Dynamic Orthogonal Continual Fine-tuning for Mitigating Catastrophic Forgettings

Zhixin Zhang[1] **Zeming Wei**[1*] **Meng Sun**[1*]

[1]Peking University

Abstract

Catastrophic forgetting remains a critical challenge in continual learning for large language models (LLMs), where models struggle to retain performance on historical tasks when fine-tuning on new sequential data without access to past datasets. In this paper, we first reveal that the drift of functional directions during the fine-tuning process is a key reason why existing regularization-based methods fail in long-term LLM continual learning. To address this, we propose Dynamic Orthogonal Continual (DOC) fine-tuning, a novel approach that tracks the drift of these functional directions and dynamically updates them during the fine-tuning process. Furthermore, by adjusting the gradients of new task parameters to be orthogonal to the tracked historical function directions, our method mitigates interference between new and old tasks. Extensive experiments on various LLM continual learning benchmarks demonstrate that this approach outperforms prior methods, effectively reducing catastrophic forgetting and providing a robust tool for continuous LLM fine-tuning. Our code is available at https://github.com/meloxxxxxx/DOC.

1 Introduction

Recently, Large Language Models (LLMs) have achieved significant milestones in various tasks based on their extensive capacity and knowledge. In particular, fine-tuning LLMs with task-specific data has emerged as a popular learning paradigm in their diverse applications. In this context, LLM Continual Learning (Wu et al. 2024b), which fine-tunes LLMs with evolving tasks and data, has become a crucial technique for updating their knowledge to keep pace with new environments and goals. However, a critical challenge of continual learning is catastrophic forgetting (Wu et al. 2022), where the model forgets the knowledge it acquired from previous tasks after receiving new updates.

Existing continual learning approaches for LLMs can be categorized into the following types (Wu et al. 2024b; Zheng et al. 2024): Rehearsal-based (de Masson d'Autume et al. 2019; Mok et al. 2023; Huang et al. 2021), Architecture-based (Jang et al. 2023; Peng et al. 2024; Wu et al. 2024a), Prompt-based (Wang et al. 2022; Qin and Joty 2022; Razdaibiedina et al. 2023), and Regularization-based approaches (Farajtabar et al. 2019; Wang et al. 2023a; Li and Hoiem 2017; Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017). While the first three approaches may suffer from significant computational or memory overhead issues (*e.g.*, training additional modules or storing historical data), regularization-based continual learning for LLMs does not suffer from these issues and has been acknowledged as an efficient approach (Wu et al. 2024b). More formally, we denote that all existing regularization-based methods abide by the following outline:

- **Step (1)** Record the functional directions, mainly including the *gradient direction* of the model parameter, on historical tasks (Olah et al. 2020, 2018; Saxena and Cunningham 2019);
- **Step (2)** Regularize new updates based on these historical functional directions.

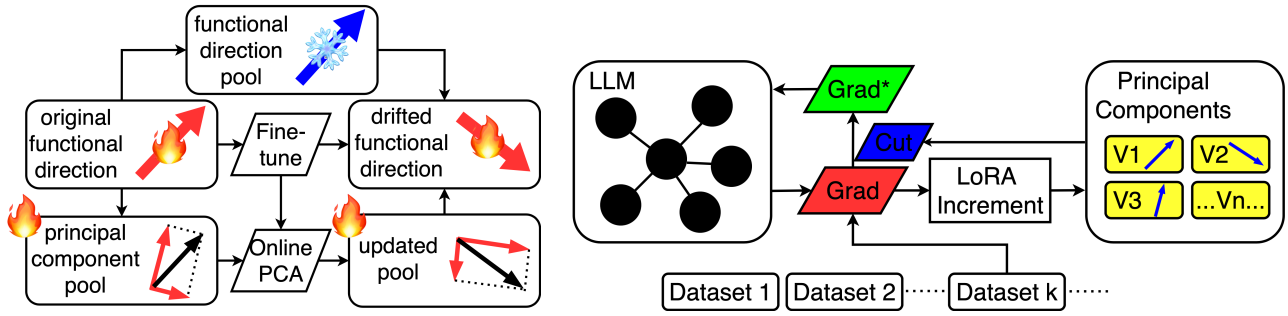
For instance, Elastic Weight Consolidation (EWC) methods (Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017) and orthogonal optimization methods, including Orthogonal Gradient Descent (OGD) (Farajtabar et al. 2019) and Orthogonal Subspace Learning (O-LoRA) (Wang et al. 2023a), employ historical gradient directions and vectors in LoRA matrices of the model for regularization.

However, current regularization-based continual learning still faces the catastrophic forgetting problem, leaving a gap for its practical deployment. In this paper, we aim to mitigate this problem by identifying a key problem in the regularizers. Specifically, we find that the drift of functional directions (Black et al. 2022) during continuous fine-tuning poses a significant issue for their regularizations. While functional directions may be valid within a local neighborhood around a static point in the parameter space, continuous fine-tuning can break this locality when moving the model weights towards other spaces, thus destroying the functionality of these directions, as shown in Figure 1(a). This observation is detailed in Section 3.1. In the settings of regularization-based methods, the difficulty lies in the lack of access to historical data, which makes it challenging to update their functional directions in the current parameter space.

Based on the observation above, we propose our method that tracks the drifting functional directions of historical

*Corresponding to Zeming Wei (weizeming@stu.pku.edu.cn) and Meng Sun (sunm@pku.edu.cn).

AAAI 2026 Workshop on Deployable AI (DAI), Oral presentation.



(a) Drifting functional directions (b) Method overview

Figure 1: An introduction to our work. Prior methods record the functional directions in a *fixed* pool and try to regularize future updates with it, which is shown on the upper half of Figure (a). Our method (the lower half) *updates* these directions with Online PCA for better regularization. Figure (b) presents an overview of our method. In a sequence of incoming datasets, we compute gradients and LoRA increments to update a set of principal components that represent drifting functional directions. We cut them off from current gradients to avoid forgetting historical functions.

tasks with the latest task data. Since LLMs primarily fine-tune within a low-rank subspace (Aghajanyan, Zettlemoyer, and Gupta 2020), all tasks share most of the functional directions in this subspace with different linear combinations. Thus, we employ a modified Online Principal Component Analysis (Cardot and Degras 2015) to extract these directions from their combinations to capture and track the evolving functional directions. Leveraging these up-to-date functional directions, we cut gradients of new task parameters to be orthogonal to the tracked historical function directions, following prior orthogonal methods including OGD (Farajtabar et al. 2019) and O-LoRA (Wang et al. 2023a), which mitigates the interference between new and old tasks. However, a key difference between our method and other orthogonal methods is that we dynamically *update* the functional directions rather than regularizing on *fixed* ones. Tracking these functional directions, which prior works often overlook, is crucial for preserving functions that lie in drifting directions. A brief overview of our method is in Figure 1(b).

Extensive experiments verify the drift of functional directions and demonstrate the effectiveness of our method in tracking them, offering a substantiated motivation for our method. Furthermore, experiments on various LLM continual learning benchmarks demonstrate that our approach significantly mitigates the catastrophic forgetting issues in online streaming data scenarios, and outperforms prior methods, *e.g.*, we respectively achieve an accuracy of 77.7 and 73.4 in standard CL benchmark (Zhang, Zhao, and LeCun 2016) and long chains of tasks for LLaMA-7B (Touvron et al. 2023), compared to 76.5 and 71.9 of O-LoRA (Wang et al. 2023a), the previous state-of-the-art regularization-based method. In summary, our contributions are as follows:

- We reveal the drift of function directions in the fine-tuning process, which explains why regularization-based approaches fail in long-term LLM continual learning.
- Based on this discovery, we propose the Dynamic Orthogonal Continual Fine-tuning (DOC) method that tracks the drift of functional directions to mitigate catastrophic forgetting issues.
- We conduct extensive experiments to validate that DOC outperforms prior methods in various LLM continual learning benchmarks, contributing an effective tool in continuous LLM fine-tuning.

2 Preliminaries

2.1 Continual Learning Setup

Continual learning for LLMs (Wu et al. 2024b; Zheng et al. 2024) is crucial for updating their knowledge and keeping pace with new goals. In a continual learning scenario, a pre-trained LLM is fine-tuned on an online stream of tasks with their task-specific data. Due to factors like storage costs and privacy protection, historical data cannot be accessed when fine-tuning on the latest one.

Definition of continual learning. Given a LLM F_θ with parameters θ, a sequence of labeled datasets $\{D_1, D_2, ..., D_N\}$, where $D_t = \{(x_t^i, y_t^i)\}_{i=1}^{n_t}$ $(t = 1, \ldots, N)$. Then F_θ is sequentially fine-tuned on $D_1, D_2, ..., D_N$. When fine-tuning on D_T, historical datasets, *i.e.* $\{D_1, D_2, ..., D_{t-1}\}$, cannot be accessed. The target is an F_θ that behaves well on all datasets:

$$\arg\min_{\theta} \sum_{t=1}^{N} \sum_{i=1}^{n_i} L_t(F_\theta(x_t^i), y_t^i), \tag{1}$$

where L_t is the task-specific loss function of the t-th task. **Note that for concision, we substitute L for L_t when fine-tuning on D_t in the following statements**.

2.2 Low-Rank Adaptation (LoRA)

When fine-tuning LLMs for specific tasks, there exists a low intrinsic dimension for the parameter update of the model (Aghajanyan, Zettlemoyer, and Gupta 2020). For a weight matrix $W_{m\times n}$ of a pre-trained LLM, LoRA (Hu et al. 2021) employs low-rank matrixes $B_{m\times r}$ and $A_{r\times n}$ $(r \ll min(m, n))$ to constrain its update by representing it with a low-rank decomposition:

$$W^* = W + BA, \tag{2}$$

where W^* is the new parameter after fine-tuning. As a result, the propagation process is modified:

$$W^*x = (W + BA)x = Wx + BAx, \tag{3}$$

where x is the input to the module with parameter W.

3 Motivation and the Proposed Method

In this section, we first reveal that the drift of functional directions is the key issue for existing regularization-based methods in 3.1, then propose a method to track drifting functional directions and validate the effectiveness of our tracking method in 3.2, and finally cut the parameter increment of new tasks to be orthogonal to historical ones in 3.3.

3.1 Motivation: Analysis of Existing Regularization Methods

Our method is developed using a regularization-based approach in consideration of its little computational or memory overhead issues (Wu et al. 2024b). While prior research (Zheng et al. 2024) has demonstrated that existing regularization methods are efficient on short task sequences, their performance is relatively limited in long sequences, leaving a gap for their practical deployment. In the following parts, we propose an analysis to identify the primary cause of this defect.

Intrinsic functional directions of LLMs. Functional directions (Olah et al. 2020, 2018; Saxena and Cunningham 2019) have become prevalent in research on LLMs. In this paper, we define functional directions of LLMs as *the gradient direction of model parameters on certain datapoints*. Most of the prior regularization-based approaches employ functional directions to approximate the functional unit of certain tasks in LLMs. They adhere to the outline for recording functional directions and regularizing new updates on historical directions. Specifically, Orthogonal methods, including Orthogonal Gradient Descent (OGD) (Farajtabar et al. 2019) and Orthogonal Subspace Learning (O-LoRA) (Wang et al. 2023a), avoid perturbing the historical settings of the model through orthogonal approaches, and the two respectively employ gradient directions and LoRA vectors as the regularized functional directions. Elastic Weight Consolidation(EWC) methods (Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017) employ the Fisher information matrix for its consolidation, which is also computed with gradients.

Functional directions drift in the fine-tuning process. In this part, we identify that the drift of functional directions during the continuous fine-tuning process is the key issue of the regularizations above. Specifically, in the process of continually fine-tuning an LLM, the locality of linearity in its deep neural networks is broken (Black et al. 2022), thus destroying the functionality of the directions extracted in earlier steps. Consequently, regularization in these directions deviates from the original purpose in the continual fine-tuning, as demonstrated in Figure 1(a). In this part, we present the following observations regarding the drifts proposed above. We take fine-tuning Llama-2 (Touvron et al. 2023) on CL Benchmark (Zhang, Zhao, and LeCun 2016) as the example in this experiment, and measure the drift of the gradient direction during continual fine-tuning.

As shown in Figure 2(a), with the fine-tuning process conducted, the functional directions captured earlier no longer represent the current ones, exposing the ineffectiveness of employing *fixed* singular or average gradient as the functional direction, which is conducted in EWC (Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017) and OGD (Farajtabar et al. 2019). Similarly, we also investigate the drift of column vectors in LoRA B matrices employed by O-LoRA (Wang et al. 2023a), *i.e.* $\beta_1, \beta_2, ..., \beta_r$ in $B = (\beta_1, \beta_2, ..., \beta_r)$. The results (blue line, denoted as LoRA B) show that employing column vectors in LoRA B matrices mitigates the loss of functional directions. However, it does not resolve the drift issue fundamentally. Overall, we identify that the prevalent problem in prior regularization-based methods is the drift of functional directions, indicating that we need to dynamically update the functional directions rather than relying on fixed ones.

3.2 Tracking the drift of functional directions

The difficulty of mitigating the drift of functional directions lies in updating the functional directions of historical data in the current parameter space, as there is no access to historical data in the settings of regularization-based methods. To tackle this issue, we propose our method to track the drifting functional directions of historical tasks with the latest task data.

As shown in Equation (3), LLMs primarily fine-tune within a low-rank subspace, *i.e.* the space of BAx, such that all tasks share most of the bases in this subspace, and the functional directions are different linear combinations of these bases. By extracting and updating the bases from the functional directions of the current task, we update the shared bases of historical functional directions, thus updating historical functional directions themselves.

Tracking method overview. To achieve the conception above, we select the LoRA increment as the functional direction, and employ Principal Component Analysis (PCA) to extract the bases. The following parts propose respective elaborations.

LoRA increment as functional directions. Following prior regularization-based methods, including OGD(Farajtabar et al. 2019) and O-LoRA(Wang et al. 2023a), we extract fine-tuning increments as the functional directions of certain continual learning tasks. More specifically, the functional direction we select is the increment of LoRA in Equation (3), that is:

$$\mathbf{d}(W_m x_m) = \mathbf{d}(B_m A_m x_m) = \mathbf{d}(B_m A_m) x_m \quad (4)$$

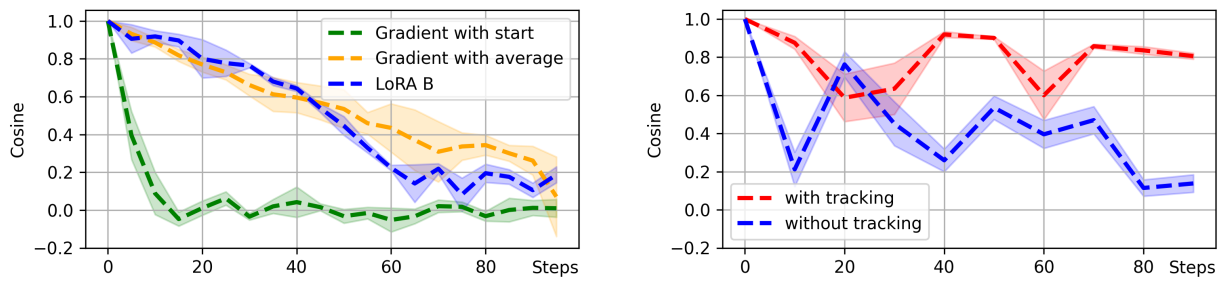
$$= (\mathbf{d}B_m) A_m x_m + B_m (\mathbf{d}A_m) x_m \triangleq p_m, \quad (5)$$

where x_m is the input *vector* to the m-th LoRA module with parameter B_m and A_m. Let α be the learning rate, then $\mathbf{d}B = \alpha \nabla_B L$, $\mathbf{d}A = \alpha \nabla_A L$, L is the task-specific loss function. We represent the update direction of LoRA with the following concatenated vector:

$$h = \mathbf{concat}(p_1, p_2, ..., p_M), \quad (6)$$

where M is the number of LoRA modules. The concatenation captures the relation between the LoRA increment of different layers. More computational details on x and h are shown in Appendix A.3.

Online PCA. To extract the basis of functional directions from their linear combinations, we employ the Online Prin-



(a) (b)

Figure 2: Quantification of functional direction drift regarding a particular datapoint (x, y). Figure (a) shows the cosine similarity between current and historical functional directions. The green line shows $\mathbf{cos}\langle G_T, G_1\rangle$, where $G_T = \nabla_\theta L(F_{\theta_T}(x), y)$, θ_T is the model parameter in the T th fine-tuning step. The yellow line shows $\mathbf{cos}\langle G_T, \bar{G}_T\rangle$, where $\bar{G}_T = \frac{1}{T}\sum_{t=1}^{T} G_t$. The blue line shows the average similarity of $\beta_1, \beta_2, ..., \beta_r$ in LoRA B matrices with their start value, *i.e.* $\frac{1}{r}\sum_{n=1}^{r} \mathbf{cos}\langle\tilde{\beta}_n, \beta_n\rangle$, where $\tilde{\beta}$ is the current one, β is the start one. Figure (b) shows the effect of tracking functional directions. We initialize the principal components during the first dataset, and measure the drift in the following steps. The red line shows the drift with $\mathbf{cos}\langle\mathbf{coord}(h_T^*), \mathbf{coord}(h_1^*)\rangle$, where h_T^* is the LoRA increment (computed with Equation (6)) in the T-th step. For contradiction, we freeze the update of principal components (the blue line). The results are the average of randomly-chosen datapoints, with standard deviation shown.

cipal Component Analysis (Online PCA) (Cardot and Degras 2015), which requires only the latest data in memory, conforming to the settings of regularization-based continual learning.

The target of Online PCA is as follows. Let $\{h_1, h_2, ..., h_n\}$ be functional directions computed with Equation (6) on a sequence of incoming data. On receiving a new functional direction h_t, Online PCA seeks to update principal components $\{v_t^1, v_t^2, ..., v_t^{K_t}\}$ as the basis of functional directions $\{h_1, h_2, ..., h_t\}$. Moreover, when processing the latest data h_t, there is no access to historical datas $\{h_1, h_2, ..., h_{t-1}\}$. This realizes our goal of updating historical functional directions with current ones, *i.e.* updating the representation of historical functional directions with the current functional direction. Please refer to Algorithm 1 for a summary and Figure 1(a) for a brief demonstration.

There are multiple approaches to implement Online PCA, including Incremental PCA (Arora et al. 2012; Levey and Lindenbaum 2000) and stochastic approximation methods (Sanger 1989; Krasulina 1970; Oja and Karhunen 1985; Oja 1992). Our method draws inspiration from Candid Covariance-free Incremental PCA (CCIPCA) (Weng, Zhang, and Hwang 2003), since its edge lies in the ability to add components freely, which is suited for emerging new tasks. It also has a lower computational overhead compared to other techniques. Please refer to Appendix for more technical details on our Online PCA method.

The effectiveness of tracking. To evaluate the effectiveness of tracking, we investigate the drift of the functional direction in the subspace of the updated principal components. Specifically, we compute the LoRA increment h_T^* of a particular datapoint in the T-th fine-tuning step, and compute its coordinate in the subspace of extracted principal components, that is

$$\mathbf{coord}(h_T^*) = \left((h_T^*, v_T^1), (h_T^*, v_T^2), ..., (h_T^*, v_T^K)\right). \quad (7)$$

where $(h_T^*, v_T^k) = \frac{h_T^* \cdot v_T^k}{\|v_T^k\|}$ is the projection of h_T^* on v_T^k.

Algorithm 1 DOC (Our method)

Input: Model F_θ, where $\theta = (A, B)$ includes LoRA A, B modules; learning rate α; the t-th incoming dataset D_t, expected maximum principal component number K for each new task

Initialization: Principal components $v_T^1, v_T^2, ..., v_T^{K_T}$ extracted from historical fine-tunings, T is the number of finished fine-tuning steps.

Output: Fine-tuned parameter θ^*

1: **for** data point(batch) (x_t^i, y_t^i) in D_t **do**
2: extract gradients: $\nabla_B L = \nabla_B L(F_\theta(x_t^i), y_t^i)$ $\nabla_A L = \nabla_A L(F_\theta(x_t^i), y_t^i)$
3: compute current LoRA increment h_{T+i} with Equation (6)
4: use h_{T+i} to update principal components with Online PCA Algorithm on the basis of existing $v_{T+i-1}^1, v_{T+i-1}^2, ..., v_{T+i-1}^{K_{T+i-1}}$, get $v_{T+i}^1, v_{T+i}^2, ..., v_{T+i}^{K_{T+i}}$ $(K_{T+i-1} \leq K_{T+i} \leq K \cdot t)$
5: cut $\nabla_B L$ with Equation (14), get $(\nabla_B L)_{\text{cut}}$
6: update parameter: $B = B - \alpha \cdot (\nabla_B L)_{\text{cut}}$ $A = A - \alpha \cdot \nabla_A L$
7: **end for**
8: **return** $\theta^* = (A, B)$

As shown in Figure 2(b), by tracking principal components, drifting functional directions can be followed and thus remain in correspondence with their original states; if we forbid tracking, functional directions are gradually lost.

3.3 Cut fine-tuning directions for function preservation

Following prior orthogonal space fine-tuning approaches, including OGD (Farajtabar et al. 2019) and O-LoRA (Wang et al. 2023a), for regularization-based continual learning, we

		Standard CL Benchmark				Long chain of tasks			
		Order 1	Order 2	Order 3	**Average** (↑)	Order 4	Order 5	Order 6	**Average** (↑)
Baselines	LoRA	67.7	65.4	66.2	66.4	61.2	63.6	60.7	61.8
	EWC	72.3	65.0	70.4	69.2	59.7	61.2	65.4	62.1
	LwF	71.6	66.0	69.7	69.1	60.8	62.6	63.3	62.2
	O-LoRA	78.2	76.4	**74.7**	76.5	**71.7**	73.8	70.2	71.9
	DOC (ours)	**80.5**	**78.6**	73.9	**77.7**	71.6	**74.1**	**74.4**	**73.4**
	DOC-ablation	70.7	69.5	67.3	69.1	60.0	62.5	64.9	62.4
Oracle methods	Replay	67.9	68.2	71.0	69.0	62.3	65.0	61.4	62.9
	PerTaskLoRA	76.9	76.9	76.9	76.9	76.8	76.8	76.8	76.8
	MTL	83.4	83.4	83.4	83.4	80.3	80.3	80.3	80.3
	ProgPrompt	77.4	76.9	77.9	77.4	76.8	76.2	77.1	76.7

Table 1: Average Accuracy (AA) of different continual methods on LLaMA-7B.

try to make the parameter increment of new tasks orthogonal to historical ones. This avoids changing the functional directions representing historical tasks, thus protecting historical functions. Specifically, we make the current LoRA increment h_T orthogonal to historical ones, whose basis are principal components $\{v_T^1, v_T^2, ..., v_T^K\}$. The goal is as follows:

$$h_T \perp v_T^k \quad k = 1, 2, ..., K. \tag{8}$$

Note that in Equation (6) we have

$$h_T = \textbf{concat}\left(\mathbf{d}(B_m A_m x_m) \quad m = 1, 2, ..., M\right), \tag{9}$$

so we disassemble the concatenation to realize the orthogonality in Equation (8). The disassembly is as follows:

$$v_T^k = \textbf{concat}(v_T^k(m) \quad m = 1, 2, ..., M). \tag{10}$$

Then we only need to make

$$\mathbf{d}(B_m A_m x_m) \perp v_T^k(m) \quad m = 1, 2, ...M \quad k = 1, 2, ..., K \tag{11}$$

Note that we substitute BAx for $B_m A_m x_m$ and $\tilde{v}^k$ for $v_T^k(m)$ in the following statements for concision. As $\mathbf{d}(BAx) = (\mathbf{d}B)Ax + B(\mathbf{d}A)x$, we realize the orthogonality in Equation (11) respectively for $(\mathbf{d}B)Ax$ and $B(\mathbf{d}A)x$:

$$(\mathbf{d}B)Ax \perp \tilde{v}^k, \quad B(\mathbf{d}A)x \perp \tilde{v}^k \quad \text{for} \quad k = 1, 2, ..., K.$$

For $(\mathbf{d}B)Ax$, note that

$$(\mathbf{d}B)Ax = (\mathbf{d}\beta_1, \mathbf{d}\beta_2, ..., \mathbf{d}\beta_r)(Ax) \in \langle \mathbf{d}\beta_1, \mathbf{d}\beta_2, ..., \mathbf{d}\beta_r \rangle \tag{12}$$

So we only need to cut $\mathbf{d}\beta_i = \alpha \cdot \nabla_{\beta_i} L \quad (i = 1, 2, .., r)$ to be orthogonal to $\tilde{v}^k \quad (k = 1, 2, ..., K)$. That is:

$$\nabla_{\beta_i} L \perp \tilde{v}^k \quad i = 1, 2, ..., r \quad k = 1, 2, ..., K \tag{13}$$

Then we reach the following gradient cut:

$$(\nabla_{\beta_i} L)^* = \nabla_{\beta_i} L - \sum_{k=1}^{K} \frac{\nabla_{\beta_i} L \cdot v_T^k}{\|v_T^k\|^2} \cdot v_T^k \quad i = 1, 2, ..., r. \tag{14}$$

Now we get $(\nabla_B L)_{\text{cut}} = ((\nabla_{\beta_1} L)^*, (\nabla_{\beta_2} L)^*, ..., (\nabla_{\beta_r} L)^*)$. Note that the cut above removes the correlation with input x since Equation (12), making the orthogonality hold true for all kinds of input x. This is significant in preserving historical functions on all tasks.

For $B(\mathbf{d}A)x$, assume that we have employed $(\nabla_B L)_{\text{cut}}$ in previous steps, then their aggregated $B = (\beta_1, \beta_2, ..., \beta_r)$ satisfies the orthogonality for the former steps. Similar to Equation (12), we have

$$B(\mathbf{d}A)x \in \langle \beta_1, \beta_2, ..., \beta_r \rangle, \tag{15}$$

so the orthogonality holds for $B(\mathbf{d}A)x$. We keep $B(\mathbf{d}A)x$ intact as a momentum for optimization, which means we keep the original $\mathbf{d}A$ and $\nabla_A L$.

Please note that the above orthogonal cut does not harm the gradient descent, as described in the paper of OGD (Farajtabar et al. 2019). In summary, our complete method is formulated as Algorithm 1. Please refer to Figure 1(b) for a brief demonstration.

4 Experiments

In this section, we test our method across various LLM continual learning benchmarks through extensive experiments to explore the practical impact on real-world continual deployment with online streaming data.

4.1 Setup

Datasets and Models. Following ProgPrompt (Razdaibiedina et al. 2023) and O-LoRA (Wang et al. 2023a), we employ CLBenchmark (AG News, Amazon reviews, Yelp reviews, DBpedia, Yahoo answers) (Zhang, Zhao, and LeCun 2016) to evaluate our methods, adding GLUE (MNLI, QQP, RTE, SST2) (Wang et al. 2019), SuperGLUE (WiC, CB, COPA, MultiRC, BoolQ) (Wang et al. 2020), and IMDB review (Maas et al. 2011) for long-chain tasks. The models we use are LLaMA-7B, LLaMA-13B (Touvron et al. 2023), and T5-Large (Raffel et al. 2023).

Metrics. Following ProgPrompt and O-LoRA, we employ **Average Accuracy (AA)** to evaluate the overall performance of continual learning, that is $\mathbf{AA}(T) = \frac{1}{T}\sum_{t=1}^{T} a_{t,T}$ where $a_{t,T}$ is the test accuracy on the t-th task after fine-tuning on the T-th task.

In order to measure the catastrophic forgetting, we employ Backward Transfer Rate (BWT) and Forward Transfer

		Standard CL Benchmark				Long chain of tasks			
		Order 1	Order 2	Order 3	**Average** (↑)	Order 4	Order 5	Order 6	**Average** (↑)
Baselines	LoRA	69.2	68.0	65.7	67.6	59.9	64.7	62.0	62.2
	EWC	72.7	66.9	66.0	68.5	63.4	60.2	66.7	63.4
	LwF	71.0	70.4	72.8	71.4	64.5	62.6	65.3	64.1
	O-LoRA	77.9	79.8	77.6	78.4	70.8	73.2	72.2	72.0
	DOC (ours)	**79.5**	**81.2**	**79.7**	**80.1**	**72.4**	**74.0**	**76.5**	**74.3**
	DOC-ablation	69.0	74.6	70.9	71.5	62.6	62.3	66.0	63.6
Oracle methods	Replay	70.1	69.4	68.2	69.2	64.3	65.4	63.6	64.4
	PerTaskLoRA	77.4	77.4	77.4	77.4	78.5	78.5	78.5	78.5
	MTL	85.7	85.7	85.7	85.7	83.6	83.6	83.6	83.6
	ProgPrompt	76.2	80.9	78.5	78.5	79.9	80.0	78.0	79.3

Table 2: Average Accuracy (AA) of different continual methods on LLaMA-13B.

	Standard CL Benchmark		Long chain of tasks	
	BWT(↑)	FWT(↑)	BWT	FWT
LoRA	$-14.6_{+0.0}$	$0.6_{+0.0}$	$-16.2_{+0.0}$	$0.2_{+0.0}$
EWC	$-10.6_{+4.0}$	$0.2_{-0.4}$	$-14.3_{+1.9}$	$-1.5_{-1.7}$
LwF	$-10.9_{+3.7}$	$0.5_{-0.1}$	$-15.0_{+1.2}$	$-0.6_{-0.8}$
O-LoRA	$-1.9_{+12.7}$	$1.4_{+0.8}$	$-5.2_{+11.0}$	$\mathbf{0}_{\mathbf{-0.2}}$
DOC (ours)	$\mathbf{-0.6_{+14.0}}$	$\mathbf{1.6_{+1.0}}$	$\mathbf{-3.4_{+12.8}}$	$-0.1_{-0.3}$
DOC-Ablation	$-8.8_{+5.8}$	$-1.5_{-2.1}$	$-13.7_{+2.5}$	$-1.7_{-1.9}$
Replay	$-10.5_{+4.1}$	$0_{-0.6}$	$-14.7_{+1.5}$	$0.2_{+0.0}$
ProgPrompt	$-0.2_{+14.4}$	$0.8_{+0.2}$	$-0.2_{+16.0}$	$0.1_{-0.1}$

Table 3: Average BWT and FWT scores of different continual methods on LLaMA-7B

Rate (FWT) (Wu et al. 2022):

$$\mathbf{BWT}(T) = \frac{1}{T-1}\sum_{t=1}^{T-1}(a_{t,T} - a_{t,t}), \tag{16}$$

$$\mathbf{FWT}(T) = \frac{1}{T-1}\sum_{t=2}^{T}(a_{t,t} - \tilde{a}_t). \tag{17}$$

Commonly, in a continual learning scenario, a negative BWT score indicates forgetting, and a negative FWT reveals that we regularize the fine-tuning process and decrease the fine-tuning performance $a_{t,t}$ compared to a standard fine-tuning performance $\tilde{a}_t$.

Overall, a regularization-based method pursues a higher BWT score representing less forgetting, at the cost of a smaller decrease in FWT score, representing less damage to the fine-tuning of each task.

Implementation details. For LLaMA-7B and LLaMA-13B, we set learning rate α = 1e-4, with a batch size of 8. For T5-Large, we let α = 1e-3 with a batch size of 64, following O-LoRA. Please refer to Appendix A.2 for more details, including step number, task sequence, instructions, *etc.*

Compared methods. To ensure a fair comparison, we primarily focus on the recent **state-of-the-art regulation-based methods**, including EWC, LwF, and O-LoRA. We also consider fine-tuning the model with task-specific datasets sequentially using LoRA (Hu et al. 2021), which is a vanilla baseline and the expected lower bound of continual learning. Note that other-based methods require additional settings and are **not comparable** to ours, which is detailed in Appendix A.4. Furthermore, we present the results from several other oracle methods that are not suitable for continuous fine-tuning settings, but they can serve as upper bounds:

- **Replay** replay samples from historical tasks when fine-tuning on new tasks.
- **PerTaskLoRA** train LoRA modules solely for each task.
- **MTL** train the model on all tasks as multi-task learning.
- **ProgPrompt** (Razdaibiedina et al. 2023) a state-of-the-art method that updates an extending prompt in the streaming data, but task ID is required during inference.

4.2 Main Results

Following ProgPrompt and O-LoRA, there are three independent runs with different task orders for different chains of tasks, as detailed in Appendix A.2.

Overall performance. The results of Average Accuracy (AA) are shown in Table 3.2, Table 3.3, and Table 4.1. We refer to the paper of O-LoRA (Wang et al. 2023a), the up-to-date regularization-based method, for the results of other approaches on T5-Large, as the settings and hyperparameters of our experiments are equal. The results show that our method outperforms prior ones, especially in long-chain tasks. We respectively achieve an accuracy of 77.4 and 73.0 in the standard CL benchmark and long chains of tasks for LLaMA-7B, compared to 75.8 and 69.6 for O-LoRA, the previous state-of-the-art regularization-based method.

Mitigating Catastrophic Forgetting. The BWT and FWT results are shown in Table 3.3. The BWT score of our method is higher than that of prior approaches. We reach -0.6 and -3.4 for standard and long continual learning tasks, compared to -1.9 and -5.2 of O-LoRA, indicating that our method suffers less from catastrophic forgetting. Our FWT score, compared to other methods, indicates that we mitigate catastrophic forgetting at a slight cost to fine-tuning performance.

		Standard CL Benchmark				Long chain of tasks			
		Order 1	Order 2	Order 3	Average (↑)	Order 4	Order 5	Order 6	Average (↑)
Baselines	LoRA	44.6	32.7	53.7	43.7	2.3	0.6	1.9	1.6
	EWC	48.7	47.7	54.5	50.3	45.3	44.5	45.6	45.1
	LwF	54.4	53.1	49.6	52.3	50.1	43.1	47.4	46.9
	O-LoRA	75.4	75.7	**76.3**	75.8	72.3	64.8	71.6	69.6
	DOC (ours)	**78.8**	**78.8**	74.5	**77.4**	**72.7**	**72.4**	**74.0**	**73.0**
	DOC-ablation	62.1	62.9	60.4	61.8	55.6	52.5	57.7	55.3
Oracle methods	Replay	55.2	56.9	61.3	57.8	55.0	54.6	53.1	54.2
	PerTaskLoRA	70.0	70.0	70.0	70.0	78.1	78.1	78.1	78.1
	MTL	80.0	80.0	80.0	80.0	76.5	76.5	76.5	76.5
	ProgPrompt	75.2	75	75.1	75.1	78.0	77.7	77.9	77.9

Table 4: Average Accuracy (AA) of different continual methods on T5-large

	LLaMA-7B	LLaMA-13B	T5-Large
LoRA	0.38s	0.97s	0.68s
EWC	0.42s	1.20s	0.76s
LwF	0.40s	1.23s	0.76s
O-LoRA	0.42s	1.29s	0.78s
DOC(ours)	0.46s	1.23s	0.80s
ProgPrompt	0.32s	0.43s	0.31s

(a)

	K	32	48	64	96
DOC (ours)	$r = 16$	76.1	77.6	76.5	76.5
	$r = 64$	77.9	78.5	78.4	**78.7**
LoRA	$r = 16$	65.4			
	$r = 64$	67.4			
O-LoRA	$r = 16$	77.0			
	$r = 64$	76.8			

(b)

Table 5: (a) Average clock time of one fine-tuning step; (b) Average Accuracy (AA) results of standard CL Benchmark on LLaMA-7B with different LoRA rank r and maximum principal component number K for each new task. The results are the average of task orders 1-3.

In summary, our method mitigates forgetting with a much higher BWT score, at the cost of a little fine-tuning performance with a slightly lower FWT score, eventually reaching effective overall performances and higher AA scores.

4.3 Discussions

Computational costs. The cost of storing all principal components (with maximum principal component number $K \leq 100$) is within 100MB, roughly equivalent to a few sets of LoRA modules, and is negligible compared to the cost of fine-tuning the model itself. We employ vGPU-48GB as our device, with PyTorch 2.1.0 and CUDA 12.1, and the clock time of one training step with different regularization methods is shown in Table 4.1. As the Online PCA technique we employ has an explicit update expression (shown in Appendix A.1), it does not incur much extra computational costs.

The choice of hyperparameters. We present the following empirical study regarding different choices of LoRA rank, say r, for fine-tuning, and the maximum principal component number for each new task, say K, for functional direction tracking. The results are shown in Table 4.1. Overall, adequate principal components cooperating with higher LoRA ranks are able to cover and protect more critical functional directions for historical tasks, thus ensuring a better historical functional preservation and task accuracy. The results show that the variation between different choices of hyperparameters is little, revealing the robustness of our method.

Ablation study. We conduct a trial on freezing the update of principal components to investigate the impact of functional direction tracking. Specifically, we cease updating principal components after their initialization during the first 10% fine-tuning steps for each task. The results are shown in the **DOC-ablation** line in Table 3.2,3.3, 3.3, and 4.1. The decrease in continual learning performance in the ablation experiment indicates that it is tracking the functional directions that mitigate catastrophic forgetting and enhance the performance of continual learning.

5 Conclusion

In this paper, we introduce a novel regularization-based approach that leverages functional direction tracking for continual learning in language models. We identify that the drift of functional directions is the key issue for regularization-based continual learning approaches, and the proposed method systematically addresses the drift issue by updating the functional directions dynamically with Online PCA during the fine-tuning process. Empirical evaluations verify the effectiveness of our tracking method and underscore its efficacy in enhancing continual learning performance. For limitations and future directions, please refer to Appendix A.5 and A.6.

References

Aghajanyan, A.; Zettlemoyer, L.; and Gupta, S. 2020. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. arXiv:2012.13255.

Arora, R.; Cotter, A.; Livescu, K.; and Srebro, N. 2012. Stochastic optimization for PCA and PLS. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 861–868.

Black, S.; Sharkey, L.; Grinsztajn, L.; Winsor, E.; Braun, D.; Merizian, J.; Parker, K.; Guevara, C. R.; Millidge, B.; Alfour, G.; and Leahy, C. 2022. Interpreting Neural Networks through the Polytope Lens. arXiv:2211.12312.

Cardot, H.; and Degras, D. 2015. Online Principal Component Analysis in High Dimension: Which Algorithm to Choose? arXiv:1511.03688.

de Masson d'Autume, C.; Ruder, S.; Kong, L.; and Yogatama, D. 2019. Episodic Memory in Lifelong Language Learning. arXiv:1906.01076.

Farajtabar, M.; Azizan, N.; Mott, A.; and Li, A. 2019. Orthogonal Gradient Descent for Continual Learning. arXiv:1910.07104.

Hendrycks, D.; Burns, C.; Basart, S.; Critch, A.; Li, J.; Song, D.; and Steinhardt, J. 2021a. Aligning AI With Shared Human Values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021b. Measuring Massive Multitask Language Understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685.

Huang, Y.; Zhang, Y.; Chen, J.; Wang, X.; and Yang, D. 2021. Continual Learning for Text Classification with Information Disentanglement Based Regularization. arXiv:2104.05489.

Jang, J.; Kim, S.; Ye, S.; Kim, D.; Logeswaran, L.; Lee, M.; Lee, K.; and Seo, M. 2023. Exploring the Benefits of Training Expert Language Models over Instruction Tuning. arXiv:2302.03202.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Kumaran, D.; and Hadsell, R. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526.

Krasulina, T. P. 1970. Method of stochastic approximation in the determination of the largest eigenvalue of the mathematical expectation of random matrices. *Automation and Remote Control*, 215–221. Originally published in *Avtomatika i Telemekhanika*, 1970, no. 2, pp. 50–56.

Levey, A.; and Lindenbaum, M. 2000. Sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, 9(8): 1371–1374.

Li, Z.; and Hoiem, D. 2017. Learning without Forgetting. arXiv:1606.09282.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning Word Vectors for Sentiment Analysis. In Lin, D.; Matsumoto, Y.; and Mihalcea, R., eds., *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. Portland, Oregon, USA: Association for Computational Linguistics.

Michaud, E. J.; Liu, Z.; Girit, U.; and Tegmark, M. 2024. The Quantization Model of Neural Scaling. arXiv:2303.13506.

Mok, J.; Do, J.; Lee, S.; Taghavi, T.; Yu, S.; and Yoon, S. 2023. Large-scale Lifelong Learning of In-context Instructions and How to Tackle It. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 12573–12589. Toronto, Canada: Association for Computational Linguistics.

Oja, E. 1992. Principal components, minor components, and linear neural networks. *Neural Networks*, 5(6): 927–935.

Oja, E.; and Karhunen, J. 1985. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106(1): 69–84.

Olah, C.; Cammarata, N.; Schubert, L.; Goh, G.; Petrov, M.; and Carter, S. 2020. Zoom In: An Introduction to Circuits. *Distill*. Https://distill.pub/2020/circuits/zoom-in.

Olah, C.; Satyanarayan, A.; Johnson, I.; Carter, S.; Schubert, L.; Ye, K.; and Mordvintsev, A. 2018. The Building Blocks of Interpretability. *Distill*. Https://distill.pub/2018/building-blocks.

Peng, B.; Tian, Z.; Liu, S.; Yang, M.; and Jia, J. 2024. Scalable Language Model with Generalized Continual Learning. arXiv:2404.07470.

Qin, C.; and Joty, S. 2022. LFPT5: A Unified Framework for Lifelong Few-shot Language Learning Based on Prompt Tuning of T5. arXiv:2110.07298.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2023. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683.

Razdaibiedina, A.; Mao, Y.; Hou, R.; Khabsa, M.; Lewis, M.; and Almahairi, A. 2023. Progressive Prompts: Continual Learning for Language Models. arXiv:2301.12314.

Sanger, T. D. 1989. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6): 459–473.

Saxena, S.; and Cunningham, J. P. 2019. Towards the neural population doctrine. *Current Opinion in Neurobiology*, 55: 103–111. Machine Learning, Big Data, and Neuroscience.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wang, A.; Pruksachatkun, Y.; Nangia, N.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2020. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. arXiv:1905.00537.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. arXiv:1804.07461.

Wang, M.; Adel, H.; Lange, L.; Strötgen, J.; and Schütze, H. 2024. Rehearsal-Free Modular and Compositional Continual Learning for Language Models. arXiv:2404.00790.

Wang, X.; Chen, T.; Ge, Q.; Xia, H.; Bao, R.; Zheng, R.; Zhang, Q.; Gui, T.; and Huang, X. 2023a. Orthogonal Subspace Learning for Language Model Continual Learning. arXiv:2310.14152.

Wang, Z.; Liu, Y.; Ji, T.; Wang, X.; Wu, Y.; Jiang, C.; Chao, Y.; Han, Z.; Wang, L.; Shao, X.; and Zeng, W. 2023b. Rehearsal-free Continual Language Learning via Efficient Parameter Isolation. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 10933–10946. Toronto, Canada: Association for Computational Linguistics.

Wang, Z.; Zhang, Z.; Lee, C.-Y.; Zhang, H.; Sun, R.; Ren, X.; Su, G.; Perot, V.; Dy, J.; and Pfister, T. 2022. Learning to Prompt for Continual Learning. arXiv:2112.08654.

Weng, J.; Zhang, Y.; and Hwang, W.-S. 2003. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8): 1034–1040.

Wu, C.; Gan, Y.; Ge, Y.; Lu, Z.; Wang, J.; Feng, Y.; Shan, Y.; and Luo, P. 2024a. LLaMA Pro: Progressive LLaMA with Block Expansion. arXiv:2401.02415.

Wu, T.; Caccia, M.; Li, Z.; Li, Y.-F.; Qi, G.; and Haffari, G. 2022. Pretrained Language Model in Continual Learning: A Comparative Study. In *International Conference on Learning Representations*.

Wu, T.; Luo, L.; Li, Y.-F.; Pan, S.; Vu, T.-T.; and Haffari, G. 2024b. Continual Learning for Large Language Models: A Survey. arXiv:2402.01364.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual Learning Through Synaptic Intelligence. arXiv:1703.04200.

Zhang, X.; Zhao, J.; and LeCun, Y. 2016. Character-level Convolutional Networks for Text Classification. arXiv:1509.01626.

Zhang, Y.; and Weng, J. 2001. Convergence analysis of complementary candid incremental principal component analysis.

Zheng, J.; Qiu, S.; Shi, C.; and Ma, Q. 2024. Towards Lifelong Learning of Large Language Models: A Survey. arXiv:2406.06391.

A Appendix

A.1 Online PCA in our method

In our method, we extract the basis of functional directions from their linear combinations using Online PCA (Cardot and Degras 2015). Specifically, we employ a modified Candid Covariance- free Incremental PCA (CCIPCA) (Weng, Zhang, and Hwang 2003) to implement Online PCA.

The CCIPCA technique. Let $\Gamma = \frac{1}{T-1}HH^\top$ be the covariance matrix $H = (h_1, h_2, ..., h_N)$ with standardized datas $h_1, h_2, ..., h_N$. Recall the goal of the PCA is to find the eigenvector u and the eigenvalue λ of Γ that satisfy

$$\Gamma u = \lambda u. \tag{18}$$

The idea of CCIPCA is as follows. For the first eigenvector v^1, assume that estimates $v_0^1, ..., v_{T-1}^1$ of $v = \lambda u$ have been constructed in previous steps $t = 1, 2, ..., T-1$. We substitute $h_t h_t^\top$ to Γ and $\frac{v_{t-1}^1}{\|v_{t-1}^1\|}$ to u in the eigenequation (18) for $t = 1, ..., T$, and average the results:

$$v_T^1 = \frac{1}{T}\sum_{t=1}^{T} h_t h_t^\top \frac{v_{t-1}^1}{\|v_{t-1}^1\|}. \tag{19}$$

Note that CCIPCA requires no historical datas $\{h_1, h_2, ..., h_{T-1}\}$, as equation (19) can be conveniently written in recursive form as:

$$v_{T+1}^1 = \frac{T-l}{T+1}v_T^1 + \frac{1+l}{T+1}h_{T+1}h_{T+1}^\top \frac{v_T^1}{\|v_T^1\|}, \tag{20}$$

where an amnesic factor $l \geq 0$ is introduced to handle non-stationary data generation, and the initialization is $v_0 = h_1$. The almost-sure convergence of equation (20) has been proved by (Zhang and Weng 2001). For estimating more than one eigenvector, say $v^1, v^2, ..., v^K$, to update the K-th eigenvector v_{T+1}^K, simply replace the input vector h_{T+1} in equation (20) with the following residual cutting:

$$h_{T+1}^* = h_{T+1} - \sum_{k=1}^{K-1} \frac{h_{T+1} \cdot v_T^k}{\|v_T^k\|^2} \cdot v_T^k. \tag{21}$$

Modified CCIPCA for tracking functional directions. To deal with the issue of functional direction drift, we introduce a tracking factor $\epsilon \in (0, 1)$ to equation (20) for a faster update:

$$v_{T+1}^1 = \eta \cdot v_T^1 + (1-\eta) \cdot h_{T+1}h_{T+1}^\top \frac{v_T^1}{\|v_T^1\|}, \tag{22}$$

where $\eta = \frac{T-l}{T+1} \cdot (1-\epsilon)$. Note that the convergence of equation (22) is disturbed for tracking. Algorithm 2 summarizes the modified tracking CCIPCA method, which additionally employs a residual threshold $\delta \in (0, 1)$ to append new components automatically (lines 7-10).

An example of functional direction tracking. We present the following example on the working process of functional direction tracking as a reference for Algorithm 2. Still, we take fine-tuning Llama-2 on CLBenchmark as an example. As shown in Figure 3, we update the principal components

Algorithm 2 Online PCA for Tracking Functional Directions

Parameter: Maximum principal component number K_{max}, amnesic factor l, tracking factor ϵ, residual threshold δ
Initialization: current principal component number $n = 0$
Input: The incoming model state data h_{T+1}
Output: Updated principal components $v_{T+1}^1, ..., v_{T+1}^K$

1: residual $h_{T+1}^* = h_{T+1}$
2: $\eta = \frac{T-l}{T+1} \cdot (1-\epsilon)$
3: **for** k in $range(n)$ **do**
4: update v_{T+1}^k using h_{T+1}^* with equation (22)
5: update h_{T+1}^* with equation (21)
6: **end for**
7: **if** $n < K_{\text{max}}$ and $\frac{\|h_{T+1}^*\|}{\|h_{T+1}\|} > \delta$ **then**
8: add a new component $v_{T+1}^{n+1} = h_{T+1}^*$
9: $n = n + 1$
10: **end if**

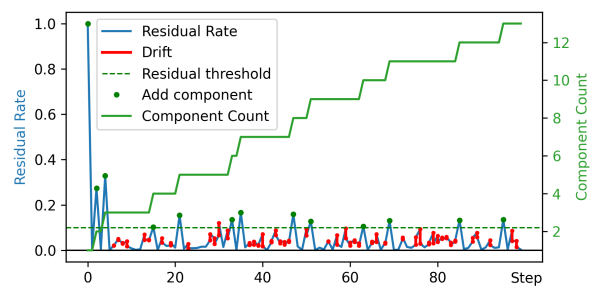


Figure 3: The update of principal components. If the residual rate is over the threshold, we add a new component for it. The red line shows that we track the drift by adjusting the tracking factor ϵ, whose increase mostly reduces the residual rate for better drift tracking.

based on the residual rate $\frac{\|h_t^*\|}{\|h_t\|}$ with residual threshold δ, abiding by lines 7-10 in Algorithm 2. Note that a lower residual rate indicates more complete coverage of LoRA increment with existing principal components. The tracking factor ϵ is adjusted dynamically following the increase and decrease of the residual rate, which is executed by redoing lines 2-6 in Algorithm 2 with adjusted ϵ and η. The results show that we continuously keep the residual rate less than 10%, covering 90% of the LoRA increment.

A.2 Addtional experimental details

For the Online PCA above, we let the amnestic factor $l = 2$, following the recommendation of (Weng, Zhang, and Hwang 2003). The empirical value of the tracking factor is that $\epsilon \in (0, 0.1)$, which is adjusted with the increase and decrease of the residual rate, and the residual threshold $\delta = 0.1$ for adding components automatically. For each new task, we enhance the maximum principal component number K_{max} by 48.

We follow O-LoRA(Wang et al. 2023a) and Progressive Prompt (Razdaibiedina et al. 2023) for the following contin-

ual learning settings:

Dataset details. Table 6 shows details of the datasets we employ for continual learning experiments, along with their evaluation metrics. Overall, we used datasets from CL benchmark (Zhang, Zhao, and LeCun 2016), GLUE (Wang et al. 2019), and SuperGLUE (Wang et al. 2020) benchmarks, adding the IMDB movie reviews (Maas et al. 2011). We randomly sample 100-10000 samples for each dataset, depending on their size, and fine-tune for 1000 steps for each incoming dataset in streaming data.

Task sequence of continual learning. The task orders used for our CL experiments across LLaMA and T5 models are shown in Table 7.

Prompts for different tasks. Table 8 shows prompts for different tasks. NLI denotes natural language inference, including MNLI, RTE, CB. SC denotes sentiment analysis, including Amazon, Yelp, SST-2, IMDB. TC denotes topic classification, including AG News, DBpedia, Yahoo.

A.3 Details for computation

Extract input vector x with token average. In our method, we extract LoRA increment $\mathbf{d}Wx$ as the functional direction, where x is the input *vector* of the module with the parameter matrix W. Note that in a transformer model, the input, say X, to W is several vectors, that is:

$$X = (x_1, x_2, ..., x_n) \tag{23}$$

where N is the number of input tokens, x_n is the input vector at the place of the n-th token. Common methods to represent inputs $x_1, x_2, ..., x_n$ with a single vector x include computing their average or taking the last vector. For stability of computation, we employ the average method, that is:

$$x = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{24}$$

Standarization of LoRA increment h for PCA. We employ the LoRA increment h(computed with equation (6)) as the functional direction in our method. As there is no scale difference in gradients, we omit normalization, following (Cardot and Degras 2015). Note that we are concerned with only the directions of h, so we also conduct no centralization for h at the beginning. Note that in this case, the first few principal component represents the weighted historical average (19), and the residual cut in equation (21) will deduct the average and thus reach certain centralization. Also, the effect of other normalization methods designed for LoRA increments or gradients deserves further investigation.

A.4 Additional Related works

The following methods have been developed for LLM continual learning. They can be categorized into the following types: Rehearsal-based , Architecture-based , Prompt-based, and Regularization-based approaches. A brief summary is in Table 10.

Rehearsal-based approach (de Masson d'Autume et al. 2019; Mok et al. 2023; Huang et al. 2021) try to remind the model of historical tasks and thus avoid forgetting. However, there are growing restoration costs as tasks accumulate, and privacy issues in gaining historical training data.

Architecture-based approach (Jang et al. 2023; Wang et al. 2024; Peng et al. 2024) train multiple expert models for each task. However, when it comes to unseen tasks, there is no proper expert to use, which destroys the generalization ability of models.

Prompt-based approach L2P(Wang et al. 2022), LFPT5(Qin and Joty 2022), and Progressive Prompts(Razdaibiedina et al. 2023) add prompts during the inference of the model. This approach is lightweight, but when the fine-tuning information gets large, the prompt will not be able to cover it.

Regularization-based Approach EWC(Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017), LwF(Li and Hoiem 2017), OGD(Farajtabar et al. 2019), and O-LoRA(Wang et al. 2023a) limit the update of model parameters to preserve the historical ability of the model. Their edge is that no historical data or extra architecture is required. We lay emphasis on the orthogonal methods, including OGD and O-LoRA, as we employ orthogonal cuts to avoid changing historical parameter settings and preserve historical functions.

Orthogonal methods The key point of orthogonal methods is to avoid wrecking the parameter subspace of historical tasks when fine-tuning on the latest task, and the method is to make the parameter space of new tasks orthogonal to the historical ones. Representative methods, including Orthogonal Gradient Descent (OGD)(Farajtabar et al. 2019) and Orthogonal Subspace Learning(O-LoRA)(Wang et al. 2023a), have been proven effective in preventing catastrophic forgetting. OGD forces the gradient descent to be orthogonal to the gradient directions of historical tasks. That is:

$$G_T \perp G_t \quad t = 1, 2, ..., T-1 \tag{25}$$

where $G_t = \nabla_\theta L_t$ is the gradient direction of the t th task. O-LoRA tries to make the LoRA B matrix in equation (2) orthogonal to that of historical LoRA modules. That is:

$$\beta_T^i \perp \beta_t^j \quad t = 1, 2, ...T-1 \quad i, j = 1, 2, ..., r \tag{26}$$

where β_t^i is the i th colomun vector of $B_{m\times r}$ matrix fine-tuned in the t th task, that is $B = (\beta_1, \beta_2, ..., \beta_r)$.

A.5 Limitations

While the proposed method has an outstanding performance in empirical evaluations, we discuss its potential limitations as follows.

Scalability. In more complex scenarios with a large number of tasks, such as hundreds of tasks, the principal component pool expands as we add new components during the fine-tuning process, imposing a growing load for computation. The empirical scale of the expansion is approximately 40 components for each task, as shown in Figure 4. The size of these components is approximately 15MB and is acceptable in the settings of our experiments. However, in the case of hundreds of tasks, the performance and applicability of our method require further investigation.

Table 6: The details of 15 datasets used in the CL experiments, following O-LoRA and Progressive Prompt. NLI denotes natural language inference, QA denotes the question and answer task.

Dataset name	Category	Task	Domain	Metric
1. Yelp	CL Benchmark	sentiment analysis	Yelp reviews	accuracy
2. Amazon	CL Benchmark	sentiment analysis	Amazon reviews	accuracy
3. DBpedia	CL Benchmark	topic classification	Wikipedia	accuracy
4. Yahoo	CL Benchmark	topic classification	Yahoo Q&A	accuracy
5. AG News	CL Benchmark	topic classification	news	accuracy
6. MNLI	GLUE	NLI	various	accuracy
7. QQP	GLUE	paragraph detection	Quora	accuracy
8. RTE	GLUE	NLI	news, Wikipedia	accuracy
9. SST-2	GLUE	sentiment analysis	movie reviews	accuracy
10. WiC	SuperGLUE	word sense disambiguation	lexical databases	accuracy
11. CB	SuperGLUE	NLI	various	accuracy
12. COPA	SuperGLUE	QA	blogs, encyclopedia	accuracy
13. BoolQA	SuperGLUE	boolean QA	Wikipedia	accuracy
14. MultiRC	SuperGLUE	QA	various	accuracy
15. IMDB	SuperGLUE	sentiment analysis	movie reviews	accuracy

Table 7: Different orders of task sequences used for continual learning experiments. Orders 1-3 correspond to the standard CL benchmark, orders 4-6 are long chain of tasks, following O-LoRA and Progressive Prompt.

Order	Task Sequence
1	dbpedia → amazon → yahoo → ag
2	dbpedia → amazon → ag → yahoo
3	yahoo → amazon → ag → dbpedia
4	mnli → cb → wic → copa → qqp → boolqa → rte → imdb → yelp → amazon → sst-2 → dbpedia → ag → multirc → yahoo
5	multirc → boolqa → wic → mnli → cb → copa → qqp → rte → imdb → sst-2 → dbpedia → ag → yelp → amazon → yahoo
6	yelp → amazon → mnli → cb → copa → qqp → rte → imdb → sst-2 → dbpedia → ag → yahoo → multirc → boolqa → wic

Task identification. Although our method requires no task identification during inference, it is still required during the continual fine-tuning process. Exploring methods for task-agnostic training would be a valuable future direction. This is further discussed in Future Directions.

Generalization Ability. As our method is targeted at preserving historical functions, it has no recognition of unseen tasks. Its generalization ability deserves further investigation. We propose the following empirical demonstration of the impact of our method on the generalization ability of the model. Following O-LoRA (Wang et al. 2023a), we start with a fine-tuned LLaMA-7B language model on the Alpaca (Taori et al. 2023) dataset. After conducting continual learning on the CL benchmark (Zhang, Zhao, and LeCun 2016), we test the model on the MMLU benchmark (Hendrycks et al. 2021b,a), composed of unseen tasks. The results are shown in Table 9. Compared to the original model, SeqLoRA and our method (DOC) suffer from forgetting (accuracy respectively drops from 36.0 to 26.2 and 29.4). This is because of the lack of information about unseen tasks during continual learning. In the experimental settings, the issues above limit the practicality of DOC.

Note that we fine-tune the model on Alpaca at the beginning, so that continual learning also triggers the forgetting of Alpaca. What if we mitigate this forgetting with CL methods? We further investigate the effect of continual fine-tuning the model on the Alpaca and CL benchmark with CL methods applied **throughout from start to end**, which makes the Alpaca **visible** to the methods. Note that MMLU is still unseen during the continual fine-tuning process in this setting. The results show the enhanced performance (an accuracy of 34.6 for DOC and 32.1 for O-LoRA, compared to 29.4 and 30.1 in the former experiment where Alpaca is invisible to the methods). An explanation is that the methods avoid forgetting Alpaca, which is a general dataset that assists in the initialization of crucial functional directions of the model, thus aiding in the preservation of crucial functions for unseen tasks. The results also indicate that the generalization of our method, which preserves the ability on unseen tasks with a general dataset, is better compared to O-LoRA. It inspires the practical deployment of DOC to initialize on a general dataset beforehand.

A.6 Future directoins

The interpretability of principal components. We employ PCA in our method for functional direction tracking.

Table 8: Instructions for different tasks, following O-LoRA and Progressive Prompt.

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

Table 9: The accuracy on the MMLU benchmark of LLaMA-7B before and after continual learning (CL) on the CL benchmark. The results are the average of task orders 1-3. Note that with MMLU being a four-classification problem, a 25% accuracy equates to random guessing.

	MMLU Accuracy
Original model	32.3
Alpaca LoRA fine-tuned model	36.0
Seq LoRA CL after Alpaca LoRA	26.2
O-LoRA CL after Alpaca LoRA	30.1
DOC CL after Alpaca LoRA	29.4
O-LoRA throughout Alpaca and CL	32.1
DOC throughout Alpaca and CL	**34.6**

Another edge of PCA is that the components extracted are statistically independent of each other; thus, each component represents an individual unit, as proposed by (Michaud et al. 2024). The individuality of these components provides chances for model deconstruction and better interpretability, and it is possible to find the exact meaning of each component, *e.g.*, semantic function, logic function, certain knowledge, *etc.*, through empirical methods. It is a promising direction for interpretable learning based on model deconstruction.

Automated task ID recognition As mentioned before, exploring methods for task-agnostic training would be valuable. It deserves further investigation into the characteristics of the principal components extracted from a specific task, which assists in the distinction of different tasks.

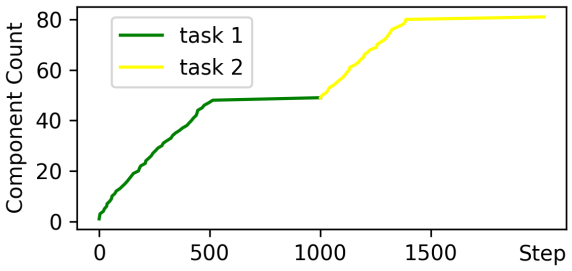


Figure 4: The expansion of principal components in the first 2 tasks. Note that we do not limit the maximum principal component number in this experiment, *i.e.*, $K_{\max} = +\infty$. As the number of principal components increases, it reaches a point where no extra component is required, indicating that the current components are adequate to cover a large enough part of the LoRA increment.

Table 10: The comparison of continual learning methods. Specifically, **RF** indicates whether the method is rehearsal-free. **TIF** indicates whether the task ID is free during inference. Compared to regularization-based methods, other methods have extra settings or computational overheads.

		RF	**TIF**	**Inference costs**
Rehearsal-based	MBPA++ (de Masson d'Autume et al. 2019)		✓	
	IDBR (Huang et al. 2021)		✓	
Architecture-based	EIP (Wang et al. 2023b)	✓	✓	Expert selection
	SLM (Peng et al. 2024)	✓	✓	
	Expert LMs (Jang et al. 2023)		✓	
	MoCL (Wang et al. 2024)	✓	✓	
Prompt-based	L2P (Wang et al. 2022)	✓	✓	Additional prompts
	LFPT5 (Qin and Joty 2022)		✓	
	ProgPrompt (Razdaibiedina et al. 2023)	✓		
Regularizatoin-based	EWC (Kirkpatrick et al. 2017)	✓	✓	
	LwF (Li and Hoiem 2017)	✓		
	OGD (Farajtabar et al. 2019)	✓	✓	
	O-LoRA (Wang et al. 2023a)	✓	✓	
	DOC(ours)	✓	✓	