NEURAL CONSOLIDATION PROCESS: SELECTIVE TAGGING AND VALIDATED CAPTURE FOR CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning in deep neural networks is severely bottlenecked by catastrophic forgetting. While the human brain elegantly resolves this stability-plasticity dilemma through systems memory consolidation, i.e., gradually transferring short-term memory into stable neocortical traces, mimicking this in artificial networks without destructive overwriting remains an open challenge. In this paper, we propose **Neural Consolidation Process (NeuralCP)**, a novel continual learning framework inspired by the biological Synaptic Tagging and Capture (STC) mechanism. NeuralCP models short-term memory using a lightweight LoRA module to rapidly encode task-specific knowledge. Instead of accumulating task-specific adapters or relying on rigid parameter-space regularization, NeuralCP employs function-space attribution to explicitly tag modules responsible for critical representation shifts. We then introduce a replay-validated capture mechanism, utilizing a minimal calibration set strictly to filter for highly beneficial and consistently recallable memory tags. Finally, these validated tags are consolidated into the backbone via residual distillation within a stable-plastic subspace decomposition, dedicatedly protecting previously consolidated knowledge while maintaining future adaptability. Crucially, NeuralCP introduces zero additional parameters or overhead at inference. Extensive evaluations across eight sequential tasks (including GLUE, HellaSwag, GSM8K, and BoolQ) demonstrate that NeuralCP promotes average continual learning scores by 22.72, 7.39, and 8.05 points on Qwen3-0.6B, 4B, and 8B respectively, setting a new state-of-the-art. Besides, NeuralCP exhibits consistent improvements and scalability on visual Transformers.

1 INTRODUCTION

Large Language Models (LLMs) have achieved remarkable success across a wide range of tasks, yet their ability to continually acquire new knowledge without catastrophically forgetting previously learned capabilities remains severely limited French (1999); Kemker et al. (2018); Robins (1995); Serra et al. (2018). In sequential task settings, naive fine-tuning inherently overwrites existing representations, fundamentally restricting the deployment of foundation models in dynamic, real-world environments. Existing continual learning (CL) methods attempt to mitigate this through several distinct paradigms. Modular methods, such as LoRA-Mixer Li et al. (2025) and MixLoRA Huang et al. (2024); Li et al. (2024); Wu et al. (2024); Shazeer et al. (2017), avoid overwriting by isolating task-specific experts, but their parameter count and routing complexity scale unsuitably with the number of tasks. Regularization-based approaches (e.g., EWC Aich (2021), EWCLoRA Zheng et al. (2026); Qiang et al. (2024)) penalize updates to mathematically important weights, yet they typically operate rigidly in parameter space and often over-constrain future adaptation. Orthogonalization methods Qiu et al. (2025b;a), including SDLoRA Wu et al. (2025) and O-LoRA Büyükakyüz (2024), attempt to project updates into less conflicting geometric subspaces, but these constraints become brittle when tasks naturally require the reuse of overlapping functional pathways. Among existing solutions, replay-based continual learning draws the most direct inspiration from the human brain, mimicking biological neural replay to preserve past performance. However, standard replay methods largely treat this as a simple data-rehearsal or gradient-mixing exercise. The fundamental question of how new knowledge can be safely consolidated into the base model, without destructively overwriting

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.

established parameter spaces or causing representation drift, remains an unresolved, open problem in artificial neural networks. To address this gap, we look beyond high-level behavioral replay and examine the specific, microscopic mechanisms of human memory. The human brain elegantly solves the stability-plasticity dilemma via systems memory consolidation. When humans acquire new information, transient experiences (short-term memories) are not immediately and indiscriminately written into the neocortex. Instead, according to the well-established Synaptic Tagging and Capture (STC) hypothesis Redondo & Morris (2011); Moncada et al. (2015); Wang et al. (2010), short-term task adaptation temporarily ***"tags"*** specific synapses. Only the memory tags that are functionally salient and later ***"validated***, i.e., reactivated during neural replay, are ***"captured*** and selectively consolidated into stable, long-term neocortical networks. The rest are allowed to naturally fade. This profound biological mechanism suggests a critical paradigm shift for CL: the core challenge is not merely deciding how to constrain global parameter updates, but selectively identifying, validating, and consolidating only the task-induced changes that genuinely deserve durable retention.

Inspired by this neurobiological mechanism, we propose the **Neural Consolidation Process (NeuralCP)**, a novel continual learning framework that closely mimics the brain's synaptic tagging and selective consolidation pipeline. Rather than treating replay as mere data rehearsal or gradient mixing, NeuralCP treats continual task learning as a strict progression from transient adapters (represented as LoRA modules in our paper) to stable neocortical memory (base model parameters). First, NeuralCP trains a lightweight LoRA module to rapidly capture task-specific knowledge, serving as the short-term memory. Second, NeuralCP performs precise functional attribution, analyzing hidden-state shifts rather than static parameter weights, to ***tag*** the specific network modules responsible for task-salient representational shifts. Third, we repurpose replay not for standard rehearsal, but for ***validated capture***. Using a minimal calibration set, NeuralCP reactivates these tagged tags to strictly test if they are beneficial and consistently recallable.Finally, to safely consolidate these validated tags into long-term memory, NeuralCP projects updates through a stable-plastic subspace decomposition, where stability preserves previously consolidated knowledge while plasticity reserves capacity for absorbing new task-specific information, and applies residual distillation. This ensures that new knowledge is delicately woven into the base model's plastic pathways, deliberately protecting previously consolidated stable knowledge while preserving future adaptability. Crucially, unlike modular methods that rely on an ever-growing bank of experts, NeuralCP integrates validated knowledge directly into the base model. After the consolidation phase, NeuralCP introduces zero additional parameters and incurs no extra inference overhead. We validate NeuralCP through extensive experiments in both language and vision settings. For LLM continual learning, we evaluate models ranging from 0.6B to 8B parameters on eight sequential tasks spanning GLUE Wang et al. (2019), HellaSwag, GSM8K Cobbe et al. (2021), and BoolQ. NeuralCP consistently establishes a new state-of-the-art, improving average continual learning performance by 22.72, 7.39, and 8.05 points on Qwen3-0.6B Yang et al. (2025), Qwen3-4B, and Qwen3-8B, respectively, compared to their base models. We further verify its generality and scalability on vision transformers Dosovitskiy et al. (2021) under class-incremental learning scenarios. These results compellingly demonstrate that continual learning fundamentally benefits from selectively consolidating validated, function-space tags—proving that biological memory mechanisms hold the key to seamlessly updating foundation models.

2 Related Works

Continual Learning Continual learning aims to enable models to learn from a sequence of non-i.i.d. tasks while preserving previously acquired knowledge. A key challenge is catastrophic forgetting, where adapting to new tasks degrades performance on earlier ones. Depending on the availability of task identity, continual learning can be categorized into several settings, with class-incremental learning being among the most challenging scenarios van de Ven et al. (2022). Existing approaches mainly fall into three categories. Replay-based methods alleviate forgetting by revisiting previous data or generated representations, but typically require memory buffers, data storage, or additional generative models van de Ven et al. (2020). Regularization-based methods, such as EWC Aich (2021), SI Zenke et al. (2017a), and MAS Aljundi et al. (2018), estimate parameter importance and constrain updates to preserve previous knowledge Kirkpatrick et al. (2017); Zenke et al. (2017b); Aljundi et al. (2018). Projection-based methods reduce interference by isolating task-specific parameter or gradient subspaces Liang & Li (2024); Wu et al. (2025); Qiang et al. (2024); Büyükakyüz (2024); Qian et al. (2025). Recent studies have also revealed severe forgetting during continual fine-tuning of large language models, motivating scalable approaches for efficient adaptation and knowledge preservation Luo et al. (2023).

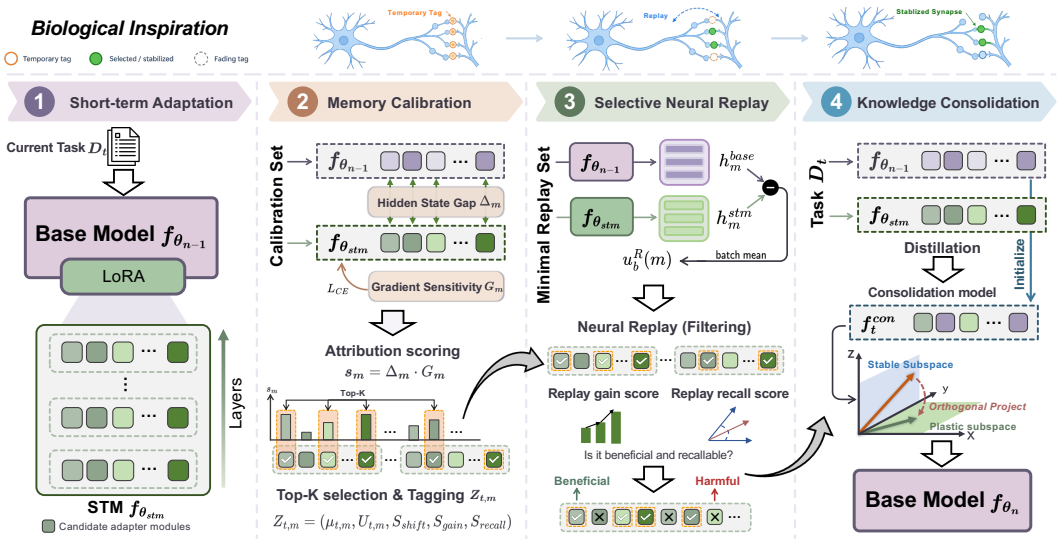


Figure 1: NeuralCP Overview: For each incoming task, NeuralCP first trains a short-term LoRA model, then identifies task-relevant modules through functional attribution and replay-validated memory tags, and finally integrates acceptable tags into the selected modules through residual distillation. Each module has a stable-plasticity space to prevent forgetting caused by selecting the same module for different tasks.

Parameter-Efficient Continual Learning Parameter-efficient continual learning adapts large pre-trained models by updating only a small number of additional parameters while keeping the backbone fixed. Prompt-based methods, such as L2P Wang et al. (2022b), DualPrompt Wang et al. (2022a), and CODA-Prompt Smith et al. (2023), maintain task-aware prompt components to reduce interference Wang et al. (2022b;a); Smith et al. (2023). Adapter- and LoRA-based approaches provide a more structured parameter-efficient adaptation mechanism. InfLoRA learns task-specific low-rank subspaces to reduce interference Liang & Li (2024), O-LoRA enforces orthogonality among LoRA subspaces for language model continual learning Wang et al. (2023), and SD-LoRA improves the stability–plasticity trade-off by decoupling LoRA components Wu et al. (2025). However, many existing PEFT approaches require maintaining task-specific prompts or LoRA modules that grow with the number of tasks, introducing additional memory and inference overhead. In contrast, our method uses LoRA as a temporary adaptation module and consolidates validated knowledge into the backbone, eliminating extra parameters after consolidation.

Memory-Inspired Continual Learning Another line of work studies continual learning from the perspective of memory formation and consolidation. EWC protects previous knowledge by preserving parameters with high Fisher importance Kirkpatrick et al. (2017), while SI and MAS estimate parameter importance through online trajectory information and output sensitivity, respectively Zenke et al. (2017a); Aljundi et al. (2018). Replay-based methods have also been interpreted as memory mechanisms that revisit stored or generated representations rather than directly replaying raw data van de Ven et al. (2020). Recent theoretical studies further highlight the importance of adaptive consolidation and multi-timescale plasticity in continual learning Zenke & Laborieux (2024). Our work shares this memory consolidation perspective but differs by explicitly identifying and validating which task-induced updates should be preserved, and then writing them back into the backbone for long-term accumulation without external memory or task-dependent inference.

3 METHODS

Our Neural Consolidation Process (NeuralCP) represents a novel paradigm shift in continual learning: rather than solely focusing on how to mathematically constrain parameter updates to prevent interference Aich (2021); Zheng et al. (2026); Büyükakyüz (2024), we argue the core challenge is determining which specific task-induced changes genuinely deserve long-term retention without destructively overwrite past knowledge. This perspective is deeply inspired by the biological Synaptic Tagging and Capture (STC) mechanism Redondo & Morris (2011); Moncada et al. (2015), wherein transient cognitive experiences form temporary synaptic tags, but only functionally salient and repeatedly validated memory traces are permanently captured by the neocortex. To mimic

this biological mechanism in artificial neural networks, NeuralCP is structured into three carefully designed stages: (1) short-term task adaptation utilizing task-specific lightweight LoRA module to represent transient memory, following a functional attribution process to tag task-induced shifts, (2) replay-based validation to explicitly identify valuable changes, and (3) stable–plastic selective consolidation to safely integrate these traces into long-term memory. The full pipeline of NeuralCP is illustrated in Figure 1.

3.1 Short-Term Memory Tagging

Consider a standard continual learning scenario where a sequence of tasks $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$ arrives incrementally. Let $f_{\theta_{t-1}}$ denote the base model after learning tasks up to $t-1$. When presented with a new task $\mathcal{D}_t$, NeuralCP instantiates a lightweight, transient LoRA adapter on top of $f_{\theta_{t-1}}$, effectively modeling the rapid formation of short-term memory, in line with parameter-efficient continual learning approaches Büyükakyüz (2024); Qiang et al. (2024); Liang & Li (2024).

Short-Term LoRA Adapter For a newly arriving task $\mathcal{D}_t$, we instantiate a LoRA module on top of the current base model: $f_t^{\text{stm}} = f_{\theta_{t-1},\phi_t}$, where ϕ_t denotes the LoRA parameters for task t.

$$\phi_t^\star = \arg\min_{\phi_t} \; \mathcal{L}_{\text{task}}\big(f_{\theta_{t-1},\phi_t}; \mathcal{D}_t\big)\,. \tag{1}$$

where $\mathcal{L}_{\text{task}}$ is the task-specific supervision. ϕ_t represents the parameter shifts of the base network for accommodating current task, acting as a short-term memory capturing task-specific knowledge.

Module-wise Functional Attribution of Task-Induced Shifts The short-term LoRA adapter effectively captures task-specific knowledge by introducing distributed parameter shifts across the base network. Crucially, not all of these parametric shifts are equally vital for task performance; many merely represent redundant co-adaptations or optimization noise Wang et al. (2010). To isolate and consolidate only the most salient memory traces, NeuralCP performs functional attribution on each constituent module (e.g., the attention-layer adapters) within the short-term LoRA adapter ϕ_t. We evaluate the divergence in latent activations between the frozen base model $f_{\theta_{t-1}}$ and the short-term adapted model f_{θ_{t-1},ϕ_t} using a minimal calibration set $\mathcal{C}_t \subset \mathcal{D}_t$. Empirically, restricting $|\mathcal{C}_t|$ to 256 samples ensures that the computational overhead remains negligible without compromising the fidelity of the attribution. For each candidate adapter module m, we compute two complementary functional metrics: the adapter-induced hidden-state shift Δ_m and the task-level sensitivity G_m:

$$\Delta_m = \mathbb{E}_{x\sim\mathcal{C}_t} \left\| h_m^{\text{stm}}(x) - h_m^{\text{base}}(x) \right\|_2, \quad G_m = \mathbb{E}_{(x,y)\sim\mathcal{C}_t} \left\| \frac{\partial \mathcal{L}_{\text{task}}(f_t^{\text{stm}}(x), y)}{\partial h_m^{\text{stm}}} \right\|_2. \tag{2}$$

Here, $h_m^{\text{stm}}(x)$ and $h_m^{\text{base}}(x)$ denote the hidden activations of the short-term adapted model and the base model at module m, respectively. Δ_m, quantifies the magnitude of the representational shift induced by the transient adapter, while G_m measures the sensitivity of the task-specific loss to the module's output. To identify modules that are both heavily modified and functionally critical, we independently normalize these signals and compute their product:

$$S_m = \widetilde{\Delta}_m \cdot \widetilde{G}_m, \tag{3}$$

where $\widetilde{\Delta}_m$ and $\widetilde{G}_m$ represent the normalized scores. S_m ensures that highly ranked modules are genuinely instrumental to task performance. Furthermore, to explicitly protect previously integrated knowledge, we apply a conflict-aware correction:

$$S_m^{\text{conf}} = S_m - \lambda_{\text{conf}} C_m, \qquad C_m = \min\left(1,\ \frac{r_m}{r_{\max}}\right). \tag{4}$$

Here, C_m serves as a conflict penalty that quantifies the geometric overlap between the candidate representational shift at module m and its previously consolidated stable subspace. This penalty is explicitly tied to the module's remaining capacity, where r_m denotes the current stable rank already committed to memory, and $r_{\max}$ represents the maximum stable rank allowed per module (see Appendix F for formal definitions). Finally, NeuralCP isolates the final set of salient memory traces, $\mathcal{S}_t$, by applying a top-K selection criterion based on the penalized score S_m^{conf}. By constraining the integration to this sparse subset of modules, NeuralCP elegantly transforms what is traditionally an indiscriminate global parameter update into a highly targeted, conflict-minimized memory allocation.

Tagging Standard continual learning methods typically write task-induced parameter shifts directly into the base network with certain constraints. In contrast, NeuralCP defers this integration. For each module in salient memory trace set, $m \in \mathcal{S}_t$, NeuralCP encapsulates the shift into an explicit representational summary, which we term a *memory tag*, $z_{t,m}$:

$$z_{t,m} = \left(\mu_{t,m}, U_{t,m}, s^{\text{shift}}_{t,m}, s^{\text{gain}}_{t,m}, s^{\text{recall}}_{t,m}\right). \tag{5}$$

Here, $\mu_{t,m}$ represents the mean shift vector, $U_{t,m}$ is a low-rank basis capturing its principal directions (see Appendix G), and the three scalar scores, $s^{\text{shift}}_{t,m}$, $s^{\text{gain}}_{t,m}$, and $s^{\text{recall}}_{t,m}$, quantify the magnitude, utility, and consistency of the adaptation, respectively. To compute these metrics, we partition the previously defined calibration set $\mathcal{C}_t$ into B batches. For a given batch b, we compute the hidden-state shift u_b using $u_b = \mathbb{E}_{x \in b}(h^{\text{stm}}_m(x) - h^{\text{base}}_m(x))$. And we average u_b of all batches and get $\mu_{t,m}$. The shift score simply inherits the overall magnitude from the attribution stage: $s^{\text{shift}}_{t,m} = \Delta_m$. The gain score measures the direct task-level benefit derived from the representational shift, and is computed as:

$$s^{\text{gain}}_{t,m} = \frac{1}{B}\sum_{b=1}^{B} \tilde{r}_b\, g_b, \quad g_b = \max(0, \mathcal{L}^{(b)}_{\text{base}} - \mathcal{L}^{(b)}_{\text{stm}}), \quad \tilde{r}_b = \frac{r_b}{\max_{b'} r_{b'}}. \tag{6}$$

where g_b denotes the non-negative loss improvement provided by the short-term adapter on batch b, and $r_b = \|u_b\|_2$ is the norm of the batch residual. This formulation deliberately favors memory tags that are both strongly activated and functionally beneficial. Finally, the recall score evaluates the cross-batch consistency of the adaptation. We define the recall score via cosine similarity:

$$s^{\text{recall}}_{t,m} = \frac{1}{2}\left(1 + \frac{1}{B}\sum_{b=1}^{B} \cos(u_b, , \mu_{t,m})\right)$$

Consequently, a memory tag achieves a high recall score only if its residual direction is robustly reproduced across diverse inputs. The collection of all generated tags for the current task is denoted as $\mathcal{Z}^{\text{new}}_t = \{z_{t,m} \mid m \in \mathcal{S}_t\}$.

3.2 Replay-validated Capture

We use a global memory buffer $\mathcal{M}$ to store the memory tags. Let $\mathcal{M}_{t-1}$ denote this buffer prior to learning task t (empirically bounded to a capacity of 512 tags). The buffer is first updated to incorporate the current task's tags: $\mathcal{M}_t = \mathcal{M}_{t-1} \cup \mathcal{Z}^{\text{new}}_t$. Unlike conventional replay-based continual learning methods, which typically rehearse past data to mitigate forgetting via gradient updates, NeuralCP repurposes replay as a strict validation gate. It is utilized exclusively to verify whether a stored memory tag remains functionally beneficial and consistently recallable on unseen data. To maximize efficiency, NeuralCP does not validate the entire buffer. Instead, it restricts validation to tags associated with the modules explicitly targeted by the current task:

$$\mathcal{Z}_t = \{z_{\tau,m} \in \mathcal{M}_t \mid m \in \mathcal{S}_t,\ \tau \le t\}. \tag{7}$$

Consequently, $\mathcal{Z}_t$ encompasses both the newly generated tags and any historical tags whose module identities **overlap** with $\mathcal{S}_t$. This formulation naturally facilitates the reuse of past knowledge on shared functional pathways, without indiscriminately expanding the consolidation footprint to irrelevant historical modules. Validation is executed on a held-out set $\mathcal{R}_t \subset \mathcal{D}_t$ of 256 samples, perfectly disjoint from the calibration set $\mathcal{C}_t$. For each candidate tag $z_{\tau,m} \in \mathcal{Z}_t$, NeuralCP forwards both the frozen base model and the short-term adapted model on $\mathcal{R}_t$ to compute the validation residual at module m:

$$u^{\mathcal{R}}_b(m) = \mathbb{E}_{x \in b}\left[h^{\text{stm}}_m(x) - h^{\text{base}}_m(x)\right], \qquad b \subset \mathcal{R}_t. \tag{8}$$

Crucially, this residual is not used to instantiate a new tag, but rather to assess the generalizability of an existing tag on held-out samples. Analogous to the initial tagging phase, NeuralCP computes a replay gain score $s^{\text{gain}}_{\mathcal{R}}(z_{\tau,m})$ and a replay recall score $s^{\text{recall}}_{\mathcal{R}}(z_{\tau,m})$ over $\mathcal{R}_t$. The gain score verifies whether the adapter-induced residual remains beneficial on the held-out set. The recall score computes the cosine similarity between $u^{\mathcal{R}}_b(m)$ and the stored prototype $\mu_{\tau,m}$ to confirm consistent activation (for historical tags where $\tau < t$, this prototype is retrieved directly from the memory buffer). A candidate tag $z_{\tau,m} \in \mathcal{Z}_t$ is authorized for capture if and only if it satisfies predefined robustness thresholds:

$$s^{\text{gain}}_{\mathcal{R}}(z_{\tau,m}) \ge \gamma_{\text{gain}} \quad \text{and} \quad s^{\text{recall}}_{\mathcal{R}}(z_{\tau,m}) \ge \eta_{\text{rec}}, \tag{9}$$

where $\gamma_{\text{gain}} = 0.5$ and $\eta_{\text{rec}} = 0.6$ across all experiments. The subset of accepted tags is denoted as $\widehat{\mathcal{Z}}_t$. Finally, the corresponding modules authorized for consolidation are defined as:

$$\mathcal{S}_t^{\text{cap}} = \{m \in \mathcal{S}_t \mid \exists z_{\tau,m} \in \widehat{\mathcal{Z}}_t\}. \tag{10}$$

Only these replay-validated modules are advanced to the subsequent consolidation stage. By rigorously filtering the candidates, this replay-validated capture ensures that noisy or idiosyncratic tags are discarded, while robust, generalizable memory traces are safely integrated into the backbone.

3.3 Stable-Plastic Selective Consolidation

After replay validation, NeuralCP consolidates the accepted memory tags into the base network. Let f_t^{con} be a trainable consolidation model initialized from the current base model $f_{\theta_{t-1}}$, and let $f_t^{\text{stm}} = f_{\theta_{t-1},\phi_t^\star}$ be the short-term adapted model for task t. We freeze both $f_{\theta_{t-1}}$ and $\phi_t^\star$, and update only the replay-validated modules $\mathcal{S}_t^{\text{cap}}$ in f_t^{con}.

Plastic Subspace To limit interference with previously consolidated knowledge, NeuralCP maintains a module-level stable–plastic subspace decomposition, following the orthogonal-subspace principle in O-LoRA Büyükakyüz (2024). For each module m, an orthonormal stable basis $B_m^{(t-1)} \in \mathbb{R}^{r_m \times d_m}$ spans the input-side directions already committed to long-term memory, while its orthogonal complement defines the plastic subspace for new adaptation. The basis construction and update rule are provided in Appendix I. Let $W_m \in \mathbb{R}^{d_m^{\text{out}} \times d_m}$ denote the effective weight matrix of module m. Since the rows of $B_m^{(t-1)}$ are orthonormal, we project the gradient onto the plastic subspace as

$$\nabla_{W_m}^{\text{pl}} \mathcal{L}_{\text{cp}} = \nabla_{W_m} \mathcal{L}_{\text{cp}} \left(I_{d_m} - {B_m^{(t-1)}}^\top B_m^{(t-1)} \right). \tag{11}$$

This removes update components along the stored stable directions, protecting consolidated memory while leaving the complementary directions available for new-task adaptation. The projection alone does not guarantee zero functional change on previous tasks; retention is further encouraged by replay-validated module selection and the selective consolidation objectives below.

Consolidation via Residual-Aware Distillation NeuralCP writes the accepted tags into the plastic subspace by matching task-induced residuals. We define the logit-level increments as

$$\Delta f_t^{\text{con}}(x) = f_t^{\text{con}}(x) - f_{\theta_{t-1}}(x), \qquad \Delta f_t^{\text{stm}}(x) = f_t^{\text{stm}}(x) - f_{\theta_{t-1}}(x), \tag{12}$$

and the hidden-state increments as

$$\Delta h_{m,t}^{\text{con}}(x) = h_m^{\text{con}}(x) - h_m^{\text{base}}(x), \qquad \Delta h_{m,t}^{\text{stm}}(x) = h_m^{\text{stm}}(x) - h_m^{\text{base}}(x). \tag{13}$$

At the logit level, the consolidation model is trained to reproduce the increment introduced by the short-term adapter:

$$\mathcal{L}_{\text{logit}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_t} \left[\left\| \Delta f_t^{\text{con}}(x) - \text{sg}\left[\Delta f_t^{\text{stm}}(x)\right] \right\|_2^2 \right], \tag{14}$$

where $\text{sg}[\cdot]$ denotes stop-gradient. Since $f_{\theta_{t-1}}$ is shared and frozen, this residual form is algebraically equivalent to matching the short-term teacher at the logit level; it mainly makes the incremental target explicit rather than introducing a fundamentally different teacher-matching loss. The selective behavior of NeuralCP is instead enforced by restricting optimization to $\mathcal{S}_t^{\text{cap}}$ and applying the stable–plastic projection. For hidden states, the consolidation loss is applied only to the replay-validated modules:

$$\mathcal{L}_{\text{hidden}} = \frac{1}{|\mathcal{S}_t^{\text{cap}}|} \sum_{m \in \mathcal{S}_t^{\text{cap}}} \mathbb{E}_{x\sim\mathcal{D}_t} \left[\left\| \Delta h_{m,t}^{\text{con}}(x) - \text{sg}\left[\Delta h_{m,t}^{\text{stm}}(x)\right] \right\|_2^2 \right]. \tag{15}$$

The final selective consolidation objective is $\mathcal{L}_{\text{cp}} = \mathcal{L}_{\text{logit}} + \mathcal{L}_{\text{hidden}}$. Together, replay-validated module selection, stable–plastic gradient projection, and selective hidden-state consolidation restrict the write-back operation to task-relevant directions that remain available for adaptation. After optimization, the newly consolidated residual directions are appended to the corresponding module-specific stable bases, yielding the updated backbone f_{θ_t}. The transient LoRA adapter is then discarded, so the final model introduces no additional inference parameters or latency.

4 EXPERIMENTS

Datasets We conducted experiments on nine datasets covering language and vision. For language, we used GLUE, ARC-E, GSM8K, BoolQ, and HellaSwag, covering general comprehension, commonsense reasoning, mathematical reasoning, and QA abilities. For visual learning, we evaluated using incremental category learning on the ImageNet-R. See Table 8 for more dataset details.

Baselines For language models, we compare NeuralCP with several representative continual learning baseline, including EWCLoRA Zheng et al. (2026), SDLoRA Wu et al. (2025), InfLoRA Liang & Li (2024), BiLoRA Qiang et al. (2024), O-LoRA Büyükakyüz (2024), and LoRA-Mixer Li et al. (2025). These baseline cover a variety of major design paradigms, such as regularization-based methods, orthogonality or subspace-based methods, and parameter isolation methods.

Metrics and Implementation Details For language tasks, we use the lm-evaluation-harness for evaluation, reporting exact-match accuracy for GSM8K under the 5-shot setting and standard accuracy for all other tasks. Unless otherwise specified, LoRA uses rank $r = 8$, $\alpha = 16$, dropout 0.05, learning rate 2×10^{-4}, and weight decay 0.05, with both LoRA adaptation and consolidation trained for one epoch; the consolidation learning rate is 2×10^{-5}. For vision experiments, we follow EWCLoRA Zheng et al. (2026) and use ViT-B/16 with LoRA rank 10, $\alpha = 20$, and 20 epochs for both stages. We keep the top 12 modules in functional attribution and set the tag rank to 8. More implementation details and training time are provided in Appendix B.

4.1 COMPARISON

We first evaluate NeuralCP on five sequential language tasks. As shown in Table 1, NeuralCP achieves the best average performance across four backbones, outperforming the strongest baselines by 6.60, 2.61, 3.24, and 4.31, respectively. The consistent gains over LoRA merging, regularization-based, and subspace-based methods demonstrate the effectiveness of selectively identifying, validating, and consolidating task-relevant changes. The improvement is more significant on smaller models, where catastrophic forgetting is more severe. For example, NeuralCP improves SST-2, MRPC, and QNLI on Qwen3-0.6B by 27.52, 36.31, and 31.21 points over the base model. Meanwhile, NeuralCP remains effective as model scale increases and achieves robust performance under different task orders (Appendix J), demonstrating its scalability and robustness.

To evaluate NeuralCP under more heterogeneous continual learning streams, we extend the Qwen3-8B setting with three diverse tasks: BoolQ, HellaSwag, and GSM8K, covering question answering, commonsense reasoning, and mathematical reasoning. As shown in Table 3, NeuralCP achieves the best average score of 86.46, outperforming the strongest baseline by 4.24 points. These results demonstrate that replay-validated consolidation remains effective beyond homogeneous task sequences and can preserve useful knowledge across diverse reasoning tasks.

Performance Evolution in the Task Sequence. To assess whether NeuralCP preserves prior knowledge, we evaluate all previously seen tasks after each new task. As shown in Figure 2a and Figure 2b, NeuralCP maintains strong retention throughout the task sequence. For example, ARC-E decreases only from 70.96 to 69.38 after all five tasks, while MRPC drops slightly from 79.01 to 78.45 after learning QNLI. These results show that NeuralCP can acquire new knowledge while protecting previously consolidated representations. The trajectories also reveal positive transfer rather than simple freezing. RTE improves from 68.59 to 73.65 after MRPC training, and SST-2 partially recovers after learning MRPC. This suggests that later tasks can reactivate and refine reusable functional pathways, consistent with replay validation and stable–plastic consolidation.

Comparison with LoRA-MoE Method. A central motivation of NeuralCP is to avoid the growing expert bank used by modular continual learning methods. We therefore compare NeuralCP with LoRA-Mixer Li et al. (2025), a representative LoRA-MoE approach. Using Qwen3-0.6B with five task-specific LoRAs (rank 8), LoRA-Mixer introduces 3.5% additional parameters. As shown in Table 2 and Table 12, NeuralCP achieves higher average performance with no additional parameters after consolidation, while reducing inference time from 319.81s to 261.07s and improving throughput from 33.49 to 41.03 samples/s. Additional results on larger backbones are provided in Appendix D.

Table 2: Comparison between LoRA-Mixer and NeuralCP in performance and deployment efficiency.

Method	Avg	Time/s	Samples/s
LoRA-Mixer	73.77	319.81	33.49
NeuralCP	**75.42**	**261.07**	**41.03**

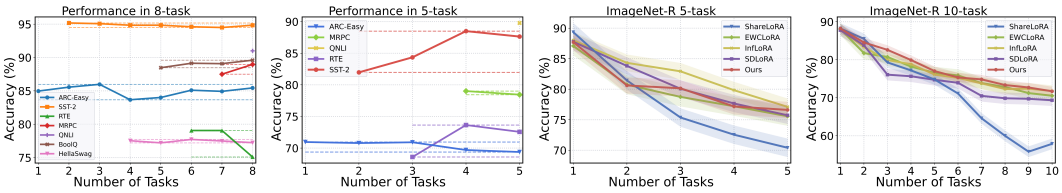
Table 1: The results on Qwen3-0.6B, Qwen3-4B, Qwen3-8B, and LLaMA3.2-1B demonstrate the effectiveness of NeuralCP across different architectures and scales.

Method	ARC-E	SST2	RTE	MRPC	QNLI	Avg
Qwen3-0.6B						
Base	60.69	57.11	53.43	42.65	49.62	52.70
Merged-LoRA	62.45	79.87	50.28	46.97	56.77	59.27
OLoRA	61.02	81.25	59.88	60.14	59.04	64.27
SDLoRA	64.23	78.59	60.86	60.75	63.44	65.57
EWCLoRA	62.02	82.73	59.48	68.14	63.04	67.88
BiLoRA	64.87	83.76	60.49	67.78	67.19	68.82
Ours	**67.34**	**84.63**	**65.34**	**78.96**	**80.83**	**75.42**
LLaMA3.2-1B						
Base	68.81	50.80	61.73	62.14	49.48	58.59
Merged-LoRA	66.45	84.22	61.85	67.42	79.88	71.96
OLoRA	66.43	86.39	66.37	70.56	84.28	74.81
SDLoRA	68.56	82.19	65.46	74.19	83.65	74.81
EWCLoRA	68.27	**90.02**	68.59	71.65	87.74	77.25
BiLoRA	68.68	88.45	67.41	72.85	82.96	76.07
Ours	**70.16**	88.53	**71.48**	**79.36**	**89.79**	**79.86**
Qwen3-4B						
Base	80.51	89.91	75.81	67.98	69.85	76.81
Merged-LoRA	80.16	86.98	76.77	69.14	74.81	77.57
OLoRA	79.81	90.77	75.54	74.97	82.47	80.71
SDLoRA	82.28	87.68	75.19	73.48	82.97	80.32
EWCLoRA	81.33	90.45	76.37	75.42	81.25	80.96
BiLoRA	77.46	88.45	74.83	75.22	80.89	79.37
Ours	**83.48**	**93.05**	**78.62**	**79.72**	**86.14**	**84.20**
Qwen3-8B						
Base	83.48	91.74	76.90	66.18	78.25	79.31
Merged-LoRA	82.57	90.71	74.88	71.62	82.19	80.39
OLoRA	81.38	89.38	78.68	73.54	80.57	80.71
SDLoRA	83.28	92.11	77.62	76.64	82.39	82.41
EWCLoRA	82.53	90.45	77.46	77.46	87.36	83.05
BiLoRA	82.96	91.48	76.77	75.85	83.69	82.15
Ours	**83.63**	**94.84**	**80.14**	**85.78**	**92.40**	**87.36**

Table 3: Performance comparison on the 8-task using Qwen3-8B. "Merged LoRA" merges LoRA weights from different tasks into the backbone.

Method	ARC-E	SST2	RTE	MRPC	QNLI	BoolQ	HellaSwag	GSM8K	Avg
Base	83.48	91.74	76.90	66.18	78.25	84.43	68.45	82.31	78.97
Merged-LoRA	79.32	91.78	70.40	76.22	83.54	87.37	72.68	75.89	79.65
OLoRA	80.28	90.45	75.32	71.68	85.52	85.28	70.37	78.62	79.69
SDLoRA	79.46	87.49	72.87	73.89	86.34	87.23	73.46	80.48	80.15
EWCLoRA	80.25	92.33	74.26	79.36	86.35	86.49	74.66	80.57	81.78
BiLoRA	80.77	91.18	75.88	81.51	89.40	85.98	73.75	79.31	82.22
Ours	**85.44**	**94.84**	**82.09**	**88.97**	**91.01**	**89.61**	**77.25**	**82.50**	**86.46**

Continual Learning on ImageNet-R. Finally, we test whether the proposed selective consolidation principle generalizes beyond language models. We conduct experiments on ImageNet-R under a standard class-incremental learning setting, where classes are introduced sequentially and the model must retain previously learned classes while adapting to new ones. We consider two protocols that divide the full class set into 5 or 10 tasks. For each incoming task, we train one LoRA module and its task-specific classifier head, keep previous heads frozen, copy the learned head into the consolidation model, and then apply NeuralCP to identify and update only the replay-validated task-relevant modules. Figure 2c and Figure 2d show the results under the 5-task and 10-task settings, respectively. In both cases, NeuralCP maintains stronger performance across the task sequence and exhibits slower degradation as more tasks are introduced. The advantage is especially clear in the 10-task setting,



(a) (b) (c) (d)

Figure 2: Performance of NeuralCP in language and vision continual learning. (a) Per-task accuracy evolution on Qwen3-8B under an 8-task setting. (b) Per-task accuracy evolution on LLaMA3.2-1B under a 5-task setting. (c) Comparison of Top-1 accuracy with baselines on 5-task ImageNet-R. (d) Comparison of Top-1 accuracy with baselines on 10-task ImageNet-R. ShareLoRA denotes the setting where all tasks share a single LoRA.

where longer task streams create stronger interference. These results suggest that NeuralCP captures a general selective-consolidation mechanism rather than an LLM-specific adaptation strategy.

4.2 Ablation

Ablation on key components. We ablate the key stages of NeuralCP to verify the contribution of the complete tag–validate–consolidate pipeline. Table 4 reports results on Qwen3-0.6B under the five-task. The full NeuralCP achieves the best average score of 75.42. Removing functional attribution causes the largest drop (67.11), demonstrating the importance of accurately identifying task-relevant changes. Removing memory tagging and replay validation also degrades performance (66.84 and 70.07), confirming that only validated task-induced changes should be consolidated. Finally, removing stable–plastic decomposition reduces the score to 67.80, highlighting its role in protecting previous knowledge during consolidation.

Table 4: Ablation on key components of NeuralCP using Qwen3-0.6B under the five-task.

Variant	ARC-E	SST2	RTE	MRPC	QNLI	Avg
w/o Functional Attribution	58.87	73.45	60.21	72.46	70.58	67.11
w/o Memory Tagging	60.15	76.23	56.88	69.75	71.19	66.84
w/o Replay Validation	63.28	79.88	59.37	73.55	74.28	70.07
w/o Stable–Plastic Decomposition	61.33	75.93	60.65	72.41	68.70	67.80
Full NeuralCP	**67.34**	**84.63**	**65.34**	**78.96**	**80.83**	**75.42**

Sensitivity to hyperparameters. We evaluate the sensitivity of NeuralCP to its main selection-related hyperparameters on ImageNet-R, including the number of selected modules Top-K, the capture threshold τ_{cap}, and the recall threshold τ_{recall}. As shown in Table 5, NeuralCP remains reasonably stable across different settings, while balanced intermediate values achieve the best performance. In particular, Top-$K = 12$, $\tau_{\text{cap}} = 0.50$, and $\tau_{\text{recall}} = 0.60$ obtain the highest final average accuracy. Smaller values tend to be overly conservative and may discard useful task-induced changes, while larger values may admit less reliable tags or expand the consolidation region too aggressively. These results demonstrate that NeuralCP is robust; more results can be found in the appendix A.

Table 5: Robustness of NeuralCP.

Top-K	Acc	τ_{cap}	Acc	τ_{recall}	Acc
6	68.47	0.30	69.24	0.40	69.76
12	**70.98**	0.50	**70.98**	0.60	**70.98**
24	69.32	0.70	70.11	0.80	70.42

5 Conclusion

We presented Neural Consolidation Process (NeuralCP), a continual learning framework that views sequential adaptation as selective memory formation. Instead of retaining all task updates, NeuralCP identifies and consolidates task-relevant memory tags into the backbone, while stable–plastic decomposition preserves previous knowledge. Experiments on language and vision benchmarks demonstrate that NeuralCP effectively mitigates catastrophic forgetting without additional parameters or inference overhead after consolidation. NeuralCP still has some limitations. Its multi-stage design introduces extra training costs compared with single-stage approaches, and it relies on small-scale calibration and replay data despite avoiding large historical storage. Reducing such dependence while maintaining selective consolidation remains an important direction for future work.

AI USE STATEMENT

In writing this paper, we used generative AI tools to help polish the text, check grammar, improve the readability of technical explanations, and provide feedback on the organization of figures and experimental demonstrations. Furthermore, we utilized AI tools to assist discussion, explore different ways of expression, and enhance the clarity of the article. However, the core scientific contributions of this paper, including the proposed framework, theoretical motivation, mathematical derivation, experimental methods, and the implementation of the proposed methods, were not generated using any generative AI tools. All experimental results, analyses, and conclusions were independently verified by the authors. The authors bear full responsibility for all content in this paper, including the methodology, arguments, experimental results, and the correctness of the final manuscript.

REFERENCES

Abhishek Aich. Elastic weight consolidation (ewc): Nuts and bolts, 2021. URL https://arxiv.org/abs/2105.04093.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget, 2018. URL https://arxiv.org/abs/1711.09601.

Kerim Büyükakyüz. Olora: Orthonormal low-rank adaptation of large language models, 2024. URL https://arxiv.org/abs/2406.01775.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition, 2024. URL https://arxiv.org/abs/2307.13269.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts, 2024. URL https://arxiv.org/abs/2404.15159.

Wenbing Li, Zikai Song, Hang Zhou, Yunyao Zhang, Junqing Yu, and Wei Yang. Lora-mixer: Coordinate modular lora experts through serial attention routing, 2025. URL https://arxiv.org/abs/2507.00029.

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning, 2024. URL https://arxiv.org/abs/2404.00228.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

Diego Moncada, Fabricio Ballarini, and Haydée Viola. Behavioral tagging: a translation of the synaptic tagging and capture hypothesis. *Neural plasticity*, 2015(1):650780, 2015.

Yu-Yang Qian, Yuan-Ze Xu, Zhen-Yu Zhang, Peng Zhao, and Zhi-Hua Zhou. Treelora: Efficient continual learning via layer-wise loras guided by a hierarchical gradient-similarity tree, 2025. URL https://arxiv.org/abs/2506.10355.

Rushi Qiang, Ruiyi Zhang, and Pengtao Xie. Bilora: A bi-level optimization framework for overfitting-resilient low-rank adaptation of large pre-trained models, 2024. URL https://arxiv.org/abs/2403.13037.

Haomiao Qiu, Miao Zhang, Ziyue Qiao, Weili Guan, Min Zhang, and Liqiang Nie. Splitlora: Balancing stability and plasticity in continual learning through gradient space splitting, 2025a. URL https://arxiv.org/abs/2505.22370.

Haomiao Qiu, Miao Zhang, Ziyue Qiao, and Liqiang Nie. Train with perturbation, infer after merging: A two-stage framework for continual learning, 2025b. URL https://arxiv.org/abs/2505.22389.

Roger L Redondo and Richard GM Morris. Making memories last: the synaptic tagging and capture hypothesis. *Nature reviews neuroscience*, 12(1):17–30, 2011.

Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2): 123–146, 1995.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pp. 4548–4557. PMLR, 2018.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11909–11919, 2023.

Gido M. van de Ven, Hava T. Siegelmann, and Andreas S. Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11(1):4069, 2020. doi: 10.1038/s41467-020-17866-2.

Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022. doi: 10.1038/s42256-022-00568-3.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019. URL https://arxiv.org/abs/1804.07461.

Szu-Han Wang, Roger L Redondo, and Richard GM Morris. Relevance of synaptic tagging and capture to the persistence of long-term potentiation and everyday spatial memory. *Proceedings of the National Academy of Sciences*, 107(45):19537–19542, 2010.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.715.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Dualprompt: Complementary prompting for rehearsal-free continual learning, 2022a. URL https://arxiv.org/abs/2204.04799.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning, 2022b. URL https://arxiv.org/abs/2112.08654.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts, 2024. URL https://arxiv.org/abs/2404.13628.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. Sd-lora: Scalable decoupled low-rank adaptation for class incremental learning, 2025. URL https://arxiv.org/abs/2501.13198.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Friedemann Zenke and Axel Laborieux. Theories of synaptic memory consolidation and intelligent plasticity for continual learning, 2024. URL https://arxiv.org/abs/2405.16922.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence, 2017a. URL https://arxiv.org/abs/1703.04200.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 2017b.

Yaoyue Zheng, Yin Zhang, Joost van de Weijer, Gido M van de Ven, Shaoyi Du, Xuetao Zhang, and Zhiqiang Tian. Revisiting weight regularization for low-rank continual learning, 2026. URL https://arxiv.org/abs/2602.17559.

A ADDITIONAL ABLATION RESULTS.

In this appendix, we provide additional ablation studies on key design choices of NeuralCP. Specifically, we analyze: (i) the SVD rank used in tag/subspace representation, (ii) the amount of data retained and replayed during replay-capture, (iii) the replay-capture gating hyperparameters, namely the capture threshold and the minimum recall-score threshold, and (iv) optimization-related settings including the consolidation learning rate and the LoRA rank. These results complement the ImageNet-R hyperparameter study in the main text by providing a more detailed analysis on the language side (Qwen3-0.6B). In all experiments, we keep all settings fixed except for the ablated factor and report the final average performance.

As shown in Table 6, NeuralCP is generally robust to a moderate range of hyperparameter choices, while still exhibiting clear preferences for several key settings. For the tag SVD rank, performance improves noticeably from $r_{\text{tag}} = 4$ to $r_{\text{tag}} = 16$, indicating that overly low-rank tags are insufficient to capture task-relevant functional shifts. Increasing the rank further to 24 does not provide additional gains, suggesting that a moderate rank already offers enough expressive capacity while avoiding redundancy. For replay-capture data, enlarging the retained and replayed set consistently improves performance, with the best result obtained at 384 samples, confirming the value of replay validation for filtering reliable tags before consolidation.

The two gating hyperparameters show a similar trade-off. For the capture threshold, overly permissive capture ($\tau_{\text{cap}} = 0.30$) leads to a clear drop in performance, indicating that indiscriminate consolidation harms long-term retention. Increasing the threshold improves performance and peaks at $\tau_{\text{cap}} = 0.50$, while a further increase becomes slightly too conservative. A similar pattern is observed for the recall threshold: low values admit unstable tags, whereas overly strict values begin to suppress useful memory candidates. Overall, the best performance is achieved around $\tau_{\text{cap}} = 0.50$ and $\tau_{\text{recall}} = 0.60$, which is consistent with the ImageNet-R results in the main text and supports our claim that NeuralCP is not overly fragile to threshold choices, while still benefiting from moderate selectivity.

We also include additional ablations on the consolidation learning rate and the LoRA rank. The results show that NeuralCP remains stable across a reasonable range of learning rates, with the best performance achieved at $2\text{e}{-5}$. Both smaller and larger values lead to mild degradation, suggesting that the consolidation stage benefits from a moderately conservative update scale. Similarly, the LoRA rank exhibits a non-monotonic trend: increasing the rank from 4 to 8 improves performance, but a further increase to 16 does not yield additional gains. This suggests that a moderate LoRA rank already provides sufficient adaptation capacity, while larger ranks may introduce unnecessary task-specific variation. Taken together, these results indicate that NeuralCP is not highly sensitive to optimization hyperparameters, provided that the consolidation process remains selective and sufficiently stable.

Table 6: Additional ablation studies of NeuralCP on Qwen3-0.6B. We report only the average accuracy (Avg).

Setting	Avg
A. Tag SVD rank (r_{tag})	
$r_{\text{tag}} = 4$	73.28
$r_{\text{tag}} = 8$	75.42
$r_{\text{tag}} = 16$	75.91
$r_{\text{tag}} = 24$	75.34
B. Replay-capture data	
128	75.21
256	75.42
384	75.88
C. Capture threshold (τ_{cap})	
$\tau_{\text{cap}} = 0.30$	73.19
$\tau_{\text{cap}} = 0.40$	74.25
$\tau_{\text{cap}} = 0.50$	75.42
$\tau_{\text{cap}} = 0.60$	74.77
D. Recall threshold (τ_{recall})	
$\tau_{\text{recall}} = 0.40$	73.56
$\tau_{\text{recall}} = 0.50$	74.95
$\tau_{\text{recall}} = 0.60$	75.42
$\tau_{\text{recall}} = 0.70$	75.31
E. Consolidation learning rate	
$1\text{e}{-5}$	74.68
$2\text{e}{-5}$	75.42
$5\text{e}{-5}$	74.91
F. LoRA rank	
$r = 4$	73.85
$r = 8$	75.42
$r = 16$	74.97

B IMPLEMENTATION DETAILS.

We report the training time of NeuralCP across different backbone models under the same experimental setting (five tasks: ARC-E, SST2, RTE, MRPC, and QNLI). All models use the same training configuration, with the only difference being the backbone size. As shown in Table 7, the total training time increases moderately with model scale, ranging from 3.5 hours on Qwen3-0.6B to 6.7 hours on Qwen3-8B. For all experiments, the LoRA training batch size was fixed at 16, while the consolidation batch size was fixed at 8. These experiments were all conducted on a single A800 GPU. We did not test faster training times, so the training times currently reported are not the fastest. In actual training, training time can be accelerated based on hardware resources.

Table 7: Training time of NeuralCP on different backbone models.

Model	Training Time
Qwen3-0.6B	3.5 h
LLaMA3.2-1B	4.1 h
Qwen3-4B	5.3 h
Qwen3-8B	6.7 h

We use LoRA ($r = 8$, $\alpha = 16$, dropout $= 0.05$), trained in bfloat16 on CUDA with FlashAttention and gradient checkpointing. Each stage is trained for 1 epoch: teacher training uses batch size 16, gradient accumulation 4, learning rate 2×10^{-4}, and weight decay 0.05; consolidation uses batch size 8 and learning rate 2×10^{-5}. We apply top-k module selection.

C DATASET DETAILS.

The following are details for all training datasets. For the class-incremental experiments on ImageNet-R, we consider two task partition settings. The first uses `init_cls=20`, `increment=20`, and `shuffle=false`, which divides the 200 classes into 10 sequential tasks with 20 classes per task. The second uses a 5-task setting with 40 classes per task. Tables 9 and 10 summarize the detailed sample statistics under the two protocols.

Under the 10-task setting, the task sizes are moderately imbalanced, with total samples per task ranging from 2,175 to 3,606. In total, this split contains 24,002 training samples and 5,998 test samples, yielding 30,000 samples overall. Under the 5-task setting, the data are grouped into coarser tasks, each covering 40 classes, with training samples ranging from 4,083 to 5,473 and test samples ranging from 1,024 to 1,364. The total number of samples remains unchanged. We report these statistics for transparency and reproducibility, as different task granularities may influence the difficulty of class-incremental learning and the observed forgetting behavior.

Table 8: Statistics of datasets used in experiments.

Dataset	**Train**	**Dev**	**Test**	**Task**
ARC-E	2251	570	2376	Multiple-choice QA
GSM8K	7473	–	1319	Math reasoning
SST-2	10000	872	1821	Sentiment classification
MRPC	3668	408	1725	Paraphrase detection
QNLI	104743	5463	5463	Question answering/NLI
RTE	2490	277	3000	Natural language inference
HellaSwag	39905	10042	10042	Commonsense reasoning
BoolQ	9427	3270	3270	Boolean question answering

D COMPARISONS OF INFERENCE TIME BETWEEN NEURALCP AND LORA-MIXER.

To fairly compare the inference efficiency of different methods, we statistically analyzed end-to-end inference time under the same hardware environment and evaluation settings. Specifically, we

Table 9: Task statistics of the ImageNet-R 10-task split under `init_cls=20`, `increment=20`, and `shuffle=false`.

Task	Class Range	Train	Test	Total
Task 1	0–19	2585	646	3231
Task 2	20–39	2888	718	3606
Task 3	40–59	1739	436	2175
Task 4	60–79	2344	589	2933
Task 5	80–99	2816	703	3519
Task 6	100–119	2215	555	2770
Task 7	120–139	2274	568	2842
Task 8	140–159	1821	456	2277
Task 9	160–179	2489	621	3110
Task 10	180–199	2831	706	3537
Total	**0–199**	**24002**	**5998**	**30000**

Table 10: Task statistics of the ImageNet-R 5-task split with 40 classes per task.

Task	Class Range	Train	Test	Total
Task 1	0–39	5473	1364	6837
Task 2	40–79	4083	1025	5108
Task 3	80–119	5031	1258	6289
Task 4	120–159	4095	1024	5119
Task 5	160–199	5320	1327	6647
Total	**0–199**	**24002**	**5998**	**30000**

evaluated the same set of tasks (SST2, ARC-E, MRPC, RTE, QNLI) using a uniform batch size (16) and recorded three metrics: total time, total number of evaluation samples, and average throughput. Average throughput was defined as the total number of samples divided by the total time, measuring how much data could be processed per second. All methods employed a consistent data processing and evaluation process to ensure comparability of results.

E Residual Parameterization and Selective Consolidation

NeuralCP represents the task-induced adaptation relative to the current backbone. Let $f_{\theta_{t-1}}$ denote the backbone before learning task t, and let $f_t^{\mathrm{stm}} = f_{\theta_{t-1},\phi_t^\star}$ denote the short-term adapted model. We define the task-induced logit increment as

$$\Delta_t(x) = f_t^{\mathrm{stm}}(x) - f_{\theta_{t-1}}(x). \tag{16}$$

The consolidation model is trained to reproduce this increment:

$$\mathcal{L}_{\mathrm{logit}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_t}\left[\left\|\left(f_t^{\mathrm{con}}(x) - f_{\theta_{t-1}}(x)\right) - \mathrm{sg}[\Delta_t(x)]\right\|_2^2\right]. \tag{17}$$

Because the reference backbone is shared and frozen, the residual terms cancel algebraically:

$$\mathcal{L}_{\mathrm{logit}} = \mathbb{E}_{(x,y)\sim\mathcal{D}_t}\left[\left\|f_t^{\mathrm{con}}(x) - f_t^{\mathrm{stm}}(x)\right\|_2^2\right]. \tag{18}$$

Thus, the residual formulation is equivalent to direct short-term teacher matching in terms of both objective value and student gradient. Its role is to make the incremental target explicit relative to the current backbone, rather than to introduce a fundamentally different distillation loss. The same equivalence holds for the hidden-state objective. For a selected module m,

$$\begin{aligned}&\left\|\left(h_m^{\mathrm{con}}(x) - h_m^{\mathrm{base}}(x)\right) - \mathrm{sg}\left[h_m^{\mathrm{stm}}(x) - h_m^{\mathrm{base}}(x)\right]\right\|_2^2 \\ &\quad = \left\|h_m^{\mathrm{con}}(x) - h_m^{\mathrm{stm}}(x)\right\|_2^2.\end{aligned} \tag{19}$$

Table 11: Comparison between LoRA-Mixer and NeuralCP on LLaMA3.2-1B.

Method	Avg	Time/s	Samples/s
LoRA-Mixer	78.25	281.01	38.12
NeuralCP	79.86	221.26	48.41

Table 12: Comparison between LoRA-Mixer and NeuralCP on Qwen3-4B.

Method	Avg	Time/s	Samples/s
LoRA-Mixer	83.77	398.51	26.88
NeuralCP	84.20	304.21	35.22

Consequently, the forgetting-control effect of NeuralCP does not arise from the residual parameterization alone. It is provided by the combination of replay-validated module selection and stable–plastic gradient projection. For each module m, let

$$P_m^{\mathrm{pl}} = I_{d_m} - B_m^{(t-1)\top} B_m^{(t-1)}, \tag{20}$$

where the rows of $B_m^{(t-1)}$ are orthonormal. The projected gradient is

$$g_m^{\mathrm{pl}} = g_m P_m^{\mathrm{pl}}. \tag{21}$$

Since $B_m^{(t-1)} B_m^{(t-1)\top} = I$, the projected update satisfies

$$g_m^{\mathrm{pl}} B_m^{(t-1)\top} = g_m \left(I - B_m^{(t-1)\top} B_m^{(t-1)} \right) B_m^{(t-1)\top} = 0. \tag{22}$$

Therefore, the update contains no component along the previously stored stable parameter directions. In addition, modules outside $\mathcal{S}_t^{\mathrm{cap}}$ receive no consolidation updates:

$$\delta W_m = 0, \qquad m \notin \mathcal{S}_t^{\mathrm{cap}}. \tag{23}$$

For replay-validated modules, the update is restricted to the corresponding plastic subspace.

To relate these constraints to forgetting, consider the old-task loss $\mathcal{L}_{\mathrm{old}}$ and let $\theta_t = \theta_{t-1} + \delta\theta$. If $\mathcal{L}_{\mathrm{old}}$ is locally L-smooth around θ_{t-1}, then

$$\mathcal{L}_{\mathrm{old}}(\theta_t) \le \mathcal{L}_{\mathrm{old}}(\theta_{t-1}) + \langle \nabla \mathcal{L}_{\mathrm{old}}(\theta_{t-1}), \delta\theta \rangle + \frac{L}{2} \|\delta\theta\|^2. \tag{24}$$

This bound shows that preserving previous tasks depends on the support, magnitude, and alignment of the consolidation update. Replay validation restricts consolidation to modules whose current task-induced responses are beneficial and consistent with stored memory tags, while stable–plastic projection removes update components along previously consolidated parameter directions. These mechanisms reduce the effective consolidation region and limit interference in parameter space.

Hence, the residual notation in NeuralCP provides an explicit representation of the task-induced increment, whereas replay validation and stable–plastic projection provide the actual mechanisms for selective write-back and forgetting control. Together, they enable the model to absorb the current task-induced adaptation while preserving previously consolidated parameter directions.

F Computation of the Module Score S_m

In this appendix, we provide the implementation form of the module score S_m and its conflict-aware version S_m^{conf} used in Section 3.1. For each candidate module m, we compute a hidden-shift score and a gradient-sensitivity score on a calibration set $\mathcal{D}_{\mathrm{cal}}$. Given an input batch $x \in \mathcal{D}_{\mathrm{cal}}$, let $h_{m,t}^{(B)}(x)$ and $h_{m,t}^{(T)}(x)$ denote the hidden activations at token position t from the base model and the LoRA teacher, respectively. Let $M_t \in \{0,1\}$ be the attention mask. The hidden-shift score of module m is

$$s_m^{\mathrm{shift}} = \frac{1}{|\mathcal{D}_{\mathrm{cal}}|} \sum_{x \in \mathcal{D}_{\mathrm{cal}}} \frac{\sum_t M_t \left\| h_{m,t}^{(T)}(x) - h_{m,t}^{(B)}(x) \right\|_2}{\sum_t M_t + \varepsilon}. \tag{25}$$

We also compute the gradient sensitivity of the teacher at module m:

$$s_m^{\text{grad}} = \frac{1}{|\mathcal{D}_{\text{cal}}|} \sum_{x \in \mathcal{D}_{\text{cal}}} \frac{\sum_t M_t \left\| \nabla_{h_{m,t}^{(T)}(x)} \mathcal{L}_{\text{task}}(x) \right\|_2}{\sum_t M_t + \varepsilon}. \tag{26}$$

Following the implementation, both scores are min–max normalized across candidate modules:

$$\tilde{s}_m^{\text{shift}} = \frac{s_m^{\text{shift}} - \min_j s_j^{\text{shift}}}{\max_j s_j^{\text{shift}} - \min_j s_j^{\text{shift}} + \varepsilon}, \qquad \tilde{s}_m^{\text{grad}} = \frac{s_m^{\text{grad}} - \min_j s_j^{\text{grad}}}{\max_j s_j^{\text{grad}} - \min_j s_j^{\text{grad}} + \varepsilon}. \tag{27}$$

The module score used for selection is then

$$S_m = \tilde{s}_m^{\text{shift}} \cdot \tilde{s}_m^{\text{grad}}. \tag{28}$$

If conflict-aware selection is enabled, the final score becomes

$$S_m^{\text{conf}} = S_m - \lambda_{\text{conf}} C_m, \qquad C_m = \min\left(1, \frac{r_m}{r_{\max}}\right), \tag{29}$$

where λ_{conf} is the conflict penalty coefficient, r_m is the current stable rank stored for module m, and $r_{\max}$ is the maximum stable rank allowed per module.

Here, $C_m \in [0, 1]$ measures the relative occupancy of the module's stable subspace bank. A larger C_m indicates that the module already contains more occupied stable directions and is therefore considered more prone to interference during further adaptation.

G Construction of Tag Residual Basis $U_{t,m}$

For each selected module $m \in \mathcal{S}_t$, we construct the low-rank basis $U_{t,m}$ from the batch-level residuals $\{u_b\}_{b=1}^B$ computed on the calibration set $\mathcal{C}_t$. Let $U = [u_1, u_2, \ldots, u_B]^\top \in \mathbb{R}^{B \times d}$ be the matrix of stacked residuals. We perform singular value decomposition (SVD):

$$U = V \Sigma W^\top, \tag{30}$$

and retain the top-r right singular vectors:

$$U_{t,m} = W_{:,:r} \in \mathbb{R}^{d \times r}, \tag{31}$$

where $r = 8$ in all experiments. This basis captures the dominant directions of variation in the teacher-induced residual changes at module m.

H Algorithm.

Here we present the overall algorithmic pipeline of NeuralCP.

I Update Rule of the Module-wise Memory Bank B_m

We denote by $B_m^{(t)}$ the stable memory bank of module m after finishing task t. In the implementation, each accepted memory tag for module m carries a residual basis $R_{m,k}^{(t)} \in \mathbb{R}^{r_k \times d}$, where k indexes the accepted tags produced at task t.

The memory bank is updated only after replay-and-capture validation. For each accepted tag, we merge its basis into the existing bank by row-wise concatenation, orthonormalization, and rank truncation:

$$\bar{B}_m^{(t)} = \text{Orth}\left(\begin{bmatrix} B_m^{(t-1)} \\ R_{m,k}^{(t)} \end{bmatrix}\right), \tag{32}$$

Algorithm 1 Neural Consolidation Process (NeuralCP)

Require: Sequential tasks $\{\mathcal{D}_1, \ldots, \mathcal{D}_T\}$, initial backbone f_{θ_0}, module budget/top-K

Initialize stable memory bank $\{B_m^{(0)}\}_m$
for $t = 1, \ldots, T$ **do**
  Short-term teacher adaptation:
  Initialize LoRA teacher $f_t^{\mathrm{stm}} = f_{\theta_{t-1},\phi_t}$
  Train ϕ_t on current task $\mathcal{D}_t$ to obtain $\phi_t^\star$
  Functional attribution:
  Build calibration set $\mathcal{C}_t \subset \mathcal{D}_t$
  for each candidate module m **do**
    Compute hidden-shift score Δ_m
    Compute gradient-sensitivity score G_m
    Compute module score S_m
    if stable memory is available **then**
      Apply conflict-aware correction to obtain S_m^{conf}
    end if
  end for
  Select task-relevant module set S_t under top-K or parameter budget
  Memory tagging and replay validation:
  for each selected module $m \in S_t$ **do**
    Extract teacher-induced residual tag
    Construct memory tag $z_{t,m} = (\mu_{t,m}, U_{t,m}, s_{t,m}^{\mathrm{shift}}, s_{t,m}^{\mathrm{gain}}, s_{t,m}^{\mathrm{recall}})$
  end for
  Store tags in memory buffer
  Replay high-value tags on held-out set $\mathcal{R}_t \subset \mathcal{D}_t$
  Accept only replay-validated tags $\hat{Z}_t$
  Stable–plastic consolidation:
  Initialize student $f_t^{\mathrm{stu}} \leftarrow f_{\theta_{t-1}}$
  for each selected module $m \in S_t$ **do**
    Project update direction into plastic subspace:

$$g_m^{\mathrm{pl}} = \big(I - (B_m^{(t-1)})^\top B_m^{(t-1)}\big) g_m$$

  end for
  Optimize student with

$$\mathcal{L}_{\mathrm{NeuralCP}} = \lambda_{\mathrm{task}} \mathcal{L}_{\mathrm{task}} + \lambda_{\mathrm{hidden}} \mathcal{L}_{\mathrm{hidden}}$$

  Append accepted residual bases to the stable bank and update backbone:

$$f_{\theta_t} \leftarrow f_t^{\mathrm{stu}}, \qquad \{B_m^{(t)}\}_m \leftarrow \{B_m^{(t-1)}\}_m \cup \hat{Z}_t$$

end for
return Final consolidated backbone f_{θ_T}

followed by

$$B_m^{(t)} = \mathrm{Trunc}_{r_{\max}}\left(\bar{B}_m^{(t)}\right). \tag{33}$$

If no previous bank exists for module m, we initialize it as

$$B_m^{(t)} = \mathrm{Trunc}_{r_{\max}}\left(\mathrm{Orth}\left(R_{m,k}^{(t)}\right)\right). \tag{34}$$

Here, $\mathrm{Orth}(\cdot)$ denotes row-space orthonormalization implemented with QR decomposition, and $\mathrm{Trunc}_{r_{\max}}(\cdot)$ keeps at most the first $r_{\max}$ basis vectors. Therefore, the bank update is an append–orthonormalize–compress procedure rather than a moving average.

When multiple accepted tags belong to the same module during task t, the above update is applied sequentially for each accepted basis. This means that B_m accumulates stable directions that have survived replay validation, while keeping its rank bounded by $r_{\max}$.

J RANDOMIZED SEQUENTIAL EXPERIMENTAL RESULTS.

To further evaluate the robustness of NeuralCP to task order, we conduct additional experiments on Qwen3-4B and Qwen3-8B using a shuffled task sequence: `SST2` → `ARC-E` → `MRPC` → `RTE` → `QNLI`. The results are reported in Table 14 and Table 13. We observe that NeuralCP continues to achieve strong performance under this altered order on both model scales, indicating that its effectiveness does not rely on a specific task sequence. This suggests that the proposed selective consolidation mechanism is relatively insensitive to task order and can robustly identify, validate, and preserve useful task-induced tags even when the order of incoming tasks changes.

Table 13: Evaluation results on Qwen3-8B.

Task	Version	Filter	n-shot	Metric	Value
ARC-E	1	none	0	acc	85.56
MRPC	1	none	0	acc	83.35
QNLI	1	none	0	acc	91.58
RTE	1	none	0	acc	78.81
SST2	1	none	0	acc	94.95

Table 14: Evaluation results on Qwen3-4B.

Task	Version	Filter	n-shot	Metric	Value
ARC-E	1	none	0	acc	82.66
MRPC	1	none	0	acc	77.45
QNLI	1	none	0	acc	86.33
RTE	1	none	0	acc	77.89
SST2	1	none	0	acc	92.59

K THE RESULTS OF IMAGENET-R.

Table 15 reports the 10-task class-incremental results on ImageNet-R. NeuralCP achieves the best overall average accuracy, outperforming all compared baselines with an Avg. of 78.03. It also attains the best final accuracy $\overline{A}_{10}$, slightly surpassing the strongest competing methods. These results suggest that the proposed selective consolidation mechanism is effective not only for language models but also for vision continual learning, where it can better preserve previously acquired knowledge while maintaining adaptation to later tasks.

Table 15: Results on ImageNet-R. Best results are in bold and second-best results are underlined.

Methods	$\overline{A}_{10}$ (↑)	Avg. (↑)
Joint Train	80.37	84.69
L2P	71.01	76.24
DualPrompt	68.17	73.69
InfLoRA	<u>71.70</u>	77.17
SD-LoRA	69.35	75.14
BiLoRA	74.28	<u>77.38</u>
Shared LoRA	57.88	71.45
EWC-LoRA	70.53	76.97
NeuralCP	**71.77**	**78.03**

L TRAINING LOSS VISUALIZATION.

The training process of NeuralCP is also highly stable. We conduct experiments on the GLUE benchmark using Qwen3-0.6B and LLaMA3.2-1B, and visualize the logit residual loss during consolidation. As shown in Figure 3 and Figure 4, the residual loss evolves smoothly and remains well-behaved throughout training, indicating that the proposed optimization process is stable. This stability is beneficial for continual learning, as it reduces the risk of destructive updates and supports more reliable consolidation of task-relevant knowledge into the backbone.

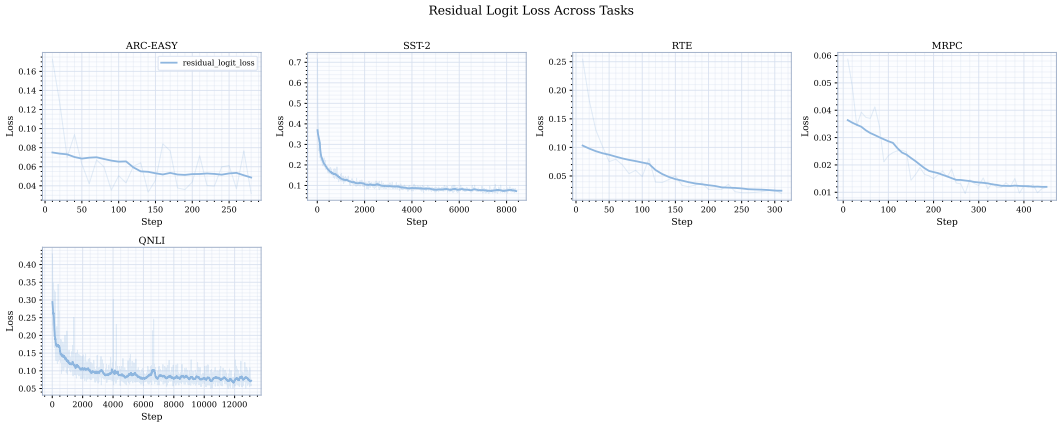


Figure 3: Logit residual loss during consolidation on the 5-task GLUE sequence using LLaMA3.2-1B.

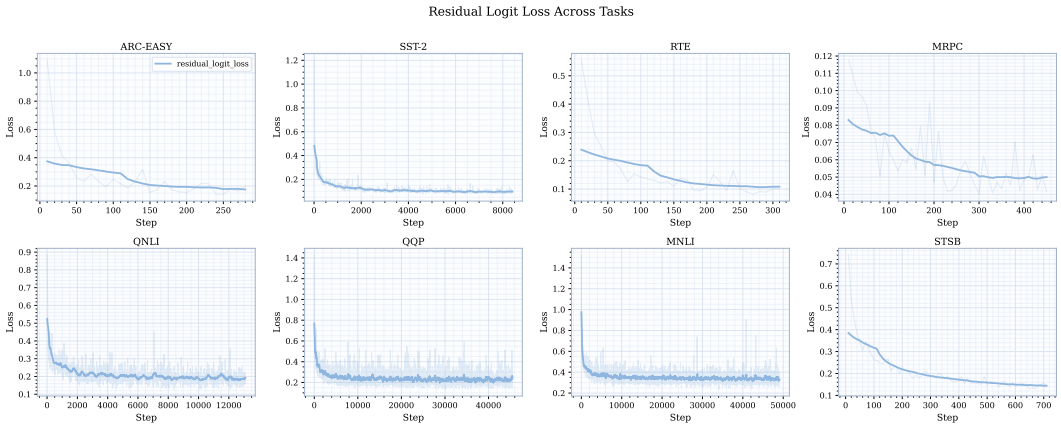


Figure 4: Logit residual loss during consolidation on the 8-task GLUE sequence using Qwen3-0.6B.

M OPTIMIZER CHOICE.

We next examine whether NeuralCP depends on a conservative optimizer to avoid forgetting. As shown in Table 16, AdamW consistently outperforms SGD, improving the average score from 69.32 to 75.42. The improvement is especially clear on RTE, MRPC, and QNLI, where AdamW brings gains of 5.41, 9.45, and 15.22 points, respectively. This suggests that stronger optimization is beneficial in NeuralCP because task-relevant updates can be better captured during consolidation. At the same time, the selective tagging, replay validation, and stable–plastic protection mechanisms help prevent these stronger updates from becoming destructive overwrites.

Table 16: Effect of optimizer choice for NeuralCP on Qwen3-0.6B.

Variant	ARC-E	SST2	RTE	MRPC	QNLI	Avg
AdamW	**67.34**	**84.63**	**65.34**	**78.96**	**80.83**	**75.42**
SGD	66.92	84.63	59.93	69.51	65.61	69.32