McLoRA: Multiplicative Composition for LoRA-Based Continual Learning

Anonymous authors
Paper under double-blind review

Abstract

Low-Rank Adaptation (LoRA) suffers from catastrophic forgetting in continual learning. Existing methods for addressing this problem still retain the standard additive composition, in which the LoRA update does not receive direct guidance from the backbone output. This paper proposes McLoRA (Multiplicative Composition LoRA), which replaces the standard additive composition with element-wise multiplicative composition between the low-rank branch and the frozen backbone output. This design makes the learning gradient of each output dimension in the low-rank branch explicitly correlated with the output magnitude of the corresponding backbone dimension, thereby encouraging updates to focus on dimensions that respond more strongly to the current task while suppressing updates on weakly activated dimensions. Empirically, McLoRA produces more concentrated updates with less cross-task interference. Experiments on multiple large language models and sequential learning benchmarks show that, under the same parameter budget and computational complexity as LoRA, McLoRA significantly alleviates catastrophic forgetting when a single shared LoRA module is trained sequentially, while maintaining comparable single-task performance.

1 Introduction

Low-Rank Adaptation (LoRA) (Hu et al., 2022) is a widely used parameter-efficient fine-tuning (PEFT) method (Houlsby et al., 2019; Li & Liang, 2021). It trains a lightweight low-rank branch with the backbone frozen, greatly reducing training costs while achieving performance competitive with or better than that of full fine-tuning. However, it suffers from catastrophic forgetting in continual/sequential learning.

Much work has been done to address the problem. A line of work maintains multiple independent LoRA modules and reduces inter-task interference by selecting task-relevant modules via routing or attention mechanisms, or by imposing mathematical constraints such as orthogonality (Duzhen Zhang et al., 2025; Jialin Liu et al., 2024; Wang et al., 2023). These methods effectively suppress forgetting, but the number of parameters grows approximately linearly with the number of tasks. Another line introduces shared LoRA modules or shared low-rank subspaces across tasks and reduces forgetting through routing-matrix and orthogonality constraints within a shared adapter, task-shared/task-specific adapter decomposition, or shared-subspace updates that preserve past-task knowledge (Xin Zhang et al., 2025; He et al., 2025; Kaushik et al., 2026).

Yet all these methods retain the original additive paradigm of LoRA, in which the output of the low-rank branch is added to the backbone output. In this formulation, the learning signal of the LoRA branch is driven mainly by upstream error signals and its own projections. The backbone output does not explicitly appear in the gradient of the LoRA fine-tuning process; consequently, the updates lack direct guidance from the backbone output.

In fact, backbone output magnitudes constitute a useful task-dependent signal for allocating updates: dimensions with larger backbone output magnitudes tend to exhibit higher sensitivity to the current task. As illustrated in Figure 1, replacing high-magnitude backbone outputs with their calibration means causes a substantially larger loss increase than replacing random dimensions or low-magnitude ones. This observation confirms that backbone output magnitudes serve as a reliable indicator for guiding update allocation.

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

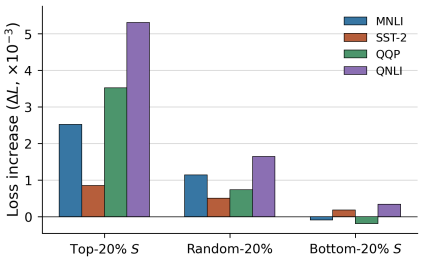


Figure 1: Loss changes on four tasks after intervening on three different dimension groups respectively. Results are based on RoBERTa-large with single-task LoRA. See Appendix A for details.

Consequently, an ideal mechanism ought to account for the per-dimension magnitudes of the backbone output and preferentially allocate updates to those dimensions that are more strongly activated by the current task, thereby rendering LoRA updates more efficient. Conversely, updating weakly activated dimensions yields marginal benefit to the current task and poses a higher risk of interfering with other tasks: dimensions that are weakly activated for the current task may be strongly activated for other tasks, and updating them therefore elevates the risk of catastrophic forgetting.

To this end, we propose McLoRA (Multiplicative Composition LoRA), which replaces additive composition with element-wise multiplication between the backbone output and the low-rank branch output. This multiplicative composition makes the learning gradient of each output dimension in the low-rank branch linearly correlated with the output magnitude of the corresponding backbone dimension. As a result, updates in the McLoRA branch are preferentially allocated to dimensions that respond more strongly to the current task. McLoRA naturally encourages updates to concentrate on dimensions that are more relevant to the current task, reducing unnecessary interference with dimensions that may be important for other tasks, thereby helping mitigate catastrophic forgetting.

We present both a theoretical analysis of McLoRA and quantitative investigations into its internal mechanism and statistical behavior. Experimental results on multiple sequential/continual learning benchmarks show that McLoRA with a single shared LoRA module consistently performs better than LoRA. Moreover, replacing additive composition with multiplicative composition can also improve representative additive LoRA variants. A bounded sigmoid-McLoRA extension remains effective. We additionally extend our approach to a multi-adapter sequential/continual learning protocol, where the benefit of multiplicative composition is preserved. Meanwhile, McLoRA remains comparable to LoRA in single-task fine-tuning.

In summary, our contributions are threefold:

We propose McLoRA, a multiplicative composition method in which the low-rank branch output is element-wise multiplied with the frozen backbone output. This design makes the gradient of each output dimension linearly correlated with the corresponding backbone output magnitude, enabling task-sensitive, dimension-wise update allocation.

Our quantitative analyses reveal that McLoRA produces more concentrated updates, allocates less energy to weakly activated dimensions, exhibits smaller cross-task overlap in update supports, and shows less interference with old-task-critical dimensions—providing mechanistic insights into its improved forgetting resistance.

We extensively validate McLoRA across diverse settings. It significantly mitigates catastrophic forgetting in the shared single-adapter continual learning setting while remaining competitive with LoRA in single-task fine-tuning. Beyond that, it improves additive LoRA variants (e.g., AdaLoRA, DoRA) and transfers effectively to multi-adapter protocols.

2 Related Work

LoRA. LoRA (Hu et al., 2022) is a leading PEFT method. Subsequent studies improve it via adaptive rank allocation (AdaLoRA (Zhang et al., 2023)), weight decomposition (DoRA (Shih-Yang Liu et al., 2024)), or weight-space multiplicative adaptation (LoRMA (Bihany et al., 2025),

LieRA (Si et al., 2025)). In contrast, McLoRA performs multiplicative modulation at the output level, making backbone output magnitudes an explicit factor in the low-rank gradient. Appendix B provides a comparison table of standard LoRA and various LoRA variants, showing the specific differences among them.

Multi-LoRA Methods for Continual Learning. Multi-LoRA methods maintain multiple independent LoRA modules and reduce inter-task interference through task-relevant module selection via routing or attention mechanisms, orthogonality-based constraints, and other mechanisms. AM-LoRA (Jialin Liu et al., 2024) and BranchLoRA (Duzhen Zhang et al., 2025) combine multiple LoRA modules with an attentional selector or router. O-LoRA (Wang et al., 2023) enforces orthogonality across task-specific low-rank subspaces. N-LoRA (Yang et al., 2025) identifies parameter collision as a key source of forgetting and sparsifies LoRA updates to encourage different tasks to modify different parameter locations. LB-CL (Qiao & Mahdavi, 2024) applies orthogonal gradient projection to reduce forgetting and cross-task interference.

Shared-LoRA Methods for Continual Learning. More recent work has begun to explore Shared-LoRA methods, which introduce shared LoRA modules or shared low-rank subspaces across tasks. C-LoRA (Xin Zhang et al., 2025) is a single adaptive LoRA with routing and orthogonality constraints. CL-LoRA (He et al., 2025) combines task-shared and task-specific adapters in a dual-adapter architecture. Share (Kaushik et al., 2026) learns and dynamically updates a shared low-rank LoRA subspace, where essential directions from new tasks are integrated into the evolving subspace while past-task knowledge is preserved to reduce catastrophic forgetting. SD-LoRA (Wu et al., 2025) decouples LoRA updates into normalized directions and learnable magnitudes, freezes previously learned directions, and re-optimizes their magnitudes while learning a new direction for each task. The accumulated directions form a cross-task reusable direction bank spanning a shared subspace, while new directions expand the bank with additional capacity for residual task-specific variation.

These methods improve compactness through cross-task sharing, but all of them retain the additive composition of standard LoRA. The backbone output does not explicitly appear in the gradients of the low-rank parameters, and updates lack direct guidance from the backbone output.

3 METHOD

This section first introduces McLoRA, and then compares it with standard LoRA quantitatively on the differences of the gradient updates.

3.1 MCLORA STRUCTURE AND TRAINING

The structure of McLoRA is illustrated in Figure 2. Let the backbone weight be W. Given input x, the backbone output is $h_{\text{base}} = Wx$. The McLoRA branch is defined by equation 1.

$$g(x) = \mathbf{1} + B(Ax), \tag{1}$$

where $A \in \mathbb{R}^{r\times d}$ and $B \in \mathbb{R}^{k\times r}$ are two low-rank matrices. B is initialized as an all-zero matrix, and A is initialized with Kaiming uniform initialization (He et al., 2015).

The final output is obtained by element-wise multiplication between the backbone output and the McLoRA branch, as shown in equation 2.

$$h_{\text{final}} = h_{\text{base}} \odot g(x) = (Wx) \odot (\mathbf{1} + B(Ax)). \tag{2}$$

Backpropagation. Let $\delta = \partial L/\partial h_{\text{final}}$. We have

$$\frac{\partial L}{\partial g} = \frac{\partial L}{\partial h_{\text{final}}}\frac{\partial h_{\text{final}}}{\partial g} = \delta \odot (Wx). \tag{3}$$

Backpropagating to low-rank parameters yields

$$\frac{\partial L}{\partial B} = (\delta \odot (Wx))(Ax)^\top, \qquad \frac{\partial L}{\partial A} = \big(B^\top(\delta \odot (Wx))\big)x^\top \tag{4}$$

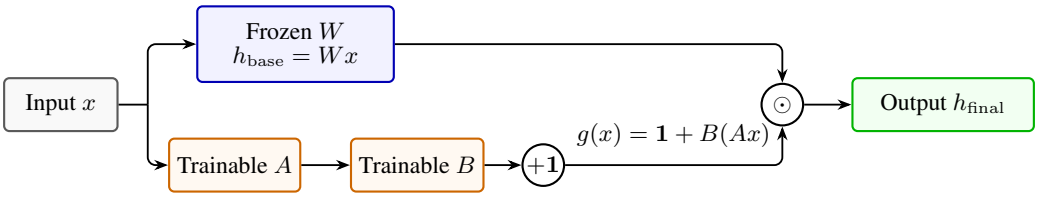


Figure 2: Architecture of McLoRA.

Crucially, the learning signals for A and B depend explicitly on the backbone output $h_{\text{base}} = Wx$ dimension by dimension. This direct dependence imposes a dimension-level constraint on parameter updates: dimensions with larger loss sensitivity ($|\delta_i|$) and larger backbone output magnitude ($|(Wx)_i|$) receive stronger updates.

Under standard backpropagation and within the class of element-wise compositions that introduce no additional parameters and preserve the backbone output when the low-rank update is zero, McLoRA is the minimal natural form for directly modulating the low-rank branch gradients of standard LoRA by the corresponding backbone output Wx; see Appendix C for the derivation.

Bounded extension. The multiplicative signal $g(x) = \mathbf{1} + B(Ax)$ in McLoRA is unbounded and might be negative, which reverses the sign of the corresponding backbone output and causes instability. Our post-hoc diagnostics show that sign reversals and instability are rare in practice. Detailed results are provided in Appendix D. Nevertheless, we introduce a bounded variant, termed sigmoid-McLoRA:

$$g_\sigma(x) = 2\sigma(B_s(A_s x)), \qquad h_{\text{final}} = (Wx) \odot g_\sigma(x), \tag{5}$$

where σ denotes the sigmoid function. Each element of $g_\sigma(x)$ lies in $(0, 2)$, and the zero initialization of B_s gives the neutral value $g_\sigma(x) = \mathbf{1}$. Subsequent experiments show that sigmoid-McLoRA supports the same qualitative conclusion as McLoRA.

To focus on the comparisons between additive and multiplicative composition, we use the original McLoRA formulation as our primary model and treat sigmoid-McLoRA as a bounded extension.

3.2 Comparison with Standard LoRA

McLoRA's outputs and gradients differ from those of LoRA. Appendix B (Table 8) lists the detailed output formulation and low-rank parameter gradients of McLoRA, standard LoRA, and several other LoRA variants.

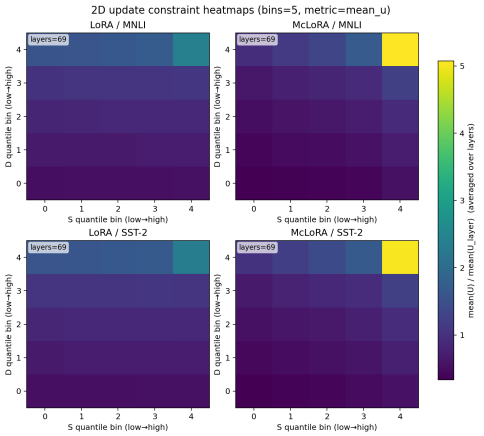
Below we take matrix B as an example to quantitatively compare how the gradient updates of McLoRA and standard LoRA are influenced by the backbone output Wx. To this end, we define the following statistics:

$$S = \mathbb{E}|Wx|, \qquad D = \mathbb{E}|\delta|, \qquad U_i = \sum_t \left\| \frac{\partial L_t}{\partial B_{i,:}} \right\|_2 \tag{6}$$

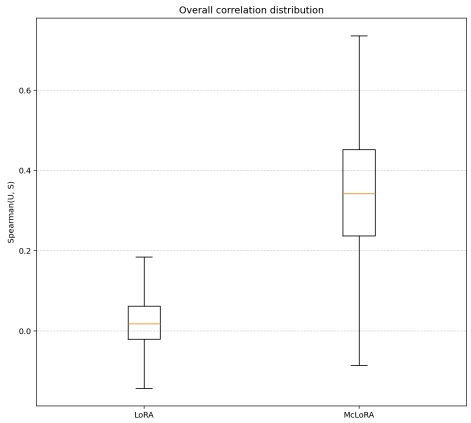
We first partition output dimensions into two-dimensional bins according to the quantiles of S and D, and compute the average U in each bin for each layer. To remove scale differences across layers, we normalize the bin-wise averages within each layer by the layer-wise mean of U. We then average the normalized heatmaps across layers to obtain the overall results shown in Figure 3(a).

As shown in Figure 3(a), strong updates in LoRA are mainly distributed along regions with high D, whereas strong updates in McLoRA are more concentrated in the upper-right region with both high S and high D. This indicates that the multiplicative constraint induced by $\partial L/\partial g = \delta \odot (Wx)$ behaves as expected. Similar patterns are observed in some other tasks on RoBERTa-large and LLaMA-2-7B, which are provided in Appendix E.

We then directly examine the correlation between U and S. To control for the influence of $|\delta|$, we normalize δ by fixing its ℓ_2 norm, and compute the aggregated statistics of U. We visualize the distribution of Spearman correlations across all layers in Figure 3(b).



(a) Cross-layer S–D binned maps of update locations of RoBERTa-large on MNLI and SST-2.



(b) Distribution of layer-wise $\mathrm{Spearman}(U, S)$.

Figure 3: Comparisons of backbone-output-guided update allocation in McLoRA and LoRA.

Table 1: Dimension-wise update statistics of RoBERTa-large. The definitions of metrics are given in Appendix F.

Method	Task	Gini↑	Norm. Entropy↓	Top-1% Energy↑	Top-5% Energy↑	k_{80}↓	$r_{low S}$↓	Support Overlap↓	Support Jaccard↓	Energy on old_unique↓
LoRA	MNLI	0.2809	0.9745	5.22%	15.72%	657.0	16.97%	505.7	0.619	8.01%
LoRA	SST-2	0.2697	0.9771	4.99%	14.92%	665.2	17.87%	505.7	0.619	8.15%
McLoRA	MNLI	**0.4818**	**0.9044**	**16.97%**	**33.87%**	**516.9**	**10.70%**	**391.8**	**0.590**	**6.27%**
McLoRA	SST-2	**0.4516**	**0.9169**	**15.02%**	**30.59%**	**537.2**	**11.62%**	**391.8**	**0.590**	**6.02%**

As shown in Figure 3(b), McLoRA achieves a significantly higher Spearman correlation $\mathrm{Spearman}(U, S)$ than LoRA across layers, which indicates that updates in McLoRA are more likely to occur on dimensions with larger backbone output magnitudes. This reflects the stronger dimension selectivity introduced by explicitly incorporating Wx into the gradient signal of McLoRA.

We further employ a set of complementary statistics to quantitatively compare McLoRA and standard LoRA in two respects: (1) the concentration of their gradient updates across different dimensions, and (2) the overlap across different tasks of the dimension-wise update supports. The results are reported in Table 1, with detailed descriptions of each statistic provided in Appendix F.

As shown in Table 1, compared to LoRA, McLoRA produces more concentrated updates, allocates less update energy to weakly activated dimensions, exhibits smaller overlap in major update supports across tasks in terms of both the overlap size and the Jaccard score, and intrudes less into old-task-specific critical dimensions. These findings provide supporting evidence that McLoRA reduces potential cross-task interference and mitigates forgetting in sequential and continual learning.

4 Experiments

We focus on assessing McLoRA's multi-task continual learning capability. We begin with evaluations under the single-adapter setting on multiple continual learning benchmarks with varying difficulty levels, where we not only evaluate McLoRA but also compare its bounded variant. We then move to the multi-adapter setting, where we evaluate McLoRA alone. Experimental results demonstrate that, under both the single-adapter and multi-adapter settings and across different benchmarks, McLoRA and its bounded variant consistently achieve significant advantages in the comparisons. Finally, as a supplement, we also conduct single-task evaluations and find that McLoRA and LoRA are comparable; detailed results are provided in Appendix H.

4.1 SINGLE-ADAPTER SETTING

4.1.1 DATASETS

We evaluate McLoRA on multiple groups of continual learning benchmarks. See Appendix G for more details.

First, following Kar et al. (2022), we construct four sequential learning task sequences (denoted as S1, S2, S3, and S4) from subsets of GLUE (Wang et al., 2018), covering different sequence lengths and task orders.

Second, following Razdaibiedina et al. (2023), we construct three sequential task orders from the Standard CL Benchmark using Amazon, AG News, Yahoo, and DBpedia, denoted as S5, S6, and S7.

Third, we evaluate McLoRA over four longer task sequences, denoted as LS1, LS2, LS3 and LS4. LS1 $\sim$ LS3 are three permutations of the 15-task long-sequence classification benchmark (Razdaibiedina et al., 2023). LS4 is the 38-task InstrDialog++ stream from CITB (Zihan Zhang et al., 2023). For LS4, we additionally consider Hidden-abrupt and Blurry-20% variants to evaluate continual learning without explicit task-boundary information.

Finally, we construct a sequential language modeling benchmark with the order WikiText (Merity et al., 2017) $\rightarrow$ PTB (Marcus et al., 1993) $\rightarrow$ Enwik8 and track the perplexity (PPL) trajectory on each corpus across different stages of sequential training.

4.1.2 BASELINES

We conducted extensive model comparisons.

First, we compare LoRA and McLoRA on six backbone models: LLaMA-2-7B (Touvron et al., 2023), Qwen1.5-7B (Bai et al., 2023), Qwen3-8B-Base (An Yang et al., 2025), T5-large (Raffel et al., 2020), GPT-2 medium (Radford et al., 2019), and RoBERTa-large (Liu et al., 2019).

Second, we replace the additive LoRA modules in AdaLoRA and DoRA with McLoRA to create Mc-AdaLoRA and Mc-DoRA, respectively, testing whether the additive-to-multiplicative replacement generalizes beyond standard LoRA.

Third, we evaluate sigmoid-McLoRA, as a sign-preserving alternative to the original McLoRA formulation. We compare it with LoRA and McLoRA under the same parameter budget and training configuration.

4.1.3 EVALUATION METRICS

Following the ACC/BWT evaluation protocol for continual learning (Lopez-Paz & Ranzato, 2017), we report two metrics for sequential learning experiments. **Final Avg** is the average performance over all tasks at the end of the sequence. **Forgetting** is the negative backward transfer (BWT) defined as

$$\text{Forgetting} = \frac{1}{T-1} \sum_{t=1}^{T-1} \left(s_{t,t} - s_{t,T} \right), \tag{7}$$

where $s_{t,t}$ denotes the evaluation score on task t immediately after training on task t, and $s_{t,T}$ denotes its final score after completing the whole sequence. For S1 $\sim$ S7 and LS1 $\sim$ LS3, the task-level evaluation score is accuracy. For LS4 and its Hidden-abrupt and Blurry-20% variants, we follow CITB (Zihan Zhang et al., 2023) and use ROUGE-L across all 38 tasks; accordingly, **Final Avg** denotes the final average ROUGE-L, and **Forgetting** is computed from the corresponding ROUGE-L scores.

For language modeling tasks, we report perplexity (PPL) on each corpus at different stages of sequential training.

4.1.4 IMPLEMENTATION DETAILS

A single shared LoRA-style adapter is used throughout the entire task sequence. For S1 $\sim$ S7, each task has its own task-specific classification head; when training the t-th task, we update the

Table 2: Results on different backbones and task sequences.

Backbone	Method	Metric	S1	S2	S3	S4	S5	S6	S7	Overall Avg
LLaMA-2-7B	McLoRA	Final Avg (↑)	82.1	**72.4**	**80.8**	**80.5**	**86.0**	**89.7**	**90.8**	**83.2**
		Forgetting (↓)	13.9	**24.8**	**12.4**	**13.4**	**8.5**	**3.7**	**2.2**	**11.3**
	LoRA	Final Avg (↑)	**84.3**	67.5	63.7	67.1	81.0	88.0	90.4	77.4
		Forgetting (↓)	**11.8**	31.4	35.5	30.8	15.5	6.1	3.0	19.2
Qwen1.5-7B	McLoRA	Final Avg (↑)	**90.1**	**67.1**	**73.6**	**67.2**	**76.8**	**85.3**	**90.3**	**78.6**
		Forgetting (↓)	**3.4**	**31.5**	**22.7**	**31.4**	**20.7**	**9.3**	**2.8**	**17.4**
	LoRA	Final Avg (↑)	79.1	64.9	63.1	61.3	72.3	82.8	89.7	73.3
		Forgetting (↓)	19.2	34.0	36.4	38.7	26.8	12.9	3.6	24.5
Qwen3-8B-Base	McLoRA	Final Avg (↑)	**74.8**	**66.5**	**67.9**	**67.1**	**82.4**	**85.6**	**89.8**	**76.3**
		Forgetting (↓)	**23.1**	**30.6**	**25.5**	**28.0**	**13.0**	**8.9**	**3.4**	**18.9**
	LoRA	Final Avg (↑)	65.0	64.4	61.2	59.2	80.7	82.9	87.7	71.6
		Forgetting (↓)	41.3	36.1	39.4	42.5	15.6	12.8	6.3	27.7
RoBERTa-large	McLoRA	Final Avg (↑)	**87.8**	**81.8**	**73.0**	**73.2**	**86.8**	**86.7**	**89.4**	**82.7**
		Forgetting (↓)	**2.7**	**11.8**	**22.0**	**21.8**	**6.6**	**6.7**	**3.1**	**10.7**
	LoRA	Final Avg (↑)	85.5	73.1	72.2	72.3	78.6	83.4	86.6	78.8
		Forgetting (↓)	6.2	22.4	23.4	23.7	17.5	11.1	6.8	15.9

Table 3: Results on different backbones and long task sequences.

(a) Results on LS1 $\sim$ LS4 with explicit task boundaries using T5-Large and LLaMA-2-7B.

Backbone	Method	Metric	LS1	LS2	LS3	Avg (LS1 $\sim$ LS3)	LS4
T5-Large	McLoRA	Final Avg (↑)	**71.9**	**63.2**	**61.0**	**65.4**	**32.6**
		Forgetting (↓)	**6.9**	**17.2**	**19.8**	**14.6**	**22.6**
	LoRA	Final Avg (↑)	51.2	35.5	27.4	38.0	19.2
		Forgetting (↓)	29.2	44.8	53.2	42.4	34.1
LLaMA-2-7B	McLoRA	Final Avg (↑)	**73.2**	**72.7**	**73.9**	**73.3**	**56.8**
		Forgetting (↓)	**7.0**	**6.7**	**9.2**	**7.6**	**3.6**
	LoRA	Final Avg (↑)	67.3	68.1	68.8	68.1	55.3
		Forgetting (↓)	16.0	10.9	14.4	13.8	5.0

(b) Boundary-free results on LS4 using T5-Large.

Protocol	Method	Final Avg (↑) Forgetting (↓)
Hidden-abrupt	McLoRA	**29.1**
		23.7
	LoRA	18.5
		31.7
Blurry-20%	McLoRA	**35.3**
		18.3
	LoRA	19.4
		32.2

current classification head together with the shared adapter. LS1 $\sim$ LS4 are formulated as text-to-text generation and use the backbone's shared language-modeling head. We record each task's post-learning score and evaluate all tasks at the end of the sequence.

Following the task-free continual-learning formulation of Aljundi et al. (2019) and Jin et al. (2021), the boundary-free variants of LS4 do not provide the learner with explicit task IDs or boundary indicators, and maintain the optimizer state continuously across transitions. Hidden-abrupt directly concatenates the latent task blocks, whereas Blurry-20% mixes samples from adjacent tasks within a 20%-width transition region. For these variants, each latent task's post-learning score is recorded at the first optimizer-step checkpoint after all of its training examples have appeared.

For sequential/continual learning experiments, both LoRA and McLoRA use rank $r = 8$ and are optimized with AdamW (Loshchilov & Hutter, 2019).

For language modeling experiments, we use GPT-2 medium, LLaMA-2-7B, and Qwen1.5-7B as backbones. GPT-2 medium and Qwen1.5-7B use rank $r = 8$, while LLaMA-2-7B uses rank $r = 16$.

See Appendix G for more experimental details and task-wise hyperparameter settings.

4.1.5 Main Results

Table 2 presents the results on seven continual learning sequences (S1 $\sim$ S7) across different backbone models and additionally reports the average over all seven sequences. Overall, McLoRA

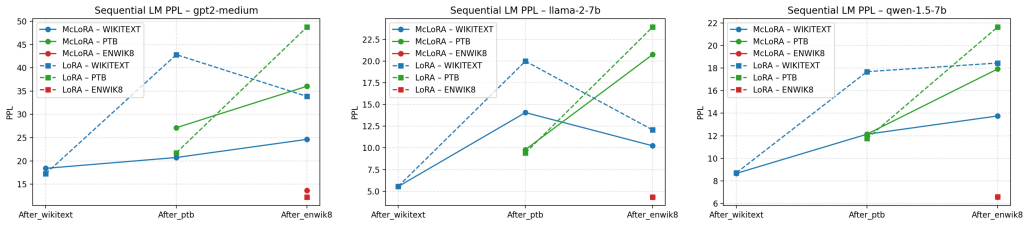


Figure 4: PPL trajectories. All solid lines lie below their corresponding dashed lines.

Table 4: Results on LLaMA-2-7B and RoBERTa-large over GLUE-5 task orders S2 $\sim$ S4.

(a) LLaMA-2-7B.

Method	Metric	S2	S3	S4	Avg
Mc-AdaLoRA	Final Avg (↑)	**80.4**	83.3	**87.0**	**83.6**
	Forgetting (↓)	**11.3**	8.0	**3.8**	**7.7**
AdaLoRA	Final Avg (↑)	78.7	**86.0**	81.0	81.9
	Forgetting (↓)	15.3	**5.9**	12.6	11.3
Mc-DoRA	Final Avg (↑)	**62.2**	**79.5**	**81.7**	**74.5**
	Forgetting (↓)	**37.7**	**15.8**	**12.4**	**22.0**
DoRA	Final Avg (↑)	60.7	70.8	78.2	69.9
	Forgetting (↓)	39.7	27.6	18.4	28.6

(b) RoBERTa-large.

Method	Metric	S2	S3	S4	Avg
Mc-AdaLoRA	Final Avg (↑)	**85.4**	**79.4**	75.6	**80.1**
	Forgetting (↓)	**6.4**	**8.5**	16.4	**10.4**
AdaLoRA	Final Avg (↑)	82.2	71.3	**79.4**	77.6
	Forgetting (↓)	11.5	22.8	**14.5**	16.3
Mc-DoRA	Final Avg (↑)	83.1	**74.2**	**75.0**	**77.4**
	Forgetting (↓)	9.7	**19.7**	**18.9**	**16.1**
DoRA	Final Avg (↑)	**85.2**	72.4	71.3	76.3
	Forgetting (↓)	**7.8**	23.3	25.0	18.7

achieves higher Final Avg in 27 out of 28 settings and lower Forgetting in 27 out of 28 settings. Averaged over all 28 settings, McLoRA improves Final Avg by 4.9 points and reduces Forgetting by 7.2 points compared with LoRA.

The largest gain is observed on the S3 sequence with LLaMA-2-7B, where McLoRA improves Final Avg by 17.1 points and reduces Forgetting by 23.1 points. The only exception where McLoRA slightly underperforms LoRA occurs on the shortest task sequence (S1) with LLaMA-2-7B, where the forgetting pressure is relatively low.

Table 3(a) extends the comparison to long task streams. McLoRA improves both Final Avg and Forgetting in all backbone–sequence combinations significantly. Averaged over the three 15-task orders, McLoRA improves Final Avg by 27.4 points and reduces Forgetting by 27.8 points on T5-Large. The corresponding improvements on LLaMA-2-7B are 5.2 and 6.2 points, respectively. The same pattern persists on the more diverse 38-task InstrDialog++ stream, where McLoRA outperforms LoRA on both metrics with both backbones.

Table 3(b) further evaluates McLoRA without explicit task-boundary information. McLoRA remains consistently better under both transition protocols. Under Hidden-abrupt, it improves Final Avg ROUGE-L by 10.6 points and reduces Forgetting by 8.0 points. Under Blurry-20%, the corresponding improvements are 15.9 and 13.9 points. These results show that McLoRA's gains persist both without explicit boundary signals and under overlapping task distributions near transitions.

Figure 4 shows that on the sequential language modeling benchmark, McLoRA exhibits smaller PPL rebounds on earlier corpora after training on later ones, indicating weaker forgetting, consistent with the classification results.

Table 4 shows that replacing additive composition with multiplicative composition consistently improves AdaLoRA and DoRA on both LLaMA-2-7B and RoBERTa-large.

We further evaluate the bounded extension sigmoid-McLoRA. As shown in Table 5, Sigmoid-McLoRA achieves significantly higher performance than standard LoRA, on par with or even slightly better than McLoRA.

The overall training overhead of McLoRA is comparable to that of LoRA. Further details are provided in Appendix G.6.

Table 5: Results on Qwen1.5-7B and RoBERTa-large over GLUE-5 task orders S2 $\sim$ S4.

(a) Qwen1.5-7B.

Method	Metric	S2	S3	S4	Avg
McLoRA	Final Avg (↑)	67.1	73.6	67.2	69.3
	Forgetting (↓)	31.5	22.7	31.4	28.5
sigmoid-McLoRA	Final Avg (↑)	**68.8**	**78.8**	**72.7**	**73.4**
	Forgetting (↓)	**29.7**	**15.9**	**23.8**	**23.1**
LoRA	Final Avg (↑)	64.9	63.1	61.3	63.1
	Forgetting (↓)	34.0	36.4	38.7	36.4

(b) RoBERTa-large.

Method	Metric	S2	S3	S4	Avg
McLoRA	Final Avg (↑)	81.8	73.0	73.2	76.0
	Forgetting (↓)	11.8	22.0	21.8	18.5
sigmoid-McLoRA	Final Avg (↑)	**82.8**	**75.0**	**76.1**	**78.0**
	Forgetting (↓)	**9.9**	**19.3**	**17.4**	**15.5**
LoRA	Final Avg (↑)	73.1	72.2	72.3	72.5
	Forgetting (↓)	22.4	23.4	23.7	23.2

Table 6: Final Avg on T5-large for three task orders.

Method	Order 1	Order 2	Order 3	Avg
IncLoRA	66.0	64.9	68.3	66.4
O-LoRA	75.4	75.7	76.3	75.8
LB-CL	76.9	76.5	76.8	76.7
AM-LoRA	78.1	79.8	76.2	78.0
N-LoRA	**79.2**	78.4	78.8	78.8
Inc-McLoRA	79.0	**79.9**	**79.1**	**79.3**

4.2 Multi-Adapter Setting

Following the incremental LoRA protocol in prior work (Wang et al., 2023; Yang et al., 2025; Qiao & Mahdavi, 2024; Jialin Liu et al., 2024), we construct **Inc-McLoRA** by simply replacing additive LoRA with McLoRA in multi-adapter IncLoRA, without additional regularization or routing.

We evaluate on the Standard CL Benchmark with T5-large (Raffel et al., 2020).

Table 6 compares Inc-McLoRA with IncLoRA and some other multi-adapter models, including O-LoRA, LB-CL, AM-LoRA, and N-LoRA. Inc-McLoRA achieves the best average performance among all compared methods. This demonstrates that the advantage of multiplicative composition can generalize from the single-adapter setting to multi-adapter scenarios.

5 Conclusion

This paper proposes McLoRA, a multiplicative LoRA. Unlike the standard additive LoRA, McLoRA modulates the frozen backbone output via element-wise multiplication, making the low-rank branch update explicitly guided by backbone output magnitudes. Theoretically, we show that McLoRA's update signal is jointly driven by $\delta \odot (Wx)$. Empirically, McLoRA produces more concentrated updates, allocates less energy to weakly activated dimensions, exhibits smaller cross-task overlap, and intrudes less into old-task-critical dimensions—mechanistically explaining its reduced interference and stronger forgetting resistance.

Experiments show that McLoRA significantly mitigates catastrophic forgetting in single-adapter continual learning under the same parameter budget, improves additive LoRA variants (AdaLoRA, DoRA), and extends effectively to multi-adapter protocols. The bounded sigmoid-McLoRA remains effective, and single-task performance stays on par with LoRA.

Overall, McLoRA offers a simple, effective, and extensible alternative to standard LoRA composition, providing a new perspective for improving forgetting resistance in LoRA-based continual learning. We discuss the limitations of McLoRA in Appendix I.

AI use statement

In this work, we used generative AI tools for implementing the proposed method. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate math-

ematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on the research methodology or experiments, assist with translation, clean or reformat datasets, or interpret the results. Generating synthetic datasets and supporting qualitative or thematic data analysis were not applicable to this work.

Additionally, we used generative AI tools to create or edit software code, and to edit the manuscript to improve its grammar, clarity, and readability. We have reviewed all AI-assisted work. All AI-assisted implementations and code modifications were manually reviewed and tested by the authors to verify their correctness and consistency with the intended method and experimental pipeline. All AI-assisted language edits were reviewed to ensure that they preserved the intended technical meaning and did not introduce or alter scientific claims, experimental results, or citations. We take responsibility for the final content of this work, including text, claims, code, and other artifacts produced with the aid of generative AI.

Reproducibility statement

We support the reproducibility of this work by providing detailed descriptions of the proposed method and experimental protocols. Section 3.1 specifies the architecture, initialization, forward computation, and gradient computation of McLoRA, while Section 4.1 describes the sequential learning evaluation settings, including the datasets, baselines, evaluation metrics, and implementation protocols. Appendix G provides dataset documentation, complete task orders, input construction procedures, target modules, and task-wise hyperparameter settings for the sequential learning experiments. Appendices G.7 and G.8 document the experimental settings for the method extensions. Appendix H presents the single-task experimental setup, results, dataset documentation, and training details. The protocols and metric definitions used in the mechanism analyses are documented in Appendices A, E, and F. Finally, Appendix J reports the licenses and terms of use of the employed artifacts, model sizes, computational resources, and software and package versions. Together, these sections provide the information needed to implement the proposed method and reproduce the reported evaluations.

References

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

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

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023. URL https://arxiv.org/abs/2309.16609.

Harsh Bihany, Shubham Patel, and Ashutosh Modi. LoRMA: Low-rank multiplicative adaptation for LLMs. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10113–10133,

Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.527. URL https://aclanthology.org/2025.findings-acl.527/.

Duzhen Zhang, Yong Ren, Zhong-Zhi Li, Yahan Yu, Jiahua Dong, Chenxing Li, Zhilong Ji, and Jinfeng Bai. Enhancing multimodal continual instruction tuning with BranchLoRA. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5743–5756, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.287. URL https://aclanthology.org/2025.acl-long.287/.

Jiangpeng He, Zhihao Duan, and Fengqing Zhu. Cl-lora: Continual low-rank adaptation for rehearsal-free class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30534–30544, June 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

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

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5254–5276, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.319. URL https://aclanthology.org/2023.emnlp-main.319/.

Jialin Liu, Jianhua Wu, Jie Liu, and Yutai Duan. Learning attentional mixture of loras for language model continual learning, 2024. URL https://arxiv.org/abs/2409.19611.

Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 29193–29205. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/f45a1078feb35de77d26b3f7a52ef502-Paper.pdf.

Sudipta Kar, Giuseppe Castellucci, Simone Filice, Shervin Malmasi, and Oleg Rokhlenko. Preventing catastrophic forgetting in continual learning of new natural language tasks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 3137–3145, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539169. URL https://doi.org/10.1145/3534678.3539169.

Prakhar Kaushik, Ankit Vaidya, Shravan Chaudhari, Rama Chellappa, and Alan Yuille. Shared lora subspaces for almost strict continual learning, 2026. URL https://arxiv.org/abs/2602.06043.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint*

Conference on Natural Language Processing (Volume 1: Long Papers), pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://aclanthology.org/J93-2004/.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Fuli Qiao and Mehrdad Mahdavi. Learn more, but bother less: parameter efficient continual learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 97476–97498. Curran Associates, Inc., 2024. doi: 10.52202/079017-3092. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b0bc711f48724237b38823c4d9cee10b-Paper-Conference.pdf.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UJTgQBc91_.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/liu24bn.html.

Chongjie Si, Zhiyi Shi, Xuehui Wang, Yichen Xiao, Xiaokang Yang, and Wei Shen. Generalized tensor-based parameter-efficient fine-tuning via lie group transformations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 197–207, October 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel

Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446/.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.715. URL https://aclanthology.org/2023.findings-emnlp.715/.

Yichen Wu, Hongming Piao, Long-Kai Huang, Renzhen Wang, Wanhua Li, Hanspeter Pfister, Deyu Meng, Kede Ma, and Ying Wei. Sd-lora: Scalable decoupled low-rank adaptation for class incremental learning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 58337–58353, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/92f43b1d33fae4aa1958f75317f0cec1-Paper-Conference.pdf.

Xin Zhang, Liang Bai, Xian Yang, and Jiye Liang. C-lora: Continual low-rank adaptation for pre-trained models, 2025. URL https://arxiv.org/abs/2502.17920.

Shuo Yang, Kun-Peng Ning, Yu-Yang Liu, Jia-Yu Yao, Yong-Hong Tian, Yi-Bing Song, and Li Yuan. Is parameter collision hindering continual learning in LLMs? In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 4243–4259, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.286/.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

Zihan Zhang, Meng Fang, Ling Chen, and Mohammad-Reza Namazi-Rad. CITB: A benchmark for continual instruction tuning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9443–9455, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.633. URL https://aclanthology.org/2023.findings-emnlp.633/.

A Activation Intervention Analysis

In the introduction, we motivate McLoRA by arguing that backbone output magnitudes provide a task-dependent response profile, and that dimensions with larger backbone output magnitudes are more likely to be influential for the current task. To empirically support this motivation, we conduct an activation-intervention analysis on single-task LoRA checkpoints.

We use RoBERTa-large single-task LoRA checkpoints trained on MNLI, SST-2, QQP, and QNLI. For each of the LoRA target sites, i.e., the query, key, and value projections in each Transformer

Table 7: Activation-intervention analysis on RoBERTa-large single-task LoRA checkpoints.

Task	Intervention	Mean selected S	ΔLoss ($\times 10^{-3}$)	Acc. Drop ($\times 10^{-3}$)
MNLI	Top-20% S	1.7468	2.521	1.667
	Random-20%	0.9766	1.142	1.076
	Bottom-20% S	0.5791	-0.089	-0.201
SST-2	Top-20% S	1.7590	0.849	0.557
	Random-20%	0.9470	0.502	0.377
	Bottom-20% S	0.5279	0.182	-0.350
QQP	Top-20% S	1.6979	3.520	-0.861
	Random-20%	0.9389	0.739	-1.463
	Bottom-20% S	0.5412	-0.190	-1.431
QNLI	Top-20% S	1.7261	5.307	1.167
	Random-20%	0.9710	1.643	0.449
	Bottom-20% S	0.5754	0.338	0.264

layer, we first collect the frozen backbone output on a calibration set and compute the dimension-wise output-magnitude profile

$$S_i = \mathbb{E}_{x \sim \mathcal{D}_{\text{cal}}} |h_{\text{base},i}(x)|. \tag{8}$$

We also compute the calibration mean of each dimension,

$$\mu_i = \mathbb{E}_{x \sim \mathcal{D}_{\text{cal}}} h_{\text{base},i}(x). \tag{9}$$

For each target site, we rank output dimensions according to S_i and construct three intervention groups: the top 20% dimensions with the largest output magnitudes, a random 20% subset, and the bottom 20% dimensions with the smallest output magnitudes. During evaluation, we do not modify any model parameters. Instead, we replace the selected dimensions of the frozen backbone output with their calibration means,

$$h_{\text{base},i}(x) \leftarrow \mu_i. \tag{10}$$

This intervention removes the input-specific information carried by the selected dimensions while keeping their values close to the normal backbone-output range. We then measure the loss increase relative to the clean model. A larger loss increase indicates that the intervened dimensions have greater influence on current-task predictions.

For each task, the calibration set contains 5,000 training examples. MNLI is evaluated on 2,000 examples from the validation-matched split, SST-2 is evaluated on the full validation set, and QQP and QNLI are each evaluated on 1,000 examples from their respective validation splits. Results are averaged over all 72 target sites. For the random group, we use three random masks and report the average.

As shown in Table 7, intervening on dimensions with the largest backbone output magnitudes leads to the largest loss increase on all four tasks, while intervening on dimensions with the smallest backbone output magnitudes has the smallest effect. This suggests that dimensions with larger backbone output magnitudes tend to have greater causal influence on current-task prediction. Therefore, using backbone output magnitudes as a conditioning signal for low-rank update allocation is empirically well motivated.

B Comparison of LoRA Variants

Table 8 compares standard LoRA and representative LoRA variants discussed in Section 2 in terms of their output formulation and low-rank parameter gradients. For a unified comparison, we denote the upstream gradient by $\delta = \partial L / \partial h$.

Table 8: Comparison of standard LoRA and representative LoRA variants in terms of output formulation and low-rank parameter gradients. For DoRA, G_{DoRA} denotes the gradient with respect to the low-rank update after the magnitude-direction normalization. For LieRA, $G_{\mathrm{LieRA}} = \big((\delta x^\top) \odot W\big) \odot \exp(BA)$. LoRMA and LieRA introduce multiplicative adaptation in the weight space, whereas McLoRA introduces multiplicative modulation in the output space. McLoRA is the only method in this table for which the corresponding backbone output $(Wx)_i$ directly appears as an explicit dimension-wise factor in the gradients of the low-rank parameters.

Method	**Output Formulation**	**Gradient of B**	**Gradient of A**
LoRA	$h = Wx + B(Ax)$	$\frac{\partial L}{\partial B} = \delta(Ax)^\top$	$\frac{\partial L}{\partial A} = B^\top \delta x^\top$
AdaLoRA	$h = Wx + B(Ax)$ with adaptive rank	$\frac{\partial L}{\partial B} = \delta(Ax)^\top$ on active ranks	$\frac{\partial L}{\partial A} = B^\top \delta x^\top$ on active ranks
DoRA	$h = \frac{m}{\|W+BA\|}(W+BA)x$	$\frac{\partial L}{\partial B} = G_{\mathrm{DoRA}}(Ax)^\top$	$\frac{\partial L}{\partial A} = B^\top G_{\mathrm{DoRA}} x^\top$
LoRMA	$h = [(I+BA)W]x$	$\frac{\partial L}{\partial B} = \delta(AWx)^\top$	$\frac{\partial L}{\partial A} = B^\top \delta(Wx)^\top$
LieRA	$h = [W \odot \exp(BA)]x$	$\frac{\partial L}{\partial B} = G_{\mathrm{LieRA}} A^\top$	$\frac{\partial L}{\partial A} = B^\top G_{\mathrm{LieRA}}$
McLoRA	$h = (Wx) \odot (\mathbf{1} + B(Ax))$	$\frac{\partial L}{\partial B} = (\delta \odot Wx)(Ax)^\top$	$\frac{\partial L}{\partial A} = B^\top(\delta \odot Wx)x^\top$

C Derivation of the Minimal Natural Form

Let

$$u = Wx, \qquad v = BAx, \qquad h_i = F(u_i, v_i), \qquad \delta_i = \frac{\partial L}{\partial h_i}. \tag{11}$$

Under standard backpropagation, standard additive LoRA uses $h_i = u_i + v_i$, and therefore

$$\frac{\partial L}{\partial v_i} = \delta_i \frac{\partial h_i}{\partial v_i} = \delta_i. \tag{12}$$

Here, we make the notion of "minimal" precise as follows: relative to the local gradient δ_i of additive LoRA, the composition introduces only the corresponding backbone-output factor u_i, without introducing any other modulation term. Therefore, for any upstream gradient δ_i, we require

$$\frac{\partial L}{\partial v_i} = \delta_i u_i. \tag{13}$$

By the chain rule used in standard backpropagation,

$$\frac{\partial L}{\partial v_i} = \delta_i \frac{\partial F(u_i, v_i)}{\partial v_i}. \tag{14}$$

Thus, the exact local-gradient condition above is equivalent to

$$\frac{\partial F(u_i, v_i)}{\partial v_i} = u_i. \tag{15}$$

Integrating with respect to v_i gives

$$F(u_i, v_i) = u_i v_i + C(u_i). \tag{16}$$

We further require the backbone output to remain unchanged when the low-rank signal is zero:

$$F(u_i, 0) = u_i. \tag{17}$$

It follows that $C(u_i) = u_i$, and hence

$$F(u_i, v_i) = u_i(1 + v_i), \tag{18}$$

Table 9: Post-hoc diagnostics of the McLoRA multiplicative signal on the T5-Large InstrDialog++ sequence. The table reports selected task-end checkpoints from the complete 38-checkpoint analysis.

Checkpoint	$P(g(x) < 0)$	$P(\|g(x)\| < 0.1)$	Sign-change Rate
1	0.000%	0.000%	–
3	0.923%	0.541%	1.043%
20	2.073%	1.115%	2.019%
30	4.757%	1.939%	**3.198%**
32	**5.521%**	2.071%	2.263%
38	5.267%	**2.104%**	1.532%

or, equivalently,

$$h = (Wx) \odot (1 + BAx). \tag{19}$$

Therefore, under element-wise differentiable composition, no additional composition parameters, standard backpropagation, and preservation of the backbone output when the low-rank signal is zero, McLoRA is the minimal natural form for directly modulating the LoRA low-rank branch gradients by the corresponding backbone output.

This conclusion is restricted to the design class, standard-backpropagation setting, and exact gradient condition specified above; it is not stated as a uniqueness result over all possible architectures.

D Sign Diagnostics of the McLoRA Multiplicative Signal

We conduct a post-hoc diagnostic of the multiplicative signal in the original McLoRA formulation using T5-Large on the 38-task InstrDialog++ sequence. We analyze all 38 task-end checkpoints. For each task, we take the first four examples from the released test set in its original order, resulting in a fixed set of 152 probe examples. At each checkpoint, we collect the multiplicative signal $g(x)$ from all 144 target q/v layers and all non-padding token positions.

We report the proportion of negative signal values, $P(g(x) < 0)$; the proportion of values close to zero, $P(|g(x)| < 0.1)$; and the adjacent-checkpoint sign-change rate. The first two quantities are computed exactly over all non-padding signal values collected at each checkpoint, corresponding to approximately 4.97 billion values per checkpoint. The sign-change rate is computed on a deterministic aligned sample of 1,400,832 coordinates per checkpoint. Using the same probe inputs, we compare the sign of the multiplicative signal at the same token, layer, and dimension between consecutive task-end checkpoints. Table 9 reports representative checkpoints; all 38 checkpoints are included in the analysis.

Negative signal values and sign changes therefore occur in practice but remain limited. Across all checkpoints, the highest observed negative-signal proportion is 5.521%, and the highest adjacent-checkpoint sign-change rate at fixed coordinates is 3.198%.

We further examine the gradient norms recorded during training to determine whether these sign reversals are accompanied by numerical instability. Among 368 recorded observations, the median, 95th percentile, and 99th percentile are 0.973, 3.395, and 7.247, respectively. The maximum value is an isolated observation of 24.696. We observe no systematic increase in gradient norms as the task sequence progresses and no NaN/Inf values, overflow, or skipped optimization steps.

These results establish that the unbounded multiplicative signal in the original McLoRA formulation exhibits limited but nonzero sign reversals, while the overall training process remains numerically stable.

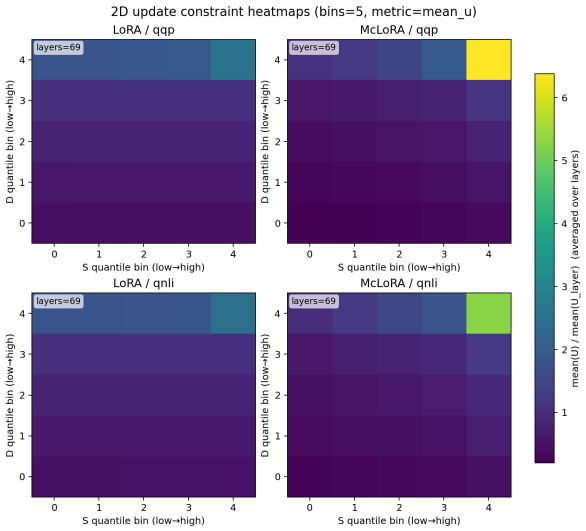


Figure 5: Cross-layer two-dimensional S–D binned maps of update locations for LoRA and McLoRA on RoBERTa-large with QQP and QNLI, following the same protocol as Figure 3(a).

E Additional Mechanism Validation Across Task Pairs and Architectures

To further verify that the update-allocation behavior discussed in Section 3.2 is not specific to a single task pair or backbone, we report two additional settings in this appendix: (i) RoBERTa-large on QQP and QNLI, and (ii) a decoder-only LLaMA-2-7B backbone on MNLI and SST-2. For both settings, we follow exactly the same definitions of S, D, and U, and report the same cross-layer S–D binned maps as in Figure 3(a).

E.1 RoBERTa-large on QQP and QNLI

Figure 5 presents the corresponding results on RoBERTa-large with QQP and QNLI. Consistent with the observations in the main text, strong updates in LoRA are mainly distributed in regions with high D, whereas McLoRA concentrates them more strongly in the joint high-S/high-D region. This provides additional evidence that multiplicative composition encourages updates to align with both loss sensitivity and backbone output magnitude.

E.2 Decoder-only LLaMA-2-7B on MNLI and SST-2

Figure 6 shows the results on a decoder-only LLaMA-2-7B backbone with MNLI and SST-2. Similar qualitative trends are observed in this setting. Compared with standard LoRA, McLoRA places stronger updates in the joint high-S/high-D region, suggesting that the mechanism is not restricted to encoder-only backbones.

F Detailed Dimension-wise Update Statistics

We provide detailed definitions for the dimension-wise update statistics reported in Table 1. We compute several concentration statistics of the update profile U for each layer, including the Gini coefficient, normalized entropy, and the proportions of update energy carried by the top 1% and top 5% dimensions. We also define the effective support set $\mathrm{Supp}_p(U)$ as the smallest set of dimensions whose cumulative update energy covers a proportion p of the total update energy within a layer; throughout this paper, we use $p = 0.8$. The size of this set can be viewed as the width of the major update coverage, and is denoted by k_{80}.

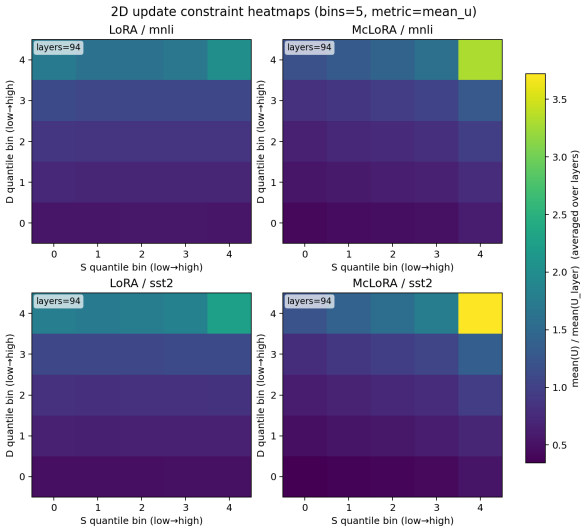


Figure 6: Cross-layer two-dimensional S–D binned maps of update locations for LoRA and McLoRA on a decoder-only LLaMA-2-7B backbone with MNLI and SST-2, following the same protocol as Figure 3(a).

As shown in Table 1, McLoRA exhibits substantially more concentrated updates: it has higher Gini coefficients, lower normalized entropy, and larger Top-1%/Top-5% energy ratios. Consistently, McLoRA requires fewer dimensions to cover 80% of the update energy (i.e., smaller k_{80}), indicating a narrower major update coverage.

Beyond overall concentration, we further measure the proportion of update energy that falls on weakly activated dimensions. Specifically, for each layer, we sort dimensions according to the output-magnitude profile S and take the bottom 20% set, denoted by $B_{0.2}(S)$, and compute

$$r_{\text{low}S} = \frac{\sum_{i \in B_{0.2}(S)} U_i}{\sum_i U_i}. \tag{20}$$

As shown in the corresponding column of Table 1, McLoRA yields lower $r_{\text{low}S}$, indicating that its updates diffuse less into weakly activated dimensions.

Furthermore, under the notion of major update energy coverage, we compare both the size of the intersection between task-level effective support sets and their corresponding Jaccard score. Table 1 shows that McLoRA yields a smaller value of

$$\left|\text{Supp}_{0.8}(U_{\text{MNLI}}) \cap \text{Supp}_{0.8}(U_{\text{SST-2}})\right|, \tag{21}$$

as well as a lower Jaccard score,

$$\frac{\left|\text{Supp}_{0.8}(U_{\text{MNLI}}) \cap \text{Supp}_{0.8}(U_{\text{SST-2}})\right|}{\left|\text{Supp}_{0.8}(U_{\text{MNLI}}) \cup \text{Supp}_{0.8}(U_{\text{SST-2}})\right|}. \tag{22}$$

Both statistics are computed separately for each layer and then averaged across layers. These results indicate a narrower major update coverage shared across tasks in terms of both the absolute overlap size and the support-size-normalized Jaccard score.

Finally, in a method-aligned manner, we further measure whether new-task updates intrude into dimensions that are uniquely critical to the old task. For LoRA, branch updates are dominated by δ, and we therefore define old-task and new-task critical support sets using $D = \mathbb{E}|\delta|$, and construct

$$\text{old_unique}^{\text{LoRA}} = \text{Supp}_{0.8}(D_{\text{old}}) \setminus \text{Supp}_{0.8}(D_{\text{new}}). \tag{23}$$

For McLoRA, multiplicative-branch updates are dominated by $\delta \odot (Wx)$, and we therefore define critical support sets using $I = \mathbb{E}|\delta \odot (Wx)|$, and construct

$$\text{old_unique}^{\text{McLoRA}} = \text{Supp}_{0.8}(I_{\text{old}}) \setminus \text{Supp}_{0.8}(I_{\text{new}}). \tag{24}$$

We then compute the proportion of new-task update energy that falls into these sets. As shown in the corresponding column of Table 1, McLoRA assigns a smaller fraction of update energy to these old-task-specific critical support sets than LoRA.

Table 10: Dataset documentation for GLUE, the Standard CL Benchmark, the 15-task Long Sequence, InstrDialog++, and the sequential language-modeling corpora.

Dataset	Task Type	Text Domain / Source
MNLI	Natural language inference	Multi-genre written and spoken English text
QQP	Paraphrase detection	Quora question pairs
QNLI	Question-answering NLI	Wikipedia passages and questions
SST-2	Sentiment classification	Movie reviews
RTE	Natural language inference	News, Wikipedia, and related entailment sources
CoLA	Linguistic acceptability	Sentences from linguistic acceptability studies
MRPC	Paraphrase detection	News sentence pairs
STS-B	Semantic textual similarity	Mixed-domain sentence pairs
CB	Natural language inference	Naturally occurring discourses from multiple genres
WiC	Word-sense classification	Lexicographic example sentences
COPA	Causal reasoning	Short everyday causal scenarios
BoolQ	Boolean question answering	Yes/no questions over Wikipedia passages
IMDB	Sentiment classification	Movie reviews
Yelp	Sentiment classification	Yelp business reviews
MultiRC	Reading comprehension	Multi-sentence passages and questions
DBpedia	Topic classification	Wikipedia/DBpedia entries
Amazon	Sentiment classification	Amazon product reviews
Yahoo	Topic classification	Yahoo question-answer text
AG News	Topic classification	News articles
InstrDialog++	Instruction-based text-to-text learning	38 Super-NaturalInstructions tasks spanning 18 NLP task categories and heterogeneous text domains
WikiText	Language modeling	Wikipedia articles
PTB	Language modeling	Wall Street Journal text
Enwik8	Language modeling	English Wikipedia text

G Experimental Details for Sequential Learning

This appendix provides dataset documentation, task orders, hyperparameter settings, and other implementation details for the sequential learning experiments in Section 4. For the main shared single-adapter LoRA/McLoRA experiments, the same backbone-task pair uses identical task-wise training hyperparameters; only the adapter composition differs. Within each benchmark family, the same task uses the same hyperparameters across all task orders in which it appears.

G.1 Dataset Documentation

All benchmarks documented in this subsection are English-language benchmarks. Table 10 summarizes the task type and text domain/source of the datasets used in the GLUE, Standard CL Benchmark, 15-task Long Sequence, InstrDialog++, and sequential language-modeling experiments. The 15-task Long Sequence reuses several datasets from GLUE and the Standard CL Benchmark and additionally includes the datasets listed in the corresponding block below.

G.2 Task Orders

The task orders are listed in Table 11. LS1 $\sim$ LS3 correspond to Orders 8 $\sim$ 10, respectively, from Razdaibiedina et al. (2023). LS4 follows Order 1 of the released InstrDialog++ stream. For compactness, Table 12 identifies each SuperNI task by its unique numeric task ID; these IDs correspond directly to the released `task<ID>_*.json` filenames.

Table 11: Task orders used in the sequential learning experiments. S1 $\sim$ S4 are GLUE sequences, S5 $\sim$ S7 are Standard CL Benchmark sequences, LS1 $\sim$ LS3 are 15-task Long Sequence settings, and LS4 is the 38-task InstrDialog++ stream.

Sequence	Task Order
S1	QQP → RTE → SST-2
S2	MNLI → QQP → QNLI → SST-2 → RTE
S3	RTE → SST-2 → QNLI → QQP → MNLI
S4	SST-2 → QQP → RTE → QNLI → MNLI
S5	DBpedia → Amazon → Yahoo → AG News
S6	DBpedia → Amazon → AG News → Yahoo
S7	Yahoo → Amazon → AG News → DBpedia
LS1	MNLI → CB → WiC → COPA → QQP → BoolQ → RTE → IMDB → Yelp → Amazon → SST-2 → DBpedia → AG News → MultiRC → Yahoo
LS2	MultiRC → BoolQ → WiC → MNLI → CB → COPA → QQP → RTE → IMDB → SST-2 → DBpedia → AG News → Yelp → Amazon → Yahoo
LS3	Yelp → Amazon → MNLI → CB → COPA → QQP → RTE → IMDB → SST-2 → DBpedia → AG News → Yahoo → MultiRC → BoolQ → WiC
LS4	InstrDialog++ Order 1 (38 tasks; see Table 12)

Table 12: Task order for the 38-task InstrDialog++ stream (LS4).

Position	Task ID	Position	Task ID
1	task864	20	task576
2	task967	21	task082
3	task1553	22	task1590
4	task1549	23	task1729
5	task639	24	task565
6	task766	25	task1730
7	task636	26	task1603
8	task1501	27	task301
9	task459	28	task298
10	task1203	29	task306
11	task1500	30	task1600
12	task611	31	task1427
13	task347	32	task848
14	task574	33	task1713
15	task598	34	task1151
16	task379	35	task927
17	task1384	36	task294
18	task1360	37	task1607
19	task1714	38	task573

G.3 Classification Sequences

For the shared single-adapter sequential classification experiments, LoRA and McLoRA use the same target modules under the same backbone. Specifically, the target modules are {query, key, value} for RoBERTa-large, {q_proj, v_proj, down_proj} for LLaMA-2-7B, and {q_proj, k_proj, v_proj, o_proj} for both Qwen1.5-7B and Qwen3-8B-Base.

Table 13 reports the task-wise hyperparameter settings for the sequential GLUE classification experiments (S1 $\sim$ S4), and Table 14 reports the corresponding settings for the Standard CL Benchmark (S5 $\sim$ S7).

Table 13: Task-wise hyperparameter settings for the shared single-adapter sequential GLUE classification experiments (S1 $\sim$ S4).

Backbone	Task	Epochs	LR	Train/Eval BS	Grad Acc	Max Length	Weight Decay
RoBERTa-large	MNLI	5	3e-4	64 / 64	1	128	0.01
	QQP	10	3e-4	64 / 64	1	512	0.01
	QNLI	5	2e-4	64 / 64	1	512	0.01
	SST-2	5	2e-5	16 / 16	1	128	0.01
	RTE	10	4e-4	8 / 8	1	512	0.01
LLaMA-2-7B	MNLI	3	2e-5	16 / 16	4	128	0.01
	QQP	5	2e-5	16 / 16	4	512	0.01
	QNLI	3	2e-5	16 / 16	4	512	0.01
	SST-2	3	2e-5	16 / 16	1	128	0.01
	RTE	5	2e-5	8 / 8	1	512	0.01
Qwen1.5-7B Qwen3-8B-Base	MNLI	3	2e-5	16 / 16	4	128	0.01
	QQP	5	2e-5	16 / 16	4	512	0.01
	QNLI	3	2e-5	16 / 16	4	512	0.01
	SST-2	3	2e-5	16 / 16	1	128	0.01
	RTE	5	2e-5	8 / 8	1	512	0.01

Table 14: Task-wise hyperparameter settings for the shared single-adapter sequential Standard CL Benchmark experiments (S5 $\sim$ S7).

Backbone	Task	Epochs	LR	Train/Eval BS	Grad Acc	Max Length	Weight Decay	Max Train Samples
RoBERTa-large	Yahoo	1	4e-4	16 / 32	1	512	0.01	100000
	Amazon	1	4e-4	32 / 64	1	256	0.01	100000
	AG News	1	4e-4	32 / 64	1	256	0.01	100000
	DBpedia	1	4e-4	32 / 64	1	256	0.01	100000
LLaMA-2-7B	Yahoo	1	1e-4	16 / 32	1	512	0.01	100000
	Amazon	1	1e-4	32 / 64	1	256	0.01	100000
	AG News	1	1e-4	32 / 64	1	256	0.01	100000
	DBpedia	1	1e-4	32 / 64	1	256	0.01	100000
Qwen1.5-7B Qwen3-8B-Base	Yahoo	1	1e-4	16 / 32	1	512	0.01	100000
	Amazon	1	1e-4	32 / 64	1	256	0.01	100000
	AG News	1	1e-4	32 / 64	1	256	0.01	100000
	DBpedia	1	1e-4	32 / 64	1	256	0.01	100000

G.4 Long Text-to-Text Sequences

For the shared single-adapter long-sequence experiments (LS1 $\sim$ LS4), all tasks are formulated in a unified text-to-text format and use the backbone's shared language-modeling head. One fixed adapter is updated throughout each task sequence, while the pretrained backbone remains frozen. LoRA and McLoRA use the same target modules under the same backbone: the attention {`q, v`} projections for T5-Large and {`q_proj, v_proj`} for LLaMA-2-7B.

LS1 $\sim$ LS3 contain 15 text-classification tasks represented as label generation, whereas LS4 contains 38 heterogeneous instruction-following tasks from InstrDialog++. Table 15 reports the hyperparameter settings for LS1 $\sim$ LS3, and Table 16 reports the corresponding settings for LS4.

For all long-sequence experiments, LoRA and McLoRA use rank $r = 8$ and are optimized with AdamW without weight decay. We use a constant learning rate with no warmup and clip the gradient norm to 1.0. LLaMA-2-7B uses gradient checkpointing, whereas T5-Large does not.

For LS1 $\sim$ LS3, we initialize the benchmark from the publicly released processed task files of Wang et al. (2023), while independently implementing LoRA and McLoRA under our single-shared-

Table 15: Hyperparameter settings for the shared single-adapter 15-task Long Sequence experiments (LS1 $\sim$ LS3).

Backbone	Tasks	Epochs per Task	LR	Train/Eval BS	Grad Acc	Effective Train BS	Max Source/Target Length	Weight Decay
T5-Large	15	1	1e-3	8 / 128	8	64	512 / 50	0
LLaMA-2-7B	15	1	2e-4	4 / 32	16	64	512 / 16	0

Table 16: Hyperparameter settings for the shared single-adapter 38-task InstrDialog++ experiment (LS4).

Backbone	Tasks	Epochs per Task	LR	Train/Eval BS	Grad Acc	Max Source/Target Length	Checkpoint Selection	Weight Decay
T5-Large	38	15	1e-3	8 / 16	2	1,024 / 128	Best dev ROUGE-L	0
LLaMA-2-7B	38	15	2e-4	4 / 4	4	1,024 / 128	Lowest dev loss	0

adapter protocol. We use all examples in the processed training and test files without additional runtime subsampling.

Each input contains the task family, dataset name, a natural-language instruction, the candidate output labels, the original input, and an `Answer:` suffix. The exact task-family instructions used for LS1 $\sim$ LS3 are listed in Table 17. Candidate output verbalizers are read directly from the released processed task files.

Predictions are produced by greedy auto-regressive generation and evaluated using normalized exact match; no candidate-constrained probability ranking is applied. Each task is trained for exactly one epoch without development-set checkpoint selection. The shared adapter is carried to the next task, while the optimizer is reinitialized at each task transition.

For LS4, we use the no-initial-instruction-tuning setting corresponding to CITB's FT-NO-INIT: training starts directly from the pretrained T5-Large or LLaMA-2-7B backbone, without initial instruction tuning on the 100 auxiliary tasks. We follow the task construction, instruction formatting, and data-splitting procedure implemented in the released CITB long-stream code.

In the boundary-aware LS4 experiments, training runs for all 15 epochs on each task. T5-Large selects the best shared-adapter checkpoint according to development ROUGE-L, while LLaMA-2-7B selects the checkpoint with the lowest development loss. The selected adapter is carried to the next task, and the optimizer is reinitialized at each task transition.

The Hidden-abrupt and Blurry-20% variants use T5-Large with the same data, adapter configuration, learning rate, effective batch size, and total training-example budget as the boundary-aware LS4 experiment. They do not provide explicit task IDs or boundary indicators to the learner, and a single optimizer state is maintained continuously throughout the stream. Hidden-abrupt directly concatenates the latent task blocks. For Blurry-20%, at each transition we pool and shuffle a boundary region whose width on each side is

$$w_t = \mathrm{round}(0.2 \min(|B_t|, |B_{t+1}|)),$$

where B_t and B_{t+1} are the adjacent repeated task blocks. This construction preserves the total number of examples and optimizer updates. No task-specific development evaluation or model selection is performed in the boundary-free variants.

For LS1 $\sim$ LS3, task performance is measured by normalized exact-match accuracy. For LS4 and its boundary-free variants, performance is measured by ROUGE-L with stemming. We record each task's post-learning score and evaluate all tasks once at the end of the sequence. In the boundary-free variants, a task's post-learning score is recorded at the first optimizer-step checkpoint after all of its training examples have appeared; these diagnostic evaluations do not select or modify the model.

Table 17: Task-family instructions used to construct the text-to-text inputs for LS1 $\sim$ LS3.

Task Family	Instruction
NLI	What is the logical relationship between the "sentence 1" and the "sentence 2"? Choose one from the option.
QQP	Whether the "first sentence" and the "second sentence" have the same meaning? Choose one from the option.
SC	What is the sentiment of the following paragraph? Choose one from the option.
TC	What is the topic of the following paragraph? Choose one from the option.
BoolQA	According to the following passage, is the question true or false? Choose one from the option.
MultiRC	According to the following passage and question, is the candidate answer true or false? Choose one from the option.
WiC	Given a word and two sentences, whether the word is used with the same sense in both sentence? Choose one from the option.
COPA	No additional task-family instruction is added; the released input containing the premise, question, and two answer choices is used directly.

Table 18: Task-wise hyperparameter settings for the shared single-adapter sequential language modeling experiments. The corpus order is fixed as WikiText $\rightarrow$ PTB $\rightarrow$ Enwik8.

Backbone	Corpus	Epochs	LR	Train/Eval BS	Grad Acc	Block Size	Stride
GPT-2 medium	WikiText	3	2e-4	32 / 32	1	512	256
	PTB	3	2e-4	32 / 32	1	512	256
	Enwik8	3	2e-4	32 / 32	1	512	256
LLaMA-2-7B	WikiText	3	2e-4	4 / 4	4	512	256
	PTB	3	2e-4	4 / 4	4	512	256
	Enwik8	3	2e-4	4 / 4	4	512	256
Qwen1.5-7B	WikiText	3	2e-4	4 / 4	4	512	256
	PTB	3	2e-4	4 / 4	4	512	256
	Enwik8	3	2e-4	4 / 4	4	512	256

G.5 Sequential Language Modeling

For sequential language modeling, the corpus order is fixed as WikiText $\rightarrow$ PTB $\rightarrow$ Enwik8. LoRA and McLoRA use the same target modules under the same backbone. Specifically, the target modules are {`c_attn`} for GPT-2 medium, {`q_proj, v_proj`} for LLaMA-2-7B, and {`q_proj, k_proj, v_proj, o_proj`} for Qwen1.5-7B. The task-wise hyperparameter settings are summarized in Table 18.

G.6 Training-Efficiency Profiling

Table 19 reports the empirical training efficiency of LoRA and McLoRA using RoBERTa-large on MNLI, SST-2, RTE, QNLI, and QQP. Both methods use rank $r = 8$, target modules {`query, key, value`}, dropout 0, batch size 16, gradient accumulation 4, FP16, and AdamW. For each run, we first perform 20 profiling warm-up steps and then measure a consecutive window of 100 CUDA-synchronized training steps, corresponding to 6,400 processed samples. Timing includes `zero_grad`, forward, backward, `optimizer.step`, and batch transfer, but excludes model loading, tokenization, and data preprocessing.

Across the five tasks, McLoRA increases the observed training-step time and peak allocated memory by macro-averages of 3.2% and 2.7%, respectively. Overall, McLoRA has training speed and peak GPU memory comparable to LoRA, with no appreciable overall training overhead.

Table 19: Training-efficiency profiling of LoRA and McLoRA using RoBERTa-large under matched settings.

Task	Method	Trainable Params	Training Step (ms)	Throughput (samples/s)	Peak Allocated (GB)
MNLI	LoRA	2.232M	192.492	332.57	2.353
	McLoRA	2.232M	195.864	326.94	2.415
SST-2	LoRA	2.231M	187.327	341.82	1.412
	McLoRA	2.231M	186.189	343.78	1.432
RTE	LoRA	2.231M	195.317	327.74	4.501
	McLoRA	2.231M	199.580	320.69	4.660
QNLI	LoRA	2.231M	187.401	341.57	3.757
	McLoRA	2.231M	194.685	328.77	3.878
QQP	LoRA	2.231M	180.900	353.80	2.400
	McLoRA	2.231M	196.587	325.70	2.469

Table 20: Task-wise training hyperparameters for the AdaLoRA/Mc-AdaLoRA and DoRA/Mc-DoRA experiments reported in Table 4. All experiments use the shared single-adapter GLUE-5 setting on S2 $\sim$ S4.

Backbone	Variant Family	Task	Epochs	LR	Train/Eval BS	Grad Acc	Max Length	Weight Decay
LLaMA-2-7B	AdaLoRA/DoRA variants	MNLI	3	2e-5	16 / 16	4	128	0.01
		QQP	5	2e-5	16 / 16	4	512	0.01
		QNLI	3	2e-5	8 / 16	8	512	0.01
		SST-2	3	2e-5	16 / 16	1	128	0.01
		RTE	5	2e-4	8 / 8	1	512	0.01
RoBERTa-large	AdaLoRA variants	MNLI	5	3e-4	64 / 64	1	128	0.01
		QQP	10	3e-4	64 / 64	1	512	0.01
		QNLI	5	2e-4	64 / 64	1	512	0.01
		SST-2	5	2e-5	16 / 16	1	128	0.01
		RTE	10	4e-4	8 / 8	1	512	0.01
	DoRA variants	MNLI	5	3e-4	64 / 64	1	128	0.01
		QQP	10	3e-4	64 / 64	1	512	0.01
		QNLI	5	2e-4	32 / 64	2	512	0.01
		SST-2	5	2e-5	16 / 16	1	128	0.01
		RTE	10	4e-4	8 / 8	8	512	0.01

G.7 Experimental Details for Variant Extensions

This subsection summarizes the training settings for the variant-extension experiments in Tables 4 and 5. Unless otherwise specified, these experiments follow the same shared single-adapter sequential GLUE-5 setting as in Section 4.1 and are conducted on the three GLUE-5 task orders S2 $\sim$ S4 defined there. We therefore do not repeat the task sequences here.

Table 20 reports the task-wise training hyperparameters for the AdaLoRA/Mc-AdaLoRA and DoRA/Mc-DoRA experiments. Within each variant pair, the additive and multiplicative versions share the same task-wise training hyperparameters.

Table 21: Training settings for the supplementary multi-adapter boundary-extension experiment under the O-LoRA protocol. We follow the same benchmark, task orders, and general training protocol as O-LoRA (Wang et al., 2023).

Backbone	Orders	Epochs per Task	LR	Train/Eval BS	Grad Acc	Max Source Length	Max Target Length	Generation Max Length	LR Scheduler	Warmup Steps
T5-large	Order 1 $\sim$ 3	1	1e-3	8 / 128	1	512	50	50	constant	0

For LLaMA-2-7B, AdaLoRA and Mc-AdaLoRA use the same task-wise training hyperparameters, the same rank setting ($r = 8$, target rank of 4), the same target modules {q_proj, v_proj, down_proj}, and the same AdaLoRA schedule parameters (init_r=8, tinit=200, tfinal=1000, and deltaT=10). Likewise, DoRA and Mc-DoRA use the same task-wise training hyperparameters, the same rank setting ($r = 8$), the same target modules, and use_dora=True.

For RoBERTa-large, AdaLoRA and Mc-AdaLoRA share the same task-wise training hyperparameters, the same rank-scheduling parameters (init_r=8, target rank of 4, tinit=200, tfinal=1000, and deltaT=10), and the same target modules {query, key, value}. DoRA and Mc-DoRA likewise share the same task-wise training hyperparameters, the same rank setting ($r = 8$), use_dora=True, and the same target modules.

For sigmoid-McLoRA, the training setup is exactly the same as that of the corresponding LoRA/McLoRA experiments in the shared single-adapter GLUE-5 setting. These experiments are conducted on Qwen1.5-7B and RoBERTa-large over S2 $\sim$ S4; the only change is the bounded sigmoid parameterization described in Section 3.1.

G.8 Boundary Extension under the O-LoRA Protocol

For the supplementary multi-adapter boundary-extension experiment in Section 4.2, we follow the benchmark setting and general training protocol of O-LoRA (Wang et al., 2023). A new adapter is introduced for each task, while previously learned adapters are frozen. The backbone model is T5-large. For Inc-McLoRA, each newly introduced adapter uses rank $r = 8$, matching the compared methods. The adapter is applied to the default T5 LoRA target modules used by the PEFT implementation.

The three task orders used here coincide exactly with S5 $\sim$ S7, but this setting is reported separately because it follows a different multi-adapter training and evaluation protocol. We denote them as Order 1 $\sim$ 3 to align with the naming convention of prior multi-adapter work. We restate the orders here for convenience:

- **Order 1:** DBpedia $\rightarrow$ Amazon $\rightarrow$ Yahoo $\rightarrow$ AG News;
- **Order 2:** DBpedia $\rightarrow$ Amazon $\rightarrow$ AG News $\rightarrow$ Yahoo;
- **Order 3:** Yahoo $\rightarrow$ Amazon $\rightarrow$ AG News $\rightarrow$ DBpedia.

Table 21 summarizes the training settings used in this supplementary experiment. Following the O-LoRA setup, we use one training epoch for each task, a constant learning-rate schedule, and zero warmup. Our Inc-McLoRA replaces the additive composition in incremental LoRA with multiplicative composition, while keeping the remaining protocol unchanged.

H Experimental Details for Single-Task Fine-tuning

This appendix presents the experimental setup, results, dataset documentation, and training details for the single-task fine-tuning experiments.

H.1 Experimental Setup

For discriminative tasks, we use the GLUE benchmark, including RTE, MNLI, SST-2, QQP, QNLI, CoLA, MRPC, and STS-B, with RoBERTa-large as the backbone.

Table 22: Single-task fine-tuning results on GLUE and commonsense reasoning benchmarks.

GLUE, RoBERTa-large									
Method	**Avg**	**RTE**	**MNLI**	**SST-2**	**QQP**	**QNLI**	**CoLA**	**MRPC**	**STS-B**
McLoRA	87.6	**84.8**	**90.6**	**95.4**	91.0	93.8	62.1	91.4	91.6
LoRA	**87.8**	84.5	90.5	**95.4**	**91.2**	**94.3**	**63.3**	**91.5**	**91.8**
Commonsense, LLaMA-2-7B									
Method	**Avg**	**ARC-C**	**ARC-E**	**BoolQ**	**HellaSwag**	**OBQA**	**PIQA**	**Wino**	**SIQA**
McLoRA	65.9	49.7	**72.6**	83.9	70.9	47.2	**79.6**	**70.0**	53.0
LoRA	**66.6**	**51.5**	72.4	**85.0**	**73.4**	**49.4**	78.9	66.3	**55.5**

Table 23: Dataset documentation for the commonsense reasoning benchmarks.

Dataset	**Task Type**	**Text Domain / Source**
ARC-C / ARC-E	Science question answering	Grade-school science questions
BoolQ	Boolean question answering	Naturally occurring questions with Wikipedia passages
HellaSwag	Commonsense completion	Activity and event descriptions
OBQA	Open-book question answering	Elementary science questions
PIQA	Physical commonsense reasoning	Physical interaction scenarios
WinoGrande	Coreference commonsense reasoning	Pronoun/coreference-style commonsense examples
SIQA	Social commonsense reasoning	Social interaction scenarios

For commonsense reasoning, we use a commonly adopted suite of eight benchmarks (Hu et al., 2023), including ARC-C, ARC-E, BoolQ, HellaSwag, OBQA, PIQA, WinoGrande, and SIQA, with LLaMA-2-7B as the backbone. All tasks use the official data splits. We report the default metric for each benchmark and the average score.

LoRA and McLoRA use the same training configuration for the same backbone and task. Specifically, both methods use rank $r = 8$ on RoBERTa-large and rank $r = 16$ on LLaMA-2-7B. We use AdamW with a linear learning rate schedule. Further implementation details are provided in the following subsections.

H.2 Results

Table 22 shows that McLoRA and LoRA achieve comparable overall performance. On RoBERTa-large, the average GLUE score differs by only 0.2 points; on LLaMA-2-7B, the average commonsense reasoning score differs by only 0.7 points. Despite minor fluctuations on individual tasks, neither method exhibits a systematic advantage. Thus, replacing additive composition with multiplicative composition does not impair single-task fine-tuning, providing a fair basis for the sequential learning comparisons.

H.3 Dataset Documentation

The GLUE datasets are documented in Table 10 of Appendix G. This subsection therefore documents the commonsense reasoning benchmarks. The commonsense reasoning benchmarks used in the single-task fine-tuning experiments are English-language benchmarks. Table 23 summarizes the task type and text domain/source of these benchmarks.

Table 24: Task-wise hyperparameter settings for the RoBERTa-large single-task GLUE experiments.

Task	Epochs	LR	Train/Eval BS	Grad Acc	Max Length	Weight Decay
MNLI	5	3e-4	64 / 64	1	128	0.01
QQP	10	3e-4	64 / 64	1	512	0.01
QNLI	5	2e-4	64 / 64	1	512	0.01
SST-2	5	2e-5	16 / 16	1	128	0.01
RTE	10	4e-4	8 / 8	1	512	0.01
CoLA	10	2e-4	64 / 64	1	128	0.01
MRPC	10	3e-4	64 / 64	1	512	0.01
STS-B	15	2e-4	64 / 64	1	512	0.01

Table 25: Training hyperparameters for the LLaMA-2-7B single-task commonsense reasoning setting.

Backbone	Train Data	Epochs	LR	Train BS	Grad Acc	Max Length	Weight Decay	Warmup Ratio
LLaMA-2-7B	`commonsense_170k`	3	1e-4	8	4	512	0.01	0.03

H.4 RoBERTa-large on GLUE

Table 24 reports the task-wise hyperparameter settings for the RoBERTa-large single-task GLUE experiments. For both LoRA and McLoRA on RoBERTa-large, the effective rank budget is 8, and the target modules are {`query, key, value`}.

H.5 LLaMA-2-7B on Commonsense Reasoning

For the LLaMA-2-7B single-task setting, we train a single adapter on `zwhe99/commonsense_170k` and evaluate the resulting model on the eight commonsense reasoning benchmarks reported in Table 22. We reserve 1% of the training set as validation data, evaluate and save checkpoints at the end of each epoch, and use early stopping with a patience of 1. Table 25 summarizes the corresponding training hyperparameters.

For both LoRA and McLoRA, we use rank $r = 16$ and target modules {`q_proj, k_proj, v_proj, o_proj`}.

I Limitations

Our main setting focuses on shared single-adapter sequential learning, where the same LoRA module is updated across tasks. Although the supplementary multi-adapter extension shows promising results, the mechanism behind multiplicative composition in multi-adapter settings remains underexplored and deserves further study.

Although we evaluate McLoRA on multiple backbone models and benchmarks, further validation on broader model families, task types, and continual learning settings would help better establish the generality of McLoRA itself.

Finally, while we show that multiplicative composition improves representative additive LoRA variants, a broader study of more LoRA-based methods in diverse continual learning settings would help further clarify the applicability boundary of multiplicative composition.

Table 26: Licenses and terms of use for the main artifacts used in this paper.

Artifact	Use in this paper	License / Terms
RoBERTa-large	Backbone model	MIT
GPT-2 medium	Backbone model	Modified MIT
T5-large	Backbone model	Apache-2.0
LLaMA-2-7B	Backbone model	Llama 2 Community License
Qwen1.5-7B	Backbone model	Tongyi Qianwen License / model-card terms
Qwen3-8B-Base	Backbone model	Apache-2.0
GLUE tasks	NLU benchmark tasks	Licenses/terms vary by constituent dataset; used via public benchmark releases
SuperGLUE tasks	Long-sequence classification tasks	Licenses/terms vary by constituent dataset; used via public benchmark releases
IMDB and Yelp Reviews	Long-sequence sentiment classification	Terms follow the corresponding public dataset releases
CITB / InstrDialog++	38-task continual instruction-tuning stream	Public research release based on Super-NaturalInstructions; no repository-wide license is explicitly specified, and constituent dataset terms apply
DBpedia	Standard CL benchmark	CC-BY-SA-3.0 / terms of the dataset release used
Amazon Polarity	Standard CL benchmark	Apache-2.0
Amazon Reviews	15-task Long Sequence	Terms follow the public five-class Amazon Reviews dataset release used in the processed benchmark files
AG News	Standard CL benchmark	License not explicitly specified in the public dataset card used
Yahoo Answers Topics	Standard CL benchmark	License not explicitly specified in the public dataset card used
WikiText	Language modeling benchmark	CC-BY-SA-3.0 / GFDL
PTB	Language modeling benchmark	LDC User Agreement / Penn Treebank terms
Enwik8	Language modeling benchmark	Wikipedia-derived corpus; source terms apply
Commonsense benchmarks	Single-task evaluation	Public benchmark releases; licenses/terms follow the corresponding dataset cards or official releases

J General Artifact and Reproducibility Details

This appendix provides artifact license and usage information, computational-budget information, and software/package settings that apply to the experiments in this paper.

J.1 Licenses and Terms of Use

Table 26 summarizes the licenses or terms of use of the main artifacts used in this paper. We use all artifacts only for research evaluation, do not redistribute model weights or dataset contents, and follow the corresponding licenses and model/dataset-card terms where specified.

J.2 Model Sizes and Computational Budget

This subsection summarizes model sizes and computational budget for all experiments in the paper. The backbone models used in our experiments have approximately 355M parameters for RoBERTa-large and GPT-2 medium, 770M parameters for T5-large, 7B parameters for LLaMA-2-7B and Qwen1.5-7B, and 8.2B parameters for Qwen3-8B-Base. The experiments were run on NVIDIA A40 GPUs with 48 GB memory and NVIDIA A800 GPUs with 80 GB memory. RoBERTa-large, GPT-2 medium, and the short-sequence T5-Large experiments used one A40 GPU per run, whereas the long-sequence T5-Large experiments and all 7B- and 8B-scale experiments used one A800 GPU

per run. Including final reported runs, pilot runs, and hyperparameter checks, the total computational budget was approximately 1,400 $\sim$ 1,800 GPU hours.

J.3 Software and Package Settings

Our experiments are implemented with PyTorch, Hugging Face Transformers, Hugging Face Datasets, PEFT, and rouge-score. We use the official evaluation metrics associated with each benchmark when available. The main package versions are PyTorch 2.7.1, Transformers 4.56.1, Datasets 2.16.1, PEFT 0.17.1, rouge-score 0.1.2, CUDA 12.6, and Python 3.12.11. Unless otherwise specified, we use the default package settings.