GROLE: Instance-Level Group Relative Optimization for LoRA Experts in Incremental Learning

Yongyi Liao*, **Wencan Lai***, **Jun Fang***, **Jinjin Guo, Xiaohui Zhang, Zhiyuan Liu, Chao Liu**†, **Pengzhang Liu, Qixia Jiang**

JD Retail, Beijing, China

{liaoyongyi1, laiwencan1, fangjun8, liuchao27}@jd.com

Abstract

While Large Language Models (LLMs) demonstrate remarkable zero-shot generalization, adapting them to downstream tasks or shifting data distributions often requires continual fine-tuning—a process prone to catastrophic forgetting and limited knowledge transfer. This challenge is especially pronounced in online Incremental Learning (IL) settings, where task boundaries are blurred, and data arrives in a non-stationary stream. To address these issues, we propose GROLE (**G**roup **R**elative **O**ptimization for **L**oRA **E**xperts), a novel approach that incrementally constructs a pool of frozen, task-specific Low-Rank Adaptation (LoRA) experts. At its core, GROLE employs a lightweight, instance-level expert selector optimized through a group relative reinforcement learning objective, which dynamically combines relevant experts to maximize adaptability without compromising stability. Extensive experiments across diverse incremental learning benchmarks show that GROLE consistently outperforms state-of-the-art methods, particularly in task-free and blurred-boundary settings, achieving an optimal balance between plasticity and robustness.

1 Introduction

Large language models (LLMs) achieve remarkable generalization capabilities across a wide range of tasks (OpenAI et al., 2024; Yang et al., 2025a; Comanici et al., 2025), yet peak performance on downstream tasks still demands task-specific fine-tuning. This alignment process injects domain knowledge into the model, but it also creates a costly bottleneck: every new task requires a separate copy of the full parameter set, multiplying storage and training expenses and fragmenting the model zoo into isolated experts.

Incremental learning (IL) offers a unified alternative, where a single model acquires new knowledge incrementally over time, while preserving previously learned capabilities and leveraging them to enhance future adaptation (Ke and Liu, 2022; Wang et al., 2024). An effective incremental learning system is guided by two principal objectives. Primarily, it has to overcome catastrophic forgetting (CF), avoiding the performance degradation in earlier tasks when the model parameters are updated for fresh data. Second, the system should facilitate knowledge transfer (KT), reusing shared structures to accelerate future learning through backward transfer and even refine earlier tasks.

Recent parameter-efficient techniques, notably LoRA (Low-Rank Adaptation), reduce storage by representing task-specific updates as low-rank matrices. Most IL approaches built on LoRA still fall into two extremes: (a) maintain a single LoRA and regularize its drift, which suffers from CF under blurry boundaries (Wang et al., 2023a, 2025); or (b) allocate an independent LoRA per task, which requires task identities at inference and limits knowledge transfer (Chen et al., 2024; Yang et al., 2025b). In this work, a question naturally arises: can we enjoy the memory efficiency of LoRA while routing only the experts that are necessary for each incoming instance, without requiring task labels?

We answer this question with GROLE (Group Relative Optimization for LoRA Experts), which introduces two key innovations:

- **Frozen Expert Pool**. GROLE incrementally builds a diverse set of task-specific LoRA experts that remain frozen after creation, eliminating forgetting through parameter isolation.
- **Lightweight Adaptive Selector**. A trainable routing network dynamically combines experts for each input instance using a novel group-relative reinforcement learning objective inspired by GRPO (Shao et al., 2024).

*Equal contribution.
†Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 39170–39182
July 2-7, 2026

Crucially, this operates without value/reward networks or gradient flow through the LLM backbone.

By training the lightweight selector on a small batch of current stream data, GROLE effectively prevents forgetting caused by direct parameter updates while enhancing the model's capacity for knowledge transfer across tasks through instance-level expert merging. Experiments on text-classification benchmarks with up to 12 sequential tasks show that GROLE establishes a new state of the art, outperforming seven IL baselines and even surpassing the multi-task upper bound on the standard CL benchmark. Analysis reveals that instance-level routing consistently reduces negative transfer in blurred-boundary settings and yields robust generalization to held-out tasks.

In summary, the main contributions of our work are as follows:

- We formalize instance-level IL as a joint optimization over a frozen expert pool and a learnable weight space, providing a principled trade-off between stability and plasticity.
- We propose GROLE, the LoRA-based IL framework with instance-level selector that routes experts without task IDs, value functions, or gradient flow through the LLM.
- We demonstrate gains across multiple partitioning strategies and verify strong out-of-distribution generalization, highlighting the practical value of GROLE in open-world continual learning.

2 Related Work

2.1 Incremental Learning (IL)

IL aims to incrementally train a model on sequential data by navigating the fundamental trade-off between mitigating CF (McCloskey and Cohen, 1989; Robins, 1995) and maximizing KT (Thrun, 1995; Lopez-Paz and Ranzato, 2017). Existing work is generally categorized into three main branches: replay-based, optimization-based, and architecture-based methods. Replay-based IL methods mitigate forgetting by maintaining a representative subset of old data in a memory buffer or using a generative model to synthesize pseudo-data (Rebuffi et al., 2017; Wu et al., 2018; Riemer et al., 2019). Optimization-based IL methods focus on how to optimize the objective function or the gradient direction during training. Early works introduce objective function optimization, such as regularization and knowledge distillation, to impose constraints on model parameters (Li and Hoiem, 2016; Kirkpatrick et al., 2017). Recent research has adapted these principles to the fine-tuning of LLMs, primarily by enforcing orthogonality constraints within the low-rank parameter subspaces to minimize interference between sequential tasks (Wang et al., 2023a, 2025). Architecture-based IL methods isolate task-specific parameters to prevent interference from data across different tasks during model updates. While some approaches directly optimize task-specific parameters using task IDs (Xue et al., 2022; Gurbuz and Dovrolis, 2022), others incorporate lightweight auxiliary modules to enable the dynamic selection of corresponding parameters during inference (Kim et al., 2022; Jin and Kim, 2022).

2.2 Reinforcement Learning (RL)

The alignment of LLMs with complex, non-differentiable human objectives is increasingly framed as an RL problem. Unlike supervised learning, which requires explicit input-output pairs, RL is uniquely suited for scenarios where the objective is defined by sparse or qualitative feedback. PPO (Schulman et al., 2017) provides an actor-critic framework, which requires a reward model and a value function for advantage estimation. Moreover, PPO utilizes a surrogate objective with a clipping mechanism to prevent large policy updates. DPO (Rafailov et al., 2023) allows the model to be trained directly on preference pairs using a simple binary cross-entropy loss. KTO (Ethayarajh et al., 2024) leverages binary feedback signals for training and exhibits a higher sensitivity to negative samples. GRPO (Shao et al., 2024) eliminates the need for a critic model by averaging rewards from a group of multiple outputs generated for the same input. GSPO (Zheng et al., 2025) shifts the focus to the sequence level to improve GRPO. It defines the importance sample ratio based on sequence rather than tokens.

2.3 Parameter-Efficient Fine-Tuning (PEFT)

While scaling pre-trained models has consistently yielded performance gains, the traditional full-tuning method becomes infeasible with limited compute resources due to the massive number of parameters. To mitigate the prohibitive costs of full-parameter updates in LLMs, Parameter-Efficient Fine-Tuning (PEFT) has been widely adopted as a resource-efficient strategy for down-

stream task adaptation. Prompt-Tuning (Lester et al., 2021) freezes the backbone model and updates "soft prompts" embedding during training. Prefix-Tuning (Li and Liang, 2021) prepends learnable prefix vectors to the hidden states of every Transformer layer. P-Tuning (Liu et al., 2022) employs a prompt encoder to model the dependencies between prompt tokens. LoRA(Hu et al., 2022) adopts a low-rank decomposition strategy, and only these auxiliary low-rank branches are optimized to approximate the required weight updates. Many pre-training-based IL methods leverage low-rank adaptation to address downstream tasks with minimal trainable parameters, significantly enhancing training efficiency while conserving computational resources (Wang et al., 2023a; Chen et al., 2024; Yang et al., 2025b).

3 Methodology

In this section, we first establish a generalized formulation of the incremental learning problem at the instance level, focusing on task-free and blurred-boundary scenarios. We then introduce the concept of adaptive weights and identify the pivotal challenges inherent in optimization. Finally, we provide a novel insight that motivates our approach and detail the technical implementation of the proposed method.

3.1 Problem Formulation

General Instance-level IL Paradigm

In real-world scenarios, task boundaries are often ambiguous, and data distributions evolve continuously over time. To capture this complexity, we formalize a general instance-level incremental learning framework that includes both task-free and blurred-boundary settings.

Consider a sequential data stream $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_T\}$, where $\mathcal{D}_t$ represents the data source of t-th task over the input space $\mathcal{X}_t$ and label space $\mathcal{Y}_t$. Notably, we adopt a general definition of tasks, where a task can even be defined as the data within a single time window or step, with no pre-defined semantic boundaries. This formulation naturally encompasses scenarios ranging from discrete task divisions to task-free settings with blurred boundaries.

Meanwhile, we further denote a pre-trained backbone $\boldsymbol{W}_0$, an expert parameter pool $\mathcal{W} = \{\Delta\boldsymbol{W}_1, \Delta\boldsymbol{W}_2, \ldots, \Delta\boldsymbol{W}_n\}$, and a weight space $\mathcal{A} = \{\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \mid \text{condition}\} \subseteq \mathbb{R}^n$. For input $\boldsymbol{x}$ and label $\boldsymbol{y}$ in the stream data $\mathcal{D}$, the objective of instance-level IL can be written as follows:

$$\begin{aligned} &\min \mathcal{L}(\mathcal{D}; \mathcal{W}, \mathcal{A}) \\ &= \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_t} \left[\ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}; \mathcal{W}, \boldsymbol{\alpha}), \boldsymbol{y})\right], \end{aligned} \quad (1)$$

where $\ell(\cdot, \cdot)$ denotes the cross-entropy loss function, and $f_{\boldsymbol{\theta}}(\boldsymbol{x}; \mathcal{W}, \boldsymbol{\alpha})$ denotes for a given adapter layer, the output $\boldsymbol{h}$ is:

$$\boldsymbol{h} = (\boldsymbol{W}_0 + \sum_{i=1}^{n} \alpha_i \Delta\boldsymbol{W}_i)\boldsymbol{x} \quad (2)$$

Existing approaches to IL follow two dominant paradigms. Regularization-based methods optimize a shared expert or an expert pool by penalizing parameter deviations across tasks to mitigate catastrophic forgetting. Architecture-based methods deploy isolated experts and update specific modules, typically requiring an explicit task ID during inference. While effective in their respective settings, these methods rely on static weight allocation, thus failing to adapt to instance-level variations in data distributions. In contrast, we propose a dynamic weight optimization perspective. Rather than treating expert combination as a predetermined or task-dependent operation, our method learns to adaptively route and merge experts for each input instance.

Adaptive Weights Instance-level IL Paradigm

The primary challenges lie in the joint optimization of the expert pool and the weight space. We observe that the isolation of task-specific parameters limits KT, while the continual update of task-sharing parameters leads to CF. Consequently, we shift our focus toward maximizing the growing expert pool's potential by optimizing the weight space, ensuring experts adaptively coordinate for each instance. In this perspective, the objective can be written as follows:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathcal{L}(\mathcal{D}; \mathcal{W}, \pi_{\boldsymbol{\phi}}(\mathcal{D})), \quad (3)$$

where $\pi_{\boldsymbol{\phi}}$ denotes the adaptive selection policy network with trainable parameters $\boldsymbol{\phi}$ at the instance level. For each input instance $\boldsymbol{x} \in \mathcal{D}$, this policy generates the corresponding weight vector $\boldsymbol{\alpha} = \pi_{\boldsymbol{\phi}}(\boldsymbol{x})$, determining how to merge the expert parameters in $\mathcal{W}$.

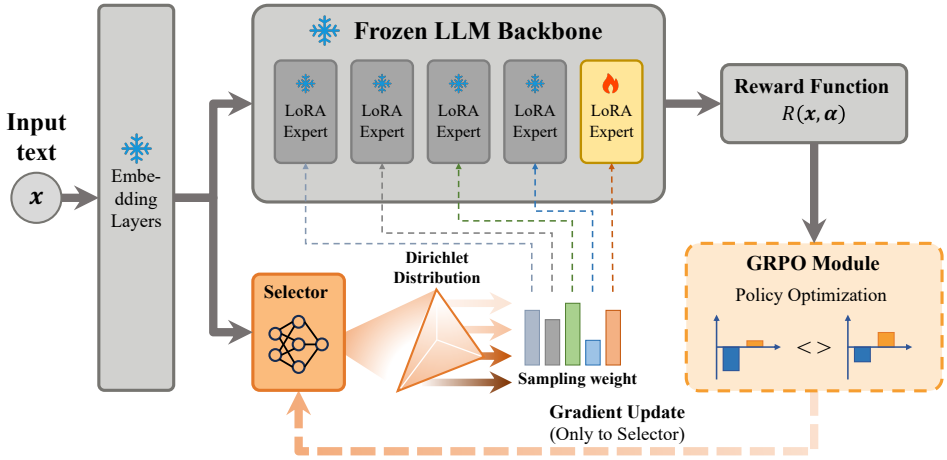


Figure 1: **The overall framework of GROLE.** The method operates in two stages: training task-specific LoRA experts via SFT (represented by the light yellow expert) and subsequently learning an instance-level merging policy via RL (represented by the orange selector).

A naive approach to learning adaptive merging weights $\boldsymbol{\alpha}$ is to treat them as standard learnable parameters, either by jointly optimizing $\boldsymbol{\alpha}$ with the base model or by generating them through a lightweight network during SFT. However, this strategy suffers from optimization instability due to the lengthy backpropagation path, where the gradients for $\boldsymbol{\alpha}$ must traverse all layers of the LLM, often leading to vanishing or unstable signals (Huang et al., 2024). To circumvent this issue, we propose a gradient-free alternative based on RL, which leverages a scalar reward derived from the model's final output to guide the optimization of $\boldsymbol{\alpha}$, thereby entirely avoiding error backpropagation through the LLM. The proposed GROLE framework primarily focuses on exploring the weight space and reformulating the objective Eq. (3) into an optimizable form.

3.2 Group Relative Optimization for LoRA Experts

We propose a two-stage method, GROLE, as illustrated in Figure 1. In the first stage, we incrementally learn a pool of task-specific LoRA experts via SFT. And in the second stage, we freeze the LLM backbone and all LoRA experts' parameters to train an adaptive merging weight selector via RL. Specifically, the selector's output is utilized for exploratory sampling during training, whereas it is normalized to directly merge LoRA experts during inference. The following sections detail our methodology, centering on the training of the selector.

Task-Specific LoRA Experts via SFT

In the first stage, we incrementally construct a set of LoRA experts to capture task-specific knowledge. Specifically, at each time step $t \in \{1, \ldots, T\}$, a new expert is instantiated for the current task D_t and added to the existing set of experts $\{\Delta \boldsymbol{W}_1, \ldots, \Delta \boldsymbol{W}_{t-1}\}$ trained on previous tasks. Given a pre-trained backbone model with frozen parameters $\boldsymbol{W}_0$, each LoRA expert i is parameterized by two low-rank matrices $\boldsymbol{A}_i \in \mathbb{R}^{d \times r}$, $\boldsymbol{B}_i \in \mathbb{R}^{r \times d}$ and $\Delta \boldsymbol{W}_i = \boldsymbol{A}_i \boldsymbol{B}_i$. While Eq. (1) permits n to differ from T, we consider the case where $n = T$ for simplicity, assigning a dedicated LoRA expert to each task. For each task t, the corresponding expert $(\boldsymbol{A}_t, \boldsymbol{B}_t)$ is optimized via SFT on D_t, while keeping both W_0 and all preceding experts $\{\Delta \boldsymbol{W}_1, \ldots, \Delta \boldsymbol{W}_{t-1}\}$ frozen. Upon completing the incremental training process, the full parameter set $\mathcal{W} = \{\Delta \boldsymbol{W}_1, \ldots, \Delta \boldsymbol{W}_T\}$ is frozen and serves as the basis for the subsequent instance-level merging stage.

Adaptive Merging Weight Selection via RL

In the second stage, we employ RL to train a lightweight selector π_ϕ that predicts instance-level merging weights $\boldsymbol{\alpha}$. To ensure compatibility, the raw input $\boldsymbol{x}$ is vectorized into $\boldsymbol{x}_{emb}$ through the embedding layers of the base model. In our RL formulation, $\boldsymbol{x}_{emb}$ serves as the state, while the

weight $\boldsymbol{\alpha}$ is defined as the action, which governs the adaptive merging of the frozen LoRA experts. Specifically, we design a sampling strategy over the selector output for exploration and exploitation during training, a reward function based on the pre-trained model and LoRA experts, and a group relative optimization objective that maximizes the expected reward without relying on gradient back-propagation through the LLM.

Sampling Strategy: To enable the selector to learn effective merging weights, we design a sampling strategy that allows it to thoroughly explore the weight space during training. First we define the weight space as the probability simplex $\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{R}^n \mid \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0\}$, which ensures stable and well-behaved merging of LoRA experts. Inspired by Lee et al. (2023), we denote the selector output $\pi_{\boldsymbol{\phi}}(\boldsymbol{x}_{emb})$ as concentration $\boldsymbol{c}$ and employ a Dirichlet distribution $\text{Dir}(\boldsymbol{c})$ for sampling. The Dirichlet distribution possesses two desirable properties. First, the sampling weight vector $\boldsymbol{\alpha} \sim \text{Dir}(\boldsymbol{c})$ naturally adhere to the probability simplex constraints, such that $\sum_{i=1}^{n} \alpha_i = 1$ and $\alpha_i \geq 0$. Second, $\boldsymbol{\alpha}$ is controlled by concentration $\boldsymbol{c}$: lower concentrations yield more dispersed samples, while higher concentrations produce samples concentrated near the mean $\boldsymbol{c}/||\boldsymbol{c}||_1$. During training, the selector's output concentration inherently emerges as small initially to foster exploration, and gradually increases to maximize exploitation as the model converges. Compared to traditional grid search, the Dirichlet distribution naturally balances exploration and exploitation while inherently satisfying the simplex constraints of the weight space.

Reward Function: In RL, the alignment between the reward model and the optimization objective is critical for performance. To address this, we adopt a streamlined reward function design: the SFT model trained in the first stage serves directly as the reward model. This ensures that the reward signal is fully aligned with the downstream objective. Specifically, the reward for an action $\boldsymbol{\alpha}$ given state x is defined as:

$$R(\boldsymbol{x}, \boldsymbol{\alpha}) = -\text{clip}(\ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}; \mathcal{W}, \boldsymbol{\alpha}), \boldsymbol{y}), \delta_1, \delta_2), \quad (4)$$

where $\ell(\cdot)$ denotes the cross-entropy loss function and y represents the ground-truth label. To enhance training stability, we employ reward clipping, bounded by $[\delta_1, \delta_2]$, to constrain the reward signal within a more concentrated numerical range. This is necessitated by the fact that raw loss values across different samples can fluctuate by several orders of magnitude, which often leads to gradient instability and impedes convergence. Meanwhile, this clipping mechanism allows the model to bypass the influence of trivial samples ($loss < \delta_1$) or intractable samples ($loss > \delta_2$), effectively stabilizing the optimization process.

Optimization Objective: The optimization objective of the selector follows the GRPO framework (Shao et al., 2024), which streamlines the training process by eliminating the traditional value-critic function. Instead of estimating absolute state values, we compute the advantage of an action by comparing its reward against the collective performance of a group. For each input, we sample a group of G outputs from the current policy $\pi_{\boldsymbol{\phi}}$. Specifically, the relative advantage is computed as $A_j = \frac{r_j - \text{mean}(\boldsymbol{r})}{\text{std}(\boldsymbol{r})}$.

The objective function is formulated to maximize the group relative reward, where the advantage for each sample is derived by normalizing its reward against the mean and standard deviation of all rewards within the group. Our objective can be written as follows:

$$\mathcal{J}_{GRPO}(\boldsymbol{\phi}) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{[(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}_t, \{\boldsymbol{\alpha}_j\}_{j=1}^{G}\sim\pi_{\boldsymbol{\phi}}(\mathcal{A}|\boldsymbol{x})]}$$
$$\frac{1}{G}\sum_{j=1}^{G} \min\left[\rho_j A_j, \text{clip}(\rho_j, 1-\epsilon, 1+\epsilon) A_j\right], \quad (5)$$

where $\rho_j = \pi_{\boldsymbol{\phi}}(\boldsymbol{\alpha}_j \mid \boldsymbol{x}) / \pi_{\boldsymbol{\phi}_{old}}(\boldsymbol{\alpha}_j \mid \boldsymbol{x})$ denotes the importance sample ratio, G denotes the group size, and ϵ denotes the clipping hyperparameter. Additionally, due to the random initialization of the selector and the lack of reference, we omit the KL-divergence term in the standard GRPO loss.

Eq. (5) successfully reformulates the original objective in Eq. (3) into an optimizable surrogate. By maximizing $\mathcal{J}_{GRPO}(\boldsymbol{\phi})$, we can derive an instance-level adaptive weight selector $\pi_{\boldsymbol{\phi}}$.

4 Experiments

In this section, we describe the experimental setup and present the main results in comparison with existing mainstream approaches.

4.1 Experimental Setup

Dataset and Partition

The proposed method is evaluated on the following two widely used benchmarks:

- **Standard CL Benchmark** (Zhang et al., 2015) consists of five text classification datasets: AG News, Amazon reviews, Yelp reviews, DBpedia, and Yahoo Answers.
- **Large Number of Tasks** (Razdaibiedina et al., 2023) consists of 15 datasets, including five datasets from the standard CL benchmark, four from GLUE benchmark (MNLI, QQP, RTE, SST2) (Wang et al., 2018), five from SuperGLUE benchmark (WiC, CB, COPA, MultiRC, BoolQ) (Wang et al., 2019), and the IMDB movie reviews dataset (Maas et al., 2011).

To faithfully simulate the blurred-boundary scenarios inherent in real-world online environments—where data streams often encompass a concurrent mixture of samples from multiple tasks—we employ two distinct partitioning strategies (Zeng et al., 2023) to simulate realistic task sequences:

- **Shards:** Decompose each task into k disjoint shards and re-aggregate them randomly to construct a fragmented task sequence, written as Srd-k.
- **Dirichlet:** Sample a probability vector from distribution Dirichlet(β) and assign samples to sequential task slots according to these proportions, written as Dir-β.

In our experiments, we instantiate three distinct partitioning configurations: Srd-1, Srd-4, and Dir-0.3. It is worth noting that Srd-1 represents a traditional sequential setup with a random task order, consistent with numerous prior studies such as O-LoRA (Wang et al., 2023a). In contrast, Srd-4 and Dir-0.3 introduce complex task overlaps, providing a more rigorous test of the model's robustness against blurred-boundary data streams. Considering the imbalance in sample sizes across different datasets, we curated a subset of 4 datasets for the standard CL benchmark and 12 for the large number of tasks to ensure distributional stability. More dataset and partition details are provided in Appendix A.1

Metrics

To evaluate the proposed method, we define **Average Accuracy** (**AA**) at the t-th training stage across the entire task sequence T. Let $a_{t,j}$ represent the testing accuracy on task j after the model has been trained on task t. To simultaneously capture both forward and backward transfer effects (Lopez-Paz and Ranzato, 2017), the metric is formulated as $AA_t = \frac{1}{T}\sum_{j=1}^{T} a_{t,j}$.

By evaluating over the full set of T tasks at each step t, this metric serves a dual purpose as follows:

- For $j < t$, it quantifies CF by measuring the performance retention on previously learned tasks.
- For $j > t$, it quantifies KT by reflecting the model's ability to generalize to unseen future tasks based on its current knowledge.

Baselines

PerTaskLoRA trains an independent LoRA for each task to achieve optimal per-task performance, while **MTL** (Multi-Task Learning) trains a unified LoRA jointly optimizing over all available task data. These two non-incremental methods typically provide the performance upper bound for IL. **Replay** mitigates catastrophic forgetting by rehearsing a subset of historical samples retained in a memory buffer. **SeqLoRA** continuously fine-tunes a single LoRA module across the entire sequence, whereas **IncLoRA** progressively instantiates a dedicated LoRA expert for each new task, expanding the expert pool over time. These three serve as foundational IL baselines. **O-LoRA** (Wang et al., 2023a) learns new tasks in orthogonal low-rank subspaces without data replay. **MultiLoRA** (Wang et al., 2023b) identifies and mitigates the over-dominance of specific unitary transforms in LoRA's weight updates. **MoELoRA** (Chen et al., 2024) employs a task-ID-guided gating mechanism to modulate expert contributions, while **MTL-LoRA** (Yang et al., 2025b) relies on ID-dependent parameters to isolate task-specific features. We primarily focus our comparison on these four baselines. More implementation details are provided in Appendix A.2

4.2 Main Results

Overall Performance: We evaluate the performance of GROLE against comprehensive baselines across six diverse task scenarios, encompassing two benchmarks and three partitioning strategies. As shown in Table 1, on the standard CL benchmark, our method achieves a 4.32% improvement over the best baseline. This advantage is even more pronounced on the large number of tasks, where the performance gain reaches 9.47%. GROLE also exhibits consistent performance across different partition strategies, illustrating the selector's efficacy in adapting to complex, blurred-boundary tasks.

Table 1: Overall performance comparison on the standard CL benchmark and large number of tasks. The average accuracy (AA_T) after training on the final task is reported.

	Standard CL Benchmark				Large Number of Tasks			
	Srd-1	**Srd-4**	**Dir-0.3**	**Avg**	**Srd-1**	**Srd-4**	**Dir-0.3**	**Avg**
Replay	72.70	72.32	72.78	72.60	62.44	76.54	77.08	72.02
SeqLoRA	69.25	73.78	72.92	71.98	71.92	73.23	75.10	73.42
IncLoRA	73.08	55.70	49.55	59.44	65.60	41.15	33.40	46.72
O-LoRA	73.65	73.45	68.15	71.75	72.69	57.77	44.50	58.32
MultiLoRA	66.90	70.10	69.60	68.87	74.08	70.98	70.21	71.76
MoELoRA	70.13	75.30	66.63	70.68	76.35	58.00	67.52	67.29
MTL-LoRA	74.78	76.60	70.68	74.02	72.23	73.13	64.98	70.11
GROLE	**78.10**	**78.83**	**78.08**	**78.34**	**82.79**	**82.63**	**83.25**	**82.89**
PerTaskLoRA	77.60	77.60	77.10	77.43	85.17	82.73	82.44	83.45
MTL	77.75	77.75	77.75	77.75	83.90	83.90	83.90	83.90

Furthermore, against established upper bounds (PerTaskLoRA and MTL), GROLE surpasses the standard CL benchmark by 0.6% while remaining highly competitive on the large number of tasks, trailing by a narrow margin of 1%. Collectively, these results confirm GROLE's capacity to simultaneously mitigate CF and facilitate robust KT.

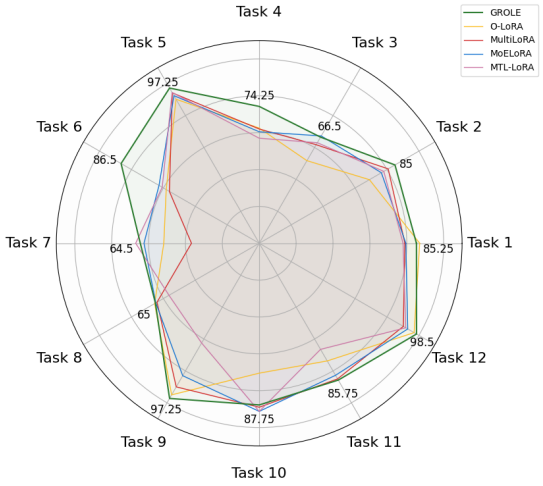
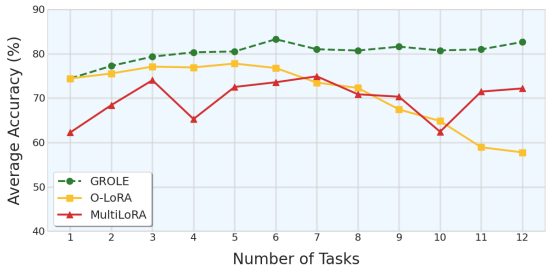
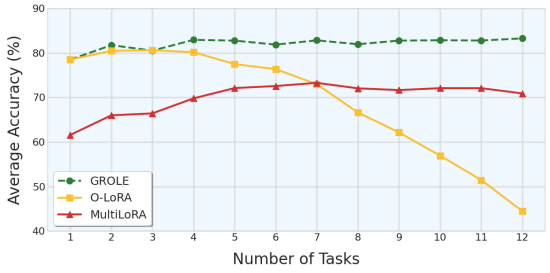


Figure 2: Per-task performance ($a_{T,j}$) visualization on the large number of tasks with Srd-1.

Per-Task Performance: To further analyze the effectiveness of GROLE on individual tasks, we visualize per-task testing accuracy $a_{T,j}$ on a large number of tasks benchmarked with Srd-1 after training the final task. This allows for a detailed comparison of performance across individual task. As shown in Figure 2, our approach not only achieves higher average performance but also maintains consistently strong results across nearly every task. Notably, while MTL-LoRA marginally outperforms GROLE by 2.50% on Task 7, it suffers a drastic performance decay of 34.75% on Task 9. This pattern of instability is prevalent among other baselines, further underscoring the challenge of maintaining task-level balance. In contrast, GROLE achieves a superior trade-off, demonstrating both formidable overall performance and consistent robustness across diverse tasks.



(a) Srd-4



(b) Dir-0.3

Figure 3: Visualization of negative transfer on the large number of tasks.

Analysis of Negative Transfer: Focusing on task-free baselines in Table 1(thereby excluding ID-dependent MoELoRA and MTL-LoRA), methods like O-LoRA suffer sharp declines under blurred boundaries. Tracking the AA_t evolution in Figure 3 reveals that while orthogonal regularization initially mitigates forgetting, it impairs generalization

in scenarios with high inter-task overlap. Specifically, enforcing strict orthogonality on shared features isolates knowledge into disjoint subspaces, preventing effective representation reuse. This rigidity hinders cross-task synergies, directly leading to negative transfer. Conversely, MultiLoRA suffers from training instability due to implicit competition among unrouted modules and sensitivity to scaling initialization. The 'dominance of top singular vectors' (Wang et al., 2023b) further aggravates these fluctuations, preventing robust convergence. In contrast, GROLE demonstrates superior stability, achieving continuous positive transfer by effectively leveraging cross-task synergies without such rigid constraints.

Table 2: Zero-shot generalization performance on the Out-of-Distribution task.

	Standard CL Benchmark			
	Srd-1	**Srd-4**	**Dir-0.3**	**Avg**
O-LoRA	**70.80**	61.80	68.20	66.93
MultiLoRA	58.10	56.30	39.60	51.33
GROLE	70.10	**70.40**	**71.00**	**70.50**
MTL	55.90	55.90	55.90	55.90

Robustness to OOD Tasks: While our primary metric the average accuracy AA_t over the T sequential tasks provides a measure of KT, it may still reflect performance on tasks implicitly observed during training, especially under blurred boundaries where data distributions overlap. To better assess generalization, we hold out one task *AG News* from the standard CL benchmark as out-of-distribution (OOD) task, entirely during training and evaluate the model on it only at test time. As shown in Table 2 (where ID-dependent methods are inapplicable), GROLE outperforms the best baseline by 3.60% and maintains consistent performance across diverse partitioning strategies. This result indicates that our method not only excels at mitigating CF and facilitating KT but also possesses robust OOD generalization capability, demonstrating that GROLE fully unleashes the potential of all experts and exhibits adaptability to unseen tasks.

Impact of Group Size: The group size defines the exploration scope in the weight space, determining how many candidate weight vectors are considered for each instance. Intuitively, this hyperparameter may affect model performance. To assess this impact, we evaluate GROLE with group sizes $\{4, 8, 16, 24, 32\}$ and compare their performance in terms of AA_T, where 16 is the value selected for our main experiments. Results indicate that performance with different group sizes exhibits minimal variation, fluctuating by only about 0.6% between the best ($G = 16$) and the worst ($G = 8$), which shows the performance is not sensitive to the choice of group size and further highlights the robustness of GROLE.

Table 3: Impact of the sampling group size G on performance (AA_T) across the Standard CL Benchmark.

Group Size	**Standard CL Benchmark**			
	Srd-1	**Srd-4**	**Dir-0.3**	**Avg**
$G = 4$	77.63	78.15	78.03	77.94
$G = 8$	77.60	77.63	77.95	77.73
$\boldsymbol{G = 16}$	78.10	**78.83**	78.08	**78.34**
$G = 24$	**78.15**	78.65	**78.20**	78.33
$G = 32$	77.70	77.80	78.10	77.87

Table 4: Performance with varying numbers of activated experts.

	Standard CL Benchmark			
	Srd-1	**Srd-4**	**Dir-0.3**	**Avg**
Top-1	76.83	76.95	76.25	76.68
Top-2	77.60	78.33	77.28	77.76
Top-all	**78.10**	**78.83**	**78.08**	**78.34**
	Large Number of Tasks			
	Srd-1	**Srd-4**	**Dir-0.3**	**Avg**
Top-1	81.31	78.85	80.63	80.26
Top-2	82.38	80.52	82.58	81.83
Top-3	**82.81**	82.15	83.06	82.67
Top-6	82.71	82.43	83.21	82.78
Top-all	82.79	**82.63**	**83.25**	**82.89**

Top-k Activation: To validate the effectivity of RL and the necessity of multi-expert merging, we apply top-k selection to the selector's predicted weights, activating only the top-k LoRA experts for merging. As shown in Table 4, results show that GROLE maintains strong performance even with only the highest-weighted expert activated. Moreover, as more experts are activated, the performance of GROLE is also consistently improving, validating the stability of selector.

Sampling Strategy: We compared Dirichlet sampling against Gaussian sampling during selector training. As shown in Table 5, Dirichlet sampling consistently outperforms Gaussian sampling across all benchmarks, with 0.81% average gains

Table 5: Comparison of two different sampling strategies.

	Standard CL Benchmark			
	Srd-1	**Srd-4**	**Dir-0.3**	**Avg**
Gaussian	77.07	77.65	77.88	77.53
Dirichlet	**78.10**	**78.83**	**78.08**	**78.34**
	Large Number of Tasks			
	Srd-1	**Srd-4**	**Dir-0.3**	**Avg**
Gaussian	79.35	80.81	82.96	81.04
Dirichlet	**82.79**	**82.63**	**83.25**	**82.89**

on standard CL benchmark and 1.85% average gains on the large number of tasks, confirming the effectiveness of Dirichlet sampling. This superiority stems from the Dirichlet distribution's inherent properties: its support over the simplex naturally produces sparse, interpretable weight vectors that balance exploration and exploitation in searching of weight space.

5 Conclusion

In this work, we propose GROLE, a novel incremental learning framework that addresses the challenges of task-free and blurred-boundary scenarios. GROLE maintains a growing pool of frozen, task-specific LoRA experts and incorporates a lightweight selector optimized by group relative advantages. This two-stage paradigm ensures strict parameter isolation to eliminate catastrophic forgetting and achieves strong knowledge transfer across tasks through adaptively merging experts at the instance level. Specifically, this modular architecture facilitates the systematic expansion of the expert library, while the selector ensures the system's agility to navigate evolving scenarios, thereby utilizing the growing expert pool while preserving prior knowledge. The experimental results demonstrate that GROLE achieves a balance between stability and plasticity on existing benchmarks, making it eminently suitable for incremental learning within complex, open-world environments.

Limitations

Despite the performance gains achieved by GROLE, it faces a common challenge with many IL approaches: degradation of training and inference efficiency as the number of tasks T scales. A growing pool of LoRA experts may introduce functional redundancy, which enlarges the search space and complicates the optimization of the selector. Consequently, developing mechanisms to prune or consolidate the expert pool, thereby minimizing capability overlap and compressing the weight space, remains a critical and promising direction for future research.

References

Cheng Chen, Junchen Zhu, Xu Luo, Heng Tao Shen, Jingkuan Song, and Lianli Gao. 2024. CoIN: A benchmark of continual instruction tuning for multimodel large language models. In *Advances in Neural Information Processing Systems 37, NeurIPS*, pages 57817–57840.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Model alignment as prospect theoretic optimization. In *Proceedings of the 41st International Conference on Machine Learning, ICML*.

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.

Mustafa Burak Gurbuz and Constantine Dovrolis. 2022. NISPA: neuro-inspired stability-plasticity adaptation for continual learning in sparse networks. In *Proceedings of the 39th International Conference on Machine Learning, ICML*, volume 162, pages 8157–8174.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *Proceedings of the 10th International Conference on Learning Representations, ICLR*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. LoraHub: Efficient cross-task generalization via dynamic lora composition.

Hyundong Jin and Eunwoo Kim. 2022. Helpful or harmful: Inter-task association in continual learning. In *Proceedings of the 17th European Conference on*

Computer Vision ECCV, volume 13671, pages 519–535.

Zixuan Ke and Bing Liu. 2022. Continual learning of natural language processing tasks: A survey. *Preprint*, arXiv:2211.12701.

Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, and Bing Liu. 2022. A theoretical study on solving continual learning. In *Advances in Neural Information Processing Systems 35 NeurIPS*.

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.

Minseop Lee, Sanghyeon Lee, Yeonghwan Jeon, Hyuncheol Jo, and Byoung-Ki Jeon. 2023. Optimizing video recommender system with bandit-based ensemble from online user action. In *Proceedings of the AAAI 2024 EcoSys Workshop*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 3045–3059.

Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing continuous prompts for generation. *Preprint*, arXiv:2101.00190.

Zhizhong Li and Derek Hoiem. 2016. Learning without forgetting. In *Proceedings of 14th European Conference on Computer Vision, ECCV*, pages 614–629.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 61–68.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems 30, NeurIPS*, pages 6467–6476.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations, ICLR*.

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.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. GPT-4 technical report. *Preprint*, arXiv:2303.08774.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36, NeurIPS*, pages 53728–53741.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *Proceedings of the 11th International Conference on Learning Representations, ICLR*.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition, CVPR*, pages 2001–2010.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. 2019. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *Proceedings of the 7th International Conference on Learning Representations, ICLR*.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Sebastian Thrun. 1995. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems 8, NeurIPS*, pages 640–646.

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. In *Advances in Neural Information Processing Systems 32, NeruIPS*, volume 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.

Chenxu Wang, Yilin Lyu, Zicheng Sun, and Liping Jing. 2025. Continual gradient low-rank projection fine-tuning for LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics, ACL*, pages 14815–14829.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383.

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. In *Findings of the Association for Computational Linguistics, EMNLP*, pages 10658–10671.

Yiming Wang, Yu Lin, Xiaodong Zeng, and Guannan Zhang. 2023b. MultiLoRA: Democratizing lora for better multi-task learning. *Preprint*, arXiv:2311.11501.

Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. 2018. Memory Replay GANs: Learning to generate new categories without forgetting. In *Advances in Neural Information Processing Systems 31 NeurIPS*, pages 5966–5976.

Mengqi Xue, Haofei Zhang, Jie Song, and Mingli Song. 2022. Meta-attention for vit-backed continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 150–159.

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. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Yaming Yang, Dilxat Muhtar, Yelong Shen, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Weiwei Deng, Feng Sun, Qi Zhang, and 1 others. 2025b. MTL-LoRA: Low-rank adaptation for multi-task learning. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence, AAAI*, 20, pages 22010–22018.

Dun Zeng, Siqi Liang, Xiangjing Hu, Hui Wang, and Zenglin Xu. 2023. FedLab: A flexible federated learning framework. *Journal of Machine Learning Research*, 24(100):1–7.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28, NeurIPS*.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. Group sequence policy optimization. *Preprint*, arXiv:2507.18071.

A Appendix

A.1 Dataset and Partition Details

Since the imbalance sample sizes across different datasets shown in Table 6, for the standard CL benchmark, we select four tasks {"Yelp", "Amazon", "Dbpedia", "Yahoo"} as $\{T_1, \dots T_4\}$ and {"AG News"} as OOD task. For the large number of tasks, we select {"Yelp", "Amazon", "MNLI", "QQP", "IMDB", "SST-2", "Dbpedia", "AG News", "Yahoo", "MultiRC", "BoolQA", "WiC"} as $\{T_1, \dots T_{12}\}$. As shown in Table 7 and Table 9, we list the details of different tasks.

Moreover, we randomly select 4000 samples as a trainset for LoRA experts, 1000 samples as a subtrainset for the selector, and 1000 samples as a testset for the standard CL benchmark, while 1600 samples as a trainset for LoRA experts, 400 samples as a subtrainset for the selector, and 400 samples are a testset for a large number of tasks. As shown in Figure 4, we visualize the trainset of two benchmarks with three different partition strategies.

A.2 Implementation Details

Our experiments are deployed with 4 NVIDIA H800 GPUs. We employ Llama3.1-8B (Grattafiori et al., 2024) as the backbone model. For training the LoRA experts, we adopt the AdamW (Loshchilov and Hutter, 2019) optimizer with a cosine learning rate scheduler, setting the initial learning rate to 1×10^{-4}. The LoRA configuration includes a rank of $r = 8$ and is applied to all available modules. We train the LoRA modules for 3 epochs with a per-device batch size of 1. For the selector module, we implement a three-layer MLP with hidden dimensions of $[256, 64]$. We employ ReLU as the internal activation function and apply Softplus at the output layer. During training, the

Table 6: Size of different datasets.

	Dbpedia	Yahoo	Amazon	Yelp	AG News	MNLI	QQP	IMDB	MultiRC	BoolQA	SST-2	WiC	RTE	COP	CB
Train Size	14000	10000	5000	5000	4000	3000	2000	2000	2000	2000	2000	2000	2000	400	250
Test Size	7600	7600	7600	7600	7600	7600	7600	7600	4848	3270	872	638	277	100	56

Table 7: Datasets, Categories, Tasks, Domains, and Metrics.

Dataset Name	Category	Task	Domain	Metric
Yelp	CL Benchmark	Sentiment analysis	Yelp reviews	Accuracy
Amazon	CL Benchmark	Sentiment analysis	Amazon reviews	Accuracy
Dbpedia	CL Benchmark	Topic classification	Wikipedia	Accuracy
Yahoo	CL Benchmark	Topic classification	Yahoo Q&A	Accuracy
AG News	CL Benchmark	Topic classification	News	Accuracy
MNLI	GLUE	NLI	Various	Accuracy
QQP	GLUE	Paragraph detection	Quora	Accuracy
RTE	GLUE	NLI	News, Wikipedia	Accuracy
SST-2	GLUE	Sentiment analysis	Movie reviews	Accuracy
WiC	SuperGLUE	Word sense disambiguation	Lexical databases	Accuracy
CB	SuperGLUE	NLI	Various	Accuracy
COPA	SuperGLUE	QA	Blogs, encyclopedia	Accuracy
BoolQA	SuperGLUE	Boolean QA	Wikipedia	Accuracy
MultiRC	SuperGLUE	QA	Various	Accuracy
IMDB	SuperGLUE	Sentiment analysis	Movie reviews	Accuracy

selector is optimized using the AdamW optimizer with a learning rate of 1×10^{-3} and a weight decay of 1×10^{-4}. We set the dropout rate to 0.3, using a batch size of 1 and a group size of 16. The training epochs are configured as 1 for the standard CL benchmark and 10 for the large number of tasks. We employ the cross-entropy loss and set the reward clipping parameters to $\delta_1 = 0.1, \delta_2 = 1$.

A.3 Supplyment

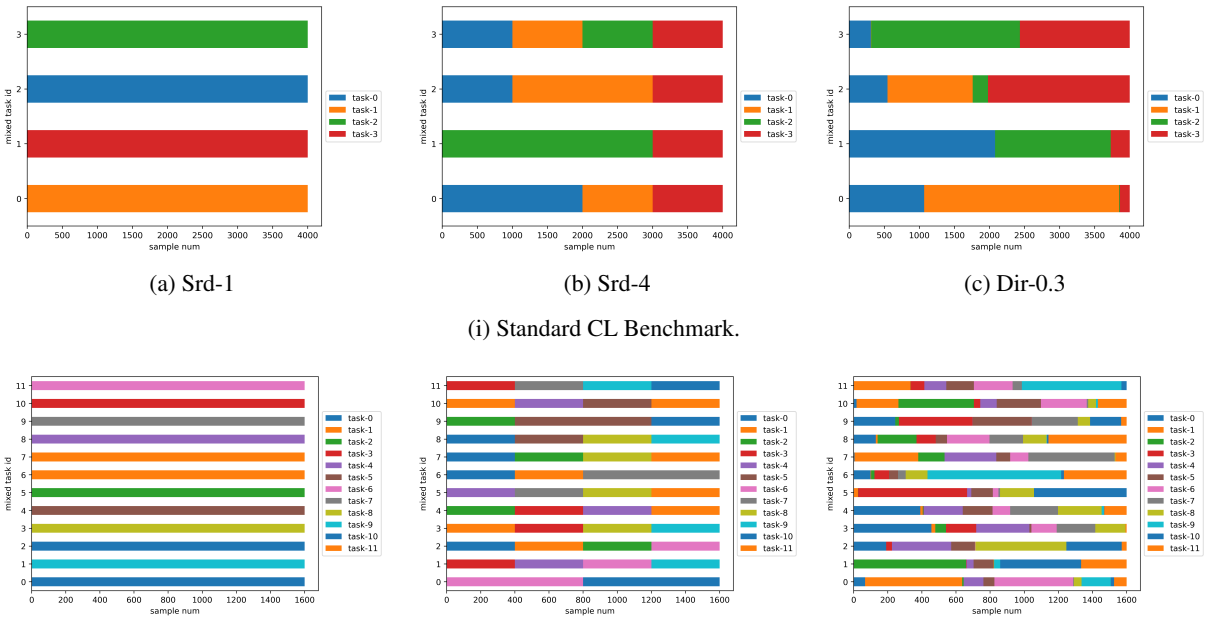
Table 8: GPU hours (minutes per epoch) of two stage training: LoRA experts and selector.

	Standard CL Benchmark			
	Srd-1	**Srd-4**	**Dir-0.3**	**Avg**
Stage 1	15.80	16.12	15.91	15.94
Stage 2	6.88	6.87	6.89	6.88
	Large Number of Tasks			
	Srd-1	**Srd-4**	**Dir-0.3**	**Avg**
Stage 1	19.47	22.52	22.13	21.37
Stage 2	18.12	18.18	18.06	18.12

Training Efficiency: As a gradient-free method based on RL, GROLE not only circumvents the instability issues associated with LLM gradient back-propagation, but also achieves reasonable training efficiency in selector optimization. Furthermore, despite operating at the instance level, our selector fully supports batched training and inference, which further reduces training elapse. As shown in Table 8, stage 2 (selector training) requires less than half the time of stage 1 (LoRA experts training) on the standard CL benchmark, demonstrating the computational efficiency of training selector.

Table 9: Instructions for different tasks.

Task	Instructions
NLI	What is the logical relationship between the "sentence 1" and the "sentence 2"? Choose one of the options.
QQP	Whether the "first sentence" and the "second sentence" have the same meaning? Choose one of the options.
SC	What is the sentiment of the following paragraph? Choose one of the options.
TC	What is the topic of the following paragraph? Choose one of the options.
BoolQA	According to the following passage, is the question true or false? Choose one of the options.
MultiRC	According to the following passage and question, is the candidate's answer true or false? Choose one of the options.
WiC	Given a word and two sentences, whether the word is used with the same sense in both sentences? Choose one of the options.



(a) Srd-1 (b) Srd-4 (c) Dir-0.3

(i) Standard CL Benchmark.

(d) Srd-1 (e) Srd-4 (f) Dir-0.3

(ii) Large Number of Tasks.

Figure 4: Visualization of trainset distribution on two benchmarks with three partition strategies.