

Beyond Adapter Expansion: Packing Multiple Tasks into Spare LoRA Capacity

Shuangjie Fan¹, Chengxiang Huang², Zhiyuan Zhao³,
Piotr Koniusz⁴, Junyu Gao³, Di Hu^{5,*}

¹Northeastern University

²Beijing University of Posts and Telecommunications

³Northwestern Polytechnical University

⁴The University of New South Wales

⁵Renmin University of China

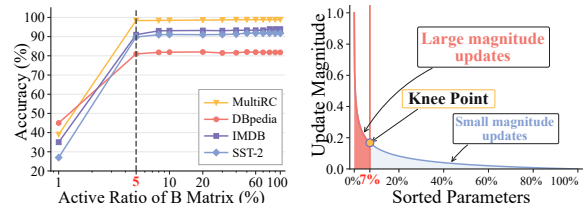
¹ tpartf@gmail.com * dihu@ruc.edu.cn

Abstract

Many continual learning methods for LLMs mitigate forgetting by expanding task-specific modules, yet this leads to memory costs that grow with the number of tasks. We reveal that such expansion may be unnecessary: a single LoRA adapter possesses substantial per-task parameter redundancy, where retaining only 3–10% of LoRA-B entries suffices to recover over 95% of task accuracy. Moreover, the per-parameter update magnitudes follow a heavy-tailed distribution whose geometric *knee point* naturally separates critical parameters from dispensable ones. Building on this insight, we propose Continual Adaptation via Subspace Trimming (CAST), which adaptively allocates a compact task-specific subspace within a shared LoRA adapter using the knee point criterion, freezes previously assigned parameters to eliminate cross-task interference, and routes inputs to relevant subspaces via learned semantic vectors without requiring task identities. Experiments on LLaMA-3.1-8B and Qwen3-8B demonstrate that CAST achieves accuracy comparable to PerTask-LoRA across 15 sequential tasks while maintaining a fixed adapter budget with over 24% idle capacity preserved. Our code is available at <https://github.com/tPartf/CAST>.

1 Introduction

Large language models (LLMs) are increasingly deployed in dynamic environments where they must adapt to a stream of new tasks without forgetting previously acquired knowledge (Wu et al., 2024). A dominant strategy for mitigating catastrophic forgetting is to assign dedicated parameters to each task, such as independent adapters or task-specific prompts (Hu et al., 2021; Wang et al.,



(a) Accuracy vs. Ratio.

(b) Update magnitudes.

Figure 1: (a) Accuracy when retaining only the highest-magnitude **B** entries as a function of the retention ratio. (b) Sorted per-parameter update magnitudes.

2022). While effective at preserving task knowledge, this paradigm implicitly assumes that each task requires its own set of parameters, causing the total parameter footprint to grow with the number of tasks. This raises a natural question: *does each task truly demand a dedicated parameter set, or does a single adapter already contain sufficient capacity for many tasks?*

To investigate this question, we conduct empirical analyses on text classification tasks using LLaMA-3.1-8B with a standard LoRA adapter, and uncover two findings that challenge the expansion assumption. **Firstly**, we find that a single LoRA adapter contains far more capacity than any individual task requires. As shown in Figure 1a, retaining only 3%–10% of the highest-magnitude LoRA-B entries recovers over 95% of full adapter accuracy, revealing that the vast majority of parameters are dispensable for a given task. **Secondly**, we find that this redundancy is not uniformly distributed but follows a structured pattern. As shown in Figure 1b, the sorted per-parameter update magnitudes exhibit a pronounced heavy-tailed distribution, where a small fraction of parameters receive large updates while the majority barely change. This heavy-tailed structure gives rise to a natural geometric **knee point** that cleanly separates the high-impact param-

eters from the long tail of negligible ones. Crucially, the detected knee point *varies* across tasks, indicating that different tasks have inherently different parameter demands and that a single fixed sparsity ratio cannot adequately serve all tasks.

Together, these findings point to a fundamentally different route for continual adaptation. Instead of expanding new adapters for every task or regularizing a shared parameter space, one can treat a single LoRA module as a *finite shared resource*: each incoming task occupies only the compact subspace it actually requires, as determined by its knee point, while previously allocated subspaces remain frozen to prevent cross-task interference. This reframes continual learning from a problem of parameter expansion into one of *adaptive subspace allocation within a fixed budget*.

We propose Continual Adaptation via Subspace Trimming (CAST) to realize this principle. CAST maintains a shared integer-indexed allocation matrix that records the ownership of each LoRA-B entry. When a new task arrives, a brief warm-up probe estimates parameter importance over the remaining unallocated slots, and the knee point criterion adaptively selects the minimal high-impact subset for the current task. Training is then confined to this allocated subspace, while all previously assigned parameters are kept frozen. To enable task-agnostic inference without storing task identities, CAST learns lightweight semantic task vectors that route each input to its relevant subspaces via cosine similarity.

Across 4- and 15-task benchmarks on LLaMA-3.1-8B and Qwen3-8B, CAST achieves accuracy comparable to PerTask-LoRA while maintaining a fixed adapter budget, with over 24% idle capacity preserved after 15 sequential tasks.

Our contributions are:

- (i) We reveal that a single LoRA adapter possesses substantial per-task parameter redundancy, and that the heavy-tailed update distribution provides a natural knee point criterion for determining the minimal task-specific parameter subset.
- (ii) We propose CAST, which performs adaptive subspace allocation within a fixed LoRA module via knee point detection, confined training, and learned semantic routing, eliminating the need for parameter expansion or task identities.
- (iii) Experiments on two LLM backbones demonstrate that CAST approaches PerTask-LoRA performance over 15-task sequences while keeping both adapter size and training memory constant.

2 Related Work

Continual learning for LLMs. CL for LLMs seeks to learn new tasks without catastrophic forgetting. Existing methods mainly fall into three categories: replay-based, regularization-based, and expansion-based approaches. Replay-based methods, such as Experience Replay (ER) (Rolnick et al., 2019), retain knowledge by reusing stored past data, but face privacy and scalability concerns. Regularization-based methods, including CLoRA (Smith et al., 2024) and DEAL (Han et al., 2025), preserve a fixed parameter budget by constraining updates, yet often sacrifice plasticity as constraints accumulate. Expansion-based methods, such as O-LoRA (Wang et al., 2023), ProgPrompt (Razdaibiedina et al., 2023), and Any-SSR (Tong et al., 2025), mitigate forgetting by adding task-specific capacity, but inevitably increase model size over time.

Parameter-Efficient Fine-Tuning. PEFT methods (Houlsby et al., 2019; Pfeiffer et al., 2021; Li and Liang, 2021; Lester et al., 2021) enable efficient LLM adaptation by updating only a small subset of parameters. LoRA (Hu et al., 2021), one of the most widely adopted PEFT methods, introduces trainable low-rank matrices and has inspired more efficient variants such as Loraprune (Zhang et al., 2024) and LoRA-drop (Zhou et al., 2024). LoRI (Zhang et al., 2025) further reduces cross-task interference by freezing the projection **A** while sparsely updating **B** matrix. Building on this idea, CAST selectively freezes and updates sparse subspaces to reduce inter-task interference under a fixed parameter budget.

3 Preliminary

3.1 Problem Formulation

We consider a CL setting (Lopez-Paz and Ranzato, 2022; Delange et al., 2021) in which a model learns from a sequence of T tasks, denoted as $\mathcal{S} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_T\}$. 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.

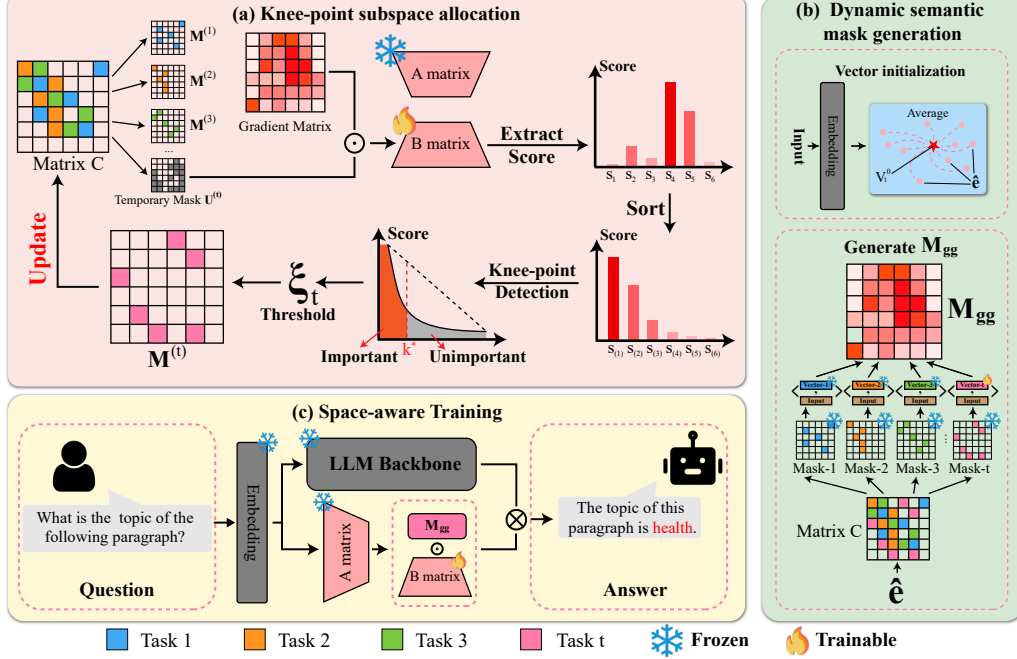


Figure 2: Overview of CAST. (a) Knee point subspace allocation. CAST first probes currently unallocated LoRA-B entries using a temporary mask $\mathbf{U}^{(t)}$, sorts their warm-up update magnitudes, and applies a geometric knee point criterion to select high-impact slots. The selected slots are written into the shared allocation matrix $\mathbf{C}$ as the task-specific mask $\mathbf{M}^{(t)}$. (b) Dynamic semantic mask generation enables task-agnostic activation. Task vectors initialized from current-task embedding centroids are used to compute semantic routing coefficients, which aggregate task-specific masks decoded from $\mathbf{C}$ into an input-specific mask $\mathbf{M}_{gg}$. (c) Space-aware training updates only the allocated LoRA-B subspace while keeping previously assigned parameters frozen. At inference, $\mathbf{M}_{gg}$ activates task-relevant subspaces without requiring task identities.

We focus on the challenging *task-agnostic* continual learning setting (van de Ven and Tolias, 2019). 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 $f_\Theta : \mathcal{X} \rightarrow \mathcal{Y}$ parameterized by Θ , which minimizes the cumulative empirical risk over all observed tasks:

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

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., 2021) 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:

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

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 achieve task-agnostic continual adaptation within a fixed LoRA adapter space. Instead of allocating an independent adapter for each incoming task, CAST reuses a single LoRA-B matrix as a shared parameter space and assigns each task only the subset of parameters it requires.

As illustrated in Figure 2, CAST proceeds in three stages. Firstly, knee point subspace allocation estimates the parameter demand of the current task and selects a compact subspace from the remaining unallocated LoRA-B slots. Secondly, dynamic semantic mask generation learns task-level semantic vectors and constructs an input-specific mask

for task-agnostic inference. Thirdly, space-aware training restricts optimization to the allocated subspace while keeping previously assigned parameters frozen.

To record subspace ownership, CAST maintains an integer-indexed allocation matrix C , where each entry corresponds to one LoRA-B parameter slot. Specifically, $c_j = 0$ denotes an unallocated slot, while $c_j = k$ indicates that the slot has been assigned to task T_k . Task-specific and input-specific masks are decoded from C on demand, avoiding the need to store separate binary masks for different tasks.

4.2 Knee Point Subspace Allocation

For each incoming task $\mathcal{T}_t$, CAST allocates a compact task-specific subspace from the remaining LoRA-B capacity. The allocation procedure consists of two steps: a warm-up probe that estimates the importance of currently unallocated parameter slots, followed by a knee point criterion that adaptively determines the allocation boundary on the sorted importance curve.

Warm-up probe for importance estimation. This step estimates the task-specific importance of currently unallocated LoRA-B entries. The core idea is that, during a short restricted warm-up, parameter slots with larger tentative displacement are more strongly required by the current task and should therefore receive higher importance scores (Sanh et al., 2020).

Let b_j denote the j -th entry of the LoRA-B matrix, and let c_j denote the corresponding entry in the allocation matrix C . The ownership of each slot is defined as

$$c_j = \begin{cases} 0, & \text{if slot } j \text{ is unallocated,} \\ k, & \text{if slot } j \text{ is assigned to task } \mathcal{T}_k. \end{cases} \quad (3)$$

Accordingly, the candidate parameter pool for task $\mathcal{T}_t$ is given by

$$\mathbf{U}^{(t)} = \mathbb{I}(\mathbf{C} = 0), \quad (4)$$

where $\mathbb{I}(\cdot)$ denotes the element-wise indicator function.

Starting from the current LoRA-B matrix $\mathbf{B}^0$, CAST performs Q warm-up gradient steps restricted to the candidate pool:

$$\mathbf{B}^{q+1} = \mathbf{B}^q - \eta \left(\nabla_{\mathbf{B}} \mathcal{L}_{\text{task}} \odot \mathbf{U}^{(t)} \right), \quad (5)$$

where $q = 0, \dots, Q-1$, η is the warm-up learning rate, $\mathcal{L}_{\text{task}}$ is the task loss, and $\odot$ denotes element-wise multiplication.

The set of unallocated candidate slots is denoted as

$$\mathcal{I}^{(t)} = \{j \mid c_j = 0\}. \quad (6)$$

The importance score of each candidate slot is then computed by its warm-up displacement:

$$s_j = |\Delta b_j| = |b_j^Q - b_j^0|, \quad j \in \mathcal{I}^{(t)}. \quad (7)$$

After obtaining the importance scores, $\mathbf{B}$ is restored to $\mathbf{B}^0$.

Knee point thresholding for adaptive allocation. Since the sorted importance scores usually exhibit a heavy-tailed distribution, CAST uses the knee point to separate high-impact slots from the low-impact tail, avoiding a manually specified sparsity ratio.

The candidate scores are first sorted in descending order:

$$s_{(1)} \geq s_{(2)} \geq \dots \geq s_{(N)}, \quad N = |\mathcal{I}^{(t)}|. \quad (8)$$

After min-max normalization, each point on the sorted curve is represented as

$$(x_i, y_i) = \left(\frac{i-1}{N-1}, \tilde{s}_{(i)} \right), \quad (9)$$

where $\tilde{s}_{(i)}$ is the normalized score.

The knee point is identified by the largest distance to the reference chord connecting $(0, 1)$ and $(1, 0)$:

$$k^* = \underset{i}{\operatorname{argmax}} |x_i + y_i - 1|. \quad (10)$$

The corresponding score is used as the adaptive allocation threshold:

$$\xi_t = s_{(k^*)}. \quad (11)$$

Finally, the allocation matrix is updated as

$$c_j \leftarrow \begin{cases} t, & \text{if } c_j = 0 \text{ and } s_j \geq \xi_t, \\ c_j, & \text{otherwise.} \end{cases} \quad (12)$$

The task-specific training mask is decoded from $\mathbf{C}$ as

$$\mathbf{M}^{(t)} = \mathbb{I}(\mathbf{C} = t). \quad (13)$$

4.3 Dynamic Semantic Mask Generation

The allocation matrix $\mathbf{C}$ records the ownership of LoRA-B slots, but task identities are unavailable during task-agnostic inference. CAST therefore maintains a lightweight semantic vector for each task, initialized by the mean embedding of a small

Table 1: Average accuracy on LLaMA-3.1-8B across standard and long-sequence continual learning benchmarks. Order-specific results are reported as mean $\pm$ standard deviation over three random seeds; the Average columns report the mean across task orders. For continual learning methods, the best performance is in **bold** and the second-best is underlined.

		Standard CL Benchmark				Long Task Sequence			
		Order 1	Order 2	Order 3	Average	Order 4	Order 5	Order 6	Average
Reference methods	Zero-shot	13.6 $\pm$ 0.00	13.6 $\pm$ 0.00	13.6 $\pm$ 0.00	13.6	32.1 $\pm$ 0.00	32.1 $\pm$ 0.00	32.1 $\pm$ 0.00	32.1
	Replay	81.2 $\pm$ 1.83	81.5 $\pm$ 1.14	78.5 $\pm$ 2.07	80.4	77.3 $\pm$ 1.09	80.5 $\pm$ 1.76	79.4 $\pm$ 1.43	79.1
	ProgPrompt	80.6 $\pm$ 1.16	79.9 $\pm$ 1.69	80.9 $\pm$ 1.03	80.5	79.4 $\pm$ 1.81	79.3 $\pm$ 1.18	80.1 $\pm$ 1.46	79.6
	PerTask-LoRA	81.5 $\pm$ 0.94	81.5 $\pm$ 0.94	81.5 $\pm$ 0.94	81.5	82.7 $\pm$ 1.56	82.7 $\pm$ 1.56	82.7 $\pm$ 1.56	82.7
CL methods	Vanilla LoRA	61.6 $\pm$ 2.13	56.4 $\pm$ 1.64	58.3 $\pm$ 3.11	58.8	61.3 $\pm$ 1.87	63.7 $\pm$ 2.94	60.4 $\pm$ 2.22	61.8
	EWC	72.3 $\pm$ 1.37	68.5 $\pm$ 2.11	70.4 $\pm$ 1.14	70.4	69.4 $\pm$ 1.93	70.2 $\pm$ 1.26	69.1 $\pm$ 1.61	69.6
	O-LoRA	78.5 $\pm$ 1.74	75.3 $\pm$ 1.27	74.7 $\pm$ 1.89	76.2	72.9 $\pm$ 1.42	73.2 $\pm$ 1.76	70.4 $\pm$ 1.33	72.2
	DEAL	79.8 $\pm$ 1.12	78.9 $\pm$ 1.83	79.1 $\pm$ 1.36	79.3	73.8 $\pm$ 1.68	74.6 $\pm$ 1.17	75.4 $\pm$ 1.54	74.6
	CAST (Ours)	81.0 $\pm$ 1.47	81.0 $\pm$ 1.04	81.5 $\pm$ 1.78	81.1	80.5 $\pm$ 1.21	80.8 $\pm$ 1.88	79.3 $\pm$ 1.13	80.2

Table 2: Average accuracy on Qwen3-8B across standard and long-sequence continual learning benchmarks. Order-specific results are reported as mean $\pm$ standard deviation over three random seeds; the Average columns report the mean across task orders. For continual learning methods, the best performance is in **bold** and the second-best is underlined.

		Standard CL Benchmark				Long Task Sequence			
		Order 1	Order 2	Order 3	Average	Order 4	Order 5	Order 6	Average
Reference methods	Zero-shot	55.6 $\pm$ 0.00	55.6 $\pm$ 0.00	55.6 $\pm$ 0.00	55.6	70.9 $\pm$ 0.00	70.9 $\pm$ 0.00	70.9 $\pm$ 0.00	70.9
	Replay	81.0 $\pm$ 1.17	81.4 $\pm$ 1.91	80.5 $\pm$ 1.34	81.0	78.8 $\pm$ 1.76	80.1 $\pm$ 1.22	79.4 $\pm$ 1.83	79.4
	ProgPrompt	80.2 $\pm$ 1.49	80.8 $\pm$ 1.16	80.7 $\pm$ 1.79	80.6	81.1 $\pm$ 1.23	81.5 $\pm$ 1.67	81.4 $\pm$ 1.12	81.3
	PerTask-LoRA	82.1 $\pm$ 1.62	82.1 $\pm$ 1.62	82.1 $\pm$ 1.62	82.1	84.3 $\pm$ 1.39	84.3 $\pm$ 1.39	84.3 $\pm$ 1.39	84.3
CL methods	Vanilla LoRA	66.3 $\pm$ 1.73	64.4 $\pm$ 2.86	66.4 $\pm$ 1.92	65.7	71.2 $\pm$ 3.13	72.5 $\pm$ 1.64	70.3 $\pm$ 2.41	71.3
	EWC	71.5 $\pm$ 1.94	70.8 $\pm$ 1.38	72.5 $\pm$ 1.87	71.6	69.8 $\pm$ 1.27	71.5 $\pm$ 2.12	71.3 $\pm$ 1.53	70.9
	O-LoRA	78.4 $\pm$ 1.32	76.4 $\pm$ 1.73	77.8 $\pm$ 1.21	77.5	75.4 $\pm$ 1.88	74.8 $\pm$ 1.31	73.4 $\pm$ 1.69	74.5
	DEAL	81.3 $\pm$ 1.81	79.9 $\pm$ 1.24	80.5 $\pm$ 1.63	80.6	79.8 $\pm$ 1.42	78.5 $\pm$ 1.93	79.4 $\pm$ 1.26	79.2
	CAST (Ours)	80.5 $\pm$ 1.26	80.3 $\pm$ 1.84	81.0 $\pm$ 1.13	80.6	82.5 $\pm$ 1.67	83.2 $\pm$ 1.32	81.7 $\pm$ 1.78	82.5

subset of current-task samples rather than random initialization, to provide a stable task-aware routing anchor.

During training, the same semantic routing mechanism is used to generate input-specific masks, while the task vectors are additionally optimized by a routing objective that aligns current task inputs with $\mathbf{v}_t$ and separates them from previous task vectors. At inference time, no task identity is required. CAST directly computes the similarity between the input embedding and all learned task vectors, and uses the resulting routing coefficients to aggregate task-specific masks decoded from $\mathbf{C}$ into $\mathbf{M}_{gg}(\mathbf{x})$.

4.4 Space-Aware Training and Inference

After subspace allocation, CAST trains only the subspace assigned to the current task while keeping previously allocated LoRA- $\mathbf{B}$ slots fixed. The training objective combines the task loss and the routing loss:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{route}}. \quad (14)$$

The LoRA- $\mathbf{B}$ update is restricted by the current task mask:

$$\mathbf{B} \leftarrow \mathbf{B} - \eta \left(\nabla_{\mathbf{B}} \mathcal{L} \odot \mathbf{M}^{(t)} \right). \quad (15)$$

During task-agnostic inference, CAST replaces the hard current-task mask with the dynamically generated semantic mask $\mathbf{M}_{gg}(\mathbf{x})$, so that task-relevant subspaces can be activated without task identities:

$$\mathbf{h} = \mathbf{W}_0 \mathbf{x} + s \cdot [(\mathbf{B} \odot \mathbf{M}_{gg}(\mathbf{x})) \mathbf{A}] \mathbf{x}. \quad (16)$$

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 four text classification datasets (Zhang et al., 2016). Specifically, AG News and Yahoo Answers focus on topic classification, Amazon Reviews (McAuley

and Leskovec, 2013) 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 D.2, D.3, and D.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., 2019), the SuperGLUE benchmark (Wang et al., 2020), 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 more challenging long-sequence continual learning scenario.

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

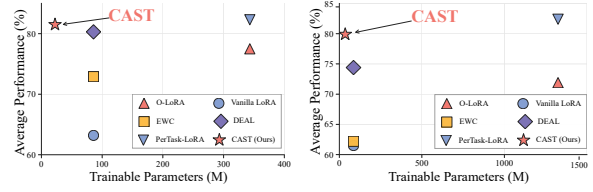
$$AA = \frac{1}{k} \sum_{i=1}^k A_{i,k}. \quad (17)$$

We additionally use **Average Forgetting (AF)** to directly quantify the degradation of previously acquired knowledge.

$$F_i = \frac{\max_{j \in \{i, \dots, T-1\}} A_{i,j} - A_{i,T}}{\max_{j \in \{i, \dots, T-1\}} A_{i,j}} \times 100\%, \quad (18)$$

where $A_{i,j}$ denotes the accuracy on task $\mathcal{T}_i$ after learning task $\mathcal{T}_j$. The forgetting rate F_i measures the relative performance degradation on task $\mathcal{T}_i$ from its historical best performance to the performance observed after learning the complete task sequence. AF is obtained by averaging the task-wise forgetting rates over all previously learned tasks. Lower AF indicates better knowledge retention.

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 Qwen3-8B (Yang et al., 2025). Results across both backbones exhibit consistent trends, demonstrating



(a) Standard CL Benchmark. (b) Long Task Sequence.

Figure 3: Average accuracy versus parameter count for CAST and baseline methods.

the robustness and generalizability of CAST. More details are described in Appendix D.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., 2023), 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. For each task order, we report the AA and standard deviation over three independent runs with different random seeds.

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 performance gap indicates 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 expansion-based methods. DEAL and O-LoRA achieve 79.3% and 76.2% average accuracy on the Standard CL benchmark, but their performance degrades on the long task sequence. This trend suggests that fixed constraints become insufficient to accommodate a growing number of tasks. In contrast, CAST maintains strong performance across both benchmarks.

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 comparable to *PerTask-LoRA* while relying on a single shared LoRA module. This result indicates that CAST can 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 Qwen3-8B 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 3 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. CAST achieves accuracy comparable to the *PerTask-LoRA* baseline while maintaining a constant parameter footprint, indicating that multiple tasks can be packed into a shared LoRA module without substantial performance loss.

Furthermore, our analysis in Section D.7 confirms that the parameter space remains far from saturation even in long sequences. Additionally,

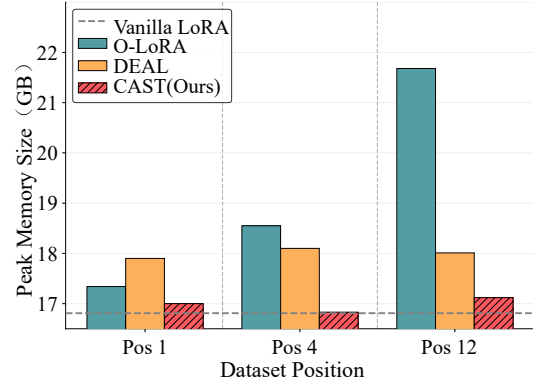


Figure 4: Peak GPU memory footprint (GB) of each method as a function of the number of learned tasks.

the allocated parameter usage for identical tasks varies across different task orders, which verifies the adaptive nature of our allocation strategy.

5.3 Forgetting Evaluation

To directly assess knowledge retention during sequential adaptation, we report AF across different task orders in Table 3. CAST consistently achieves the lowest or second-lowest forgetting across both model backbones and benchmark settings. Its advantage is particularly pronounced under the long task sequences, where the competing methods exhibit substantially greater forgetting as additional tasks are learned, while CAST maintains consistently low AF across different task orders. These results demonstrate that allocating task-specific LoRA subspaces and freezing previously assigned parameters effectively limits cross-task interference, enabling CAST to preserve previously acquired knowledge during continual adaptation.

6 Ablation Study

To further analyze the effectiveness and efficiency of CAST, we conduct a series of ablation and diagnostic studies. Due to space limitations, additional studies, including dynamic parameter allocation and adaptive knee point visualization, are provided in Appendix B. Detailed settings for all analysis and ablation experiments are summarized in Appendix D.6.

6.1 Comparison of Training Memory Footprint

We further evaluate whether CAST maintains training memory efficiency as continual tasks accumulate. This analysis measures peak GPU memory consumption at different task positions.

Table 3: Average Forgetting (AF, %) on LLaMA-3.1-8B and Qwen3-8B across standard and long-sequence continual learning benchmarks. Order-specific results are reported as mean $\pm$ standard deviation over three random seeds. Lower values indicate better knowledge retention ($\downarrow$). For continual learning methods, the best performance is in **bold** and the second-best is underlined.

Backbone	Method	Standard CL Benchmark $\downarrow$			Long Task Sequence $\downarrow$		
		Order 1	Order 2	Order 3	Order 4	Order 5	Order 6
LLaMA-3.1-8B	Vanilla LoRA	20.25 $\pm$ 1.85	23.32 $\pm$ 2.74	25.85 $\pm$ 1.62	23.45 $\pm$ 1.76	19.82 $\pm$ 1.95	24.17 $\pm$ 2.12
	EWC	10.35 $\pm$ 1.25	13.65 $\pm$ 1.65	11.55 $\pm$ 1.12	14.15 $\pm$ 1.63	13.55 $\pm$ 1.18	15.26 $\pm$ 1.52
	O-LoRA	4.02 $\pm$ 0.82	6.18 $\pm$ 0.74	6.69 $\pm$ 0.95	11.25 $\pm$ 1.35	10.82 $\pm$ 1.42	12.55 $\pm$ 1.25
	DEAL	2.45 $\pm$ 0.38	3.12 $\pm$ 0.49	2.95 $\pm$ 0.41	9.35 $\pm$ 0.91	8.52 $\pm$ 0.87	7.48 $\pm$ 0.78
	CAST (Ours)	1.03 $\pm$ 0.22	1.01 $\pm$ 0.29	0.81 $\pm$ 0.30	1.35 $\pm$ 0.28	1.18 $\pm$ 0.41	1.73 $\pm$ 0.36
Qwen3-8B	Vanilla LoRA	17.15 $\pm$ 1.66	19.45 $\pm$ 1.82	17.18 $\pm$ 1.75	13.32 $\pm$ 1.58	12.18 $\pm$ 1.41	13.95 $\pm$ 1.57
	EWC	11.45 $\pm$ 1.41	12.25 $\pm$ 1.22	10.32 $\pm$ 1.35	15.42 $\pm$ 1.35	13.85 $\pm$ 1.54	14.09 $\pm$ 1.46
	O-LoRA	4.45 $\pm$ 0.68	6.25 $\pm$ 0.73	4.93 $\pm$ 0.84	9.35 $\pm$ 1.12	10.15 $\pm$ 1.09	11.79 $\pm$ 1.23
	DEAL	0.75 $\pm$ 0.35	<u>1.89 $\pm$ 0.45</u>	1.14 $\pm$ 0.36	<u>5.35 $\pm$ 0.65</u>	<u>6.42 $\pm$ 0.82</u>	<u>5.69 $\pm$ 0.72</u>
	CAST (Ours)	<u>0.88 $\pm$ 0.18</u>	0.95 $\pm$ 0.25	0.77 $\pm$ 0.20	1.15 $\pm$ 0.31	0.84 $\pm$ 0.24	1.51 $\pm$ 0.32

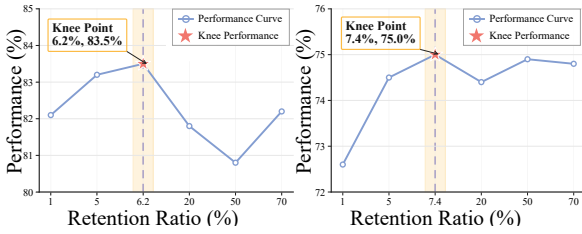


Figure 5: Accuracy on MNLI and Yahoo as a function of the parameter retention ratio. Red stars mark the knee points identified by automatic detection.

As shown in Figure 4, CAST keeps its peak memory usage nearly constant and remains close to vanilla LoRA. In contrast, O-LoRA shows increasing memory consumption as the task sequence grows, while DEAL consistently requires higher memory. These results indicate that CAST preserves the training memory efficiency of fixed-parameter LoRA under long-sequence continual learning.

6.2 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 5. The results demonstrate that the knee points detected by our approach correspond to effective retention ratios for the model.

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

6.3 Storage Scalability Analysis

We evaluate whether CAST can avoid task-dependent storage growth as the number of contin-

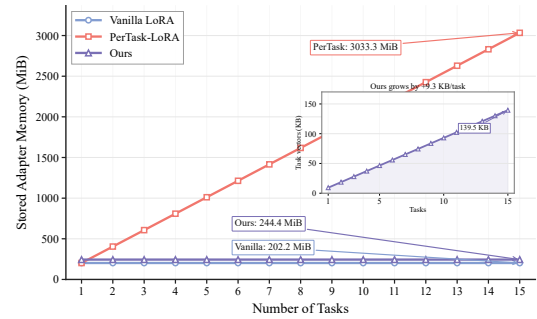


Figure 6: Storage overhead comparison on LLaMA-3.1-8B.

ual tasks increases. This analysis compares CAST with Vanilla LoRA and PerTask-LoRA.

As shown in Figure 6, PerTask-LoRA grows linearly with the task sequence length, while CAST remains close to Vanilla LoRA. After 15 tasks, CAST requires only 244.4 MiB, compared with 3033.3 MiB for PerTask-LoRA and 202.2 MiB for Vanilla LoRA, demonstrating near-constant storage cost under continual adaptation.

6.4 Effect of CAST Selection

We compare CAST-selected subsets with random subsets under the same allocation budget to verify that CAST identifies task-relevant LoRA subspaces rather than merely benefiting from sparsity.

As shown in Figure 7, CAST consistently achieves higher average accuracy than random selection across different LoRA ranks. Notably, CAST with a lower rank can even outperform random selection with a higher rank, suggesting that the proposed selection strategy improves parameter utilization efficiency by allocating more informa-

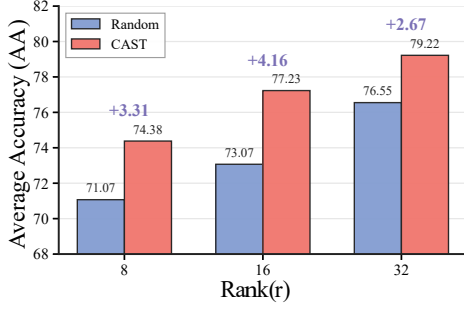


Figure 7: Average accuracy of CAST and random parameter selection at different LoRA ranks.

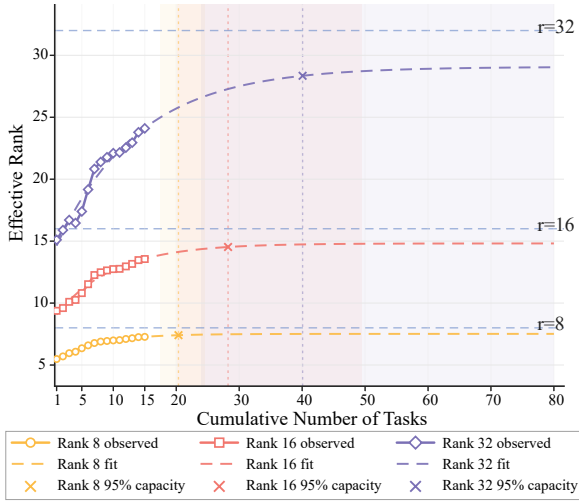


Figure 8: Effective-rank saturation analysis for multiple LoRA ranks. Solid curves denote observed effective ranks, dashed curves denote fitted saturation trends, and cross markers indicate the estimated task counts at which the effective rank reaches 95% of its fitted asymptote.

tive subspaces.

6.5 LoRA Subspace Saturation Analysis

CAST allocates multiple task-specific subspaces within a single fixed-rank LoRA module, making its total structural capacity finite. We therefore examine how the shared LoRA space is progressively occupied as tasks accumulate and whether increasing the LoRA rank delays structural saturation.

As shown in Figure 8, the effective rank increases with the number of learned tasks and gradually approaches a fitted asymptote. Based on the fitted curves, the effective rank is estimated to reach 95% of its asymptotic value after approximately 20, 28, and 40 tasks for LoRA ranks of 8, 16, and 32, respectively, as summarized in Table 4. These estimates characterize the structural occupancy of the shared LoRA space rather than the exact num-

Table 4: Estimated number of tasks required to reach 95% of the fitted asymptotic effective rank.

LoRA Rank	N_{95} (Tasks)	95% CI	R^2
8	20	[17, 24]	0.991
16	28	[20, 49]	0.974
32	40	[24, 118]	0.965

ber of tasks that CAST can learn while maintaining a given level of performance. Nevertheless, the results indicate that larger LoRA ranks delay structural saturation and provide greater capacity for continual adaptation. We next evaluate whether the remaining unallocated subspace retains functional utility for learning subsequent tasks. More details are described in D.7.

7 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 requiring task identities at inference time. Extensive experiments demonstrate its competitive performance on benchmarks, alongside strong efficiency in GPU memory consumption.

8 Limitations

Despite its effectiveness, our study has several limitations. Firstly, due to computational resource constraints, we are unable to conduct comprehensive experiments on larger-scale models, such as 70B+ variants, and our results are primarily focused on 8B-scale models. Secondly, although CAST keeps the core LoRA adapter space fixed, it still introduces a small amount of per-task routing metadata for task-agnostic inference. Third, our evaluation focuses primarily on classification and multiple-choice reasoning tasks formulated in a text-to-text format. Whether CAST generalizes to open-ended generation tasks remains to be investigated. Thus, CAST should be viewed as a fixed-adapter-budget approach with negligible routing overhead, rather than a strictly zero-overhead framework.

References

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and

- Tinne Tuytelaars. 2021. [A continual learning survey: Defying forgetting in classification tasks](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1.
- Kawin Ethayarajh. 2019. [How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings](#). *Preprint*, arXiv:1909.00512.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *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](#). *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](#). *Preprint*, arXiv:1902.00751.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. [Lora: Low-rank adaptation of large language models](#). *Preprint*, arXiv:2106.09685.
- 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. 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, S. Auer, and Christian Bizer. 2015. [Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia](#). *Semantic Web*, 6:167–195.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The power of scale for parameter-efficient prompt tuning](#). *Preprint*, arXiv:2104.08691.
- Xiang Lisa Li and Percy Liang. 2021. [Prefix-tuning: Optimizing continuous prompts for generation](#). *Preprint*, arXiv:2101.00190.
- David Lopez-Paz and Marc’Aurelio Ranzato. 2022. [Gradient episodic memory for continual learning](#). *Preprint*, arXiv:1706.08840.
- Andrew L. 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, Portland, Oregon, USA. Association for Computational Linguistics.
- 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, RecSys ’13*, page 165–172, New York, NY, USA. Association for Computing Machinery.
- Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. [Adapterfusion: Non-destructive task composition for transfer learning](#). *Preprint*, arXiv:2005.00247.
- Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madihan Khabisa, Mike Lewis, and Amjad Almahairi. 2023. [Progressive prompts: Continual learning for language models](#). *Preprint*, arXiv:2301.12314.
- David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. 2019. [Experience replay for continual learning](#). *Preprint*, arXiv:1811.11682.
- Victor Sanh, Thomas Wolf, and Alexander M. Rush. 2020. [Movement pruning: Adaptive sparsity by fine-tuning](#). *Preprint*, arXiv:2005.07683.
- James Seale Smith, Yen-Chang Hsu, Lingyu Zhang, Ting Hua, Zsolt Kira, Yilin Shen, and Hongxia Jin. 2024. [Continual diffusion: Continual customization of text-to-image diffusion with c-lora](#). *Preprint*, arXiv:2304.06027.
- 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 models](#). In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3047–3057.
- Gido M. van de Ven and Andreas S. Tolias. 2019. [Three scenarios for continual learning](#). *Preprint*, arXiv:1904.07734.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2020. [Superglue: A stickier benchmark for general-purpose language understanding systems](#). *Preprint*, arXiv:1905.00537.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. [Glue: A multi-task benchmark and analysis platform for natural language understanding](#). *Preprint*, arXiv:1804.07461.

- Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-jing Huang. 2023. [Orthogonal subspace learning for language model continual learning](#). *Preprint*, arXiv:2310.14152.
- 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, 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, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. 2025. [Lori: Reducing cross-task interference in multi-task low-rank adaptation](#). *Preprint*, arXiv:2504.07448.
- 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](#). *Preprint*, arXiv:2305.18403.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. [Character-level convolutional networks for text classification](#). *Preprint*, arXiv:1509.01626.
- Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, Tiejun Zhao, and Muyun Yang. 2024. [Lora-drop: Efficient lora parameter pruning based on output evaluation](#). *Preprint*, arXiv:2402.07721.

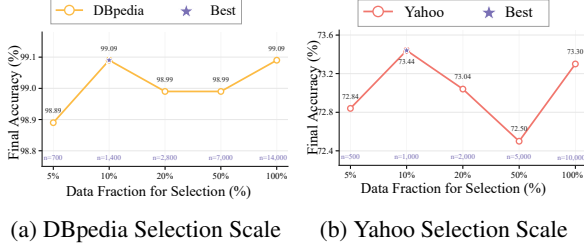


Figure 9: Final accuracy on DBpedia and Yahoo when the knee point is determined using different fractions of the training data during warm-up probing.

A Hyperparameter Sensitivity Analysis

A.1 Robustness of Mask Selection to Data Scale

To examine the stability of CAST during the warm-up selection stage, we vary the fraction of training data used to identify the knee point on DBpedia and Yahoo. After the parameter mask is selected, all settings are trained using the full training set, thereby isolating the effect of the warm-up data fraction on subspace selection.

As reported in Figure 9a and Figure 9b, using merely 10% of the training data for knee point selection yields a final accuracy that is on par with, or even slightly outperforms, selecting the mask with the full dataset. This empirical evidence demonstrates that our selection mechanism reliably identifies critical sub-networks even with limited samples, confirming that data scale during the warm-up phase does not bottleneck final model performance.

A.2 Effect of LoRA Rank

We evaluate the sensitivity of CAST to the LoRA rank r on the standard CL benchmark using LLaMA-3.1-8B. The rank r controls the size of the trainable LoRA subspace, and Table 5 reports the accuracy under different task orders with $r \in \{8, 16, 32\}$.

Increasing r from 8 to 16 improves the average accuracy from 80.2% to 80.9%, while further increasing r to 32 brings only marginal gains, indicating diminishing returns. CAST maintains strong performance even at a small rank, demonstrating its robustness to the choice of r and its parameter efficiency.

Table 5: 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

B Ablation Study

B.1 Dynamic Parameter Allocation and Subspace Capacity Analysis

To further examine whether the shared LoRA module becomes saturated under long-sequence continual learning, we analyzed the dynamic parameter allocation patterns of CAST across different task orders. As shown in Figure 10, each sector represents an individual task, where the sector length denotes the proportion of active LoRA parameters assigned to that task. The gray region indicates inactive parameters that have not been selected by any task.

The results show that after sequentially learning 15 tasks, CAST still preserves a considerable portion of inactive parameter space across all evaluated long task sequences. Specifically, the remaining inactive parameter ratios are approximately 27.3%, 27.6%, and 24.4% for Order 4, Order 5, and Order 6, respectively. This suggests that the shared LoRA module is not fully saturated even after long-sequence continual learning, and still retains additional capacity for future task adaptation.

Moreover, the distribution of active parameters varies across different task orders, indicating that CAST does not allocate parameters according to a fixed pattern. Instead, the active subspaces are adaptively selected according to task characteristics and learning order. These results provide additional evidence that CAST can exploit the intrinsic parameter redundancy of LoRA and dynamically allocate task-specific subspaces within a fixed parameter budget.

B.2 Effectiveness of Task Vectors

Semantic representations produced by pretrained language models are often anisotropic, causing embeddings to concentrate in a narrow cone and reducing the discriminability of cosine-based matching (Ethayarajh, 2019). Therefore, CAST uses a lower routing temperature to sharpen the similarity-based Softmax distribution and improve task-vector selec-

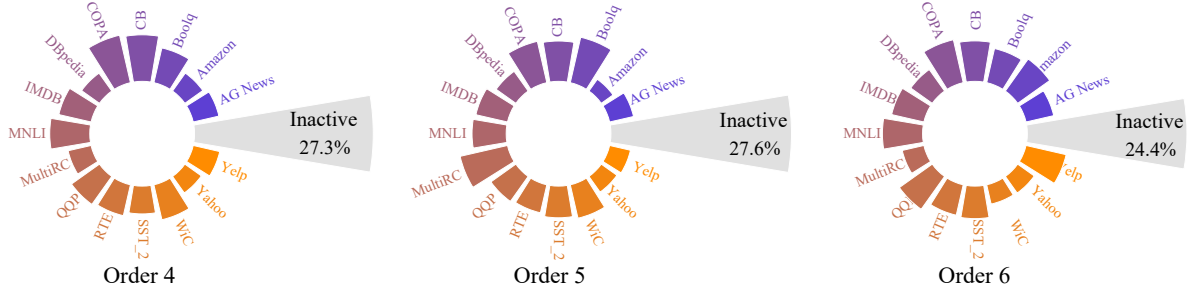


Figure 10: Dynamic parameter allocation of CAST in long-sequence continual learning. Each sector represents one task, and the sector length denotes the proportion of active LoRA parameters assigned to that task. The gray region indicates inactive parameters that have not been selected by any task.

Table 6: Comparison of training performance across different routing methods.

Selection Strategy	AA(%)	AF(%)
None	66.15	28.15
Argmax	74.29	21.29
Softmax $\gamma = 1$	76.21	6.33
Softmax $\gamma = 0.1$	78.68	4.09
Softmax $\gamma = 0.01$	80.90	1.67

tion during inference. This sharper routing distribution helps reduce ambiguity when multiple task vectors have similar cosine scores. To evaluate this design, we conduct an ablation study on Order 1 using LLaMA-3.1-8B. The results are reported in Table 6.

The results indicate that a lower temperature coefficient yields better performance. Meanwhile, the performance of the Argmax strategy is lower than that of the Softmax approach. This suggests that Softmax can effectively mitigate the issue of distribution deviation while maintaining a certain degree of marginal smoothness.

As shown in Figure 11, we also visualize the routing confusion matrix on Order 4. The visualization indicates that the router generally performs well across most tasks. High percentages are consistently visible on the main diagonal. For example, tasks like WiC, COPA, and DBpedia show routing accuracies near 100%. This suggests that the router can effectively identify and separate these specific datasets.

However, some misrouting still occurs between conceptually similar tasks. For instance, the model routes 20.6% of RTE samples to MNLI. It also routes 18.3% of MNLI samples to RTE. This overlap is reasonable, as both datasets focus on natural language

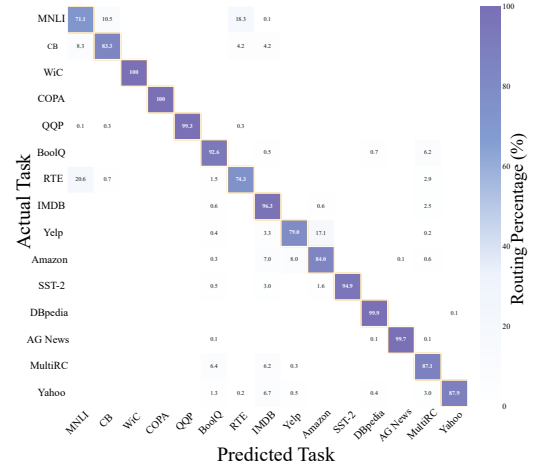


Figure 11: Routing confusion matrix on Order 4.

inference. Similarly, Yelp reviews are sometimes routed to the Amazon task. This indicates that the router might rely on shared sentiment features or vocabulary. Overall, the matrix demonstrates that the routing strategy is largely effective. It successfully directs most inputs to the correct task experts, with only minor confusion occurring on closely related domains.

B.3 Knee Point Robustness

We evaluate the robustness of knee point detection on Yahoo and DBpedia. We introduce Gaussian noise during warm-up and use the signal-to-noise ratio (SNR), measured in decibels (dB), to control the perturbation strength. A lower SNR indicates stronger noise. Robustness is evaluated by the percentage of selected parameters shared with the mask obtained under the clean setting.

As shown in Table 7, the mask overlap decreases as the noise strength increases. Nevertheless, the selected mask remains largely consistent with the clean setting under mild noise.

Table 7: Mask overlap with the clean setting under different levels of Gaussian noise during warm-up.

SNR (dB)	22.1	18.9	16.7
Mask overlap (%)	84.12	70.12	53.05

Table 8: Functional evaluation of the remaining LoRA-B capacity on LLaMA-3.1-8B under Order 4. At each checkpoint, CAST learns the subsequent task using only the remaining unallocated subspace. The performance gap is computed relative to independently trained PerTask-LoRA. Results are reported as accuracy (%).

Learned tasks	Next task	CAST	PerTask-LoRA	Gap ↓
5	BoolQ	58.30	60.90	2.60
10	SST-2	91.67	95.83	4.16
14	Yahoo	66.40	73.50	7.10

B.4 Functional Utility of the Remaining Capacity

The proportion of unallocated LoRA-B entries indicates the remaining structural capacity, but does not directly measure its utility for future adaptation. We therefore evaluate LLaMA-3.1-8B under Order 4 by training CAST on the subsequent task after 5, 10, and 14 tasks have been learned, using only the remaining unallocated subspace. Independently trained PerTask-LoRA serves as the task-specific upper-bound reference.

As shown in Table 8, the unallocated subspace remains effective for learning subsequent tasks, even late in the task sequence. Although the performance gap increases as the available capacity becomes more constrained, CAST remains competitive with independently trained PerTask-LoRA.

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 Section 4.2, we utilize the geometric knee point detection strategy to dynamically identify the knee point index.

Figure 12 illustrates the knee point detection process across the standard CL benchmark datasets. 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

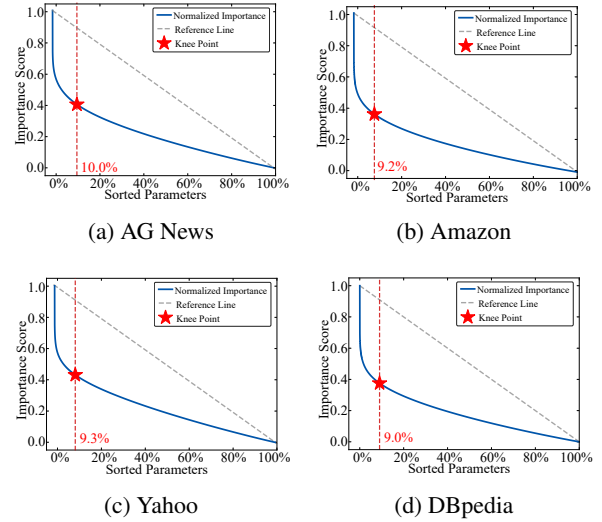


Figure 12: Sorted importance scores and identified knee points on four standard CL benchmark datasets. The red star marks the knee point.

is mathematically identified as the point with the maximum perpendicular distance to this baseline, representing the optimal cutoff for parameter retention.

D Experimental Setup

D.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.

Qwen3-8B: 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.

D.2 Dataset

Table 10 summarizes all datasets used across the continual learning benchmarks. These datasets were also employed in O-LoRA (Wang et al., 2023), where each is framed as a classification task using a unified instruction-based text-to-text format.

D.3 Instruction Prompts

We adopt the task-specific instruction prompts introduced in O-LoRA (Wang et al., 2023), as summarized in Table 11.

D.4 Task Sequence Orders

We report task orders used for our CL experiments across LLaMA-3.1-8B and Qwen3-8B models in Table 12. 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.

D.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. 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 learnable task vectors to accelerate routing convergence. The LoRA adapter **A** matrix was randomly initialized while preserving the specified rank r . The maximum sequence length was capped at 512 tokens. For each main experiment, we conduct three runs with different random seeds and report the mean performance across runs. 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.

D.6 Settings for Analysis and Ablation Studies

For the observational experiment in Figure 1, we use LLaMA-3.1-8B on four representative datasets, including DBpedia, MultiRC, IMDB, and SST-2. To examine the redundancy of LoRA parameters, we activate different proportions of LoRA-B entries

with the largest absolute update magnitudes and plot the resulting accuracy as a function of the active parameter ratio.

For the parameter-performance comparison in Figure 3, we use LLaMA-3.1-8B and record the total number of trainable parameters and the average accuracy after completing the corresponding task sequence. Specifically, Figure 3a reports the results after training on Order 1, while Figure 3b reports the results after training on Order 4.

For the training memory analysis in Figure 4, we place DBpedia at different positions in the task sequence and record the peak GPU memory consumption during training with LLaMA-3.1-8B. This experiment is used to evaluate whether the memory footprint grows as the number of learned tasks increases.

For the knee-point effectiveness analysis in Figure 5, we use LLaMA-3.1-8B on MNLi and Yahoo. We vary the fraction of data used during the warm-up selection stage, record the selected parameter ratio and downstream accuracy, and plot the corresponding curves.

For the storage scalability analysis in Figure 6, we use LLaMA-3.1-8B with LoRA applied to the attention projections **q**, **k**, **v**, **o** and the FFN projections gate, up, and down. We record the storage cost required by different methods after completing the long task sequence and plot the storage growth curves.

For the CAST selection study in Figure 7, we use LLaMA-3.1-8B under Order 4. To ensure a fair comparison, we keep the allocation budget identical for each task and compare CAST-based parameter selection with random selection across different LoRA ranks. The average accuracy is then reported for each rank.

For the LoRA subspace saturation analysis in Figure 8, we train LLaMA-3.1-8B under Order 4 and compute the average effective rank of the LoRA-B matrix after each task. We repeat this analysis under different LoRA ranks and plot the effective-rank trajectory across the task sequence.

D.7 LoRA Subspace Saturation Analysis

CAST allocates multiple task-specific subspaces within a single fixed-rank LoRA module, making its total structural capacity finite. We therefore examine how the shared LoRA space is progressively utilized as tasks accumulate and whether increasing the LoRA rank delays saturation.

For each LoRA-augmented layer ℓ , we define the cumulative update after learning n tasks as

$$\Delta \mathbf{W}_\ell^{(n)} = (\mathbf{B}_\ell \odot \mathbb{I}[1 \leq \mathbf{C}_\ell \leq n]) \mathbf{A}_\ell. \quad (19)$$

Given its singular values $\{\sigma_i\}$, we measure the utilized dimensionality using the entropy-based effective rank:

$$r_{\text{eff}} = \exp \left(- \sum_i p_i \log p_i \right), \quad p_i = \frac{\sigma_i}{\sum_j \sigma_j}. \quad (20)$$

The reported effective rank is averaged across all LoRA-augmented layers and experimental runs.

For each nominal LoRA rank r , we fit the effective-rank trajectory using the saturation function

$$f_r(n) = a_r (1 - \exp(-k_r n)), \quad (21)$$

where a_r denotes the fitted asymptotic effective rank and k_r controls the saturation rate. We define the structural saturation point as the number of tasks required to reach 95% of the fitted asymptote:

$$N_{95}^{(r)} = \frac{-\log(0.05)}{k_r}. \quad (22)$$

The 95% confidence intervals are derived from the uncertainty of the nonlinear fit and transformed through the above equation.

As shown in Figure 8, the effective rank increases with the number of learned tasks and gradually approaches a fitted asymptote. The estimated N_{95} values are approximately 20, 28, and 40 tasks for LoRA ranks of 8, 16, and 32, respectively, as summarized in Table 4. The wider confidence interval for rank 32 indicates greater uncertainty because its observed trajectory remains farther from saturation. Overall, increasing the LoRA rank delays spectral saturation and provides more update directions for continual adaptation.

These estimates characterize the structural occupancy of the shared LoRA space rather than the exact number of tasks that CAST can learn while maintaining a particular performance level. Functional capacity additionally depends on task similarity, task difficulty, and allocation sparsity. We next evaluate whether the remaining unallocated entries retain sufficient utility for learning subsequent tasks.

E Case Study

To intuitively understand the effectiveness of our proposed method, we present a qualitative case

study in Table 9, 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. 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 shows that our method transfers relevant knowledge from past tasks to facilitate current inference. While leveraging historical information, the mechanism ensures the primacy of the current task's learning. This demonstrates that our approach maintains high plasticity, synthesizing past knowledge to solve new problems without suffering from intransigence.

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 options. 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 9: Qualitative comparison between Vanilla LoRA and CAST on two representative test samples.

Table 10: 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 11: Instruction prompts provided to the model for each task.

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

Table 12: Task sequences used in experiments.

Order	Model	Task Sequence
1	Qwen3-8B, LLaMA-3.1-8B	DBpedia → Amazon → Yahoo → AG News
2	Qwen3-8B, LLaMA-3.1-8B	DBpedia → Amazon → AG News → Yahoo
3	Qwen3-8B, LLaMA-3.1-8B	Yahoo → Amazon → AG News → DBpedia
4	Qwen3-8B, LLaMA-3.1-8B	MNLI → CB → WiC → COPA → QQP → BoolQ → RTE → IMDB → Yelp → Amazon → SST-2 → DBpedia → AG News → MultiRC → Yahoo
5	Qwen3-8B, LLaMA-3.1-8B	MultiRC → BoolQ → WiC → MNLI → CB → COPA → QQP → RTE → IMDB → SST-2 → DBpedia → AG News → Yelp → Amazon → Yahoo
6	Qwen3-8B, LLaMA-3.1-8B	Yelp → Amazon → MNLI → CB → COPA → QQP → RTE → IMDB → SST-2 → DBpedia → AG News → Yahoo → MultiRC → BoolQ → WiC