

HLoRA: Hierarchical Regularization on Combined Parameter Space for LoRA Fine-Tuning

Shezheng Song*, Hao Xu*, Long Peng[†], Xiaopeng Li, Shasha Li[†], Jie Yu
College of Computer Science and Technology, National University of Defense Technology
{ssz614, xuhao19, penglong, xiaopengli, shashali, yj}@nudt.edu.cn

Abstract

Low-Rank Adaptation (LoRA) is widely used for fine-tuning large language models, but suffers from catastrophic forgetting. Existing continual learning methods mitigate this by estimating per-parameter importance independently. However, this assumption breaks down under LoRA’s factorization $\Delta W = BA$: infinitely many pairs (B, A) yield the same weight update, making importance scores on A and B separately unreliable. We propose HLoRA, which computes path-integral importance in the combined weight update space, bypassing the unreliable separate estimation on A and B . HLoRA applies hierarchical regularization at two levels: element-wise constraints preserve parameters critical for general knowledge, and layer-wise coefficients adapt regularization strength based on each layer’s aggregate importance. Experiments on scientific, medical, and physical tasks using GPT-J and LLaMA-3 show that HLoRA mitigates catastrophic forgetting while enhancing adaptability, running approximately 20x faster and requiring only 10%–15% of the storage compared to prior methods. Code is released on <https://github.com/season1blue/HLoRA>.

1 Introduction

Large Language Models (LLMs) are pretrained on large and diverse corpora, giving them broad general capabilities in reasoning, language understanding, and world knowledge (Wang and Komatsuzaki, 2021; Touvron et al., 2023b; OpenAI, 2024). However, applying LLMs to specialized domains often requires fine-tuning on domain-specific data. Low-Rank Adaptation (LoRA) (Hu et al., 2022) is now the most widely used parameter-efficient fine-tuning method, updating only low-rank matrices B and A while keeping pretrained weights

*Equal contribution.

[†]Corresponding authors.

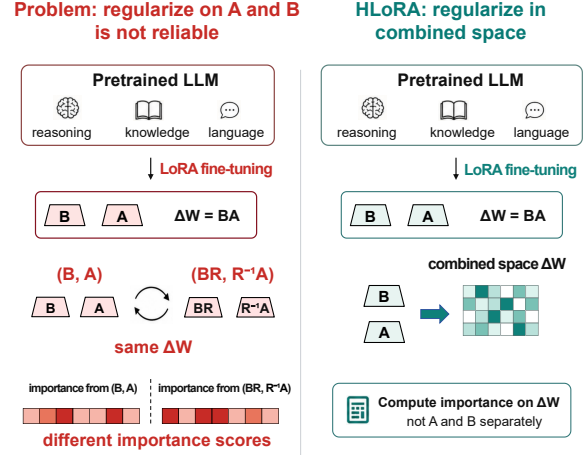


Figure 1: Dilemmas faced by traditional LoRA in mitigating catastrophic forgetting via regularization methods and our solutions.

frozen. Despite its efficiency, LoRA does not prevent **catastrophic forgetting** (Kirkpatrick et al., 2017; Kemker et al., 2018; Shao and Feng, 2022; Ren et al., 2025). Catastrophic forgetting refers to the phenomenon where a model, during fine-tuning, loses or overwrites knowledge learned during pre-training. This issue poses a severe limitation on the broader applicability of LLMs. A fixed data composition and format in fine-tuning further worsens this forgetting: on domain tasks, the model lacks the general reasoning and background knowledge needed to produce smooth, well-connected answers; on general tasks, it may even fail to answer questions it could handle before.

As shown in Figure 1, regularization-based continual learning methods such as EWC (Kirkpatrick et al., 2017) and SI (Zenke et al., 2017) address forgetting by measuring how important each parameter is and penalizing changes to important ones. These methods work well in standard neural networks where each parameter can be treated independently. However, they do not work correctly with LoRA: the actual weight update $\Delta W = BA$ lives on a low-rank manifold, not a flat parame-

ter space. For any orthogonal matrix R , the pairs (B, A) and $(BR, R^{-1}A)$ produce the same ΔW , but importance scores computed on A and B separately would give different results. This means importance estimation on the factored matrices is unreliable, as it depends on the specific factorization rather than the true contribution to model behavior.

This creates a specific challenge for mitigating forgetting in LoRA fine-tuning. A suitable method should protect the parts of the update that are important for general knowledge, but the importance should be measured in the combined update space ΔW , not separately on A and B . It should also keep the low cost of LoRA, since expensive importance computation would reduce the practical value of parameter-efficient tuning. However, existing methods do not satisfy these requirements. EW-CLoRA (Zheng et al., 2026) computes the Fisher Information Matrix over LoRA parameters, but it still estimates importance in the factored space and therefore suffers from the same factorization issue. It is also expensive: for GPT-J-6B, computing the Fisher matrix takes 22 hours on an A800 GPU and requires 23GB of storage, with costs growing for larger models.

We propose **HLoRA** (Hierarchical Low-Rank Adaptation), which solves this problem by computing parameter importance in the **combined parameter space** $\Delta W = BA$, where the actual weight update lives, instead of on A and B separately. Our main idea is to reformulate path integrals over the combined weight update, producing importance scores that reflect the actual contribution of each weight element to loss reduction, rather than depending on the arbitrary factorization of B and A . Based on this importance estimation, HLoRA applies regularization at two levels. At the element level, importance scores from the combined space control how much each parameter can change: important parameters are kept close to their general-task values, while less important ones are free to adapt. At the layer level, we observe that different layers play different roles. Some layers store general linguistic knowledge while others handle task-specific patterns. We use the total importance of each layer to adjust regularization strength, so that layers critical for general knowledge are strongly protected while other layers can adapt more freely.

Through extensive experiments on scientific, physical, and medical tasks using LLMs (GPT-J and LLaMA-3), we demonstrate that our frame-

work achieves state-of-the-art performance, mitigating catastrophic forgetting while enhancing LLM adaptability. To maintain general capabilities, it is essential to identify and quantify the importance of various parameters that contribute to these capabilities. The computation of parameter importance is typically time-consuming, and storing the associated weights requires substantial memory resources. Our experimental results demonstrate that our method is nearly **20 times faster** and requires only **10%~15% of the storage memory** compared to the previous method, demonstrating the practicality of our approach. Our contributions are as follows:

- We identify a fundamental incompatibility between per-parameter importance estimation and LoRA’s low-rank factorization: importance scores computed on A and B separately are not invariant to reparameterization. We resolve this by deriving path-integral importance in the combined space.
- We propose the element-wise and layer-wise importance metrics to dynamically adjust parameter updates, preserving critical general knowledge while allowing domain-specific expertise to be learned effectively.
- Our method achieves state-of-the-art performance across multiple datasets using mainstream backbone LLMs. It significantly reduces computational time (20x faster) and storage (10%~15%) for parameter importance estimation compared to prior methods.

2 Related Work

Continual Learning Regularization-based methods mitigate catastrophic forgetting by constraining updates to parameters important for previous tasks. EWC (Kirkpatrick et al., 2017) estimates importance using the diagonal Fisher Information Matrix, SI (Zenke et al., 2017) accumulates each parameter’s contribution to loss reduction along the optimization trajectory, and MAS (Aljundi et al., 2018) measures importance by the gradient magnitude of model outputs with respect to parameters. These methods share a common assumption: importance is computed in a *flat parameter space*, where each trainable parameter independently affects the model output. While this is suitable for standard neural networks, it becomes problematic for low-rank reparameterization methods such as

LoRA (Hu et al., 2022). In LoRA, the effective update $\delta = BA$ is a *bilinear function* of two factor matrices, making the importance of any element in A or B inherently dependent on the other factor. Other continual learning paradigms include architecture-based methods (Wu et al., 2024; Wang et al., 2023a, 2022; Chen et al., 2022), replay-based methods (Jin et al., 2022; Liu et al., 2021; Qin et al., 2022; Bai et al., 2022), and distillation-based methods (Li and Hoiem, 2017; Cao et al., 2021; Shao and Feng, 2022; Gu et al., 2022; Qin and Joty, 2021). Although effective in certain settings, they either introduce substantial storage/computation overhead or are orthogonal to the importance estimation problem addressed in this work.

Catastrophic Forgetting in LoRA Fine-Tuning

LoRA (Hu et al., 2022) is widely used for parameter-efficient adaptation of LLMs (Touvron et al., 2023a,b) due to its low training and storage cost (Zhang et al., 2023). However, sequential LoRA fine-tuning still suffers from catastrophic forgetting. Prior studies have compared LoRA with full fine-tuning in terms of knowledge retention (Biderman et al., 2024) and analyzed forgetting through the lens of intruder dimensions in the weight spectrum (Shuttleworth et al., 2024). Several methods attempt to mitigate forgetting in LoRA. LoRAMoE (Dou et al., 2023) separates knowledge through multiple LoRA experts, CLoRA (Lu et al., 2025) constrains updates via subspace regularization, and O-LoRA (Wang et al., 2023b) modifies the adapter structure to reduce task interference. However, these methods do not directly address *parameter importance estimation under low-rank factorization*: LoRAMoE relies on architectural separation, CLoRA applies uniform directional constraints, and O-LoRA redesigns the adapter without quantifying per-parameter importance. To our knowledge, no prior work reconciles per-element importance estimation, as used in EWC, SI, and MAS, with the factored parameter space of LoRA. Since the effective update $\delta = BA$ couples the two factors, independent importance scores on A or B are semantically unreliable. Our work fills this gap by estimating importance directly in the merged weight space while remaining compatible with LoRA’s low-rank structure.

3 Methodology

Regularization-based continual learning methods estimate the importance of each parameter and pe-

nalize modifications to critical ones during subsequent training (Zenke et al., 2017; Kirkpatrick et al., 2017; Aljundi et al., 2018). These methods share a core assumption: each trainable parameter independently contributes to the loss, so importance can be estimated per parameter. This assumption holds for standard neural networks where each weight is directly optimized. However, LoRA introduces a low-rank factorization $\delta = BA$ where the trainable parameters A and B are coupled through matrix multiplication. We show that this structural difference renders naive per-parameter importance estimation ill-defined, and derive a principled solution that operates in the combined weight update space.

As shown in Figure 2, our framework consists of two phases: (1) recording element-wise parameter importance in the combined space $\Delta W = BA$ during general task α training, and (2) applying hierarchical regularization, both element-wise and layer-wise, during domain task β fine-tuning.

3.1 Why Per-Parameter Importance Fails for LoRA

Consider applying a standard importance estimation method (e.g., SI (Zenke et al., 2017) or EWC (Kirkpatrick et al., 2017)) directly to the LoRA matrices A and B . Such methods would compute separate importance scores Ω_i^A for each element of A and Ω_j^B for each element of B . For EWC, this means computing the diagonal Fisher information matrix on the elements of A and B independently; for SI, this means accumulating the path integral along each element’s individual trajectory.

This approach has a fundamental flaw rooted in the non-uniqueness of the low-rank factorization. For any invertible matrix $R \in \mathbb{R}^{r \times r}$, the transformation $B' = BR$, $A' = R^{-1}A$ yields the same effective weight update:

$$B'A' = BR \cdot R^{-1}A = BA = \delta \quad (1)$$

However, the per-element importance scores computed on A' and B' generally differ from those computed on A and B , because the individual elements have changed despite the product remaining identical. This means that importance estimation on the factored parameters is not invariant to reparameterization. The same effective model behavior can yield arbitrarily different importance scores depending on the particular factorization.

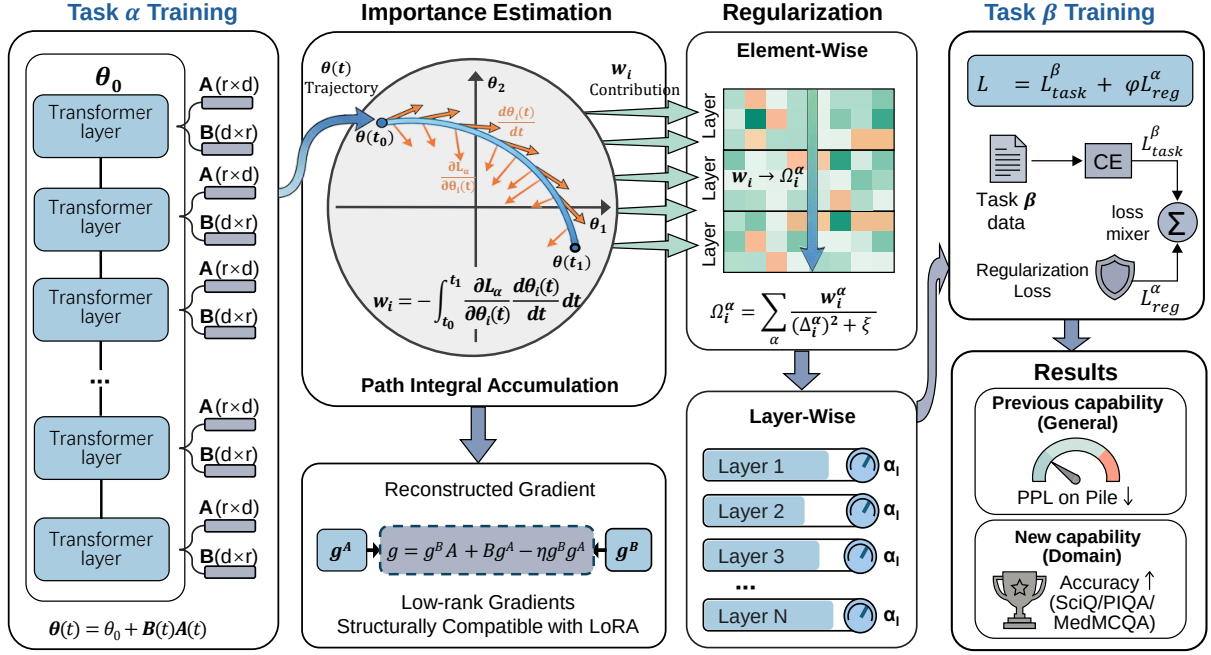


Figure 2: Overview of HLoRA: adaptive constraint combining element-wise and layer-wise importance to preserve the capabilities from the α task while learning new abilities for the β task.

This inconsistency is not merely theoretical. During training, the optimizer trajectory through (B, A) space does not uniquely determine which individual elements of A or B are “important” for the effective update $\delta = BA$. A single element of B may appear to contribute significantly to loss reduction, but this contribution is inseparable from the corresponding row of A that it multiplies. Constraining individual elements of A and B based on such scores can therefore both over-constrain unimportant directions and under-constrain important ones, leading to suboptimal preservation of general knowledge.

3.2 Importance Estimation in the Combined Space

To resolve this inconsistency, we estimate importance directly on the effective weight update $\delta = BA$ rather than on A and B separately. This ensures that importance scores reflect the actual contribution of each weight perturbation to loss reduction, independent of the particular low-rank factorization. The key idea is to track how each element of the combined parameter $\theta = \theta_0 + BA$ contributes to reducing the task loss along the entire training trajectory.

Path integral formulation. Following the path integral framework (Zenke et al., 2017), the change in loss during task α training from time t_0 to t_1 can

be decomposed as a sum of per-parameter contributions by integrating the gradient along the parameter trajectory:

$$L(\theta_{t_1}) - L(\theta_{t_0}) = \sum_i \int_{t_0}^{t_1} g(\theta_i(t)) \theta'_i(t) dt \quad (2)$$

where $g(\theta_i(t)) = \frac{\partial L}{\partial \theta_i}$ is the gradient of the loss with respect to parameter i and $\theta'_i(t) = \frac{\partial \theta_i}{\partial t}$ is the parameter velocity. Since training generally decreases the loss, we define the contribution of parameter i to loss reduction with a sign convention:

$$\omega_i = - \int_{t_0}^{t_1} g(\theta_i(t)) \theta'_i(t) dt \quad (3)$$

so that a larger ω_i indicates greater importance for maintaining task α performance. Intuitively, ω_i accumulates the “work” done by parameter i in reducing the loss throughout the training process.

Deriving the effective gradient for LoRA. Under LoRA, the effective parameters are $\theta(t) = \theta_0 + B(t)A(t)$, where θ_0 is frozen. The low-rank matrices are updated via gradient descent:

$$\begin{aligned} B(t+1) &= B(t) - \eta g^B(t) \\ A(t+1) &= A(t) - \eta g^A(t) \end{aligned} \quad (4)$$

where η is the learning rate, $g^B(t) = \frac{\partial L}{\partial B}$ and $g^A(t) = \frac{\partial L}{\partial A}$ are the gradients of the loss with respect to B and A , respectively. To compute eq. (3),

we need the change of the effective parameter θ per step. Expanding the product $B(t+1)A(t+1)$:

$$\begin{aligned} B(t+1)A(t+1) &= B(t)A(t) \\ &\quad - \eta(g^B(t)A(t) + B(t)g^A(t)) \\ &\quad + \eta^2 g^B(t)g^A(t) \end{aligned} \quad (5)$$

The effective parameter change per step is therefore:

$$\begin{aligned} \Delta\theta(t) &= B(t+1)A(t+1) - B(t)A(t) \\ &= -\eta(g^B(t)A(t) + B(t)g^A(t)) \\ &\quad - \eta g^B(t)g^A(t) \end{aligned} \quad (6)$$

In standard full fine-tuning, the parameter change is $\Delta\theta = -\eta g$, where g is the gradient. By analogy, we can view LoRA as inducing an effective gradient $\tilde{g}(t)$ on the full weight space such that $\Delta\theta(t) = -\eta \tilde{g}(t)$ (Wang et al., 2024). Comparing with eq. (6), we identify:

$$\tilde{g}(t) = g^B(t)A(t) + B(t)g^A(t) - \eta g^B(t)g^A(t) \quad (7)$$

The first two terms correspond to the product rule applied to $\frac{d}{dt}(BA)$, capturing how changes in B and A jointly affect the effective weight. The third term $\eta g^B g^A$ is a second-order correction arising from the simultaneous update of both matrices within the same step.

Substituting eq. (7) and eq. (6) into eq. (3), the path integral for LoRA importance is computed in the combined parameter space:

$$\omega_i = - \int_{t_0}^{t_1} \tilde{g}_i(t) \Delta\theta_i(t) dt \quad (8)$$

This formulation measures importance through the effective weight change $\Delta\theta(t) = B(t+1)A(t+1) - B(t)A(t)$, which is uniquely determined by the product BA rather than by the individual factors. In practice, we approximate the integral in eq. (8) by discrete summation over training steps: $\omega_i \approx -\sum_t \tilde{g}_i(t) \cdot \Delta\theta_i(t)$, which can be accumulated online during the forward-backward pass of task α training with negligible overhead.

Importance score. The element-wise importance score for parameter i after task α is:

$$\Omega_i^\alpha = \frac{\omega_i^\alpha}{(\Delta_i^\alpha)^2 + \xi} \quad (9)$$

where $\Delta_i^\alpha = (B(t^\alpha)A(t^\alpha))_i$ is the total change of the i -th effective parameter during task α (since B

is initialized to zero, $\delta(t^0) = 0$). The denominator $(\Delta_i^\alpha)^2$ normalizes the raw contribution ω_i^α by the magnitude of the parameter change, ensuring that the importance score reflects per-unit-change contribution rather than being biased toward parameters with large absolute updates. The constant $\xi > 0$ prevents division by zero for parameters that remain near their initial values.

3.3 Element-Wise Regularization in Domain Tuning

During fine-tuning on domain task β , we penalize deviations from the parameter values that are important for preserving general knowledge from task α . The element-wise regularization loss for the l -th layer is:

$$\mathcal{L}_{\text{reg},l}^\alpha = \sum_i \Omega_i^\alpha (\theta_i^t - \theta_i^\alpha)^2 \quad (10)$$

where θ_i^α is the reference parameter value after task α , and θ_i^t is the current parameter during task β training. Parameters with high importance scores Ω_i^α are constrained to remain close to their post- α values, while less important parameters are free to adapt to the new domain.

3.4 Layer-Wise Coefficient Regularization

We compute the importance of each layer based on its contribution to the parameters learned in the α task. This layer-specific importance metric allows the model to dynamically adjust the regularization across different layers. The layer-wise weighted regularization is defined as :

$$\mathcal{L}_{\text{reg}}^\alpha = \sum_l \text{softmax}(\|\Omega_l^\alpha\|_2) \mathcal{L}_{\text{reg},l}^\alpha \quad (11)$$

where $\|\Omega_l^\alpha\|_2$ denotes the L2 norm of the parameter importance matrix Ω_l^α for the l -th layer, which reflects the significance of the parameters learned in the α task. The total loss for the β task is defined as:

$$\mathcal{L}^\beta = \mathcal{L}_{\text{task}}^\beta + \varphi \mathcal{L}_{\text{reg}}^\alpha \quad (12)$$

The use of this adaptive regularization $\mathcal{L}_{\text{reg}}^\alpha$ helps mitigate catastrophic forgetting by maintaining the integrity of essential features learned in prior tasks. φ is the hyperparameter that controls the weight of the domain ($\mathcal{L}_{\text{task}}$) and general ($\mathcal{L}_{\text{reg}}$) ability of LLM.

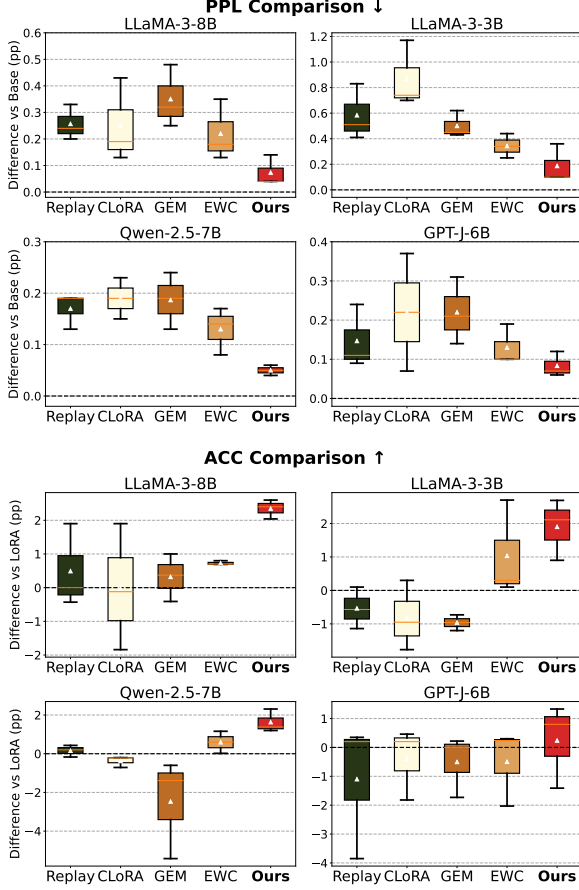


Figure 3: Single-task continual learning performance compared with LoRA. The gray baseline indicates parity with LoRA (0 pp).

4 Experiments

4.1 Metrics and Baselines

We first perform fine-tuning on a single task (Section 4.2), and then extend the experiment to multi-task fine-tuning scenarios (Section 4.3). Following previous work (Zheng et al., 2026; Schick et al., 2023), the Pile dataset (Gao et al., 2020) is taken as the evaluation dataset for general ability. And three representative tasks are selected: (1) *Medicine*: MedMCQA (Pal et al., 2022). (2) *Science*: SciQ (Welbl et al., 2017). (3) *Physics*: PiQA (Bisk et al., 2020). Perplexity ($\text{PPL}\downarrow$) of Pile dataset is used to assess general capability and accuracy ($\text{Acc}\uparrow$) is used to evaluate domain capabilities. In the experiment, the batch size is 20, and the learning rate is $8e-4$. The LoRA rank is set to 8, with alpha 32. Detailed results with different LoRA ranks can be found in Appendix B.3.

Our baseline includes: (1) *LoRA* (Hu et al., 2022): The basic continual learning setting. (2) *EWC* (Kirkpatrick et al., 2017): A method in which the Fisher matrix is computed and regularization

Table 1: Ablation of layer-wise (l) and element-wise (e) weighted regularization on single-task learning.

Method	SciQ		PiQA		Med	
	PPL↓	Acc↑	PPL↓	Acc↑	PPL↓	Acc↑
LLaMA-3-8B						
Ours	3.81	98.10	3.91	51.90	3.81	61.35
w/o l	3.90	97.70	4.09	50.30	3.91	60.05
w/o l, e	3.97	97.60	4.10	49.30	4.01	58.88
LLaMA-3-3B						
Ours	4.64	97.10	4.90	51.14	4.64	55.80
w/o l	4.75	96.80	4.96	49.70	4.74	54.41
w/o l, e	4.95	96.30	5.37	47.31	5.05	53.12
GPT-J-6B						
Ours	3.35	96.80	3.40	50.49	3.34	36.10
w/o l	3.36	96.30	3.41	49.95	3.35	35.62
w/o l, e	3.39	96.20	3.52	49.51	3.37	33.66
Qwen-2.5-7B						
Ours	3.60	97.60	3.61	49.84	3.62	60.00
w/o l	3.63	97.40	3.68	49.18	3.64	59.33
w/o l, e	3.75	96.60	3.75	48.29	3.69	58.12

constraints are applied to preserve important parameters. (3) *Replay* (Rolnick et al., 2019): A data-based method that replays samples from previous tasks, where old task data is revisited and integrated during new task learning. (4) *GEM* (Saha et al., 2021): A method using orthogonal projection, where the gradient subspace important to the old tasks is maintained through parameter updates. (5) *L2P* (Wang et al., 2022): A method using dynamic prompt selection, where the input is used to select and update prompts from the prompt pool in an instance-wise fashion. (6) *CLoRA* (Lu et al., 2025): combines low-rank adaptation with subspace regularization.

4.2 Single-Task Continual Learning

Result Analysis We evaluate the model performance **after fine-tuning on a single new task**, aiming to measure both the retention of general knowledge acquired during pretraining and the acquisition of task-specific knowledge. Figure 3 visualizes the *relative* performance of several continual learning baselines *against the standard LoRA fine-tuning*. Each box summarizes the statistics of performance differences on three evaluation domains (SciQ, PiQA, and Med), where the box plot reflects the median and variability across domains. The horizontal gray line denotes parity with LoRA, indicating zero performance difference. For PPL (top), values above the gray line mean a larger perplexity than LoRA (worse general capability),

Table 2: Continual learning performance. Order indicates the sequence of continual fine-tuning across domains. PPL↓ measures general capability; Acc₁/Acc₂ are Task-1/Task-2 accuracies. Δ_p is the PPL change and BWT (Lopez-Paz and Ranzato, 2017) is backward transfer.

LLaMA-3-8B															
Method	Order 1					Order 2					Order 3				
	PPL↓	Acc ₁ ↑	Acc ₂ ↑	Δ _p ↓	BWT↑	PPL↓	Acc ₁ ↑	Acc ₂ ↑	Δ _p ↓	BWT↑	PPL↓	Acc ₁ ↑	Acc ₂ ↑	Δ _p ↓	BWT↑
LoRA	4.11	96.50	57.63	0.12	-1.4	4.40	93.80	48.91	0.41	-4.1	4.43	44.65	49.08	0.41	-14.3
EWC	4.04	96.70	58.57	0.12	-0.8	4.24	92.90	50.05	0.31	-4.7	4.24	37.38	49.02	0.27	-22.5
REPLAY	4.37	97.00	57.54	0.38	-0.2	4.55	94.60	51.58	0.56	-3.3	4.48	53.43	48.75	0.47	-5.3
GEM	4.13	96.60	57.33	0.14	-1.4	4.41	93.30	48.48	0.42	-4.5	4.43	33.94	48.91	0.40	-25.6
L2P	5.51	87.50	11.02	1.23	-10.1	4.36	92.10	49.46	0.25	-4.6	4.59	32.77	48.64	0.31	-5.7
CLoRA	4.14	96.90	56.94	0.15	-1.1	4.39	92.50	49.13	0.40	-5.4	4.37	40.76	50.05	0.36	-18.5
Ours	3.89	97.50	58.52	0.02	0.1	4.06	95.80	50.27	0.20	-1.8	4.10	56.87	50.71	0.24	-2.3

GPT-J-6B															
Method	Order 1					Order 2					Order 3				
	PPL↓	Acc ₁ ↑	Acc ₂ ↑	Δ _p ↓	BWT↑	PPL↓	Acc ₁ ↑	Acc ₂ ↑	Δ _p ↓	BWT↑	PPL↓	Acc ₁ ↑	Acc ₂ ↑	Δ _p ↓	BWT↑
LoRA	3.53	90.50	37.81	0.10	-5.7	3.64	91.80	50.84	0.21	-4.4	3.62	30.64	48.37	0.17	-5.4
EWC	3.47	87.60	37.10	0.11	-8.8	3.48	92.00	50.53	0.12	-4.4	3.53	31.55	49.40	0.18	-4.6
REPLAY	3.73	94.70	36.88	0.30	-1.7	3.84	93.40	51.44	0.41	-3.0	3.67	30.72	49.29	0.22	-4.9
GEM	3.51	90.90	38.79	0.08	-5.5	3.64	93.30	48.75	0.21	-3.1	3.62	26.55	49.18	0.17	-9.1
L2P	4.28	27.60	21.53	0.74	-18.6	5.37	37.00	49.89	1.81	-9.2	6.15	15.77	50.00	2.18	-5.2
CLoRA	3.53	90.40	37.74	0.09	-6.0	3.65	93.10	49.51	0.21	-3.3	3.63	28.49	49.89	0.18	-7.3
Ours	3.36	95.20	37.46	0.00	-1.5	3.43	95.20	53.19	0.07	-1.5	3.43	32.27	49.40	0.08	-3.4

while values closer to or below the baseline indicate better preservation of general knowledge. For accuracy (bottom), values above the baseline indicate higher task performance than LoRA. Across different LLMs, our approach could preserve general capability (smaller PPL increase) and simultaneously improve task-specific performance (higher accuracy) after single-task continual fine-tuning.

Ablation Study As shown in Table 1, we conduct single-task continual fine-tuning ablation experiments to analyze the contribution of different components in HLoRA. Specifically, we examine two configurations: (1) *layer*, which removes the differentiation of importance among layers, and (2) *layer, element*, which further removes the element-wise importance, effectively disabling all hierarchical regularization. Both ablations increase PPL and reduce accuracy, showing that element-wise and layer-wise constraints are both necessary for balancing retention and adaptation.

4.3 Multi-Task Continual Learning

Result Analysis Table 2 reports the performance under the multi-task continual fine-tuning setting across three task orders. All metrics are evaluated *after completing* the continual fine-tuning procedure. PPL measures general capability (lower is better). Acc₁ and Acc₂ denote the accuracies on Task-1 and Task-2, respectively. Δ_P is the PPL change of Pile dataset, and BWT (Lopez-Paz and Ranzato, 2017) measures backward transfer.

Across all task orders and both backbone models, our method consistently achieves the lowest PPL and the smallest increase in PPL, indicating superior preservation of general capability after continual fine-tuning. Meanwhile, our approach also yields the highest Acc₁ and the least negative BWT values, demonstrating strong knowledge retention. Moreover, the observed trends are consistent across different task sequences. Regardless of whether the model is fine-tuned in the order of SciQ→Med, SciQ→PiQA, or Med→PiQA, our method maintains stable general performance while effectively preserving previously acquired task-specific knowledge. At the same time, competitive Acc₂ results indicate that this improved stability does not come at the cost of learning new tasks. The detailed case study is presented in Appendix C.

Ablation Study As shown in Table 3, we further investigate the effect of hierarchical regularization under the *multi-task* continual fine-tuning setting. Following the single-task analysis, we focus on how removing different levels of regularization impacts knowledge retention when multiple tasks are learned sequentially. Across different task orders and backbone models, removing layer-wise regularization (w/o l) consistently leads to larger increases in Δ_P and more negative BWT values, indicating increased general-capability drift and stronger forgetting of the first task. When both layer-wise and element-wise regularization are re-

Table 3: Ablation of layer-wise (l) and element-wise (e) weighted regularization on multi-task learning.

GPT-J-6B						
	Method	PPL↓	Acc ₁ ↑	Acc ₂ ↑	Δ_p ↓	BWT↑
Order 1	Ours	3.36	95.20	37.46	0.00	-1.5
	w/o l	3.50	94.00	37.38	0.12	-2.5
	w/o l,e	3.83	90.50	37.81	0.30	-5.7
Order 2	Ours	3.43	95.20	53.19	0.07	-1.5
	w/o l	3.53	92.90	51.20	0.13	-3.6
	w/o l,e	3.64	91.80	50.84	0.21	-4.4
Order 3	Ours	3.43	33.27	50.40	0.08	-2.4
	w/o l	3.52	28.84	50.11	0.12	-5.3
	w/o l,e	3.62	22.64	48.37	0.17	-9.5
LLaMA-3-8B						
	Method	PPL↓	Acc ₁ ↑	Acc ₂ ↑	Δ_p ↓	BWT↑
Order 1	Ours	3.89	96.50	58.52	0.02	-0.9
	w/o l	3.96	95.20	58.37	0.08	-1.7
	w/o l,e	4.11	91.50	57.63	0.12	-5.4
Order 2	Ours	4.02	95.80	50.27	0.16	-1.8
	w/o l	4.18	95.30	48.64	0.25	-2.5
	w/o l,e	4.40	92.80	48.91	0.41	-5.1
Order 3	Ours	4.01	56.87	50.71	0.15	-2.3
	w/o l	4.23	51.54	49.29	0.27	-7.9
	w/o l,e	4.43	44.65	49.08	0.41	-14.3

moved (w/o l,e), performance degradation becomes more severe, with the largest PPL increase and the most pronounced backward transfer.

4.4 Efficiency Analysis

A key motivation for HLoRA is to preserve the lightweight nature of LoRA while mitigating catastrophic forgetting. Since HLoRA accumulates importance online along the standard LoRA training trajectory, it avoids extra forward/backward passes required by prior importance-based methods such as EWC. We compare HLoRA with previous methods in terms of importance-calculation time and storage cost. As shown in Figure 4, HLoRA consistently achieves substantially lower overhead across different backbone models. For example, on LLaMA-3-8B and GPT-J-6B, EWC requires **29.18** and **27.17** hours to compute importance, while HLoRA only takes **2.25** and **1.25** hours, respectively. In terms of storage, EWC requires **31.10** GB on LLaMA-3-8B and **22.65** GB on GPT-J-6B, whereas HLoRA only requires **3.67** GB and **3.51** GB. Overall, HLoRA is nearly **20× faster** and uses only **10%~15%** of the storage memory of prior importance-based methods. The detailed complexity analysis is provided in Appendix B.1.

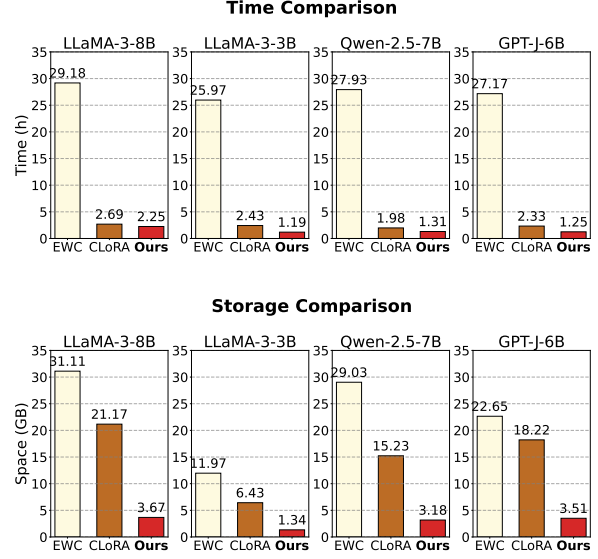


Figure 4: Comparison of computation time and storage.

4.5 Effectiveness Analysis

To demonstrate the effectiveness of our method in addressing the catastrophic forgetting problem, we conduct an analysis from both **intuitive** and **theoretical** perspectives, aiming to verify **why our method works**.

Intuitively: Formally, the parameter contribution θ_i during LoRA fine-tuning is: $\omega_i = -\int_{t_0}^{t_1} \frac{\partial L}{\partial \theta_i}(t) \cdot \frac{d\theta_i}{dt}(t) dt$. This can be interpreted as the loss reduction that can be attributed to parameter θ_i . If a parameter frequently moves in a direction that aligns with gradient and reduces loss, it accumulates a large ω_i , and is thus considered important.

Theoretically: To support this formulation, we adopt a theoretical framework based on quadratic loss functions. Specifically, consider the loss function: $E(\theta) = \frac{1}{2}(\theta - \theta^*)^T H(\theta - \theta^*)$. This setup allows for an explicit solution of the training dynamics under continuous-time gradient descent: $\frac{d\theta}{dt} = -H(\theta - \theta^*)$. From this, we can derive the trajectory $\theta(t)$ and compute the element-wise importance of all parameter updates over time. Specifically, the time-integrated covariance of parameter updates is captured by the matrix: $Q_{ij} = \tau \int_0^\infty \frac{d\theta_i}{dt} \cdot \frac{d\theta_j}{dt} dt$. This matrix Q generalizes the element-wise importance to include correlations between parameter updates. Notably, the element-wise importance $\omega_i = -\int g_i(t) \cdot \frac{d\theta_i}{dt} dt$, corresponds exactly to Q_{ii} . (Zenke et al., 2017) further show that: $\langle Q_{ij} \rangle = \frac{1}{2}\sigma^2 H_{ij}$, where σ^2 is the variance of the initial parameter offset along the Hessian eigenbasis. This equation shows that the element-wise importance (through parameter score $\omega_i = -Q_{ii}$)

captures curvature information proportional to the diagonal of the Hessian H_{ii} , a classical signal of parameter sensitivity.

In summary, the parameter importance estimation in our method is not only an **intuitive loss-contribution measure**, but also **theoretically justified by its link to sensitivity measures**.

5 Conclusion

This paper addresses the critical issue of catastrophic forgetting in Large Language Models (LLMs) during fine-tuning for applications. We propose HLoRA, a hierarchical regularization framework that preserves previously acquired general knowledge while enabling efficient adaptation to new application- or task-specific data. Our approach computes the element-wise importance of model parameters to constrain critical updates, and introduces layer-wise coefficients to dynamically balance the learning between old and new knowledge. Extensive experiments on SciQ, PiQA, and MedMCQA demonstrate that our method effectively mitigates catastrophic forgetting. Compared with prior SOTA methods, HLoRA achieves better stability across continual fine-tuning tasks and offers significantly improved computational and storage efficiency.

6 Limitations

Although HLoRA substantially reduces the computational and storage overhead of importance estimation compared with prior methods, there remains room for further optimization in both time and memory efficiency. In particular, future work could explore more compact representations of importance scores and more efficient accumulation strategies during continual fine-tuning, making the framework more scalable to larger models and longer task sequences.

7 Acknowledgements

We sincerely thank the anonymous reviewers for their insightful and detailed comments. This work is supported by the Science and Technology Innovation Program of Hunan Province under Grants 2025RC3121 and 2024RC1048.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018.

Memory aware synapses: Learning what (not) to forget. In *European conference on computer vision*, pages 144–161. Springer.

Guirong Bai, Shizhu He, Kang Liu, and Jun Zhao. 2022. Incremental intent detection for medical domain with contrast replay networks. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3549–3556.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, and 1 others. 2024. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Yue Cao, Hao-Ran Wei, Boxing Chen, and Xiaojun Wan. 2021. Continual learning for neural machine translation. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, page 3964 – 3974.

Cheng Chen, Yichun Yin, Lifeng Shang, Xin Jiang, Yujia Qin, Fengyu Wang, Zhi Wang, Xiao Chen, Zhiyuan Liu, and Qun Liu. 2022. bert2bert: Towards reusable pretrained language models. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1:2134 – 2148.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, and 1 others. 2023. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *arXiv preprint arXiv:2312.09979*.

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

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, and 1 others. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Shuhao Gu, Bojie Hu, and Yang Feng. 2022. Continual learning of neural machine translation within low forgetting risk regions. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, page 1707 – 1718.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

- Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. 2022. Lifelong pretraining: Continually adapting language models to emerging corpora. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*, page 4764 – 4780.
- Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and 1 others. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.
- Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.
- Qingbin Liu, Pengfei Cao, Cao Liu, Jiansong Chen, Xunliang Cai, Fan Yang, Shizhu He, Kang Liu, and Jun Zhao. 2021. Domain-lifelong learning for dialogue state tracking via knowledge preservation networks. *Conference on Empirical Methods in Natural Language Processing, Proceedings*, page 2301 – 2311.
- David Lopez-Paz and Marc’Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.
- Yuheng Lu, Bingshuo Qian, Caixia Yuan, Huixing Jiang, and Xiaojie Wang. 2025. Controlled low-rank adaptation with subspace regularization for continued training on large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19165–19181.
- OpenAI. 2024. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Ankit Pal, Logesh Kumar Umapathi, and Malaikanan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pages 248–260. PMLR.
- Chengwei Qin and Shafiq Joty. 2021. Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. *arXiv preprint arXiv:2110.07298*.
- Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. Elle: Efficient lifelong pre-training for emerging data. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2789–2810.
- Weijie Ren, Xinlong Li, Lei Wang, Tianxiang Zhao, and Wei Qin. 2025. Analyzing and reducing catastrophic forgetting in parameter efficient tuning. *arXiv preprint arXiv:2402.18865*.
- David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in neural information processing systems*, 32.
- Gobinda Saha, Isha Garg, and Kaushik Roy. 2021. Gradient projection memory for continual learning. *arXiv preprint arXiv:2103.09762*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.
- Chenze Shao and Yang Feng. 2022. Overcoming catastrophic forgetting beyond continual learning: Balanced training for neural machine translation. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1:2023 – 2036.
- Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. 2024. Lora vs full fine-tuning: An illusion of equivalence. *arXiv preprint arXiv:2410.21228*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. <https://github.com/kingoflolz/mesh-transformer-jax>.
- Peihao Wang, Rameswar Panda, and Zhangyang Wang. 2023a. Data efficient neural scaling law via model reusing. *Proceedings of Machine Learning Research*, 202:36193 – 36204.
- Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. 2023b. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10658–10671.
- Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. 2024. Lora-pro: Are low-rank adapters properly optimized? *arXiv preprint arXiv:2407.18242*.

- Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022. Learning to prompt for continual learning. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 139–149. IEEE.
- Johannes Welbl, Nelson F Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*.
- Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ying Shan, and Ping Luo. 2024. Llama pro: Progressive llama with block expansion. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6518–6537.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR.
- Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.
- Yaoyue Zheng, Yin Zhang, Joost van de Weijer, Gido M van de Ven, Shaoyi Du, Xuetao Zhang, and Zhiqiang Tian. 2026. Revisiting weight regularization for low-rank continual learning. *arXiv preprint arXiv:2602.17559*.

A Implementation

A.1 Backbone LLMs

Four mainstream LLMs are used for evaluation: (1) *LLaMA-3-8B* (Dubey et al., 2024) is the third-generation open-source LLM developed by Meta. (2) *LLaMA-3-3B* (Dubey et al., 2024) is a compact variant of the third-generation open-source LLaMA series. (3) *GPT-J-6B* (Wang and Komatsuzaki, 2021) is a GPT-like causal language model which is suitable for generation tasks. (4) *Qwen-2.5-7B* (Yang et al., 2024) is an open-source language model developed by Alibaba.

A.2 Dataset and Prompt

The detailed information about the dataset is listed as follows. **(1) SciQ.** The SciQ dataset contains 13,679 science exam questions, covering various scientific fields such as physics, chemistry, and biology. The dataset is divided into 11,679 questions for training, 1,000 questions for testing, and 100 questions for validation. We use 11,679 questions for training, following the official split. **(2) PiQA.** PiQA is a dataset containing 21,025 questions for physical commonsense reasoning, designed to evaluate and advance the application of natural language understanding systems in everyday physical interaction scenarios. The dataset is divided into 16,113 questions for training, 3,084 for testing, and 1,838 for validation. We use the official split and utilize 16,113 questions for training. **(3) MedMCQA.** MedMCQA dataset is a large collection of multiple-choice question and answer sets designed for real-world medical entrance exam questions, featuring over 194,000 questions selected from the All India Institute of Medical Sciences (AIIMS) entrance exam and the Graduate National Eligibility cum Entrance Test (NEET) in India. To maintain the same data scale as SciQ and PiQA, we randomly selected 20,000 samples from the original dataset as the training set and chose 4,183 samples for evaluation.

Details of the prompts used for training the datasets are provided in Table 4.

A.3 Hyperparameter Settings

Hyperparameters in Baseline Method. To ensure reproducibility of experimental results, we adopt the commonly used hyperparameter settings for all baseline methods, as summarized in Table 5. For LoRA-based approaches, we use a standard configuration with rank $r = 8$, scaling factor $\alpha = 32$,

Table 4: The prompts of different datasets.

Datasets	Prompts
SciQ	Choose an answer for the following question and give your reasons.
PiQA	Select the best option (A, B, C, D) for the physical question below and briefly justify your choice.
MedMCQA	Solve the following medical problem by choosing the correct answer from the following four choices.

Table 5: Hyperparameter settings.

LoRA	
LoRA rank r	8
LoRA α	32
Dropout	0.1
Learning Rate	$8e-4$
LR Scheduler	Linear
Batch Size	20
Epochs	5
LoRA target modules	query, value
Optimizer	AdamW
EWC	
Regularization Weight	0.5
L2P	
Top_k	3
Pool Size	10
Prompt Length	5
Diversity Loss Weight	0.5
Replay	
Buffer Size	1000
CLoRA	
Regularization Vectors num	512
Regularization Weight	1

learning rate 8×10^{-4} , and the AdamW optimizer. For EWC, the regularization weight is set to 0.5 following its original implementation. For L2P, we maintain the same prompt pool size, top- k , and diversity loss weight as reported in prior studies. Replay methods use a buffer size of 1000, and CLoRA is configured with 512 regularization vectors and weight 1. These configurations allow faithful reproduction of baseline performances for reliable comparison.

Regularization Coefficient Analysis in Our Method. Figure 5 demonstrate the effect of the regularization coefficient φ in Equation (12) on PPL and accuracy across three tasks. As φ increases, PPL gradually decreases, indicating a stronger emphasis on preserving previous ability. Higher val-

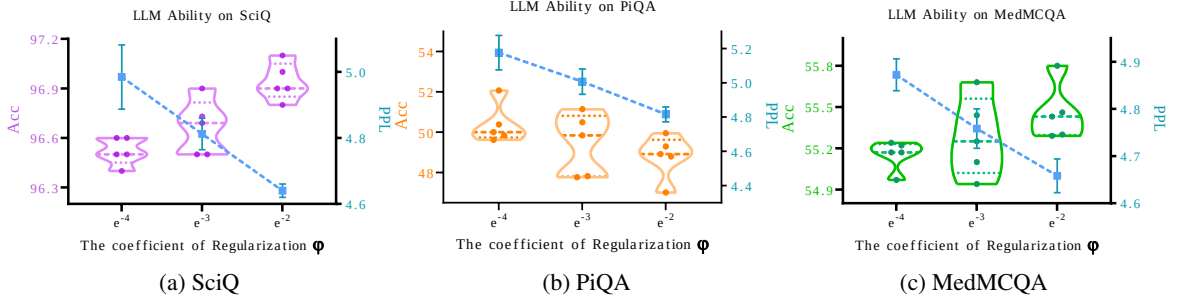


Figure 5: The influence of regularization coefficient φ on LLaMA-3-3B across datasets.

ues of φ correspond to better previous ability retention. However, as shown in Figure 5b, increasing φ negatively impacts the average accuracy on PiQA. Thus, e^{-3} is selected as the optimal value for the regularization coefficient to balance task performance and previous ability (lower PPL).

B Detailed Analysis

B.1 Efficiency Analysis

Time Complexity. HLoRA computes element-wise importance online during standard LoRA training:

$$\omega_i = - \int_{t_0}^{t_1} \frac{\partial \mathcal{L}_\alpha}{\partial \theta_i(t)} \cdot \frac{d\theta_i(t)}{dt} dt. \quad (13)$$

The terms are derived from LoRA matrices $A(t)$, $B(t)$ and their gradients, which are already available during training. Therefore, importance accumulation introduces no additional forward or backward passes. Its time complexity is $\mathcal{O}(T \cdot P)$, where T is the number of training steps and P is the number of LoRA parameters.

In contrast, EWC estimates importance via the diagonal Fisher:

$$F_{i,i} = \frac{1}{N} \sum_{j=1}^N \left(\frac{\partial \mathcal{L}_U^{(j)}}{\partial \theta_i^*} \right)^2, \quad (14)$$

which requires an additional forward and backward pass over N samples, resulting in $\mathcal{O}(N \cdot P)$ extra computation.

Storage Memory. HLoRA maintains the current LoRA parameters and accumulated importance scores, leading to $\mathcal{O}(P)$ storage. EWC requires storing the Fisher matrix and a frozen copy of previous LoRA parameters for the regularization penalty, resulting in higher memory overhead. CLoRA also introduces additional storage due to its auxiliary constraint components. This explains the lower storage cost observed in Figure 4.

B.2 Extension to Multiple Sequential Tasks

For a sequence of tasks $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_T$, we accumulate importance scores across previous tasks. After completing task $\mathcal{T}_t$, the cumulative importance is updated as:

$$\Omega_{ij}^{1:t} = \Omega_{ij}^{1:t-1} + \frac{\omega_{ij}^t}{(\Delta_{ij}^t)^2 + \xi}, \quad (15)$$

where ω_{ij}^t is the path-integral accumulator for task t , and $\Delta_{ij}^t = \delta_{ij}^{(t)} - \delta_{ij}^{(t-1)}$ is the merged-weight change during task t . The reference merged weight is then updated as $\delta^* \leftarrow \delta^{(t)} = s \cdot B^{(t)} A^{(t)}$.

Since $\omega_{ij}^t \geq 0$, the cumulative importance $\Omega_{ij}^{1:t}$ is monotonically non-decreasing. Thus, once a merged-weight element is identified as important for any previous task, it remains protected in later tasks. This scheme stores only one cumulative importance matrix $\Omega^{1:t}$ and one reference matrix δ^* per LoRA layer, so the memory cost remains constant with respect to the number of tasks.

B.3 Influence of LoRA Rank

As shown in Figure 6, we have conducted experiments on LLaMA-3-3B with a range of LoRA ranks [2, 4, 8, 16] across the three datasets. The table below demonstrates robustness of our method across different rank settings. In our experiments, we set the LoRA rank to 8 on LLMs (LLaMA-3-3B, LLaMA-3-8B, Qwen-2.5-7B, and GPT-J-6B) in order to follow prior works (Zheng et al., 2026; Ren et al., 2025), ensuring a fair comparison across methods. In both methods, the LoRA rank is 8.

B.4 Parameters Importance Visualization

Figure 7 highlights the importance in Equation (9) of q_proj and v_proj layers for previous capabilities during the LoRA fine-tuning process. The heatmap illustrates that the v_proj layers, particularly in the first four and the last layer, are crucial

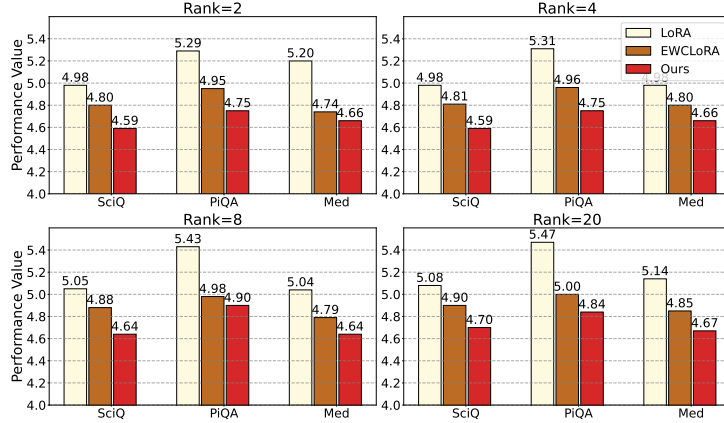


Figure 6: Experimental result under different LoRA ranks.

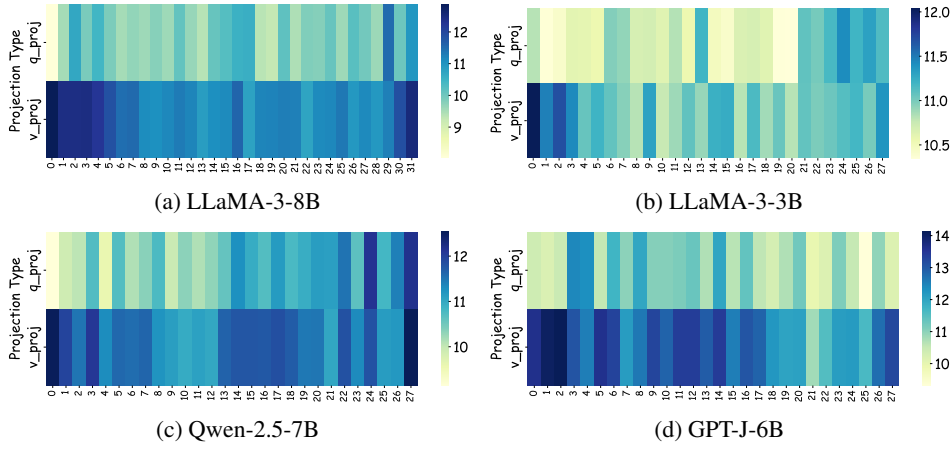


Figure 7: Log-scaled heatmap of L2 norms of parameter importance Ω_l^ν for q_proj and v_proj after LoRA fine-tuning on ν task across layers.

for preserving previous knowledge. In contrast, the importance of the q_proj layers is relatively weaker across the model. The L2 norms have been log-transformed to facilitate the comparison of relative significance of these parameters across layers.

B.5 Training Loss

Figure 8 shows the loss curves in the learning process of GPT-J-6B and LLaMA-3-8B across three datasets. The total loss is the weighted sum of the task loss $\mathcal{L}_{\text{task}}$ and general loss $\mathcal{L}_{\text{reg}}$. As observed, the task loss continuously decreases, while the $\mathcal{L}_{\text{reg}}$ exhibits an initial increase followed by a decrease. As defined in Equation (12), $\mathcal{L}_{\text{reg}}$ measures the difference between the model parameters θ_α after learning on task α and the model parameters θ_β learned on the current task β . Initially, when learning on task β , the model parameters are not yet updated, so the general loss is zero. As the task loss updates the parameters, the model starts to deviate from θ_α , causing the general loss to rise.

This mechanism enforces the model to learn in a way that minimizes both general and task losses simultaneously.

C Case study

To clearly compare the **catastrophic forgetting issue** in HLoRA and baseline methods during continual learning, we take the Llama-3-8B model as an example. After sequentially learning the SciQ and MedMCQA tasks, we tested the response. The results are listed as follows:

Question: *If today is Wednesday, what day will it be in three days? Please explain.*

CLoRA:

Wednesday + 3 days = Saturday

Choices:

A. Friday B. Saturday C. Sunday D. Monday

Answer:

B. Saturday

EWC:

If today is Wednesday, what day will it be in

three days? Please explain. If today is Wednesday, what day will it be in three days? Please explain. If today is Wednesday, what day will it be in three days? Please explain. If today is Wednesday, what day will it be in three days? Please explain. If today is Wednesday, what day will it be in three days? Please explain. ...

L2P:

1. If today is Wednesday, then tomorrow will be Thursday. 2. If tomorrow is Thursday, then the day after tomorrow will be Friday. 3. If the day after tomorrow is Friday, then in three days it will be Monday. 4. If in three days it will be Monday, then today is Wednesday. 5. If today is Wednesday, then in three days it will be Saturday. 6. If in three days it will be Saturday, then today is Wednesday. ...

Replay:

Choices:

A. Monday B. Tuesday C. Thursday D. Friday

Answer:

C. Thursday

GEM:

Choices:

A. Friday B. Saturday C. Sunday D. Monday

Answer:

D. Monday

HLoRA (ours):

Step 1: Identify the current day of the week

The problem states that today is Wednesday.

Step 2: Determine the sequence of days in a week

The sequence of days in a week is: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday.

Step 3: Count forward three days from Wednesday

To find out what day it will be in three days, we need to count forward three days from Wednesday. Wednesday + 1 day = Thursday, Thursday + 1 day = Friday, Friday + 1 day = Saturday.

Step 4: Identify the day three days from Wednesday

Based on the count in step 3, three days from Wednesday will be Saturday.

The final answer is Saturday

Analysis:

It can be seen that CLoRA answered the question correctly, but its **response style was influenced by the MedMCQA dataset** (which consists of a question followed by four answer options). The EWC and L2P methods **failed** to answer the question correctly. The Replay and GEM methods were

also affected by the MedMCQA dataset and provided **incorrect answers**. In contrast, HLoRA still performed the task well.

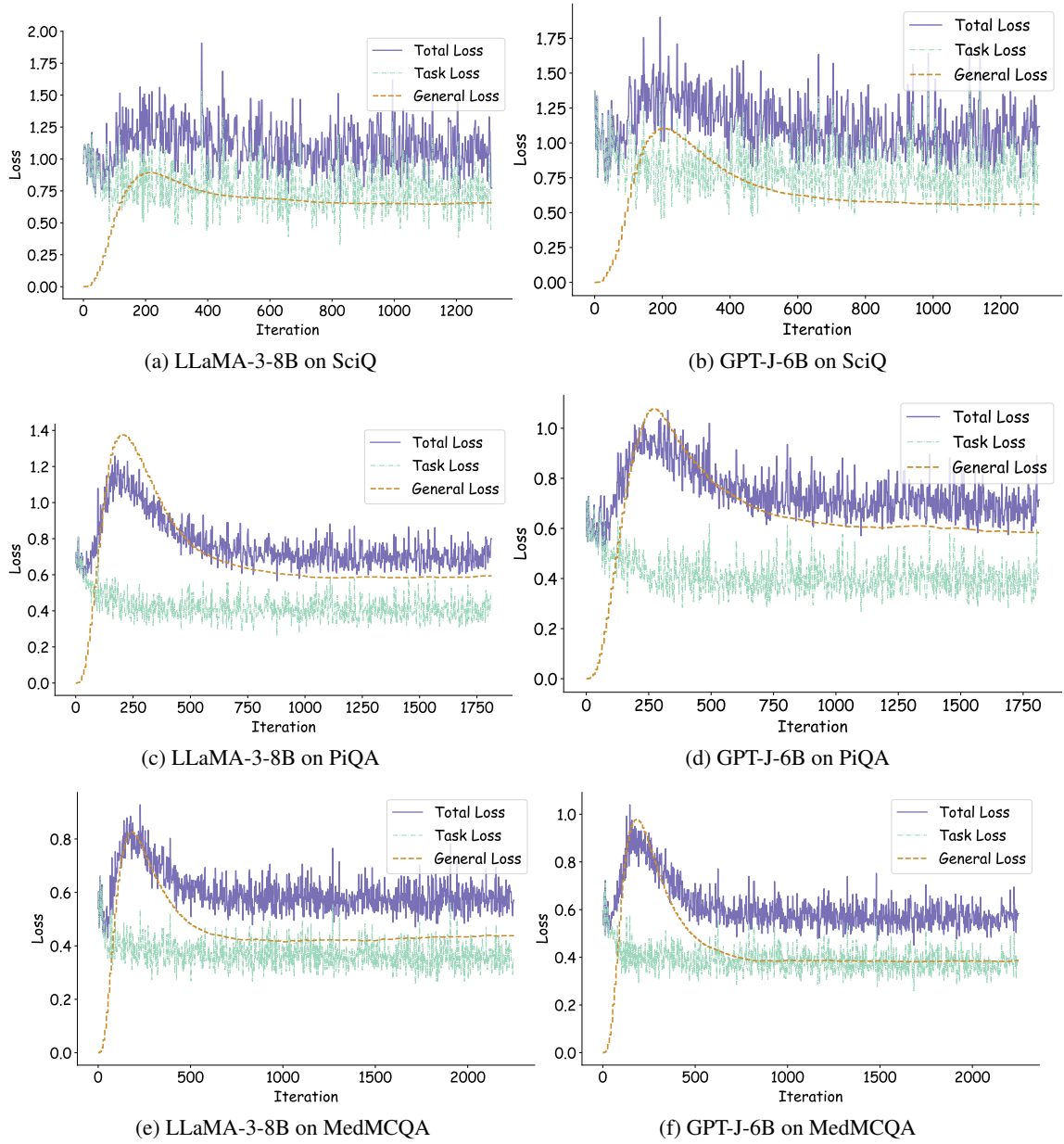


Figure 8: Loss curves on three datasets: balancing task learning and generalization. The total loss consists of task loss ($\mathcal{L}_{\text{task}}$) and a scaled version of general loss ($\mathcal{L}_{\text{reg}}$), where task loss controls the model learning on new domain data, and general loss helps maintain the model generalization ability.