0.20_{\pm 0.04}$ | $0.60_{\pm 0.06}$ | $0.00_{\pm 0.01}$ |
| | VCL | $0.14_{\pm 0.03}$ | $0.68_{\pm 0.04}$ | $0.04_{\pm 0.03}$ | $0.10_{\pm 0.03}$ | $0.84_{\pm 0.06}$ | $0.01_{\pm 0.02}$ |
| | UCL | $0.35_{\pm 0.04}$ | $0.40_{\pm 0.05}$ | $0.03_{\pm 0.02}$ | $0.29_{\pm 0.04}$ | $0.49_{\pm 0.06}$ | $0.01_{\pm 0.02}$ |
| Structure | LoRA | $0.54_{\pm 0.03}$ | $0.00_{\pm 0.00}$ | $0.01_{\pm 0.00}$ | $0.42_{\pm 0.05}$ | $0.15_{\pm 0.04}$ | $0.00_{\pm 0.01}$ |
| | PackNet | $0.64_{\pm 0.06}$ | $0.01_{\pm 0.01}$ | $0.03_{\pm 0.01}$ | $0.48_{\pm 0.05}$ | $0.15_{\pm 0.04}$ | $0.01_{\pm 0.02}$ |
| | WSN | $0.67_{\pm 0.04}$ | $0.00_{\pm 0.01}$ | $0.03_{\pm 0.02}$ | $0.50_{\pm 0.05}$ | $0.12_{\pm 0.04}$ | $0.01_{\pm 0.02}$ |
| | CompoFormer | $0.69_{\pm 0.01}$ | $\mathbf{-0.01}_{\pm 0.03}$ | $0.00_{\pm 0.00}$ | $0.52_{\pm 0.04}$ | $0.10_{\pm 0.04}$ | $0.00_{\pm 0.01}$ |
| | VQ-CD | $0.71_{\pm 0.03}$ | $0.00_{\pm 0.01}$ | $0.03_{\pm 0.02}$ | $0.44_{\pm 0.01}$ | $0.19_{\pm 0.01}$ | $0.01_{\pm 0.02}$ |
| **Ours** | **RMDT** | $\mathbf{0.75}_{\pm 0.01}$ | $0.00_{\pm 0.01}$ | $\mathbf{0.05}_{\pm 0.01}$ | $\mathbf{0.63}_{\pm 0.02}$ | $\mathbf{0.03}_{\pm 0.02}$ | $\mathbf{0.05}_{\pm 0.01}$ |
| | | **OCW20** | | | | | |
| Rehearsal | PM | $0.26_{\pm 0.08}$ | $0.57_{\pm 0.09}$ | $0.04_{\pm 0.02}$ | $0.16_{\pm 0.05}$ | $0.65_{\pm 0.07}$ | $0.01_{\pm 0.03}$ |
| | A-GEM | $0.09_{\pm 0.01}$ | $0.73_{\pm 0.04}$ | $0.04_{\pm 0.02}$ | $0.07_{\pm 0.02}$ | $0.73_{\pm 0.06}$ | $0.02_{\pm 0.02}$ |
| | OER | $0.54_{\pm 0.05}$ | $0.12_{\pm 0.04}$ | $0.03_{\pm 0.03}$ | $0.48_{\pm 0.01}$ | $0.07_{\pm 0.03}$ | $0.02_{\pm 0.01}$ |
| Regularization | EWC | $0.12_{\pm 0.02}$ | $0.69_{\pm 0.02}$ | $0.05_{\pm 0.01}$ | $0.10_{\pm 0.02}$ | $0.66_{\pm 0.05}$ | $0.02_{\pm 0.02}$ |
| | MAS | $0.25_{\pm 0.02}$ | $0.44_{\pm 0.02}$ | $0.01_{\pm 0.02}$ | $0.16_{\pm 0.03}$ | $0.45_{\pm 0.05}$ | $0.00_{\pm 0.02}$ |
| | VCL | $0.06_{\pm 0.00}$ | $0.74_{\pm 0.04}$ | $0.04_{\pm 0.01}$ | $0.04_{\pm 0.02}$ | $0.79_{\pm 0.05}$ | $0.01_{\pm 0.02}$ |
| | UCL | $0.30_{\pm 0.04}$ | $0.46_{\pm 0.05}$ | $0.04_{\pm 0.02}$ | $0.28_{\pm 0.04}$ | $0.55_{\pm 0.06}$ | $0.01_{\pm 0.02}$ |
| Structure | LoRA | $0.54_{\pm 0.01}$ | $0.02_{\pm 0.00}$ | $0.02_{\pm 0.02}$ | $0.44_{\pm 0.05}$ | $0.09_{\pm 0.04}$ | $0.01_{\pm 0.02}$ |
| | PackNet | $0.57_{\pm 0.04}$ | $0.02_{\pm 0.00}$ | $0.05_{\pm 0.02}$ | $0.50_{\pm 0.05}$ | $0.06_{\pm 0.05}$ | $0.00_{\pm 0.01}$ |
| | WSN | $0.66_{\pm 0.05}$ | $0.00_{\pm 0.01}$ | $0.04_{\pm 0.02}$ | $0.49_{\pm 0.05}$ | $0.05_{\pm 0.03}$ | $0.01_{\pm 0.02}$ |
| | CompoFormer | $0.72_{\pm 0.03}$ | $\mathbf{0.00}_{\pm 0.01}$ | $0.03_{\pm 0.02}$ | $0.53_{\pm 0.05}$ | $0.04_{\pm 0.04}$ | $0.01_{\pm 0.02}$ |
| | VQ-CD | $0.74_{\pm 0.04}$ | $0.01_{\pm 0.02}$ | $0.05_{\pm 0.02}$ | $0.46_{\pm 0.01}$ | $0.05_{\pm 0.01}$ | $0.00_{\pm 0.01}$ |
| **Ours** | **RMDT** | $\mathbf{0.76}_{\pm 0.01}$ | $\mathbf{0.00}_{\pm 0.01}$ | $\mathbf{0.06}_{\pm 0.01}$ | $\mathbf{0.65}_{\pm 0.02}$ | $\mathbf{0.02}_{\pm 0.01}$ | $\mathbf{0.06}_{\pm 0.01}$ |

### 5.1 Main Experiment

**Task-Aware Setting.** We first compare fairly under the task-aware setting natively supported by all baselines. In this setting, RMDT uses explicit task identifiers to distinguish tasks and retrieve the corresponding mask configurations during both training and inference. The routing module is used only for FWT evaluation, where it selects from previously learned mask configurations for inference because the unseen task has not yet learned a configuration. As shown in Table 1, RMDT achieves the best performance on both OCW10 and OCW20. Although rehearsal-based methods can retrain the model using the best historical trajectories stored in a replay buffer, their performance remains limited. This suggests that focusing solely on data retention while overlooking potential distribution mismatches between new and previous tasks is insufficient for effective continual learning. Regularization-based methods struggle to balance knowledge retention and new task acquisition, and their higher forgetting further indicates that soft constraints alone cannot adequately mitigate cross-task interference. Building on structural approaches, RMDT enables parameter-level knowledge reuse and protection, complemented by capacity expansion when predefined thresholds are met, achieving the best performance while substantially reducing model complexity.

**Task-Agnostic Setting.** Since our router is designed as a plug-and-play module, we replace ground-truth task identifiers with its predictions for baselines that require explicit task identifiers during

Table 2: Ablation study of RMDT components on OCW10 under the task-agnostic setting. Results are reported as mean ± standard deviation across random seeds.

| Experiment | Binary Mask | Low-Rank Expansion | Task Router | AP ↑ | F ↓ | FWT ↑ |
|---|---|---|---|---|---|---|
| w/o Binary Mask | | ✓ | ✓ | $0.27_{\pm 0.10}$ | $0.20_{\pm 0.03}$ | $0.00_{\pm 0.04}$ |
| w/o Low-Rank Expansion | ✓ | | ✓ | $0.52_{\pm 0.07}$ | $\mathbf{0.02}_{\pm 0.02}$ | $0.01_{\pm 0.03}$ |
| w/o Task Router | ✓ | ✓ | | $0.20_{\pm 0.02}$ | $0.15_{\pm 0.01}$ | $0.00_{\pm 0.00}$ |
| **Full Model** | ✓ | ✓ | ✓ | $\mathbf{0.63}_{\pm 0.02}$ | $0.03_{\pm 0.02}$ | $\mathbf{0.05}_{\pm 0.01}$ |

inference, while retaining the original inference procedures of methods that do not. Despite the performance degradation across all methods, RMDT remains competitive. This stems from the compatibility between its autoregressive architecture and the router's inference procedure: task-level routing guides fine-grained parameter selection within the policy network. This hierarchical decision process enables shared knowledge reuse while mitigating cross-task interference. Although the second half of OCW20 repeats the tasks from the first half, we initialize a new mask for each arriving task, as agents in real-world scenarios are typically not informed in advance whether they have encountered a task before. The resulting decrease in routing accuracy is expected, as the router cannot reliably distinguish between separate occurrences of the same underlying task. Nevertheless, historical weights already encode knowledge relevant to these recurring tasks, allowing the model to maintain strong performance while allocating only a small number of previously unused parameters.

### 5.2 Ablation Study

**Component Ablation.** As shown in Table 2, we investigate the contributions of binary masks, low-rank expansion, and the task router on OCW10 under the task-agnostic setting. Removing binary masks reduces average performance and increases forgetting, suggesting that fine-grained parameter selection is important for balancing knowledge reuse and protection. Supplementing the base network with low-rank modules helps accommodate additional learning requirements without modifying protected historical weights. Removing the task router leaves the model largely unable to handle task-agnostic scenarios, forcing it to select masks randomly at inference. The resulting severe performance degradation is therefore expected. With all three components combined, RMDT achieves the highest average performance, supporting the complementary roles of fine-grained knowledge selection, adaptive capacity expansion, and task identification.

**Weight Reuse Analysis.** Figure 2 illustrates how RMDT allocates base-network weights across sequential tasks on OCW10. After the first task, subsequent tasks reuse a substantial proportion of historical weights while allocating additional weights as needed. The remaining capacity decreases from 59.4% after the first task to 17.1% after the tenth task, with later tasks generally consuming less new capacity. Moreover, some previously allocated weights remain inactive for each new task, showing that RMDT selectively combines historical parameters rather than reusing them indiscriminately. These observations illustrate how the learned masks and capacity regularization balance historical weight reuse with new parameter allocation. We further visualize weight allocation across the long OCW20 task sequence in Figure 8.

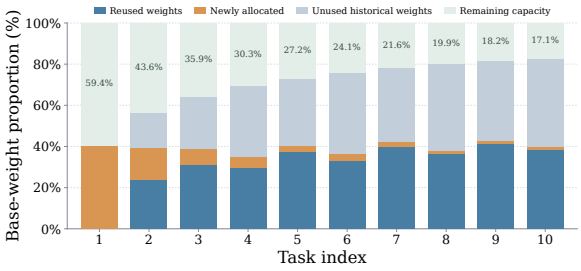



Figure 2: Parameter allocation across sequential tasks on OCW10.

**Task Routing Analysis.** We further analyze the routing components in Table 3. The classifier estimates task membership probabilities from the current trajectory, while the Gaussian prototypes assess task compatibility based on feature-space distances. Using either component alone yields lower identification accuracy than combining them, supporting their complementary roles. Historical pseudo-features help preserve task boundaries during classifier training; removing them biases training toward the current task and substantially degrades overall task identification. Task locking

Table 3: Routing ablation on OCW10. Accuracy denotes task identification accuracy.

| Routing Variant | Accuracy ↑ | AP ↑ |
|---|---|---|
| Gaussian Distance Only | 0.79 | 0.50 |
| Classifier Only | 0.76 | 0.56 |
| w/o Pseudo-Replay | 0.71 | 0.37 |
| w/o Task Locking | **0.90** | 0.57 |
| **Full Router** | **0.90** | **0.63** |

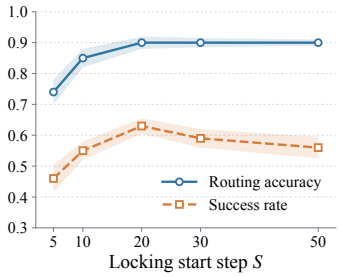



Figure 3: Effect of $S$ on routing accuracy and success rate.

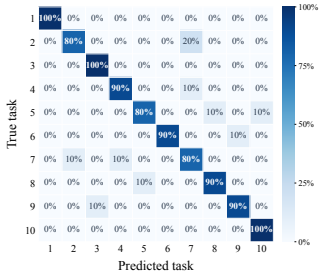



Figure 4: Task identification confusion matrix on OCW10.

stabilizes policy execution by limiting repeated switching between task-specific configurations during inference. Figure 3 examines how the locking start step $S$ affects task identification accuracy and success rate, with the maximum number of routing steps adjusted accordingly to remain consistent with $S$. As $S$ increases, the model can make more accurate predictions based on more informative interaction trajectories. However, an excessively large $S$ prolongs switching between candidate tasks, leading to erroneous decisions and reduced success rates. We also present the router's task-identification confusion matrix in Figure 4, where the numerical task labels correspond to the tasks listed in Table 5.

Table 4: Computational complexity comparison on OCW10 and OCW20.

| | OCW10 | | | OCW20 | | |
|---|---|---|---|---|---|---|
| Method | Params (M)↓ | FLOPs (G)↓ | Time (h)↓ | Params (M)↓ | FLOPs (G)↓ | Time (h)↓ |
| PackNet | 6.4 | 1.3 | 18 | 6.4 | 1.3 | 36 |
| LoRA ($r = 4$) | 5.6 | 0.7 | 15 | 6.7 | 0.8 | 36 |
| CompoFormer | 9.1 | 2.5 | 35 | 10.2 | 3.1 | 70 |
| **RMDT** | 5.9 | 0.8 | 23 | 7.0 | 0.9 | 48 |

**Computational Complexity Analysis.** Table 4 compares the parameter count, FLOPs, and total training time for RMDT and representative structure-based methods at the end of training. For RMDT, the reported parameter count includes the backbone, all retained low-rank modules, the routing classifier, and stored task configurations. FLOPs account for routing computations and policy context recomputation during inference. RMDT requires only 5.9M parameters and 0.8G FLOPs on OCW10, remaining close to LoRA while supporting fine-grained parameter allocation and task identification. Compared with CompoFormer, RMDT reduces parameter count by 35.2%, FLOPs by 68.0%, and training time by 34.3%. Due to the introduction of task-specific masks and the routing module, RMDT's training time is slightly longer than that of some baselines. However, given the performance improvement, this modest increase in computational cost is well justified.

## 6 Conclusion

In this work, we propose the Routing-Aware Masked Decision Transformer (RMDT) for task-agnostic continual offline reinforcement learning. RMDT combines weight-level binary masks with gradient freezing to enable fine-grained knowledge reuse and protection, while gated low-rank expansion provides additional capacity when needed. A trajectory-based task router further enables the selection of task-specific configurations without externally supplied task identifiers or stored historical trajectories. Experiments show competitive performance in both task-aware and task-agnostic settings, highlighting the effectiveness of hierarchical knowledge selection through parameter-level masking and task-level routing. This work provides a foundation for extending sequence-modeling policies to continual learning scenarios with unknown task identities.

**Limitation.** RMDT relies on short interaction histories for task identification, which may be ambiguous when tasks exhibit similar early behaviors. Consequently, incorrect task locking can limit policy performance even when historical knowledge is preserved, motivating more adaptive task inference in future work.

### AI USE STATEMENT

In this work, we used ChatGPT solely for language polishing, including improvements to grammar, clarity, and readability. We have not used generative AI tools for any other tasks in this work. We have reviewed all AI-assisted work and checked each suggested edit to ensure that it preserves the intended technical meaning. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

This work studies continual offline reinforcement learning using existing benchmark datasets and simulated robotic manipulation environments. The experiments do not involve human participants or the collection of personal data. The research aims to support efficient knowledge reuse and reduce the need for additional environment interactions during training. Potential real-world applications should follow appropriate safety standards and responsible deployment practices.

### REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce our method and experiments in the main text and appendices. Appendix A presents the training and inference algorithms, Appendix B describes the benchmark and task sequences, and Appendix C reports the model architecture and hyperparameters. Appendix D details state normalization, trajectory descriptor construction, Gaussian prototype updates, and historical weight preservation. The forward transfer evaluation protocol is specified in Appendix E.4. We also report results across random seeds and provide additional ablation studies and task-order evaluations to facilitate assessment of the method's robustness.

## REFERENCES

Suzan Ece Ada, Erhan Oztop, and Emre Ugur. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *IEEE Robotics and Automation Letters*, 9(4):3116–3123, 2024.

Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. *Advances in neural information processing systems*, 32, 2019.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European conference on computer vision*, pp. 144–161. Springer, 2018.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Hkf2_sC5FX.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Liting Chen, Jie Yan, Zhengdao Shao, Lu Wang, Qingwei Lin, Saravanakumar Rajmohan, Thomas Moscibroda, and Dongmei Zhang. Conservative state value estimation for offline reinforcement learning. *Advances in neural information processing systems*, 36:35064–35083, 2023.

Zhihong Deng, Zuyue Fu, Lingxiao Wang, Zhuoran Yang, Chenjia Bai, Tianyi Zhou, Zhaoran Wang, and Jing Jiang. False correlation reduction for offline reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(2):1199–1211, 2023.

Sharmita Dey, Benjamin Paassen, Sarath Ravindran Nair, Sabri Boughorbel, and Arndt F. Schilling. Continual learning from simulated interactions via multitask prospective rehearsal for bionic limb behavior modeling. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=Bmy82p2eez.

Haotian Fu, Yixiang Sun, Michael Littman, and George Konidaris. Knowledge retention in continual model-based reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

Sibo Gai, Donglin Wang, and Li He. Oer: Offline experience replay for continual offline reinforcement learning. In *ECAI 2023: 26th European Conference on Artificial Intelligence, September 30–October 4, 2023, Kraków, Poland–Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, pp. 772–779. SAGE Publications 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP, 2023.

Chen-Xiao Gao, Chenyang Wu, Mingjun Cao, Chenjun Xiao, Yang Yu, and Zongzhang Zhang. Behavior-regularized diffusion policy optimization for offline reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=pUCYJ9JJuZ.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pp. 1–7, 2025.

Jifeng Hu, Li Shen, Sili Huang, Zhejian Yang, Hechang Chen, Lichao Sun, Yi Chang, and Dacheng Tao. Continual diffuser (cod): Mastering continual offline rl with experience rehearsal. *IEEE Transactions on Neural Networks and Learning Systems*, 36(12):20429–20443, 2025. doi: 10.1109/TNNLS.2025.3598928.

Jifeng Hu, Sili Huang, Li Shen, Zhejian Yang, Shengchao Hu, Shisong Tang, Hechang Chen, Lichao Sun, Yi Chang, and Dacheng Tao. Tackling continual offline rl through selective weights activation on aligned spaces. *Advances in Neural Information Processing Systems*, 38:53617–53652, 2026a.

Jinwu Hu, Zihao Lian, Zhiquan Wen, Chenghao Li, Guohao Chen, Xutao Wen, Bin Xiao, and Mingkui Tan. Continual knowledge adaptation for reinforcement learning. *Advances in Neural Information Processing Systems*, 38:36283–36313, 2026b.

Shengchao Hu, Yuhang Zhou, Ziqing Fan, Jifeng Hu, Li Shen, Ya Zhang, and Dacheng Tao. Continual task learning through adaptive policy self-composition. *arXiv preprint arXiv:2411.11364*, 2024.

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

Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *International Conference on Machine Learning*, pp. 14096–14113. PMLR, 2023.

Haeyong Kang, Rusty John Lloyd Mina, Sultan Rizky Hikmawan Madjid, Jaehong Yoon, Mark Hasegawa-Johnson, Sung Ju Hwang, and Chang D Yoo. Forget-free continual learning with winning subnetworks. In *International conference on machine learning*, pp. 10734–10750. PMLR, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773. IEEE, 2018.

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkQqq0gRb.

Qi Qin, Wenpeng Hu, Han Peng, Dongyan Zhao, and Bing Liu. Bns: Building network structures dynamically for continual learning. *Advances in Neural Information Processing Systems*, 34: 20608–20620, 2021.

Yun Qu, Boyuan Wang, Jianzhun Shao, Yuhang Jiang, Chen Chen, Zhenbin Ye, Liu Linc, Yang Feng, Lin Lai, Hongyang Qin, et al. Hokoff: Real game dataset from honor of kings and its offline reinforcement learning benchmarks. *Advances in Neural Information Processing Systems*, 36:22166–22190, 2023.

Adrian Remonda, Cole Corbitt Terrell, Eduardo E Veas, and Marc Masana. Uncertainty-based experience replay for task-agnostic continual reinforcement learning. *Transactions on Machine Learning Research*, 2025.

Thomas Schmied, Markus Hofmarcher, Fabian Paischer, Razvan Pascanu, and Sepp Hochreiter. Learning to modulate pre-trained models in rl. *Advances in Neural Information Processing Systems*, 36:38231–38265, 2023.

Daniil Tiapkin, Denis Belomestny, Daniele Calandriello, Eric Moulines, Aleksei Naumov, Pierre Perrault, Michal Valko, and Pierre Menard. Regularized rl. In *International Conference on Learning Representations*, volume 2024, pp. 28568–28632, 2024.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE transactions on pattern analysis and machine intelligence*, 46(8):5362–5383, 2024.

Maciej Wołczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Yijun Yang, Tianyi Zhou, Jing Jiang, Guodong Long, and Yuhui Shi. Continual task allocation in meta-policy network via sparse prompting. In *International Conference on Machine Learning*, pp. 39623–39638. PMLR, 2023.

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

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Tiantian Zhang, Xueqian Wang, Bin Liang, and Bo 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, 2022a.

Yaqian Zhang, Bernhard Pfahringer, Eibe Frank, Albert Bifet, Nick Jin Sean Lim, and Yunzhe Jia. A simple but strong baseline for online continual learning: Repeated augmented rehearsal. *Advances in Neural Information Processing Systems*, 35:14771–14783, 2022b.

Hongling Zheng, Li Shen, Yong Luo, Tongliang Liu, Jialie Shen, and Dacheng Tao. Decomposed prompt decision transformer for efficient unseen task generalization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Hongling Zheng, Li Shen, Yong Luo, Deheng Ye, Shuhan Xu, Bo Du, Jialie Shen, and Dacheng Tao. Value-guided decision transformer: A unified reinforcement learning framework for online and offline settings. *Advances in Neural Information Processing Systems*, 38:71655–71684, 2026.

Qiyang Zhou, Xu Ruihang, Peng Wang, WenJie Lu, Xiaochun Cao, Naiqiang Tan, and Li Shen. HTAC: Hierarchical task-aware composition for continual offline reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=akfJfpUEBj.

# APPENDIX


**A Algorithm Details** . . . . . 15

**B Offline Continual World Benchmark** . . . . . 15

**C Hyperparameters Configuration** . . . . . 17

**D More Training Details** . . . . . 17

D.1 Inference Context and Trajectory Initialization . . . . . 17

D.2 State Normalization and Prototype Distances . . . . . 17

D.3 Gaussian Prototype Initialization and Updates. . . . . 19

D.4 Preservation of Historical Weights . . . . . 19

**E More Experiments** . . . . . 19

E.1 Visualization of the Training Process. . . . . 19

E.2 Robustness to Task Arrival Order . . . . . 19

E.3 Effect of Weight Reuse and Capacity Regularization . . . . . 21

E.4 Forward Transfer Evaluation Protocol . . . . . 22

E.5 Parameter Allocation Across the OCW20 Task Sequence . . . . . 22

E.6 Effect of the Number of Gaussian Prototypes.. . . . . 22

# A Algorithm Details

**Algorithm 1** Continual Offline Training with RMDT

**Input:** Offline datasets $\{\mathcal{D}^k\}_{k=1}^K$, updates per task $\delta$, context length $L$, maximum training routing window length $H_{\text{train}} = 30$, mask threshold $\tau_{\text{mask}}$, capacity threshold $\rho_{\min}$
**Initialize:** DT policy $\pi_\theta$, occupied mask $O = \mathbf{0}$, router $p_\psi$, Gaussian statistics $\mathcal{G} = \varnothing$, and LoRA collection $\mathcal{A} = \varnothing$
**for** $k = 1$ to $K$ **do**
  Initialize mask scores $E^k$ and task gates; expand the router with one output class
  Initialize empty Gaussian statistics $\mathcal{G}^k$
  **if** remaining base capacity $\leq \rho_{\min}$ **then**
    Add a LoRA expansion bundle to $\mathcal{A}$ and update task gates
  **end if**
  **for** $u = 1$ to $\delta$ **do**
    Sample a policy batch of context length $L$ from $\mathcal{D}^k$
    Construct causal routing histories with randomly sampled window lengths from 1 to $H_{\text{train}}$, limited by the available history
    Extract descriptors $z = \phi(h)$ using valid states and preceding actions
    Update current-task Gaussian statistics $\mathcal{G}^k$
    Sample historical pseudo-features $\widetilde{z}^i \sim \mathcal{G}^i$ for $i < k$
    Compute the task-balanced routing loss $\mathcal{L}^k_{\text{router}}$
    Update sigmoid scale $s$ and compute $M^k = \mathbb{I}[\sigma(sE^k) \geq \tau_{\text{mask}}]$ using straight-through gradients
    Predict actions using $M^k$ and the current-task LoRA gates
    Compute $\mathcal{L}^k_{\text{total}} = \mathcal{L}^k_{\text{DT}} + \lambda_{\text{cap}}\mathcal{L}^k_{\text{cap}} + \lambda_{\text{gate}}\mathcal{L}^k_{\text{gate}} + \lambda_{\text{router}}\mathcal{L}^k_{\text{router}}$
    Backpropagate; mask occupied base-weight gradients by $(1 - O)$ and exclude frozen parameters from updates
    Save the pre-update values of occupied base weights
    Clip gradients and perform an optimizer update
    Restore occupied base weights to their pre-update values
  **end for**
  Compute the final binary mask $M^k$ and store the learned task configuration and statistics
  Update cumulative occupancy $O \leftarrow O \vee M^k$
  Freeze newly learned LoRA parameters for subsequent tasks
  **if** $k = 1$ **then**
    Freeze shared policy parameters not managed by masks
  **end if**
**end for**
**Output:** Trained policy, task configurations, router, and Gaussian statistics

# B Offline Continual World Benchmark

We evaluate RMDT on the Offline Continual World (OCW) benchmark Hu et al. (2024), which builds on Meta-World Yu et al. (2020) and extends the Continual World framework Wołczyk et al. (2021) to the offline setting. The benchmark comprises 10 robotic manipulation tasks with corresponding offline datasets. RMDT learns from these datasets sequentially, without accessing raw historical trajectories or collecting additional environment interactions during training.

**OCW10 & OCW20 Setting.** The OCW10 benchmark follows a predefined task sequence designed to capture diverse transfer relationships: hammer-v2, push-wall-v2, faucet-close-v2, push-back-v2, stick-pull-v2, handle-press-side-v2, push-v2, shelf-place-v2, window-close-v2, and peg-unplug-side-v2. Each task dataset contains 1 million transitions collected from the replay buffer of a SAC Haarnoja et al. (2018) agent trained independently on that task. The transitions span the training process and therefore include behaviors at different policy proficiency levels. The OCW20 benchmark repeats the OCW10 sequence twice, yielding 20 sequential learning stages over the same 10 tasks. This setting evaluates knowledge retention and the ability to reuse previously acquired

**Algorithm 2** Task-Agnostic Inference with RMDT

**Input:** Trained RMDT, context length $L$, routing window length $H$, earliest locking step $S$, maximum routing steps $T_{\text{router}} \geq S$, confidence threshold $\tau_{\text{conf}}$, shared target return $R_{\text{target}}$
**Initialize:** Empty interaction history and prediction buffer; locked $\leftarrow$ false; $\widehat{r}_1 \leftarrow R_{\text{target}}$
Observe the initial state $s_1$
**for** decision step $t = 1, 2, \ldots$ until episode termination **do**
  **if** locked $=$ false **then**
    Extract $z_t = \phi(h_t)$ from at most $H$ recent valid states and their preceding actions
    Compute normalized squared distances $D_i(z_t)$ to the nearest valid Gaussian cluster of each candidate task
    $q_t \leftarrow \mathrm{softmax}(\log p_\psi(\cdot \mid z_t) - \beta \log(1 + D(z_t)))$
    Append $q_t$ to the prediction buffer
    $\widehat{k} \leftarrow \arg\max_i q_t^i$
    **if** $t \geq S$ **then**
      Compute $\bar{q}_t$ by averaging the most recent five predictions, or all available predictions if fewer
      **if** $\max_i \bar{q}_t^i \geq \tau_{\text{conf}}$ **or** $t \geq T_{\text{router}}$ **then**
        $\widehat{k} \leftarrow \arg\max_i \bar{q}_t^i$
        locked $\leftarrow$ true
      **end if**
    **end if**
  **end if**
  Select the binary mask and LoRA gates associated with $\widehat{k}$
  Construct the policy context from the most recent $L$ timesteps; left-pad shorter histories and mask padding positions
  Recompute all context-token representations under the selected configuration without cross-step KV caching
  Predict action $a_t$ from the current-state representation
  Execute $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
  Update the return-to-go: $\widehat{r}_{t+1} \leftarrow \widehat{r}_t - r_t$
  Update the interaction history
**end for**

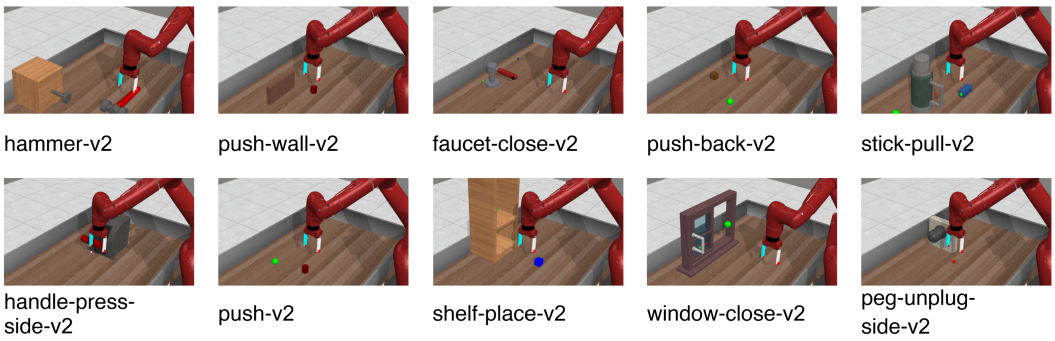



Figure 5: The Offline Continual World benchmark comprises robotic manipulation tasks from Meta-World Yu et al. (2020). The OCW10 sequence shown above is used in the main experimental results.

knowledge when tasks recur. Table 5 summarizes the manipulation objectives of the 10 OCW tasks.

**Evaluation Settings.** We consider both task-aware and task-agnostic evaluation. In the task-aware setting, the ground-truth task identity selects the corresponding binary masks and low-rank gates. In the task-agnostic setting, we withhold the task identity, and RMDT infers it from short interaction trajectories using the task router. These interactions are part of evaluation and are not used to update model parameters.

Table 5: Tasks and manipulation objectives in the OCW benchmark.

| ID | Task | Description |
|---|---|---|
| 1 | hammer-v2 | Use a hammer to drive a nail. |
| 2 | push-wall-v2 | Push a puck to a goal around a wall. |
| 3 | faucet-close-v2 | Rotate a faucet to close it. |
| 4 | push-back-v2 | Push a puck backward to a goal. |
| 5 | stick-pull-v2 | Use a stick to pull an object to a goal. |
| 6 | handle-press-side-v2 | Press a handle down from the side. |
| 7 | push-v2 | Push a puck to a goal. |
| 8 | shelf-place-v2 | Pick up an object and place it on a shelf. |
| 9 | window-close-v2 | Slide a window closed. |
| 10 | peg-unplug-side-v2 | Pull a peg out of a hole from the side. |

# C Hyperparameters Configuration

All experiments are carried out on a server with 8 NVIDIA 3090 GPUs, each with 24GB of memory. Table 6 summarizes the hyperparameter configurations used in all experiments.

# D More Training Details

## D.1 Inference Context and Trajectory Initialization

We do not reuse KV caches across inference steps. Before task locking, the router selects the active binary mask and low-rank gates at each step, and the Decision Transformer recomputes all token representations within the most recent $L$ timesteps under this configuration. Shorter histories are left-padded, with padding positions masked out in attention. Thus, all tokens within each forward pass are processed under a consistent parameter configuration, avoiding stale KV representations when the routing decision changes. This procedure requires recomputing the bounded context window at each step, rather than the entire episode history. After task locking, the selected parameter configuration remains fixed and routing inference stops, while the same window recomputation procedure continues without cross-step KV caching.

For trajectory descriptor construction, only valid positions within the routing window contribute to the statistics. At the first decision step, only the initial state is available: the first and mean states both equal $s_1$, the state displacement is zero, and the mean action is set to zero because no action has yet been executed. Therefore, $z_1 = [s_1, \mathbf{0}, s_1, \mathbf{0}]$, where $s_1$ denotes the preprocessed state supplied to the router. At subsequent steps, the mean action is computed from previously executed actions within the window, excluding the current action placeholder. We exclude padding positions from both state and action statistics.

## D.2 State Normalization and Prototype Distances

RMDT uses separate state normalization for routing and policy execution. Before training on the first task, we estimate shared per-coordinate means from all observations in its offline dataset and compute the scaling vector. These shared statistics remain fixed throughout training and evaluation. We estimate them using neither future-task datasets nor evaluation trajectories. Routing descriptors are constructed from states standardized using these shared statistics, together with actions without additional standardization. We use the same preprocessing for classifier training, Gaussian prototype estimation, and test-time routing. We apply no further descriptor-level standardization or unit-norm normalization.

The policy is trained using each task's offline state mean $\mu_k$ and scaling vector $\sigma_k = \mathrm{Std}_{\mathcal{D}^k}(s) + 10^{-6}$, stored when the task arrives and then fixed. During task-agnostic evaluation, the router first predicts $\widehat{k}$ using the shared preprocessing. The router then normalizes the original state history using

Table 6: Hyperparameter settings for RMDT. Routing configurations are reported as the default setting.

| Category | Parameter | Value |
|---|---|---|
| Transformer Architecture | Number of layers | 6 |
| | Number of attention heads | 8 |
| | Hidden dimension | 256 |
| | FFN dimension | 1024 |
| | Dropout | 0.1 |
| | Activation | ReLU |
| | Context length $L$ | 20 |
| Training | Batch size | 512 |
| | Learning rate | $10^{-4}$ |
| | Optimizer | AdamW |
| | Weight decay | $10^{-4}$ |
| | Training steps per task | 10,000 |
| | Gradient clipping norm | 0.25 |
| Binary Mask Allocation | Mask threshold | 0.5 |
| | Sigmoid scale schedule | $1 \rightarrow 50$ |
| | Capacity regularization coefficient $\lambda_{\text{cap}}$ | $10^{-4}$ |
| Low-Rank Expansion | LoRA rank $r$ | 4 |
| | LoRA scaling parameter $\alpha$ | 8 |
| | Remaining-capacity threshold | 0.2 |
| | Gate regularization coefficient $\lambda_{\text{gate}}$ | $10^{-5}$ |
| Trajectory-Based Routing | Trajectory window length $H$ | 20 |
| | Earliest decision step for locking $S$ | 20 |
| | Maximum routing decision steps | 30 |
| | Prediction aggregation window | 5 |
| | Locking confidence threshold | 0.8 |
| | Gaussian clusters per task | 4 |
| | Pseudo-features per historical task per update | 32 |
| | Gaussian distance coefficient $\beta$ | 0.1 |
| | Routing loss coefficient $\lambda_{\text{router}}$ | 1.0 |
| Evaluation | RTG | 4500 |
| | Evaluation interval (training steps) | 2,000 |
| | Maximum episode length | 500 |

$(\mu_{\widehat{k}}, \sigma_{\widehat{k}})$ for policy execution, along with the corresponding binary mask and low-rank gates. Thus, both preprocessing paths operate on the original states, and selecting policy statistics requires no ground-truth task identity. Before locking, the policy context is renormalized whenever the predicted task changes. Task-aware evaluation directly selects the statistics and parameter configuration using the provided task index, with the same trained model.

During routing, prototype matching uses a variance-normalized squared distance:

$$D_i(z) = \min_{\text{c}} \frac{1}{d_z} \sum_{q=1}^{d_z} \frac{(z_q - \mu_{c,q}^i)^2}{\max(v_{c,q}^i, 10^{-4})}. \tag{9}$$

Here, $v^i_{c,q}$ is the sample variance estimated from descriptors assigned to cluster $c$ of task $i$. This scaling reduces sensitivity to differences in coordinate variability, while the variance floor prevents division by near-zero values. The routing score combines the classifier logit $\ell_i(z)$ with this distance as $s_i(z) = \ell_i(z) - \beta \log(1 + D_i(z))$, followed by a softmax over candidate tasks. Online cluster assignment during training uses mean squared Euclidean distance in the descriptor space.

### D.3 Gaussian Prototype Initialization and Updates.

Gaussian prototype statistics are maintained incrementally during minibatch training. For each new task, the first $C$ descriptors from the first sampled minibatch initialize the $C$ cluster means in batch order, with each cluster assigned an initial count of 1 and a sum of squared deviations of 0. If the minibatch contains fewer than $C$ descriptors, initialize the remaining clusters from subsequent minibatches.

Assign the remaining descriptors in each minibatch to the nearest initialized prototype using mean squared Euclidean distance. Compute these assignments jointly before updating the cluster statistics. For a cluster with statistics $(n, \mu, M_2)$, let the assigned descriptors have count $m$, mean $\mu_B$, and sum of squared deviations $M_{2,B}$. The cumulative statistics are updated as

$$n' = n + m, \qquad \Delta = \mu_B - \mu, \tag{10}$$

$$\mu' = \mu + \frac{m}{n'}\Delta, \tag{11}$$

$$M_2' = M_2 + M_{2,B} + \frac{nm}{n'}\Delta \odot \Delta. \tag{12}$$

Clusters receiving no descriptors retain their existing statistics. The diagonal variance used for routing and pseudo-feature sampling is computed elementwise as $v = \max\big(M_2 / \max(n-1, 1), 10^{-4}\big)$. At each training iteration, we update current-task statistics without gradient computation before evaluating the routing classification loss. Historical-task statistics remain fixed. The cluster counts accumulate descriptor occurrences across training iterations, including repeated sampling.

### D.4 Preservation of Historical Weights

To preserve historical weights exactly, we mask their gradients and restore their values after each optimizer step, preventing updates from weight decay or optimizer momentum.

## E More Experiments

### E.1 Visualization of the Training Process

Figure 6 illustrates RMDT's learning curves in task-aware and task-agnostic settings, with both showing relatively stable average performance in the second half of the sequence. Figure 7 complements this illustration with task-wise results on OCW10. RMDT achieves high success rates on several tasks in both settings, including hammer, faucet-close, handle-press-side, and window-close. At the same time, task-aware evaluation yields better performance on tasks such as shelf-place and peg-unplug-side.

### E.2 Robustness to Task Arrival Order

To evaluate RMDT's robustness to task ordering, we run experiments on OCW10 under four task orders in both task-aware and task-agnostic settings. Table 7 details the task orders. Order 0 matches the order used in the main experiments, while Orders 1–3 are alternative permutations of the same 10 tasks. As shown in Tables 8 and 9, RMDT consistently achieves superior performance across both settings and all task orders, further demonstrating its robustness to task-order variations.

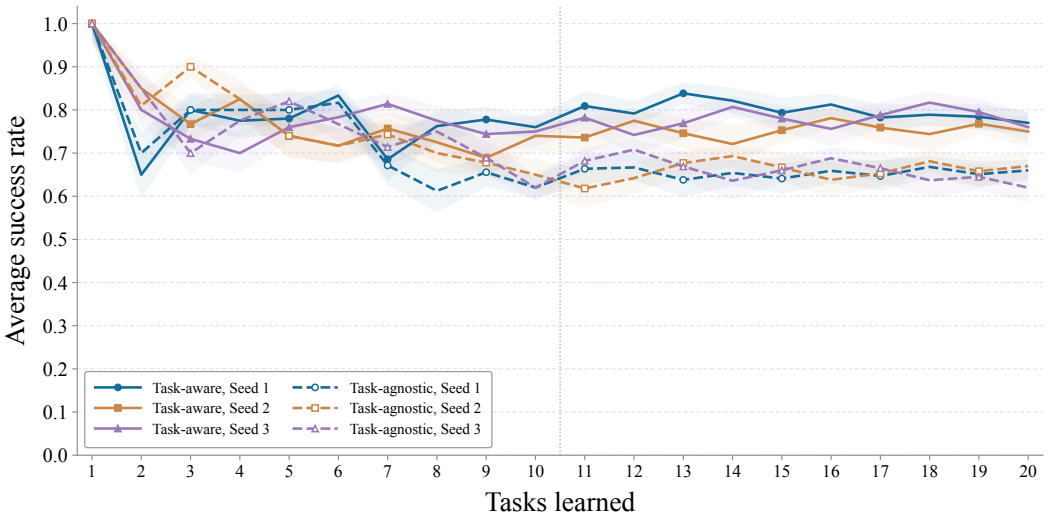
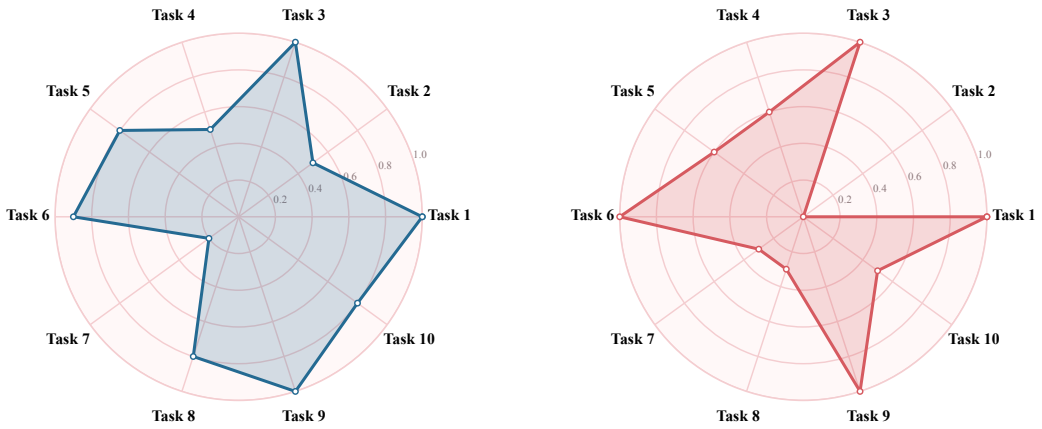



Figure 6: Average performance on seen tasks across three random seeds on OCW20 under task-aware and task-agnostic settings.




**(a) Task-aware** **(b) Task-agnostic**

Figure 7: Task-wise Performance of RMDT on OCW10.

Table 7: Task sequences used to evaluate robustness to task ordering on OCW10.

| Order | Task Sequence |
|---|---|
| Order 0 | hammer-v2, push-wall-v2, faucet-close-v2, push-back-v2, stick-pull-v2, handle-press-side-v2, push-v2, shelf-place-v2, window-close-v2, peg-unplug-side-v2 |
| Order 1 | push-v2, window-close-v2, peg-unplug-side-v2, shelf-place-v2, handle-press-side-v2, push-back-v2, hammer-v2, stick-pull-v2, push-wall-v2, faucet-close-v2 |
| Order 2 | handle-press-side-v2, peg-unplug-side-v2, push-back-v2, stick-pull-v2, push-v2, shelf-place-v2, faucet-close-v2, window-close-v2, push-wall-v2, hammer-v2 |
| Order 3 | push-wall-v2, handle-press-side-v2, push-v2, hammer-v2, peg-unplug-side-v2, stick-pull-v2, shelf-place-v2, faucet-close-v2, window-close-v2, push-back-v2 |

Table 8: Robustness to different task arrival orders on OCW10 under the task-aware setting. Results are averaged over three random seeds.

| Method | Order0 | Order1 | Order2 | Order3 | Avg. |
|---|---|---|---|---|---|
| PM | $0.26_{\pm 0.01}$ | $0.26_{\pm 0.10}$ | $0.25_{\pm 0.03}$ | $0.27_{\pm 0.01}$ | 0.26 |
| A-GEM | $0.12_{\pm 0.04}$ | $0.09_{\pm 0.03}$ | $0.08_{\pm 0.03}$ | $0.11_{\pm 0.04}$ | 0.10 |
| OER | $0.56_{\pm 0.04}$ | $0.56_{\pm 0.04}$ | $0.53_{\pm 0.05}$ | $0.55_{\pm 0.04}$ | 0.55 |
| EWC | $0.16_{\pm 0.02}$ | $0.12_{\pm 0.02}$ | $0.11_{\pm 0.02}$ | $0.15_{\pm 0.03}$ | 0.14 |
| MAS | $0.29_{\pm 0.04}$ | $0.17_{\pm 0.01}$ | $0.16_{\pm 0.06}$ | $0.20_{\pm 0.04}$ | 0.21 |
| VCL | $0.14_{\pm 0.03}$ | $0.11_{\pm 0.03}$ | $0.09_{\pm 0.03}$ | $0.12_{\pm 0.02}$ | 0.12 |
| UCL | $0.35_{\pm 0.04}$ | $0.30_{\pm 0.04}$ | $0.27_{\pm 0.05}$ | $0.31_{\pm 0.04}$ | 0.31 |
| LoRA | $0.54_{\pm 0.03}$ | $0.47_{\pm 0.02}$ | $0.35_{\pm 0.03}$ | $0.43_{\pm 0.05}$ | 0.45 |
| PackNet | $0.64_{\pm 0.06}$ | $0.67_{\pm 0.01}$ | $0.65_{\pm 0.03}$ | $0.65_{\pm 0.04}$ | 0.65 |
| WSN | $0.67_{\pm 0.04}$ | $0.66_{\pm 0.03}$ | $0.64_{\pm 0.04}$ | $0.65_{\pm 0.03}$ | 0.66 |
| CompoFormer | $0.69_{\pm 0.01}$ | $0.70_{\pm 0.02}$ | $0.68_{\pm 0.03}$ | $0.69_{\pm 0.03}$ | 0.69 |
| VQ-CD | $0.71_{\pm 0.03}$ | $0.72_{\pm 0.03}$ | $0.70_{\pm 0.03}$ | $0.71_{\pm 0.03}$ | 0.71 |
| **RMDT** | $\mathbf{0.75}_{\pm 0.01}$ | $\mathbf{0.74}_{\pm 0.02}$ | $\mathbf{0.72}_{\pm 0.03}$ | $\mathbf{0.73}_{\pm 0.02}$ | **0.74** |

Table 9: Robustness to different task arrival orders on OCW10 under the task-agnostic setting. Results are averaged over three random seeds.

| Method | Order0 | Order1 | Order2 | Order3 | Avg. |
|---|---|---|---|---|---|
| PM | $0.11_{\pm 0.03}$ | $0.11_{\pm 0.06}$ | $0.10_{\pm 0.04}$ | $0.12_{\pm 0.03}$ | 0.11 |
| A-GEM | $0.09_{\pm 0.03}$ | $0.07_{\pm 0.02}$ | $0.06_{\pm 0.02}$ | $0.08_{\pm 0.03}$ | 0.08 |
| OER | $0.49_{\pm 0.01}$ | $0.48_{\pm 0.04}$ | $0.45_{\pm 0.06}$ | $0.47_{\pm 0.05}$ | 0.47 |
| EWC | $0.11_{\pm 0.03}$ | $0.09_{\pm 0.03}$ | $0.08_{\pm 0.02}$ | $0.10_{\pm 0.03}$ | 0.10 |
| MAS | $0.20_{\pm 0.04}$ | $0.13_{\pm 0.03}$ | $0.12_{\pm 0.04}$ | $0.15_{\pm 0.03}$ | 0.15 |
| VCL | $0.10_{\pm 0.03}$ | $0.08_{\pm 0.02}$ | $0.07_{\pm 0.03}$ | $0.09_{\pm 0.02}$ | 0.09 |
| UCL | $0.29_{\pm 0.04}$ | $0.26_{\pm 0.05}$ | $0.23_{\pm 0.04}$ | $0.26_{\pm 0.03}$ | 0.26 |
| LoRA | $0.42_{\pm 0.05}$ | $0.37_{\pm 0.03}$ | $0.29_{\pm 0.05}$ | $0.34_{\pm 0.04}$ | 0.36 |
| PackNet | $0.48_{\pm 0.05}$ | $0.50_{\pm 0.03}$ | $0.47_{\pm 0.04}$ | $0.48_{\pm 0.05}$ | 0.48 |
| WSN | $0.50_{\pm 0.05}$ | $0.49_{\pm 0.05}$ | $0.46_{\pm 0.04}$ | $0.47_{\pm 0.03}$ | 0.48 |
| CompoFormer | $0.52_{\pm 0.04}$ | $0.53_{\pm 0.04}$ | $0.50_{\pm 0.05}$ | $0.51_{\pm 0.04}$ | 0.52 |
| VQ-CD | $0.44_{\pm 0.01}$ | $0.45_{\pm 0.03}$ | $0.42_{\pm 0.05}$ | $0.43_{\pm 0.04}$ | 0.44 |
| **RMDT** | $\mathbf{0.63}_{\pm 0.02}$ | $\mathbf{0.62}_{\pm 0.04}$ | $\mathbf{0.60}_{\pm 0.05}$ | $\mathbf{0.61}_{\pm 0.03}$ | **0.62** |

### E.3 Effect of Weight Reuse and Capacity Regularization

Table 10 examines the contributions of weight reuse and capacity regularization under the task-agnostic setting on OCW10. Removing weight reuse reduces AP from 0.63 to 0.58 and FWT from 0.05 to 0.02, suggesting that selectively reusing historical knowledge benefits both overall performance and transfer to subsequent tasks. Without capacity regularization, AP decreases to 0.60, while forgetting increases from 0.03 to 0.10. These results support the role of capacity regularization in balancing current-task learning with capacity preservation for future tasks. Full RMDT achieves the best mean results across all three metrics, indicating that the two components contribute to effective continual learning.

Table 10: Ablation study of weight reuse and capacity regularization on OCW10 under the task-agnostic setting. Results are reported as mean ± standard deviation across random seeds.

| Variant | AP ↑ | F ↓ | FWT ↑ |
|---|---|---|---|
| w/o Weight Reuse | $0.58_{\pm 0.06}$ | $0.08_{\pm 0.04}$ | $0.02_{\pm 0.02}$ |
| w/o Capacity Regularization | $0.60_{\pm 0.05}$ | $0.10_{\pm 0.04}$ | $0.02_{\pm 0.02}$ |
| Full RMDT | $0.63_{\pm 0.02}$ | $0.03_{\pm 0.02}$ | $0.05_{\pm 0.01}$ |

### E.4 Forward Transfer Evaluation Protocol

Forward transfer measures the improvement in task performance before learning that task relative to a randomly initialized reference. For stage $i$, we compare $p_i((i-1)\delta)$, measured after completing stage $i-1$ and before training on stage $i$, with the initialization performance $p_i(0)$. We average the differences over stages $i = 2, \ldots, K$. All evaluations use the same target return-to-go and perform no gradient updates or Gaussian prototype updates. To obtain $p_i(0)$, we evaluate the randomly initialized backbone before training the first task, when no task-specific masks or low-rank expansions have been created. We normalize state inputs using shared per-coordinate statistics estimated from the first task's offline dataset. These statistics are computed once and remain fixed throughout the experiment. To obtain $p_i((i-1)\delta)$, RMDT selects among previously learned configurations using its trajectory-based router, since the upcoming stage has no learned configuration. The router processes observations using the fixed shared normalization, and the predicted historical configuration determines the policy normalization statistics, binary mask, and low-rank gates. We use this procedure for FWT evaluation in both the task-aware and task-agnostic settings. This evaluation does not use the upcoming task's dataset-specific normalization statistics. Although we load offline datasets during setup, future-task observations do not contribute to the shared normalization, router training, or historical policy configurations.

In OCW20, stages 11–20 repeat the task sequence from stages 1–10. Consequently, the aggregate FWT captures both transfer to first-encounter tasks and reuse of historical knowledge when tasks recur. We therefore interpret OCW20 FWT as pre-stage performance improvement over initialization, encompassing both forms of transfer.

### E.5 Parameter Allocation Across the OCW20 Task Sequence

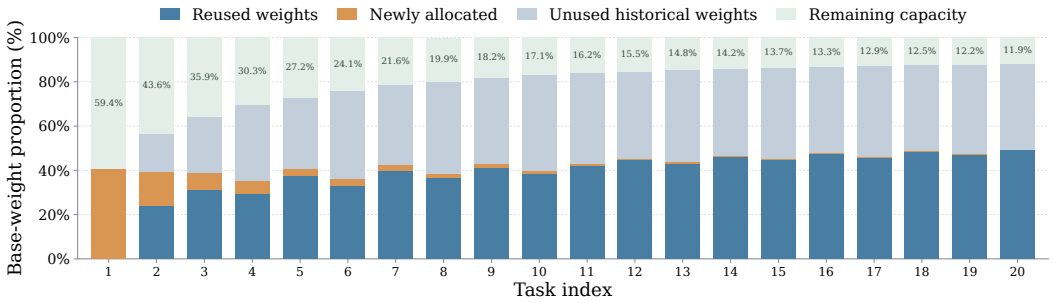



Figure 8: Parameter allocation across sequential tasks on OCW20.

### E.6 Effect of the Number of Gaussian Prototypes.

Table 11 examines the effect of the number of Gaussian prototypes per task on OCW10 under the task-agnostic setting. Increasing $C$ from 1 to 4 improves both routing accuracy and average performance, suggesting that multiple prototypes better capture within-task trajectory variations. Increasing $C$ to 8 provides only marginal gains, while $C = 16$ slightly reduces both metrics. We therefore use $C = 4$ as the default, balancing routing effectiveness with compact prototype storage.

Table 11: Sensitivity to the number of Gaussian prototypes per task on OCW10 under the task-agnostic setting.

| Prototypes $C$ | Routing accuracy ↑ | AP ↑ |
|---|---|---|
| 1 | 0.82 | 0.56 |
| 2 | 0.87 | 0.60 |
| 4 | 0.90 | 0.63 |
| 8 | 0.91 | 0.64 |
| 16 | 0.89 | 0.62 |