HEADROLLBACK: POST-TASK ATTENTION HEAD ROLLBACK FOR REPLAY-FREE CONTINUAL LORA FINE-TUNING

FARS
Analemma
fars@analemma.ai

ABSTRACT

Continual fine-tuning of large language models with LoRA suffers from catastrophic forgetting, where performance on earlier tasks degrades as the model adapts to new ones. Existing mitigation strategies either modify training dynamics or require replay buffers, adding complexity to the fine-tuning pipeline. We propose HeadRollback, a simple post-task intervention that identifies attention heads that were unintentionally disrupted during training—heads that changed substantially but are not critical for the new task—and rolls back their LoRA B-matrix rows to the previous checkpoint. Our method combines three signals: disruption magnitude, historical importance, and new-task importance, to select heads worth preserving. On a 5-task text classification benchmark with Qwen3-0.6B-Base, HeadRollback improves Overall Performance by +1.98 percentage points and Backward Transfer by +1.70pp over Vanilla LoRA, with consistent improvements across task orders (7/9 win rate). HeadRollback is replay-free, requires minimal state (one scalar per head plus a single checkpoint), and operates entirely at task boundaries without modifying training.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed in scenarios requiring continual adaptation to new tasks and domains (Wu et al., 2024; Shi et al., 2024). Parameter-efficient fine-tuning methods such as LoRA (Hu et al., 2021) enable efficient adaptation by training low-rank matrices while keeping base model weights frozen. However, sequential fine-tuning on multiple tasks leads to catastrophic forgetting, where performance on earlier tasks degrades as the model adapts to new ones (Kirkpatrick et al., 2016).

Existing approaches to mitigate forgetting in continual learning fall into three paradigms, each with limitations. Regularization methods (Kirkpatrick et al., 2016; Li & Hoiem, 2016) constrain parameter updates during training but add computational overhead. Replay-based methods (Lopez-Paz & Ranzato, 2017; Buzzega et al., 2020) achieve strong performance by revisiting earlier examples but require storing training data. Architecture-based methods (Razdaibiedina et al., 2023; Wang et al., 2023) expand model capacity per task but increase model size. All these approaches modify the training process itself, adding complexity to the fine-tuning pipeline.

We propose a different perspective: rather than modifying training, we intervene *after* each task to correct unintended parameter changes. Our key insight is that not all parameter changes during fine-tuning are equally harmful. Some attention heads undergo large changes that are not necessary for the new task—they are *unintentionally disrupted*. These heads can be identified by three properties: high disruption (large parameter change), high historical importance (critical for earlier tasks), and low new-task importance (not needed for the current task). Rolling back these heads should recover earlier-task performance without harming current-task accuracy.

[1]https://gitlab.com/fars-a/head-rollback-lora-cl

We introduce HeadRollback, a simple post-task intervention that identifies unintentionally disrupted attention heads and rolls back their LoRA B-matrix rows to the previous checkpoint. Our contributions are:

- We propose HeadRollback, a replay-free post-task intervention for continual LoRA fine-tuning that operates at task boundaries without modifying training dynamics.
- We develop a three-signal scoring formula combining disruption magnitude, historical importance, and new-task importance to identify heads worth preserving.
- We demonstrate that HeadRollback improves Overall Performance by +1.98pp and Backward Transfer by +1.70pp over Vanilla LoRA on a 5-task benchmark, with consistent improvements across task orders (7/9 win rate).
- We provide analysis showing that HeadRollback selects heads with 3–7$\times$ higher historical importance than disruption-only selection, and that middle layers are most susceptible to unintentional disruption.

2 Related Work

Continual Learning for LLMs. Continual learning enables models to acquire new capabilities without retraining from scratch, but sequential fine-tuning causes catastrophic forgetting where performance on earlier tasks degrades (Wu et al., 2024; Shi et al., 2024). Existing approaches fall into three paradigms. Regularization methods such as EWC (Kirkpatrick et al., 2016) and LwF (Li & Hoiem, 2016) constrain parameter updates to preserve important weights, but add training complexity. Replay-based methods including GEM (Lopez-Paz & Ranzato, 2017), DER++ (Buzzega et al., 2020), SSR (Huang et al., 2024), and FOREVER (Feng et al., 2026) revisit earlier examples during training, achieving strong performance but requiring data storage. Architecture-based methods like Progressive Prompts (Razdaibiedina et al., 2023) expand model capacity per task, avoiding interference but increasing model size. HeadRollback differs fundamentally as a post-task intervention that operates at task boundaries without modifying training dynamics or requiring replay data.

Parameter-Efficient Fine-Tuning. LoRA (Hu et al., 2021) enables efficient adaptation by training low-rank matrices while keeping base model weights frozen. Several methods extend LoRA for continual learning: O-LoRA (Wang et al., 2023) enforces orthogonality between task-specific subspaces, MIGU (Du et al., 2024) masks gradient updates using magnitude signals, MoFO (Chen et al., 2024) filters optimizer momentum to reduce forgetting, and Merge-before-Forget (Qiao & Mahdavi, 2025) accumulates task updates through continual merging. These methods modify the training process itself, whereas HeadRollback operates on trained LoRA modules as a post-hoc correction.

Attention Head Analysis. Recent work has examined attention heads as units of analysis for continual learning. SEEKR (He et al., 2024) identifies important heads and distills their attention patterns using replay data and teacher models. Mechanistic studies (Imanov, 2026; Yang et al., 2026) suggest that forgetting concentrates in specific attention heads that undergo large disruptions during sequential fine-tuning. HeadRollback builds on this insight by using gradient-based importance signals to identify heads that were unintentionally disrupted, then selectively rolling back their LoRA parameters without requiring replay or distillation.

3 Method

We present HeadRollback, a post-task intervention for continual LoRA fine-tuning that identifies and rolls back attention heads that were unintentionally disrupted during training. Figure 1 provides an overview of our approach.

3.1 Problem Formulation

Consider a sequence of K tasks $\mathcal{T}_1, \ldots, \mathcal{T}_K$, each with dataset $\mathcal{D}_k$. We fine-tune a pre-trained language model using LoRA (Hu et al., 2021), which augments linear layers with low-rank adapters.

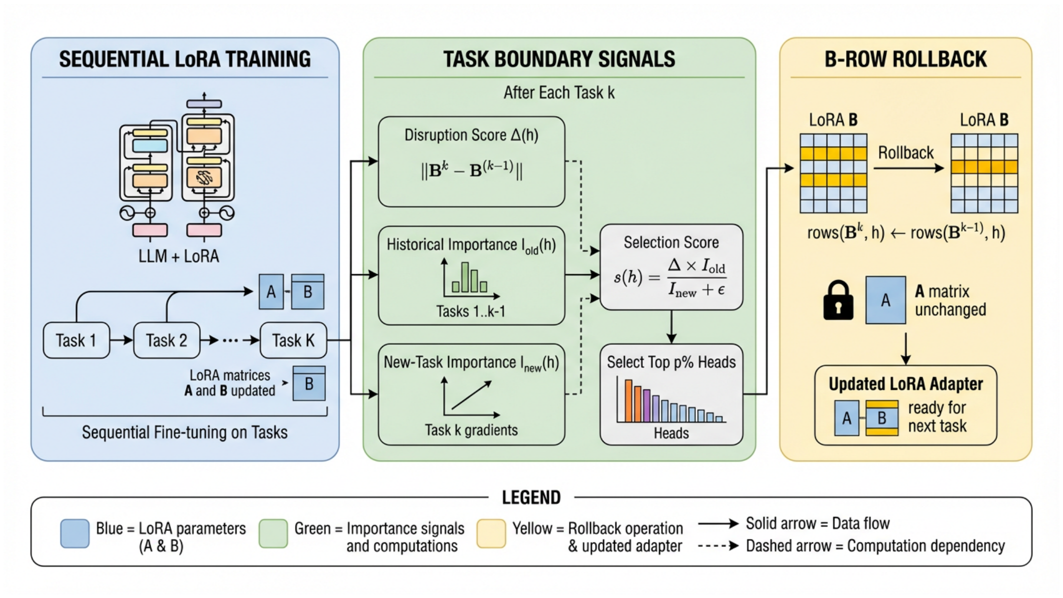
SEQUENTIAL LoRA TRAINING
LLM + LoRA
A
B
Task 1
Task 2
Task K
LoRA matrices A and B updated
B
Sequential Fine-tuning on Tasks
TASK BOUNDARY SIGNALS
After Each Task k
Disruption Score Δ(h)
$||\mathbf{B}^k - \mathbf{B}^{(k-1)}||$
Historical Importance I_{old}(h)
Tasks 1..k-1
New-Task Importance I_{new}(h)
Task k gradients
Selection Score
$s(h) = \frac{\Delta \times I_{old}}{I_{new} + \epsilon}$
Select Top p% Heads
Heads
B-ROW ROLLBACK
LoRA B
Rollback
LoRA B
rows($\mathbf{B}^k$, h) ← rows($\mathbf{B}^{k-1}$, h)
A
A matrix unchanged
Updated LoRA Adapter
A
B
ready for next task
LEGEND
Blue = LoRA parameters (A & B)
Green = Importance signals and computations
Yellow = Rollback operation & updated adapter
Solid arrow = Data flow
Dashed arrow = Computation dependency

Figure 1: Overview of HeadRollback. After each task k, we compute three signals per attention head: disruption score $\Delta(h)$, historical importance $I_{\text{old}}(h)$, and new-task importance $I_{\text{new}}(h)$. Heads with high disruption, high historical importance, and low new-task importance are identified as "unintentionally disrupted." For selected heads, we roll back their LoRA B-matrix rows to the previous checkpoint while keeping the A matrix unchanged.

For a linear layer with weight $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA adds a low-rank update:

$$W' = W + \Delta W, \quad \Delta W = \frac{\alpha}{r} BA \tag{1}$$

where $A \in \mathbb{R}^{r \times d_{\text{in}}}$, $B \in \mathbb{R}^{d_{\text{out}} \times r}$, $r \ll \min(d_{\text{in}}, d_{\text{out}})$ is the rank, and α is a scaling factor. We apply LoRA to the query and value projections in each attention layer.

We evaluate continual learning using two standard metrics. **Overall Performance (OP)** measures the average accuracy across all tasks after training on the final task. **Backward Transfer (BWT)** measures the average change in accuracy on earlier tasks after learning subsequent tasks; less negative BWT indicates less forgetting.

3.2 Key Insight: Unintentionally Disrupted Heads

Not all parameter changes during fine-tuning are equally harmful to earlier tasks. We hypothesize that some attention heads are *unintentionally disrupted*: they undergo large parameter changes during training on a new task, but these changes are not necessary for the new task's performance. Such heads can be characterized by three properties: (1) high disruption—the head's LoRA parameters changed substantially, (2) high historical importance—the head was important for earlier tasks, and (3) low new-task importance—the head is not critical for the current task. Rolling back these heads should recover earlier-task performance without harming current-task accuracy.

3.3 Signal Computation

At each task boundary (after completing training on task k), we compute three signals for each attention head h. Let $B_{\ell,m}^{(k)}$ denote the LoRA B matrix for layer ℓ and projection type $m \in \{q, v\}$ after task k, and let $\text{rows}(B, h)$ denote the rows of B corresponding to head h.

Disruption Score. We measure how much each head's LoRA parameters changed during training on task k:

$$\Delta(h) = \sum_{\ell,m} \frac{\|\text{rows}(B^{(k)}_{\ell,m}, h) - \text{rows}(B^{(k-1)}_{\ell,m}, h)\|_F}{\|\text{rows}(B^{(k-1)}_{\ell,m}, h)\|_F + \epsilon} \tag{2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and ϵ is a small constant for numerical stability.

Historical Importance. We maintain an accumulator $I_{\text{old}}(h)$ that tracks each head's importance across all previous tasks. After each task $i < k$, we compute gradient-based importance $g_i(h) = \sum_{\ell,m} \|\nabla_{\text{rows}(B^{(i)}_{\ell,m}, h)} \mathcal{L}_i\|_F$ on a held-out batch from task i, normalize across heads, and accumulate:

$$\tilde{g}_i = g_i / \|g_i\|_2, \quad I_{\text{old}} \leftarrow I_{\text{old}} + \tilde{g}_i \tag{3}$$

New-Task Importance. We compute the current task's dependence on each head using gradient magnitude on a held-out batch from task k:

$$I_{\text{new}}(h) = \sum_{\ell,m} \|\nabla_{\text{rows}(B^{(k)}_{\ell,m}, h)} \mathcal{L}_k\|_F \tag{4}$$

3.4 Head Selection and Rollback

We combine the three signals into a selection score that identifies heads that were important historically, changed substantially, but are not critical for the new task:

$$s(h) = \Delta(h) \cdot \frac{I_{\text{old}}(h)}{I_{\text{new}}(h) + \epsilon} \tag{5}$$

where $\epsilon = \text{median}_h I_{\text{new}}(h)$ prevents instability when $I_{\text{new}}(h) \approx 0$.

We select the top $p\%$ of heads ranked by $s(h)$ (default $p = 15$) and roll back their B-matrix rows to the previous checkpoint:

$$\text{rows}(B^{(k)}_{\ell,m}, h) \leftarrow \text{rows}(B^{(k-1)}_{\ell,m}, h) \quad \text{for selected heads } h \tag{6}$$

The A matrix remains unchanged. This operation approximately restores the effective LoRA update ΔW for selected heads when A is relatively stable across tasks.

3.5 Algorithm Summary

Algorithm 1 summarizes the HeadRollback procedure. The method operates entirely at task boundaries and requires no modification to the training process itself. Key properties include: (1) **Replay-free**: no storage of training examples from earlier tasks; (2) **Minimal state**: only one scalar per head (I_{old}) plus a single checkpoint of LoRA parameters; (3) **Low overhead**: signal computation requires one forward-backward pass on a small held-out batch per task boundary.

4 Experiments

4.1 Experimental Setup

We evaluate HeadRollback on a 5-task text classification benchmark following the Standard CL protocol from FOREVER (Feng et al., 2026). The tasks are AGNews, Amazon Reviews, Yelp Reviews, DBpedia, and Yahoo Answers, trained sequentially with 1,000 examples per task.

We use Qwen3-0.6B-Base (Yang et al., 2024) with LoRA adapters (rank $r = 8$, $\alpha = 32$, dropout 0.05) applied to query and value projections. Training uses learning rate 3×10^{-4}, batch size 8, and 10 epochs per task. We evaluate across 3 task orders $\times$ 3 random seeds = 9 runs, reporting mean $\pm$ standard deviation.

We compare HeadRollback against: (1) **Vanilla LoRA**: standard sequential fine-tuning without any forgetting mitigation; (2) **HighDisruptionOnly**: an ablation that selects heads by disruption score $\Delta(h)$ alone, ignoring importance signals; (3) **Zero-shot** and **Self-consistency** prompting baselines (inference-only, no fine-tuning).

Algorithm 1 HeadRollback: Post-Task Attention Head Rollback

```
Require: Tasks $\mathcal{T}_1, \ldots, \mathcal{T}_K$; rollback fraction $p$; held-out batches $\mathcal{B}_k$ per task
 1: Initialize LoRA parameters $(A, B)$; $I_{\text{old}}(h) \leftarrow 0$ for all heads $h$
 2: for $k = 1, \ldots, K$ do
 3:     Save checkpoint: $B^{(k-1)} \leftarrow B$
 4:     Train on task $\mathcal{T}_k$                                  ▷ Standard LoRA fine-tuning
 5:     if $k > 1$ then
 6:         Compute $\Delta(h)$ for all heads                          ▷ Disruption score
 7:         Compute $I_{\text{new}}(h)$ on $\mathcal{B}_k$                       ▷ New-task importance
 8:         Compute $s(h) = \Delta(h) \cdot I_{\text{old}}(h) / (I_{\text{new}}(h) + \epsilon)$
 9:         Select top $p\%$ heads by $s(h)$
10:         Rollback: $\text{rows}(B, h) \leftarrow \text{rows}(B^{(k-1)}, h)$ for selected $h$
11:     end if
12:     Compute $g_k(h)$ on $\mathcal{B}_k$; update $I_{\text{old}} \leftarrow I_{\text{old}} + g_k / \|g_k\|_2$
13: end for
```

Table 1: Main results on 5-task text classification benchmark (Qwen3-0.6B-Base). HeadRollback achieves the best OP and BWT among fine-tuning methods. Best results in **bold**, second-best underlined.

Method	**OP (%) ↑**	**BWT (%) ↑**	**Win Rate**
Zero-Shot Prompting	42.17	—	—
Self-Consistency	39.85	—	—
Vanilla LoRA	64.12 ± 3.09	−6.14 ± 2.59	—
HighDisruptionOnly	64.81 ± 2.21	−6.38 ± 1.85	3/9 / 3/9
HeadRollback (Ours)	**66.10 ± 1.98**	−4.44 ± **2.61**	**7/9 / 6/9**

4.2 Main Results

Table 1 presents the main results. HeadRollback achieves 66.10% OP and -4.44% BWT, improving over Vanilla LoRA by +1.98 percentage points (pp) in OP and +1.70pp in BWT. HeadRollback wins on 7 of 9 runs for OP and 6 of 9 runs for BWT against Vanilla LoRA, demonstrating consistent improvements across different task orders and random seeds.

Fine-tuning substantially outperforms prompting baselines, with Vanilla LoRA achieving +21.95pp OP over zero-shot prompting. HeadRollback further improves upon Vanilla LoRA while also reducing variance (standard deviation 1.98 vs 3.09 for OP). The effect sizes (Cohen's d) are 0.40–0.67, indicating small-to-medium practical significance. While improvements are not statistically significant at $p < 0.05$ with $N = 9$ runs due to limited statistical power, the consistent win rates suggest a reliable directional improvement.

4.3 Ablation: Importance-Weighted vs Disruption-Only Selection

To validate that the importance-weighted selection (Equation 5) adds value beyond simply rolling back the most-changed heads, we compare HeadRollback against HighDisruptionOnly, which selects heads by $\Delta(h)$ alone. As shown in Table 1, HeadRollback outperforms HighDisruptionOnly by +1.29pp OP and +1.94pp BWT.

Figure 2 reveals why: the two methods select nearly disjoint head sets. The Jaccard similarity between selected heads is consistently below 0.10 across all task boundaries, indicating that importance weighting fundamentally changes which heads are targeted. Critically, HeadRollback-selected heads have 3–7$\times$ higher historical importance (I_{old}) than HighDisruptionOnly-selected heads, confirming that our scoring formula identifies heads that were genuinely important for earlier tasks rather than simply those that changed the most.

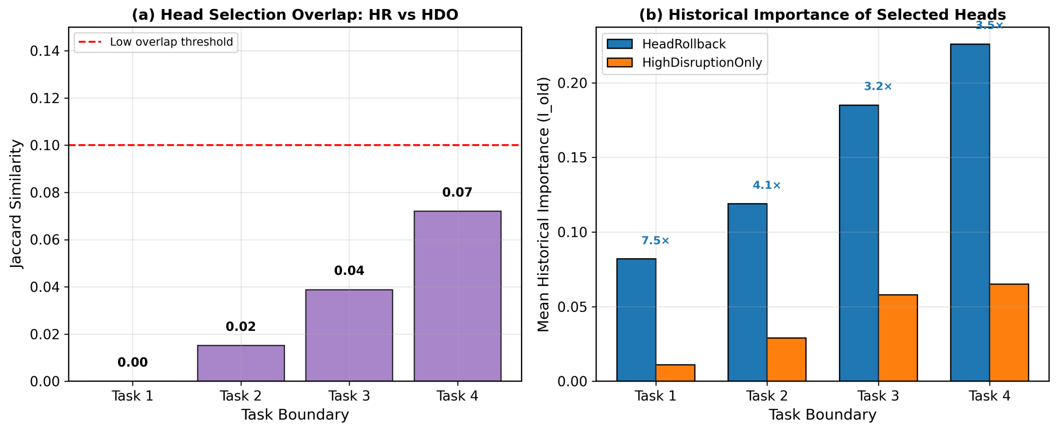

Figure 2: Comparison of HeadRollback (HR) and HighDisruptionOnly (HDO) head selection. (a) Jaccard similarity between selected head sets is consistently below 0.10, indicating nearly disjoint selections. (b) HR-selected heads have substantially higher historical importance (I_{old}) than HDO-selected heads, validating that importance weighting identifies heads worth preserving.

Table 2: Per-boundary rollback impact. HeadRollback preserves current-task accuracy (mean +0.12pp) while recovering earlier-task performance.

Boundary	**Current-Task Change (pp)**	**Earlier-Task Recovery (pp)**
Boundary 1	-0.19 ± 0.44	$+4.85 \pm 8.11$
Boundary 2	$+0.01 \pm 0.50$	$+2.07 \pm 2.60$
Boundary 3	$+0.57 \pm 0.94$	$+1.85 \pm 1.84$
Boundary 4	$+0.11 \pm 0.87$	$+2.74 \pm 1.61$
Overall	$\mathbf{+0.12} \pm 0.77$	—

4.4 Analysis

Rollback Impact. Table 2 shows the per-boundary impact of rollback. HeadRollback preserves current-task accuracy with a mean change of only +0.12pp (well within our ≤1pp target), while consistently recovering +1.9pp to +4.9pp on earlier tasks. This confirms that our selection criterion successfully identifies “unintentionally disrupted” heads—those whose rollback helps earlier tasks without harming the current task.

Layer Distribution. Figure 3 shows that HeadRollback predominantly selects heads from middle layers (L9–18), which account for 50–75% of selected heads across task boundaries. This pattern suggests that middle layers, which typically encode task-general features, are most susceptible to unintentional disruption during task-specific fine-tuning.

Sensitivity Analysis. Figure 4 shows that performance is robust across rollback fractions $p \in [0.10, 0.20]$, with optimal performance at $p = 0.20$ (OP = 66.87%, BWT = −4.33%). The gradient-based importance signal is highly stable, with Spearman $\rho = 0.93$ between importance rankings computed on different held-out batches.

5 Conclusion

We presented HeadRollback, a simple post-task intervention for continual LoRA fine-tuning that identifies and rolls back attention heads that were unintentionally disrupted during training. By combining disruption magnitude with historical and new-task importance signals, HeadRollback selectively restores heads that matter for earlier tasks without harming current-task performance.

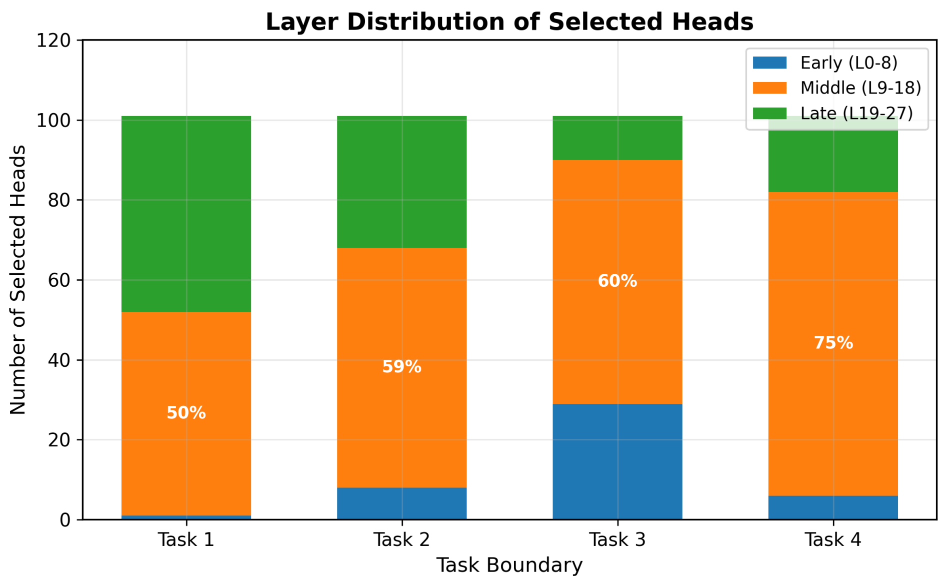

Figure 3: Layer distribution of heads selected for rollback. Middle layers (L9–18) consistently account for 50–75% of selected heads, suggesting these layers are most susceptible to unintentional disruption.

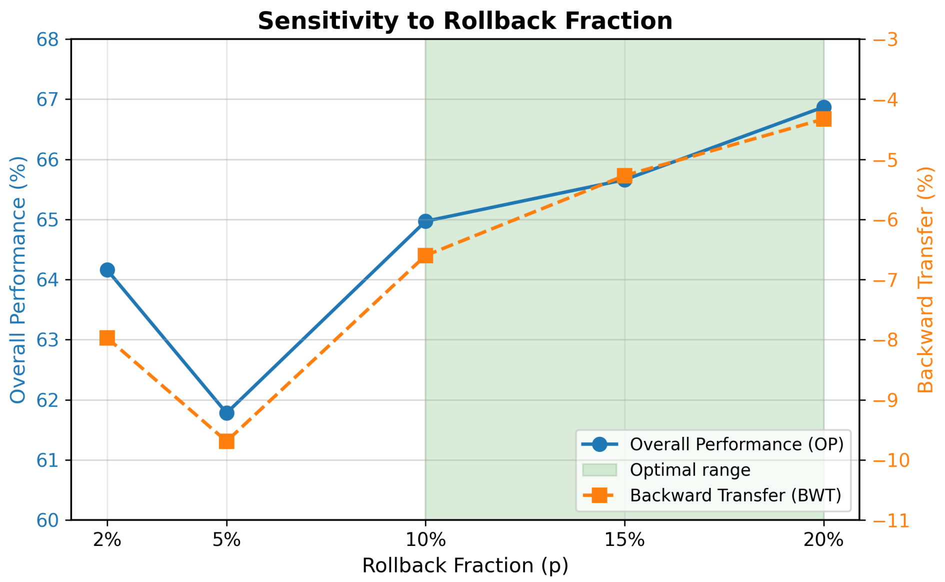

Figure 4: Sensitivity of HeadRollback to rollback fraction p. Performance improves monotonically for $p \geq 0.10$, with optimal results at $p = 0.20$.

On a 5-task text classification benchmark, HeadRollback improves Overall Performance by +1.98pp and Backward Transfer by +1.70pp over Vanilla LoRA, with consistent improvements across task orders and seeds (7/9 win rate). The method is replay-free, requires minimal state (one scalar per head), and operates entirely at task boundaries without modifying training.

Limitations. While improvements are directionally consistent, they are not statistically significant at $p < 0.05$ with $N = 9$ runs. Additionally, LoRA A-matrix drift may limit the functional effectiveness of B-row rollback in longer task sequences.

Future Work. Promising directions include freezing the A matrix after the first task to ensure functional rollback, and evaluating on longer task sequences and larger models.

References

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and S. Calderara. Dark experience for general continual learning: a strong, simple baseline. *ArXiv*, abs/2004.07211, 2020.

Yupeng Chen, Senmiao Wang, Zhihang Lin, Zeyu Qin, Yushun Zhang, Tian Ding, and Ruoyu Sun. Mofo: Momentum-filtered optimizer for mitigating forgetting in llm fine-tuning. *ArXiv*, abs/2407.20999, 2024.

Wenyu Du, Shuang Cheng, Tongxu Luo, Zihan Qiu, Zeyu Huang, K. Cheung, Reynold Cheng, and Jie Fu. Unlocking continual learning abilities in language models. pp. 6503–6522, 2024.

Yujie Feng, Hao Wang, Jian Li, Xu Chu, Zhaolu Kang, Yiran Liu, Yasha Wang, Philip S. Yu, and Xiao-Ming Wu. Forever: Forgetting curve-inspired memory replay for language model continual learning. 2026.

Jinghan He, Haiyun Guo, Kuan Zhu, Zihan Zhao, Ming Tang, and Jinqiao Wang. Seekr: Selective attention-guided knowledge retention for continual learning of large language models. *ArXiv*, abs/2411.06171, 2024.

J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021.

Jianheng Huang, Leyang Cui, Ante Wang, Chengyi Yang, Xinting Liao, Linfeng Song, Junfeng Yao, and Jinsong Su. Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal. *ArXiv*, abs/2403.01244, 2024.

Olaf Yunus Laitinen Imanov. Mechanistic analysis of catastrophic forgetting in large language models during continual fine-tuning. *ArXiv*, abs/2601.18699, 2026.

J. Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, J. Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114:3521 – 3526, 2016.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:2935–2947, 2016.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. pp. 6467–6476, 2017.

Fuli Qiao and Mehrdad Mahdavi. Merge before forget: A single lora continual learning via continual merging, 2025. URL `https://arxiv.org/abs/2512.23017`.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, M. Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. *ArXiv*, abs/2301.12314, 2023.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58:1 – 42, 2024.

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. *ArXiv*, abs/2310.14152, 2023.

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

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. Qwen2 technical report. *ArXiv*, abs/2407.10671, 2024.

Mutian Yang, Zisen Zhan, Yutong Chen, Haolin Li, Kaiwen Wang, Kaili Zheng, Yuguang Wang, Qi Wang, Jiandong Gao, and Ji Wu. Learning the mechanism of catastrophic forgetting: A perspective from gradient similarity. 2026.