One LoRA for All: Unlocking Latent Capacity of LoRA for Continual Adaptation

Anonymous ACL submission

Abstract

Continual Learning (CL) enables Large Language Models (LLMs) to incrementally acquire new knowledge without costly retraining. However, existing CL methods exhibit a fundamental trade-off. Regularization-based approaches maintain a fixed parameter budget but often suffer from degraded performance over long task sequences. Expansion-based methods, in contrast, preserve performance by dynamically extending the model architecture. However, this comes at the cost of a growing memory footprint. To resolve this trade-off, we propose **C**ontinual **A**daptation via **S**ubspace **T**rimming (CAST), a parameter-efficient framework that systematically allocates and organizes task-specific subspaces within a single LoRA module, maintaining strong performance over long task sequences without increasing the parameter budget. CAST leverages the intrinsic redundancy of LoRA to allocate sparse, task-specific subspaces and utilizes a lightweight semantic routing mechanism to facilitate task-agnostic inference. Experiments on LLaMA 3.1-8b and Qwen 3-8b show that CAST outperforms related continual learning methods while maintaining a nearly *constant training memory footprint* and preserving over 25% idle capacity even after 15 sequential tasks.

1 Introduction

Large Language Models (LLMs) (Zhao et al., 2023) have demonstrated impressive understanding and reasoning capabilities across a wide range of tasks. However, once pretrained, their parameters remain largely fixed, making it difficult for them to incorporate newly emerging knowledge or adapt to domain shifts. Since training such models from scratch is computationally prohibitive, recent studies have increasingly explored Continual Learning (CL) as a practical paradigm for updating LLMs on a stream of new tasks and data (Wu et al., 2024). A central challenge, however, is catastrophic forgetting (Wu et al., 2022), where learning new data unintentionally disrupts previously acquired knowledge, leading to substantial performance degradation on earlier tasks.

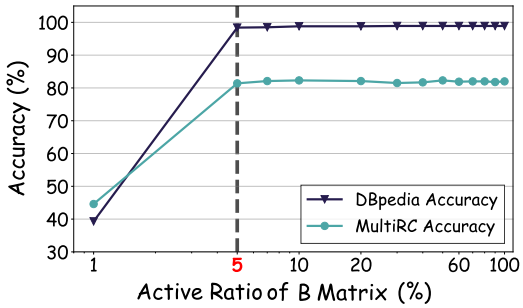


Figure 1: Effect of the active parameter ratio in $\mathbf{B}$ Matrix on downstream performance.

To mitigate catastrophic forgetting, prior studies have explored various continual learning strategies. *Replay-based methods* (de Masson d'Autume et al., 2019), while effective, often raise privacy concerns and introduce storage overheads. Alternatively, *regularization-based methods* (Zhang et al., 2025b) aim to preserve knowledge by imposing constraints on parameter updates. However, because all tasks are optimized within a shared parameter space, these methods struggle to maintain stable performance over long task sequences, as accumulated interference eventually leads to severe forgetting. Given these limitations, recent approaches have turned to architectural expansion to explicitly allocate task-specific parameters.

Expansion-based methods (Wang et al., 2023b; Razdaibiedina et al., 2023; Tong et al., 2025), particularly those built upon Parameter-Efficient Fine-Tuning (PEFT) (Houlsby et al., 2019; Pfeiffer et al., 2021; Li and Liang, 2021; Lester et al., 2021), achieve strong continual adaptation performance by assigning dedicated parameters to each task, thereby effectively eliminating interference. However, this strategy introduces a trade-off in scalability. Although Parameter-Efficient Fine-Tuning (PEFT) techniques, such as Low-Rank Adaptation (LoRA) (Hu et al., 2022), reduce the size of individ-

ual modules, many expansion-based methods rely on the frequent instantiation of additional adapters as new tasks arrive (Wang et al., 2022). This expansion leads to persistent increases in memory demands. Over long task sequences, this additive approach diminishes the core efficiency of PEFT.

Recent advances in the LoRA community reveal a crucial property: similar to common neural networks, LoRA adapters exhibit significant *parameter redundancy* (Zhang et al., 2025a; Zhu et al., 2024b; Zhang et al., 2023a). As illustrated in Figure 1, preserving a small active ratio of $\mathbf{B}$ Matrix is sufficient to recover the performance of the dense baseline, with significant degradation observed only at extreme sparsity. This observation naturally raises a question: *can such intrinsic redundancy be exploited to avoid the parameter accumulation inherent in expansion-based continual learning?*

In this work, we provide a positive answer to this question. We demonstrate that a single, fixed-size LoRA module possesses sufficient **latent capacity** to support multiple tasks, provided that task-specific knowledge is dynamically allocated within non-overlapping parameter subspaces. Building on this, we propose **C**ontinual **A**daptation via **S**ubspace **T**rimming (**CAST**), a parameter-efficient continual learning framework that preserves strong adaptation performance within a fixed LoRA parameter budget.

Empirically, we find that during adaptation, activating only 3%–10% of LoRA parameters is often sufficient to achieve performance comparable to full-parameter fine-tuning, revealing substantial latent capacity within a single adapter. Motivated by this sparsity, CAST incrementally allocates compact sets of task-relevant parameters while leaving weakly updated parameters available for future learning, enabling the continual reuse of underutilized capacity. As a result, CAST maintains stable performance over long task sequences without incurring any growth in parameters, memory, or inference cost. Extensive experiments demonstrate that CAST consistently matches the performance of independently trained task-specific models, providing strong empirical evidence that a single LoRA module can support robust continual learning when its latent capacity is properly managed.

Our main contributions are summarized as follows:

- Building upon the inherent parameter redundancy in LoRA adapters, we demonstrate how their significant latent capacity can be repurposed for continual learning without increasing model size.
- We propose CAST, a parameter-efficient continual learning framework that enables robust long-term adaptation within a single LoRA module by dynamically allocating and reusing underutilized capacity.
- Extensive experiments demonstrate that CAST matches the performance of independently trained task-specific models over long task sequences, while maintaining a constant training memory footprint.

2 Related Work

Continual learning for LLMs. CL for LLMs aims to acquire new knowledge without catastrophic forgetting. Existing approaches are generally categorized into replay-based, regularization-based, and expansion-based methods. *Replay-based methods*, such as Experience Replay (ER) (Rolnick et al., 2019), mitigate forgetting by revisiting buffered past data. However, these methods raise inherent concerns regarding data privacy and scalability, as storage overheads scale poorly with the number of tasks. *Regularization-based methods* like CLoRA (Smith et al., 2023), O-LoRA (Wang et al., 2023a), and DEAL (Han et al., 2025) preserve knowledge by imposing constraints on parameter updates to maintain a fixed parameter budget. However, as the sequence length increases, the accumulation of constraints often leads to a sharp decline in model plasticity, making it difficult to guarantee performance in complex, long-chain scenarios. *Expansion-based methods* (Wang et al., 2023b) like ProgPrompt (Razdaibiedina et al., 2023) and Any-SSR (Tong et al., 2025) adapt to new tasks by freezing historical parameters and dynamically expanding architectural capacity. While this strategy effectively alleviates catastrophic forgetting, it leads to a cumulative expansion of model parameters as the number of tasks accumulates.

Parameter-Efficient Fine-Tuning. In recent years, PEFT methods (Houlsby et al., 2019; Pfeiffer et al., 2021; Li and Liang, 2021; Lester et al., 2021) have attracted increasing research interest. Among these methods, LoRA (Hu et al., 2022) has become one of the most widely adopted approaches due to its strong performance and efficiency by

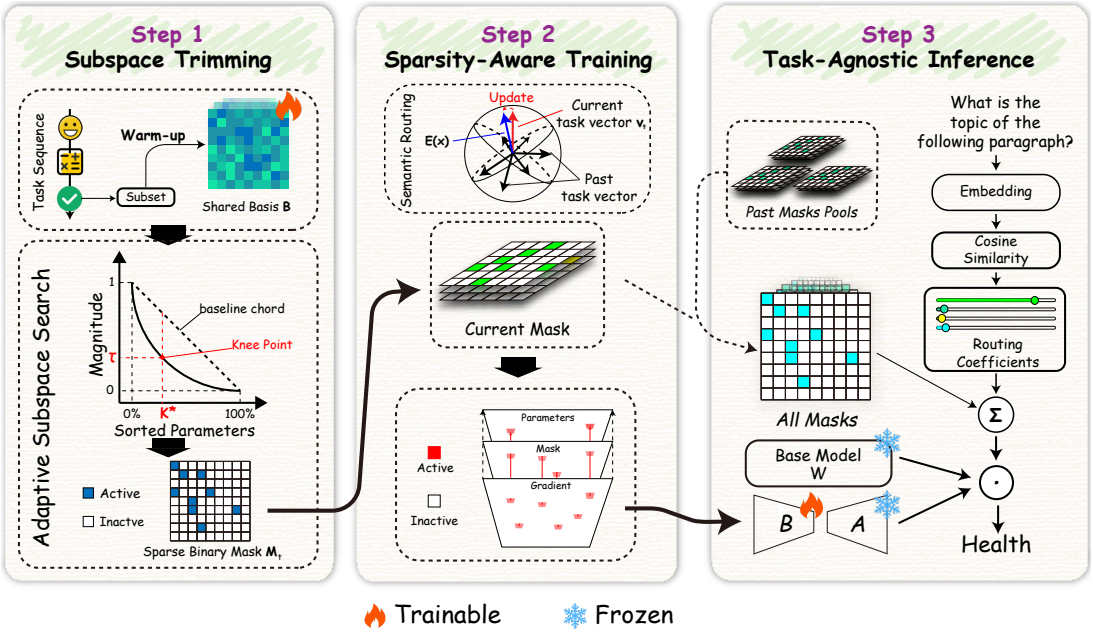


Figure 2: Overview of CAST. The pipeline proceeds in three stages. Step 1: Subspace Trimming. Task-specific sparse mask M_t is identified from the shared basis $\mathbf{B}$. Step 2: Sparsity-Aware Training. To prevent catastrophic forgetting, only the active subspace defined by M_t is updated, while past masks remain unchanged. Step 3: Task-Agnostic Inference. Input embeddings are matched with task vectors via cosine similarity to generate routing coefficients, which dynamically aggregate the relevant parameter subspaces for the final prediction.

introducing trainable low-rank matrices. To further explore more efficient fine-tuning paradigms, techniques such as Loraprune (Zhang et al., 2024) and LoRA-drop (Zhou et al., 2025) subsequently emerged. LoRI (Zhang et al., 2025a) reduces parameter redundancy through a unique asymmetric design that combines a frozen projection $\mathbf{A}$ matrix with a sparsely updated $\mathbf{B}$ matrix. Inspired by the efficiency of LoRI, CAST reduces inter-task interference by dynamically managing sparse subspaces through selective freezing and updating.

3 Preliminary

3.1 Problem formulation

We consider a CL setting in which a model learns from a sequence of T tasks, denoted as $\mathcal{S} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T\}$. Each task $\mathcal{T}_t$ arrives sequentially with its corresponding dataset $\mathcal{D}_t = \{(\mathbf{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^{N_t}$, where $\mathbf{x} \in \mathcal{X}$ denotes the input and $y \in \mathcal{Y}$ the target label.

We focus on the challenging *task-agnostic* continual learning setting. During training on task $\mathcal{T}_t$, the model has access only to $\mathcal{D}_t$ and cannot revisit full datasets from previous tasks $\mathcal{T}_{<t}$. At inference time, the task identity is unavailable; the model must therefore infer task-relevant knowledge implicitly from the input $\mathbf{x}$.

The objective is to learn a unified prediction function $f_\Theta : \mathcal{X} \to \mathcal{Y}$ parameterized by Θ, which minimizes the cumulative empirical risk over all observed tasks:

$$\min_{\Theta} \frac{1}{t} \sum_{k=1}^{t} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_k} \left[\mathcal{L}(f_\Theta(\mathbf{x}), y)\right], \quad (1)$$

where $\mathcal{L}(\cdot)$ denotes the task-specific loss. The central challenge is to acquire new task knowledge while preventing performance degradation on previously learned tasks.

3.2 Low-Rank Adaptation (LoRA)

LoRA (Hu et al., 2022) adapts large pre-trained models by injecting trainable low-rank matrices into frozen backbone weights. Given a weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA parameterizes the update as:

$$\mathbf{W} = \mathbf{W}_0 + s \cdot \mathbf{BA}, \quad (2)$$

where $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$ are trainable low-rank matrices, $r \ll \min(d_{\text{in}}, d_{\text{out}})$, and s is a scaling factor.

4 Method

4.1 Overview

CAST aims to enable long-sequence continual adaptation within a single fixed-size LoRA module. Instead of allocating new parameters for each

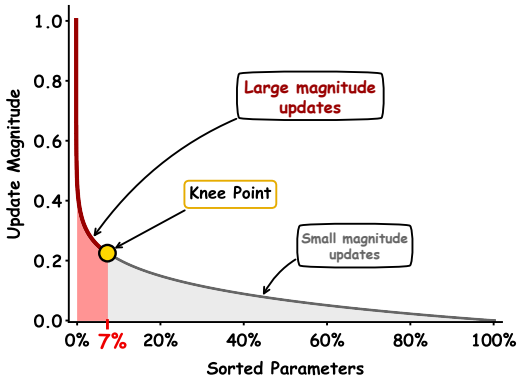


Figure 3: Sorted magnitudes of LoRA parameter updates during adaptation.

task, we treat the LoRA parameter space as a *finite shared resource* that must be progressively allocated based on task-specific effective parameter demand.

As illustrated in Figure 2, the CAST framework operates through three integrated stages: (1) **Subspace trimming**, which identifies a compact task-specific parameter subspace within a shared LoRA module; (2) **Sparsity-aware training**, which updates only the selected subspace to avoid interference with previously learned tasks; and (3) **Task-agnostic inference**, which utilizes a semantic routing mechanism to dynamically aggregate relevant subspaces based on input semantics.

4.2 Subspace trimming

This section describes how CAST performs subspace trimming, i.e., how it adaptively identifies a sparse, task-specific parameter subset within a shared LoRA module.

LoRA formulation. We freeze the pre-trained backbone parameters $\mathbf{W}_0$ and inject a single shared LoRA adapter parameterized by matrices $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$. Unlike expansion-based approaches, CAST maintains a fixed parameter budget and does not instantiate new adapters for incoming tasks.

Following prior research that the $\mathbf{A}$ matrix in LoRA tends to capture task-agnostic transformations, while task-specific variations are primarily reflected in $\mathbf{B}$ (Tian et al., 2024; Zhu et al., 2024a; Zhang et al., 2023b), we share $\mathbf{A}$ across all tasks to reduce training memory footprint and restrict subspace allocation to $\mathbf{B}$. Accordingly, CAST performs subspace trimming exclusively on $\mathbf{B}$.

For each task $\mathcal{T}_t$, CAST identifies a sparse, task-specific parameter subspace by learning a binary mask $\mathbf{M}_t \in \{0,1\}^{d_{\text{out}} \times r}$, which selectively activates a subset of parameters in the shared $\mathbf{B}$ matrix. The forward computation for task t is given by:

$$\mathbf{h}^{(t)} = \mathbf{W}_0\mathbf{x} + s \cdot (\mathbf{B} \odot \mathbf{M}_t)\mathbf{A}\mathbf{x}, \tag{3}$$

where $\odot$ denotes element-wise multiplication and s is the LoRA scaling factor.

Warm-up phase. To identify task-relevant parameters without pre-allocating a fixed sparsity budget, CAST initiates a brief warm-up phase upon the arrival of a new task $\mathcal{T}_t$. During this phase, we randomly sample a small subset of training data to probe the intrinsic parameter importance within $\mathbf{B}$.

All parameters in $\mathbf{B}$ that have not been assigned to previous tasks are temporarily activated and treated as candidates for the current task, while parameters associated with earlier tasks remain frozen. No sparsity constraint is imposed at this stage. This allows the model to adapt freely within the available parameter space and expose task-relevant parameters through natural gradient-driven updates.

Empirically, we observe that only a small fraction of parameters in $\mathbf{B}$ exhibits significant update magnitudes during the warm-up phase, indicating that task adaptation intrinsically relies on a sparse subset of the shared low-rank matrix. These update magnitudes provide a reliable signal for subsequent subspace allocation.

Adaptive subspace search. The objective of adaptive subspace search is to select a sparse subset of parameters from $\mathbf{B}$ that are essential for the current task while discarding redundant updates. We leverage the observation that the magnitude of parameter updates during the warm-up phase serves as an effective proxy for parameter importance.

This choice is motivated by the first-order Taylor expansion of the loss function. For a weight update $\Delta\mathbf{B}$, the corresponding loss reduction can be approximated as

$$\Delta\mathcal{L} \approx -\nabla_{\mathbf{B}}\mathcal{L}^{\top}\Delta\mathbf{B}. \tag{4}$$

Assuming that the update direction correlates with the gradient, parameters with larger update magnitudes $|\Delta\mathbf{B}|$ contribute more significantly to task-specific loss reduction.

After sorting the update magnitudes in descending order, we consistently observe a convex, heavy-tailed distribution. As shown in Figure 3, this structure reflects a sharp transition between a small set

of high-impact parameters and a long tail of negligible updates. Our goal is therefore to identify a principled cutoff that separates these two regions.

Direct curvature-based methods are unreliable due to the discrete and non-smooth nature of the sorted update distribution, where many parameters share identical magnitudes. To address this issue, we adopt a robust geometric knee-point detection strategy. We first normalize the sorted curve to the unit square coordinate system. Let N be the total number of parameters and $\tilde{s}_i$ be the min-max normalized magnitude of the i-th parameter. We define the coordinate point for each parameter as $(x_i, y_i) = (i/N, \tilde{s}_i)$. The knee point index k^* is then determined by maximizing the perpendicular distance to the reference chord defined by $x + y - 1 = 0$:

$$k^* = \underset{i}{\operatorname{argmax}} \left(x_i + y_i - 1\right) \tag{5}$$

The parameter magnitude at this optimal index k^* is set as the truncation threshold τ. We then construct the binary mask $\mathbf{M}_t$ where each element $m_{t,j}$ is determined by retaining parameters whose update magnitudes exceed this threshold:

$$m_{t,j} = \begin{cases} 1, & \text{if } |\Delta w_j| > \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Through this subspace trimming process, CAST preserves parameters that are critical for the current task while resetting low-impact parameters to zero. This strategy not only solidifies task-specific knowledge but also continuously releases unused capacity, ensuring that the shared LoRA module remains flexible and sustainable as new tasks arrive.

4.3 Sparsity-aware training

To preserve previously learned knowledge during adaptation to the current task $\mathcal{T}_t$, CAST restricts parameter updates to the task-specific subspace defined by the binary mask $\mathbf{M}_t$. By limiting updates to task-relevant components, this procedure enables adaptation while reducing inter-task interference.

Updates to the shared $\mathbf{B}$ matrix are performed through masked backpropagation:

$$\mathbf{B} \leftarrow \mathbf{B} - \eta \cdot \left(\nabla_{\mathbf{B}} \mathcal{L} \odot \mathbf{M}_t\right), \tag{7}$$

where $\odot$ denotes the Hadamard product. This update rule ensures that parameters assigned to previous tasks remain unchanged.

To enable task identification without explicit identifiers, we assign a learnable task vector $\mathbf{v}_t$ to each task as a reference point in the embedding space. The routing is optimized by aligning the input embedding $E(\mathbf{x})$ with $\mathbf{v}_t$ and separating it from historical task vectors using a margin-based objective. Specifically, we calculate the positive alignment:

$$s_{\text{pos}} = \langle E(\mathbf{x}), \mathbf{v}_t \rangle \tag{8}$$

We then identify the hardest negative sample from the set of frozen past vectors $\mathcal{V}\text{past}$:

$$s_{\text{neg}} = \max_{\mathbf{v}_j \in \mathcal{V}_{\text{past}}} \langle E(\mathbf{x}), \mathbf{v}_j \rangle \tag{9}$$

The routing objective uses a smooth ranking loss with a margin m:

$$\mathcal{L}_{\text{route}} = \text{Softplus}(s_{\text{neg}} - s_{\text{pos}} + m) \tag{10}$$

For the initial task where $\mathcal{V}_{\text{past}} = \emptyset$, the loss simplifies to $1 - s_{\text{pos}}$. All task vectors are constrained to unit norm to focus the optimization on directional alignment.

4.4 Task-agnostic inference

Task-agnostic inference reconstructs task-specific parameter configurations for test inputs with unknown task identities. This process utilizes the task vectors $\mathbf{v}_k$ optimized during Section 4.3 to guide mask aggregation.

For a test instance $\mathbf{x}$, we calculate routing coefficients $\boldsymbol{\alpha}$ by applying a temperature-scaled Softmax function to the cosine similarity between the input embedding $E(\mathbf{x})$ and the frozen task vectors $\mathcal{V}$:

$$\alpha_k = \text{softmax}_k \left(\frac{1}{\tau} \langle E(\mathbf{x}), \mathbf{v}_k \rangle \right). \tag{11}$$

These coefficients weight the historical masks to produce an instance-specific soft mask, defined as:

$$\mathbf{M}_{\text{agg}} = \sum_{k=1}^{t} \alpha_k \mathbf{M}_k. \tag{12}$$

The final forward pass incorporates this aggregated mask into the parameter decomposition:

$$\mathbf{h} = \mathbf{W}_0 \mathbf{x} + s \cdot \left[(\mathbf{B} \odot \mathbf{M}_{\text{agg}}) \mathbf{A} \right] \mathbf{x} \tag{13}$$

This mechanism enables dynamic task selection during inference and minimizes inter-task interference.

Table 1: Average accuracy of different methods on LLaMA 3.1-8B. Methods are grouped into reference methods and continual learning methods. For continual learning methods, the best performance is highlighted in **bold** and the second-best is underlined.

Method	Standard CL Benchmark				Long Task Sequence			
	Order 1	Order 2	Order 3	Average	Order 4	Order 5	Order 6	Average
Zero-shot	13.6	13.6	13.6	13.6	32.1	32.1	32.1	32.1
Replay	**81.2**	**81.5**	78.5	80.4	77.3	80.5	79.4	79.1
ProgPrompt	80.6	79.9	80.9	80.5	79.4	79.3	**80.1**	79.6
Vanilla LoRA	61.6	56.4	58.3	58.8	61.3	63.7	60.4	61.8
EWC	72.3	68.5	70.4	70.4	59.4	61.8	65.3	62.2
O-LoRA	78.5	75.3	74.7	76.2	72.9	73.2	70.4	72.2
DEAL	79.8	78.9	79.1	79.3	73.8	74.6	75.4	74.6
CAST (Ours)	81.0	81.0	**81.5**	**81.1**	**80.5**	**80.8**	79.3	**80.2**

Table 2: Average accuracy of different methods on Qwen 3-8B. Methods are grouped into reference methods and continual learning methods. For continual learning methods, the best performance is highlighted in **bold** and the second-best is underlined.

Method	Standard CL Benchmark				Long Task Sequence			
	Order 1	Order 2	Order 3	Average	Order 4	Order 5	Order 6	Average
Zero-shot	55.6	55.6	55.6	55.6	70.9	70.9	70.9	70.9
Replay	81.0	**81.4**	80.5	**81.0**	78.8	80.1	79.4	79.4
ProgPrompt	80.2	80.8	80.7	80.6	81.1	81.5	81.4	81.3
Vanilla LoRA	66.3	64.4	66.4	65.7	71.2	72.5	70.3	71.3
EWC	71.5	70.8	72.5	71.6	69.8	71.5	71.3	70.9
O-LoRA	78.4	76.4	77.8	77.5	75.4	74.8	73.4	74.5
DEAL	**81.3**	79.9	80.5	80.6	79.8	78.5	79.4	79.2
CAST (Ours)	80.5	80.3	**81.0**	80.6	**82.5**	**83.2**	**81.7**	**82.5**

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate CAST under both standard and long-sequence continual learning settings.

Standard Continual Learning Benchmark. We adopt a widely used continual learning benchmark for language models composed of five text classification datasets (Zhang et al., 2015). Specifically, AG News and Yahoo Answers focus on topic classification, Amazon Reviews (McAuley and Leskovec, 2013) and Yelp Reviews address sentiment analysis, and DBpedia (Lehmann et al., 2015) evaluates ontology-based document classification. Together, these datasets span diverse linguistic domains and task semantics. Following common practice, we evaluate under three different task orders to examine robustness to task sequence variations. Detailed dataset statistics and task orders are provided in Appendix E.2, E.3, and E.4.

Long Task Sequence. To assess performance under longer task sequences, we further conduct experiments on a continual learning benchmark consisting of fifteen datasets (Razdaibiedina et al., 2023). This benchmark integrates tasks from multiple sources, including the standard continual learning benchmark, the GLUE benchmark (Wang et al., 2018), the SuperGLUE benchmark (Wang et al., 2019), and the IMDB movie review dataset (Maas et al., 2011). The resulting task sequence covers a broad range of classification and reasoning tasks, and poses a substantially more challenging long-sequence continual learning scenario.

Evaluation metrics. We use **Average Accuracy (AA)** as the primary evaluation metric. Specifically, given a task sequence $\mathcal{S} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_T\}$, let $A_{i,k}$ denote the accuracy on task $\mathcal{T}_i$ after learning the k-th task. The average accuracy after learning

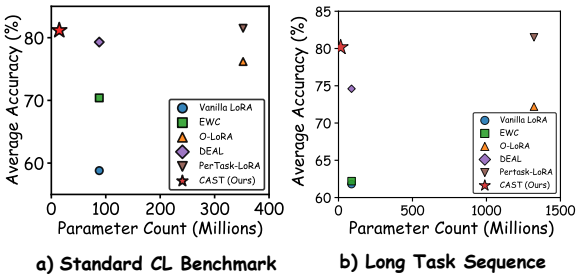


Figure 4: Comparison of various methods based on average accuracy and parameter count.

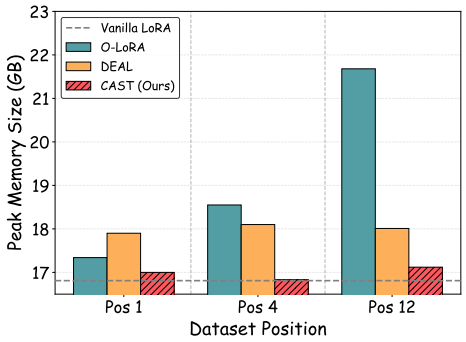


Figure 5: Comparison of Peak Training Memory Footprint (GB) across different methods evaluated at distinct sequence positions.

task $\mathcal{T}_k$ is computed as:

$$AA = \frac{1}{k}\sum_{i=1}^{k} A_{i,k}. \tag{14}$$

Implementation details. We conduct experiments on two large language models commonly adopted in prior continual learning studies: **LLaMA 3.1-8B** (Grattafiori et al., 2024) and **Qwen 3-8B** (Yang et al., 2025). Results across both backbones exhibit consistent trends, demonstrating the robustness and generalizability of CAST. More details are described in Appendix E.5.

5.2 Comparison with related methods

We compare CAST with a set of representative methods under a unified experimental protocol. The compared approaches are grouped into continual learning methods and reference methods according to their underlying assumptions.

The continual learning methods include **Vanilla LoRA**, **EWC** (Kirkpatrick et al., 2017), **O-LoRA** (Wang et al., 2023a), and **DEAL** (Han et al., 2025).

EWC, **O-LoRA**, and **DEAL** represent recent continual learning approaches that aim to improve performance during sequential adaptation. In addition, we report several reference methods that operate under different or stronger assumptions. **Zero-shot** evaluates the base model without task-specific fine-tuning. **Replay** incorporates an **additional** memory buffer containing samples from previous tasks. **ProgPrompt** (Razdaibiedina et al., 2023) relies on **additional** task identity at inference by assigning a dedicated prompt to each task. **PerTask-LoRA** trains dedicated LoRA modules for **each** task.

Table 1 reports the continual learning performance of all compared methods on the two benchmark suites. Following prior work, we report the AA over three independent runs with different task orders.

We first compare all methods against the Zero-shot baseline. Without any continual fine-tuning, the base model achieves only 13.6% average accuracy on the Standard CL benchmark and 32.1% on the long task sequence. This substantial performance gap clearly demonstrates that continual adaptation is essential for effective knowledge acquisition across sequential tasks.

Among all continual learning methods, CAST consistently achieves the strongest overall performance. On the Standard CL benchmark, CAST reaches an average accuracy of 81.1%. On the long task sequence, CAST maintains a high accuracy of 80.2%. In contrast, the vanilla sequential baseline exhibits severe forgetting and achieves only 58.8% and 61.8% average accuracy on the two benchmarks, respectively.

We further compare CAST with representative regularization-based and subspace-constrained methods. O-LoRA and DEAL demonstrate competitive anti-forgetting capability on the shorter task sequence, achieving 76.2% and 79.3% average accuracy on the Standard CL benchmark. However, their performance degrades noticeably when the task sequence is extended. This trend suggests that fixed or globally constrained subspaces become insufficient to accommodate a growing number of tasks. In contrast, CAST exhibits strong stability across both benchmarks, indicating that it can sustain reliable performance even under a long task sequence.

Finally, we compare CAST with *Pertask-LoRA*, which trains an independent LoRA adapter for each task and eliminates inter-task interference through parameter separation. Pertask-LoRA achieves 81.5% average accuracy on the Standard CL benchmark and 82.7% on the long task sequence. CAST achieves performance that is highly comparable

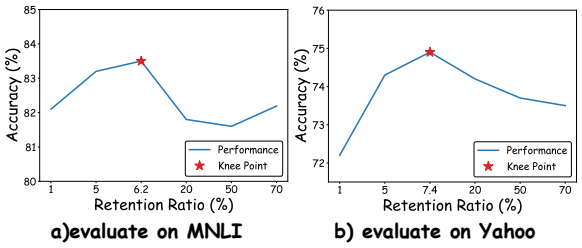


Figure 6: Comparison of accuracy across different retention ratios for two datasets.

to Pertask-LoRA while relying on a single shared LoRA module. This result indicates that CAST can effectively preserve task performance while avoiding the need to maintain separate task-specific adapters.

To evaluate the generality of these findings, we conduct the same experiments using the Qwen 3-8B backbone. As reported in Table 2, CAST exhibits performance trends consistent with those observed on LLaMA 3.1-8b. Across both standard and long task sequences, CAST remains the most reliable method among all compared approaches, confirming that its effectiveness is not specific to a particular model architecture.

Figure 4 compares the average accuracy and parameter usage of different methods on LLaMA 3.1-8B. PerTask-LoRA serves as the ideal AA upper bound, where a dedicated LoRA is trained independently for each task.

While PerTask-LoRA achieves high accuracy, its parameter count grows linearly with the task sequence, leading to prohibitive storage costs in the Long Task Sequence setting. In contrast, CAST maintains a compact and constant parameter footprint. Remarkably, CAST achieves accuracy comparable to the ideal PerTask-LoRA baseline, demonstrating that our method effectively accumulates knowledge and retains performance without the need for the extensive parameter expansion required by the ideal approach.

Furthermore, our analysis in Appendix D confirms that the parameter space remains far from saturation even in long sequences. Additionally, by observing that the allocated parameter usage for identical tasks varies adaptively across different task orders, which verify the dynamic nature of our allocation strategy.

5.3 Comparison of training memory footprint

We evaluate the peak GPU memory consumption on NVIDIA V100 GPUs using FP16 precision across sequential tasks. Figure 5 shows the training memory footprint of different methods as the number of tasks increases.

The results show that CAST maintains a nearly constant training memory footprint throughout the entire task sequence, remaining close to that of the vanilla LoRA baseline. This behavior indicates that CAST does not incur additional memory overhead as new tasks arrive.

In contrast, O-LoRA exhibits a clear increase in memory consumption, which becomes pronounced as more tasks are learned. This growth is caused by the accumulation of task-specific states, making the method increasingly impractical for long task sequences. DEAL shows a more stable memory profile but consistently requires higher memory usage due to auxiliary components introduced during training.

Overall, these results demonstrate that CAST preserves the memory efficiency of a fixed-parameter baseline while supporting continual adaptation over long task sequence.

Due to space constraints, we provide additional qualitative case studies and a detailed hyperparameter sensitivity analysis in Appendix A and Appendix B, respectively.

5.4 Effectiveness of Knee point detection

To validate the effectiveness of the knee points identified by our detection method, we conducted experiments using Llama 3.1-8b on the MNLI and Yahoo datasets as illustrated in Figure 6. The results demonstrate that the knee points detected by our approach correspond to highly effective retention ratios for the model. These specific points are highlighted by red stars in the charts. This approach effectively avoids the need for an exhaustive search across all possible ratios.

Additionally, we provide supplementary visualizations of the knee point detection process in Appendix C for further reference.

6 Conclusion

We reveal that a single LoRA module possesses intrinsic parameter redundancy suitable for reuse. Building on this, we introduce CAST to explicitly partition task-specific representations within a constrained budget. CAST delivers stable plasticity without the need for task identifiers. Extensive experiments demonstrate its competitive performance on benchmarks, alongside strong efficiency in GPU memory consumption.

7 Limitations.

Despite its effectiveness, our study has several limitations. First, due to constraints in computational resources, we are currently unable to conduct comprehensive testing on larger-scale models, such as 70B+ variants, and our results are primarily focused on the 8B-scale. Second, also due to computational constraints, the reported results are based on a single experimental run with a fixed random seed, offering only an initial observation of the method's stability.

References

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. *Preprint*, arXiv:1906.01076.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Xiao Han, Zimo Zhao, Wanyu Wang, Maolin Wang, Zitao Liu, Yi Chang, and Xiangyu Zhao. 2025. Data efficient adaptation in large language models via continuous low-rank fine-tuning. *arXiv preprint arXiv:2509.18942*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and 1 others. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: main volume*, pages 487–503.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. *arXiv preprint arXiv:2301.12314*.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in neural information processing systems*, 32.

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

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. 2024. Hydralora: An asymmetric lora architecture for efficient fine-tuning. In *Advances in Neural Information Processing Systems*, volume 37, pages 9565–9584. Curran Associates, Inc.

Kai Tong, Kang Pan, Xiao Zhang, Erli Meng, Run He, Yawen Cui, Nuoyan Guo, and Huiping Zhuang. 2025. Any-ssr: How recursive least squares works in continual learning of large language model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3047–3057.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of*

the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP, pages 353–355.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. 2023a. Orthogonal subspace learning for language model continual learning. *Preprint*, arXiv:2310.14152.

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, and 1 others. 2023b. Rehearsal-free continual language learning via efficient parameter isolation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10933–10946.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022. Learning to prompt for continual learning. *Preprint*, arXiv:2112.08654.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, Guilin Qi, and Gholamreza Haffari. 2022. Pretrained language model in continual learning: A comparative study. In *International Conference on Learning Representations*.

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. 2024. Continual learning for large language models: A survey. *Preprint*, arXiv:2402.01364.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. 2025a. Lori: Reducing cross-task interference in multi-task low-rank adaptation. *Preprint*, arXiv:2504.07448.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023a. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *Preprint*, arXiv:2308.03303.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023b. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2024. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3013–3026.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Zhixin Zhang, Zeming Wei, and Meng Sun. 2025b. Dynamic orthogonal continual fine-tuning for mitigating catastrophic forgettings. *Preprint*, arXiv:2509.23893.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, Tiejun Zhao, and Muyun Yang. 2025. Lora-drop: Efficient lora parameter pruning based on output evaluation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5530–5543.

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez De Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. 2024a. Asymmetry in low-rank adapters of foundation models. *arXiv preprint arXiv:2402.16842*.

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. 2024b. Asymmetry in low-rank adapters of foundation models. *Preprint*, arXiv:2402.16842.

A Case Study

To intuitively understand the effectiveness of our proposed method, we present a qualitative case study in Table 3, comparing our approach with Vanilla LoRA across stability verification.

In Case 1, we revisit a sample from the Yahoo dataset after the model has finished training on the subsequent DBpedia task. Vanilla LoRA fails to recall the correct label, incorrectly predicting "Business or Finance," which indicates catastrophic forgetting. In contrast, our method successfully preserves the old knowledge. Crucially, the router output for the revisited task is $[0.994, 0.001, \dots]$, demonstrating that our mechanism correctly identifies the task context and reactivates the corresponding task-specific experts with high confidence, shielding the parameters from interference.

In Case 2, we examine the model's adaptability to the new SST-2 task. The non-sparse router distribution ($[\dots, 0.177, 0.177, 0.303]$) reveals a key advantage: our method utilizes task vectors to reasonably transfer relevant knowledge from past tasks to

Case Study

Case 1: Yahoo Classification Task *Goal: Verify Stability of old knowledge.*

Instruction: What is the topic of the following paragraph? Choose one from the options: Society & Culture, Science & Mathematics, ..., Politics & Government.
Input: Question: what are the address of Citizen and Immigration Canada prcesing cantres? Answer: Go to this website to find the addresses: http://www.cic.gc.ca/
True Answer: Politics & Government

Vanilla LoRA:
- **Adapter after Yahoo:** Politics & Government
- **Adapter after Yahoo → ... → DBpedia:** Business or Finance

Ours:
- **Adapter after Yahoo:** Politics & Government (Router: $[1.0]$)
- **Adapter after Yahoo → ... → DBpedia:** Politics & Government (Router: $[0.994, 0.001, 0.004, 0.001]$)

Case 2: SST-2 Sentiment Analysis *Goal: Verify plasticity on new tasks.*

Instruction: What is the sentiment of the following paragraph? Choose one from the option. Options: Good, Bad.
Input: Whether you like rap music or loathe it , you can't deny either the tragic loss of two young men in the prime of their talent or the power of this movie.
True Answer: Good.

Ours:
- **Adapter after MNLI → ... → SST-2:** Good.
- **Router:** [0.001, 0.024, 0.014, 0.005, 0.001, 0.107, 0.014, 0.177, 0.177, 0.177, 0.303]

Table 3: **Qualitative Case Study.** We compare the prediction results of Vanilla LoRA and our method. **Case 1** (Yahoo) demonstrates that our method effectively mitigates catastrophic forgetting by reactivating the correct expert (Router ≈ 1.0) after learning subsequent tasks (up to DBpedia), whereas Vanilla LoRA fails. **Case 2** (SST-2) shows our model's plasticity in accurately handling new tasks with dynamic router allocation.

facilitate current inference. Crucially, while leveraging historical information, the mechanism ensures the primacy of the current task's learning. This demonstrates that our approach maintains high plasticity, effectively synthesizing past knowledge to solve new problems without suffering from intransigence.

B Hyperparameter Sensitivity Analysis

Effect of LoRA rank. We study the sensitivity of CAST to the LoRA rank r, which controls the number of trainable parameters. Table 7 reports the accuracy of LLaMA 3.1-8B under different task orders with $r \in \{8, 16, 32\}$.

Increasing the rank improves average accuracy from 80.2% ($r = 8$) to 80.9% ($r = 16$), while further increasing r to 32 yields a smaller gain (81.1%), indicating diminishing returns. Notably, CAST achieves consistently strong performance even with a small rank, demonstrating robustness to the choice of r and high parameter efficiency.

C Visualization of the Adaptive Subspace Search.

In this section, we provide a visual demonstration of how the adaptive trimming threshold is determined. As discussed in the Section 4.2, we utilize the geometric knee-point detection strategy to dynamically identify the knee point index.

Figure 7 illustrates the knee-point detection strategy process across standard CL benchmarks. These visualizations are based on the LLaMA 3.1-8B model evaluated under Order 2. In each Figure, the blue curve depicts the sorted parameter importance scores, while the gray dashed line serves as the reference chord. The knee point is mathematically identified as the point with the maximum perpendicular distance to this baseline, representing the optimal cutoff for parameter retention.

D Subspace Sparsity and Capacity

We analyze the parameter utilization of the shared LoRA module on the long task sequence using LLaMA 3.1–8B. A key question in fixed-capacity continual learning is whether the parameter space becomes saturated as tasks accumulate.

Table 4: Overview of datasets used in experiments.

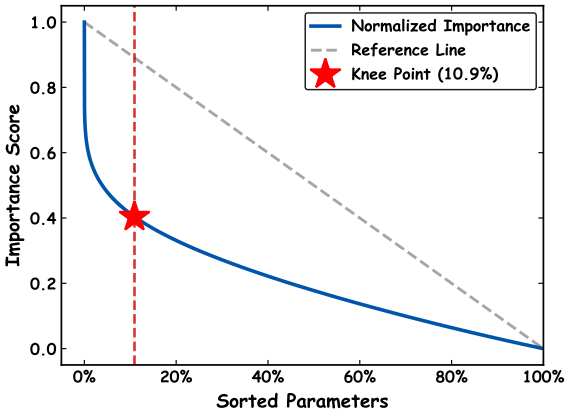
	Dataset	Source	Task Type	Evaluation Metric
1	Yelp	CL Benchmark	Sentiment Analysis	Accuracy
2	AG News	CL Benchmark	Topic Classification	Accuracy
3	DBpedia	CL Benchmark	Entity Typing	Accuracy
4	Yahoo	CL Benchmark	Topic Classification	Accuracy
5	Amazon	CL Benchmark	Sentiment Analysis	Accuracy
6	MNLI	GLUE	Natural Language Inference	Accuracy
7	QQP	GLUE	Paraphrase Detection	Accuracy
8	RTE	GLUE	Natural Language Inference	Accuracy
9	SST-2	GLUE	Sentiment Analysis	Accuracy
10	WiC	SuperGLUE	Word Sense Disambiguation	Accuracy
11	CB	SuperGLUE	Natural Language Inference	Accuracy
12	COPA	SuperGLUE	Causal Reasoning	Accuracy
13	BoolQ	SuperGLUE	Boolean QA	Accuracy
14	MultiRC	SuperGLUE	Multi-hop QA	Accuracy
15	IMDB	External	Sentiment Analysis	Accuracy

Table 5: Instruction prompts provided to the model for each task.

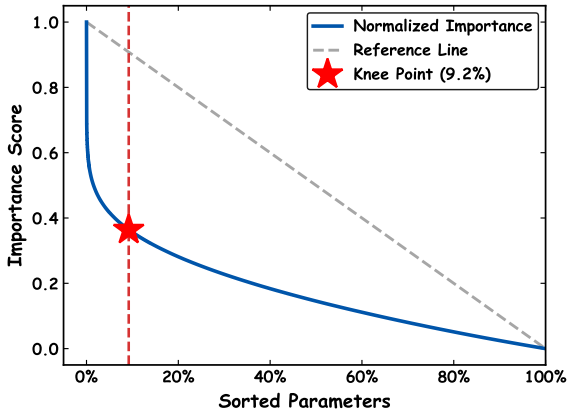
Task	Prompt
NLI	*What is the logical relationship between "sentence 1" and "sentence 2"? Choose one from the options.*
QQP	*Do "sentence 1" and "sentence 2" express the same meaning? Choose one from the options.*
SC	*What is the sentiment of the following passage? Choose one from the options.*
TC	*What is the topic of the following passage? Choose one from the options.*
BoolQA	*According to the passage, is the statement true or false? Choose one from the options.*
MultiRC	*Based on the passage and question, is the candidate's answer correct? Choose one from the options.*
WiC	*Given a word and two sentences, is the word used with the same sense in both? Choose one from the options.*

Table 6: Task sequences used in experiments.

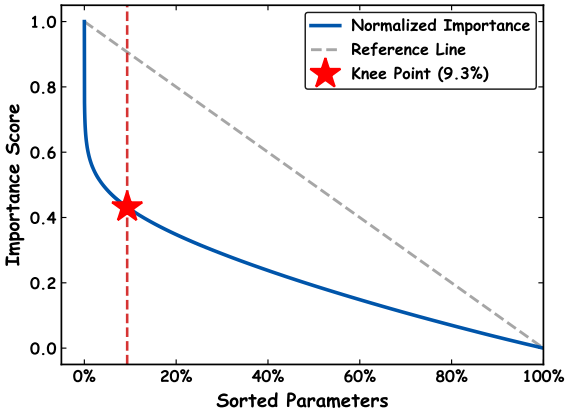
Order	Model	Task Sequence
1	Qwen 3, LLaMA 3.1	DBpedia → Amazon → Yahoo → AG News
2	Qwen 3, LLaMA 3.1	DBpedia → Amazon → AG News → Yahoo
3	Qwen 3, LLaMA 3.1	Yahoo → Amazon → AG News → DBpedia
4	Qwen 3, LLaMA 3.1	MNLI → CB → WiC → COPA → QQP → BoolQ → RTE → IMDB → Yelp → Amazon → SST-2 → DBpedia → AG News → MultiRC → Yahoo
5	Qwen 3, LLaMA 3.1	MultiRC → BoolQ → WiC → MNLI → cb → COPA → QQP → RTE → IMDB → SST-2 → DBpedia → AG News → Yelp → Amazon → Yahoo
6	Qwen 3, LLaMA 3.1	Yelp → Amazon → MNLI → CB → COPA → QQP → RTE → IMDB → SST-2 → DBpedia → AG News → Yahoo → MultiRC → BoolQ → WiC



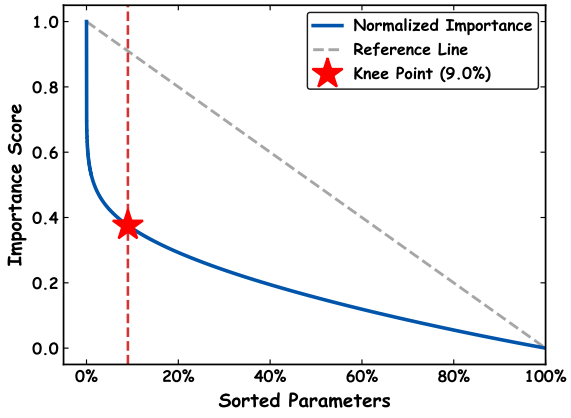
(a) AG News



(b) Amazon



(c) Yahoo



(d) DBpedia

Figure 7: **Visualization of knee-point detection strategy.** The **red star** marks the identified knee point used for trimming.

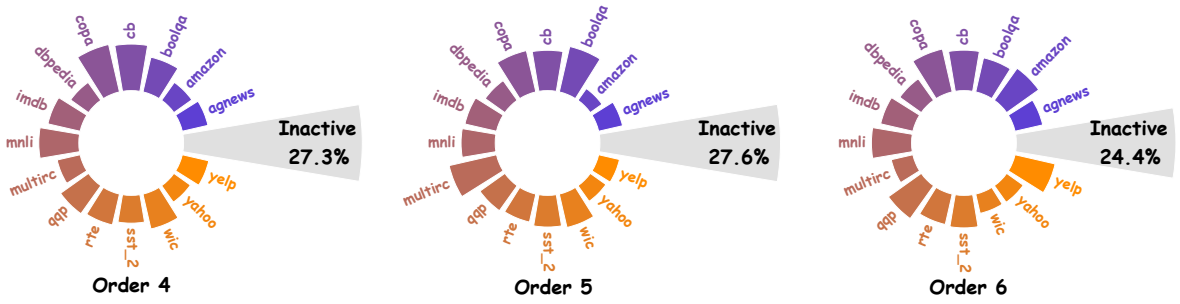


Figure 8: **Visualization of the dynamic parameter allocation across sequential learning stages (Order 4 to Order 6).** Each radial segment represents a specific task (dataset), with the radial length corresponding to the proportion of active parameters allocated to that task. The grey sector labeled *Inactive* denotes the pruned or sparse parameters.

Table 7: LoRA rank sensitivity analysis on LLaMA 3.1-8B (Accuracy, %).

r	Order1	Order2	Order3	Avg.
8	80.3	80.2	80.1	80.2
16	80.8	80.8	81.1	80.9
32	81.0	81.0	81.5	81.1

Figure 8 shows the allocation of active LoRA parameters across tasks. Even after learning 15 tasks, a substantial fraction of parameters remains inactive, with around one quarter of the parameter space never selected by any task across different task orders. Moreover, the spatial patterns of activated parameters vary across task orders, suggesting that subspace allocation is adaptive and task-dependent rather than fixed.

E Experiences Setup

E.1 Artifacts and Licenses

We use several publicly available artifacts in our experiments. Their details and licenses are summarized as follows:

LLaMA 3.1-8B: Released under the Llama 3.1 Community License Agreement. We use it via the Hugging Face library.

Qwen 3-8B: Released under the Apache license 2.0. We use it via the Hugging Face library.

The datasets used in this work, including the Standard CL Benchmark and Long Task Sequence, are accessed via Hugging Face. These datasets are released under their respective original licenses. We confirm that our use of these artifacts is strictly for non-commercial academic research, which is consistent with the intended use and the specific research-oriented licenses provided by the creators.

E.2 Dataset

Table 4 summarizes all datasets used across the continual learning benchmarks. These datasets were also employed in O-LoRA (Wang et al., 2023a), where each is framed as a classification task using a unified instruction-based text-to-text format.

E.3 Instruction Prompts

We adopt the task-specific instruction prompts introduced in O-LoRA (Wang et al., 2023a), as summarized in Table 5.

E.4 Task Sequence Orders

We report task orders used for our CL experiments across LLaMA 3.1 and Qwen 3 models in Table 6. NLI denotes natural language inference, including MNLI, RTE, and CB. SC denotes sentiment analysis, including Amazon, Yelp, SST-2 and IMDB. TC denotes topic classification, including AG News, DBpedia, and Yahoo.

E.5 Implementation Details

We implemented our framework using the DeepSpeed library with ZeRO-Stage 2 optimization and Distributed Data Parallel (DDP) on a cluster of 4 NVIDIA V100 GPUs. All models were trained in FP16 mixed precision. Due to computational constraints, we report results from a single run. To ensure numerical stability in half-precision training, we utilized an automatic loss scaling strategy with an initial scale power of 2^{12}, a hysteresis of 3, and a scale window of 1000 steps. Additionally, we adjusted the AdamW optimizer's epsilon term (ϵ) to 1e-4 to prevent underflow, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0. We employed a dual learning rate strategy: 1e-4 for the LoRA parameters and a larger 1e-2 for the learn-

able task vectors to accelerate routing convergence. The LoRA adapter $\mathbf{A}$ matrix was randomly initialized while preserving the specified rank r, and task vectors were also randomly initialized. The maximum sequence length was capped at 512 tokens. We used a batch size of 8 with 8 gradient accumulation steps. Training was conducted for a maximum of 300 steps, utilizing a WarmupDecayLR scheduler where 50% of the data was randomly sampled during the warm-up phase. To prevent overfitting, we applied early stopping with a patience of 50 steps and gradient clipping with a maximum norm of 1.0. For each task, we randomly sample 1,000 training instances and reserve 500 samples per class for validation.