OnEDIT: Online Editing with Decoupled Implicit Task for Large Language Models

Chae-Won Lee[*1], **Jae-Hong Lee**[*2], **Ji-Hun Kang**[1], **Joon-Hyuk Chang**[1†]

[1]Department of Electronic Engineering Hanyang University Seoul, Republic of Korea
[2]Division of Language & AI Hankuk University of Foreign Studies Seoul, Republic of Korea
dlcodnjs721@hanyang.ac.kr, ljh93ljh@hufs.ac.kr, kjh95@hanyang.ac.kr, jchang@hanyang.ac.kr

Abstract

Continual instruction tuning (CIT) has emerged as a promising strategy for adapting large language models (LLMs) to new tasks while preserving historical knowledge. Most existing CIT methods have focused on offline CIT (offCIT), which assumes clearly defined task boundaries and allows multiple passes over the data. However, such assumptions rarely hold in real-world scenarios, where data arrive in a streaming fashion and task boundaries are unknown. This setting introduces critical challenges: the absence of task identifiers (task IDs), a significant imbalance in task-specific information, and inaccessibility to previously seen data. In this work, we propose Online Editing with Decoupled Implicit Task (OnEDIT), an online CIT(onCIT) approach to tackle these challenges. OnEDIT leverages a fixed-size adapter for the implicit task, balancing current and past knowledge through editing operations every time step without relying on task IDs or backpropagation. Extensive experiments on CIT benchmarks demonstrate that OnEDIT consistently maintains robust and stable performance, whereas existing state-of-the-art baselines often suffer from performance degradation in online settings. It suggests that OnEDIT achieves superior generalization across diverse task orders and model scales, while maintaining high efficiency and low memory overhead.

Introduction

Large language models (LLMs) have shown rapid advances in recent years, demonstrating remarkable capabilities across a wide spectrum of tasks (Min et al. 2023; Zhao et al. 2023; Zhou et al. 2023; Dubey et al. 2024). However, as model sizes and their computational costs continue to grow (Wei et al. 2021; Yao et al. 2024), the need for more efficient learning strategies that preserve previously acquired knowledge while adapting to novel domains is necessitated. Continual instruction tuning (CIT) has emerged as a promising paradigm to extend the knowledge of powerful LLMs and adapt them to new tasks over time (Shi et al. 2024). However, sequentially learning multiple tasks often results in the well-known problem of catastrophic forgetting, whereby the performance of the model on previously learned tasks degrades as new knowledge is acquired (Jang et al. 2022; Jin et al. 2022; Chen et al. 2023; Cossu et al. 2024; Ke et al. 2022). This has led to the development of more effective approaches for CIT, which can be broadly categorized into two paradigms: offline continual instruction tuning (offCIT) and online continual instruction tuning (onCIT) (Yang et al. 2024).

OffCIT has been extensively studied, operating under the assumption that task boundaries are clearly defined and allowing task-specific and multi-run training (Song et al. 2023; Scialom et al. 2022). In contrast, onCIT assumes that the data arrive in a stream manner without explicit task boundaries and that each sample can be observed only once (de Masson d'Autume et al. 2019; Geng et al. 2021; Holla et al. 2020; Li et al. 2023, 2025). This online setting better reflects real-world deployment scenarios, but poses significant challenges: the absence of task identifiers (task IDs), severe task information imbalance and inaccesibility to previous data. To mitigate the first two issues, many state-of-the-art methods resort to memory buffers, allowing repeated access to stored data (Wang et al. 2024; Song et al. 2023; Sun et al. 2019; Holla et al. 2020). Although effective, this approach raises privacy and scalability concerns, as it requires storing data for each new task. To overcome these limitations, model expansion-based approaches have gained attention as a more practical and privacy-preserving alternative for onCIT, where previous data cannot be revisited.

Among parameter-efficient fine-tuning (PEFT) methods, low-rank adaptation (LoRA) (Hu et al. 2022) has been adopted as a fundamental technique for model expansion-based strategies. By reparameterizing pretrained weights into a low-rank decomposition, LoRA allows LLMs to adapt to downstream tasks by updating only a small subset of parameters, thereby enabling efficient CIT. Existing LoRA-based CIT methods (Zhang et al. 2023a; Bohao et al. 2024; Michieli et al. 2024) typically instantiate a new LoRA branch for each incoming task while freezing previous branches, thus mitigating catastrophic forgetting by isolating task-specific knowledge. However, these methods are often impractical in onCIT scenarios, where task IDs are not available at inference time. To address this, recent work proposes integrating new and previous LoRA branches via a additive composition (Wang et al. 2023; Liang et al. 2024; Smith et al. 2023). Unfortunately, these approaches force equal contribution across all branches, exacerbating

[*]These authors contributed equally.
[†]Corresponding author.

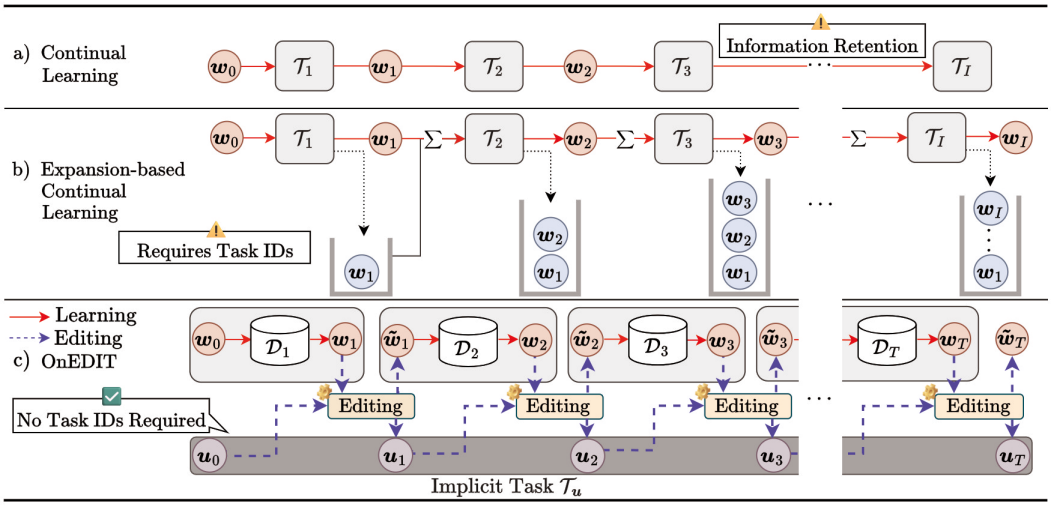

Figure 1: Overview of general continual learning approaches and the proposed Online Editing with Decoupled Implicit Task (OnEDIT) framework. (a) General continual learning sequentially adapts to a series of tasks $\mathcal{T}_1, \ldots, \mathcal{T}_I$ using a task-specific adapter $\boldsymbol{w}_i$ for each task. (b) Expansion-based methods allocate a separate adapter for each task, indexed by a known task ID $i \in \{1, \ldots, I\}$, and retain them during inference. (c) The proposed OnEDIT framework addresses the online continual instruction tuning (onCIT) setting, where task IDs are unavailable. At each time step t, the current adapter $\boldsymbol{w}_t$ is obtained by optimizing the previous adapter $\boldsymbol{w}_{t-1}$ on a mini-batch $\mathcal{D}_t$ sampled from unknown tasks. The adapter $\boldsymbol{u}_t$, assigned to the implicit task $\mathcal{T}_u$, is updated through recursive editing operations involving $\boldsymbol{w}_t$ and $\boldsymbol{u}_{t-1}$. The detailed editing operations are shown in Figure 2.

the imbalance of task information and hindering overall performance in CIT settings.

To address the key challenges in onCIT—task information balancing, absence of task IDs, and inaccessibility to past data—we propose a novel method, Online Editing with Decoupled Implicit Task (OnEDIT). OnEDIT introduces an implicit task and assigns a single adapter to this task, eliminating the need for explicit task IDs while reducing redundant information acquisition (see Figure 1). OnEDIT focuses on the current adapter trained with the latest mini-batch, iteratively integrated into the implicit task adapter every update, accumulating historical knowledge. This integration is achieved using editing operations that compare current information against historical knowledge, suppressing redundancy before updating the implicit task adapter. A subsequent regularization operation ensures that the current adapter does not diverge significantly from the implicit one, promoting stable knowledge accumulation. The main contributions of this work are summarized as follows:

- OnEDIT operates without requiring task IDs or access to previous data and mitigates catastrophic forgetting, thereby naturally balancing information across tasks while preserving historical knowledge and enabling robust adaptation to novel tasks.
- The implicit task mechanism relies solely on editing operations without backpropagation, introducing minimal additional computation, and requires only a fixed-size adapter for the implicit task, keeping memory overhead low.
- Extensive experiments on CIT benchmarks demonstrate the efficacy of OnEDIT in alleviating catastrophic forgetting and demonstrate its suitability for real-time updates in the onCIT setting.

We evaluated OnEDIT on standard CIT benchmarks in the onCIT setting, comparing its performance against existing state-of-the-art methods. Although prior approaches often suffer performance degradation in online environments, OnEDIT consistently demonstrates superior generalization across varying task orders and model scales. These improvements stem from its ability to mitigate task information imbalance through online editing with the implicit task, enabling rapid adaptation to novel tasks while effectively preserving historical knowledge.

Related Works

Continual Learning for LLMs

Recent advances in LLMs have led to growing interest in enabling them to adapt to new tasks continually without retraining from scratch. This paradigm, known as CIT, extends traditional continual learning to instruction-tuned LLMs by sequentially updating the model with new task instructions, while aiming to retain previously acquired knowledge (Shi et al. 2024). CIT methods can be broadly categorized into three types: regularization-based, memory-based, and expansion-based approaches. Regularization-based methods introduce penalty terms in the loss function to reduce the deviation from previously learned parameters (Kirkpatrick et al. 2017; Aljundi et al. 2018; Rongali et al. 2020). Al-

though simple and efficient, these methods often struggle with stability when scaling to LLMs. Memory-based approaches store a subset of previous task data or generate pseudo samples to replay during new task training (Buzzega et al. 2020; Rebuffi et al. 2017; Zhao et al. 2021). These methods require careful memory management and risk privacy leakage in sensitive applications. Expansion-based methods, on the other hand, dynamically allocate new model components for each new task while freezing existing components (Dettmers et al. 2024; Wang et al. 2023; Wu et al. 2024; Yan et al. 2023). As the adoption of LoRA (Hu et al. 2022) as a standard method, expansion-based methods have gained attention particularly to LLM because of their compatibility with PEFT (Houlsby et al. 2019; Li and Liang 2021; He et al. 2021). Despite their effectiveness, existing expansion-based methods assume that task IDs are known at inference time, enabling direct use of the correct parameter blocks (Zhang et al. 2023a; Bohao et al. 2024). However, this assumption rarely holds in real-world scenarios, making such methods less practical. Furthermore, simple summation or concatenation of all PEFT modules is prone to catastrophic forgetting and hinders knowledge transfer between tasks.

Online Continual Instruction Tuning

Most prior studies on continual instruction tuning have been conducted in offline settings, where task boundaries are explicitly defined and data are revisited multiple times. However, in a real-world implementation, the data arrive as a stream and can only be observed once, making the boundaries of the task uncertain or completely unavailable. This discrepancy limits the applicability of offline CIT methods to dynamic real-time environments. To address these challenges, recent studies have increasingly focused on online settings. Some methods tackle unclear task boundaries through rapid adaptation (de Masson d'Autume et al. 2019; Wang et al. 2020), while others, such as (Geng et al. 2021), manage prior knowledge using pruning, expansion, and masking in encoder-decoder models. Online meta-learning approaches (Javed et al. 2019; Beaulieu et al. 2020) have also been adapted for continual learning via episodic memory replay (Holla et al. 2020). Additionally, (Li et al. 2023) shows promise in handling noisy, non-stationary streams for relation learning. However, these approaches remain largely limited to the pretraining phase of LLM and do not address onCIT to expand task-specific knowledge. To this end, our proposed OnEDIT, a method that harnesses PEFT to enable onCIT without relying on explicit task IDs, makes it well suited for implementation in realistic streaming environments.

Editing Task Vectors with Arithmetic Operation

Recent studies have demonstrated that arithmetic operations between adapted weights can concretely implement semantic intents (Ilharco et al. 2022). These semantic intents include improving the performance of the downstream tasks, alleviating biases or unwanted behaviors, aligning the model with human preferences, or updating the model with new information. Such semantic intents are based on the concept of a task vector. The task vector is defined as:

$$\boldsymbol{\tau}_i = \boldsymbol{W}_i - \boldsymbol{W}_0, \tag{1}$$

where $\boldsymbol{W}_i$ represents the weights adapted for task i, and $\boldsymbol{W}_0$ denotes initial weights of the model. This approach encodes the information needed to adapt to a specific task, introducing a new paradigm for neural network editing. Inspired by studies on weight interpolation (Guo et al. 2023; Wortsman et al. 2022; Rame et al. 2022, 2023), task vectors enable task editing, performing element-wise operations to edit various models. For example, adding task vectors can enhance multi-task model performance to achieve generalized capabilities (first row in Table 1), while unlearning can help the model remove unwanted behaviors or forget specific tasks (second row in Table 1). Furthermore, when tasks share similar relationships, combining task vectors allows concrete computations of abstract concepts such as domain transfer (third row in Table 1).

Semantic Intent	Arithmetic Operation
Multi-task learning	$\boldsymbol{\tau}_\alpha + \boldsymbol{\tau}_\beta$
Unlearning	$\boldsymbol{\tau}_\alpha - \boldsymbol{\tau}_\beta$
Domain transfer	$\boldsymbol{\tau}_\gamma + (\boldsymbol{\tau}_\alpha - \boldsymbol{\tau}_\beta)$

Table 1: Semantic intent and their arithmetic operation for α, β and γ tasks.

Problem Definition and Setup

Online Scenarios

CIT aims to enable LLMs to solve tasks that emerge sequentially within a continuous data stream. In the offCIT setting, a model incrementally adapts to a sequence of tasks $(\mathcal{T}_1, \ldots, \mathcal{T}_I)$, each identified by a task ID $i \in \{1, \ldots, I\}$. We denote the dataset associated with task $\mathcal{T}_i$, consisting of N_i samples, as $\mathcal{D}_i = \{(x_i^n, y_i^n) : n = 1, \ldots, N_i\}$, where x_i^n and y_i^n are the input and target texts, respectively.

In contrast, onCIT does not receive data as task-specific datasets $\mathcal{D}_i$ associated with a given task ID i, but instead observes data as a stream of mini-batches $\mathcal{D}_t$ of size B at each time step $t \in [1, T]$, where each sample is encountered only once. Extending prior works on continual learning for language models (Liang et al. 2025; Zhao et al. 2024; Wang et al. 2023), we focus on a more challenging and realistic online setting characterized by: (1) the absence of task IDs during both training and inference, (2) single-pass learning without repeated samples, and (3) no access to previously seen data.

Imbalance in Task Information

With the widespread adoption of LoRA, model expansion-based methods that add new adapters as tasks arrive have gained increasing attention (Dettmers et al. 2024; Wang et al. 2023; Wu et al. 2024; Yan et al. 2023). For the current task i, these methods update $\boldsymbol{h}$ using adapter weights $\boldsymbol{w}_i \in \mathbb{R}^d$ as follows:

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + (\boldsymbol{w}_0 + \boldsymbol{w}_1 + \boldsymbol{w}_2 + \cdots + \boldsymbol{w}_{i-1})\boldsymbol{z}, \tag{2}$$

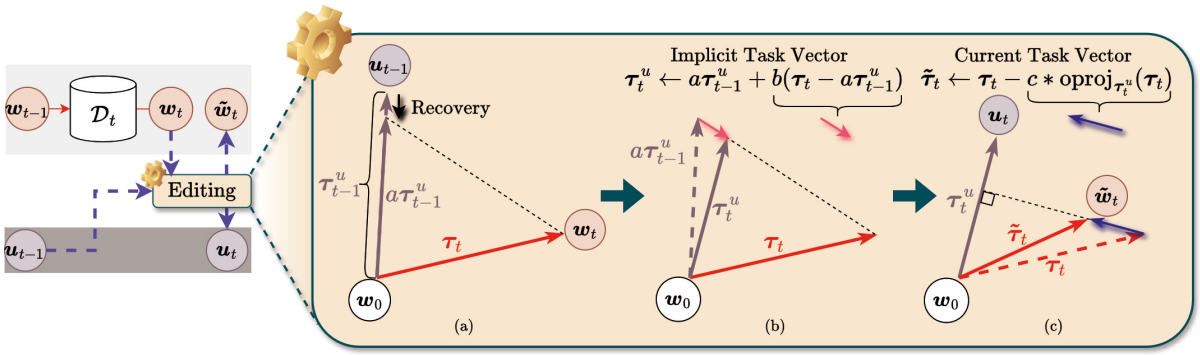

Figure 2: Illustration of editing operations in OnEDIT. (a) Recovery operation, which preserve knowledge of the initial weights $\boldsymbol{W}_0$. (b) Editing operation for the implicit task vector. (c) Editing operation for the current task vector.

where the adapters from previous tasks are stored and frozen after training, given the input $\boldsymbol{z} \in \mathbb{R}^k$ and the output $\boldsymbol{h} \in \mathbb{R}^l$ for dimensions k, l, and $d = lk$. This method accumulates adapters from all previous tasks, analogous to arithmetic that represents multitask learning (as shown in Table 1). However, such additive accumulation implicitly assumes equal importance across tasks and ignores sequential dependencies and redundancy between adapters, leading to task information imbalance, reduced learning efficiency, and poor performance on earlier tasks.

To address this balancing issue, methods that adjust the importance among task-specific adapters with weighted summation have proven to be effective. SAPT (Zhao et al. 2024) aligns task adapters through a shared attention mechanism, and GainLoRA (Liang et al. 2025) introduces separate LoRA branches per task and merges them with gating modules. Our proposed OnEDIT also performs information balancing using a weighted scheme, but instead of maintaining separate task-specific adapters, it dynamically modulates information within a single adapter $\boldsymbol{u}_t \in \mathbb{R}^d$ assigned to the implicit task $\mathcal{T}_u$. This is achieved by performing editing operations between $\boldsymbol{u}_{t-1}$ and the current task adapter $\boldsymbol{w}_t$ at each time step t, integrating new task information while suppressing redundancy.

Online Editing with Decoupled Implicit Task

Overall Process

We consider the onCIT setting, where data arrive at each time step t in the form of mini-batches $\mathcal{D}_t = \{(x_t^n, y_t^n) : n = 1, \ldots, N_t\}$, where (x_t^n, y_t^n) are input-output text pairs, without any associated task IDs. The optimization process at each time step t is defined as:

$$\boldsymbol{w}_t \leftarrow \arg\max_{\tilde{\boldsymbol{w}}_{t-1}} L_t(\{\tilde{\boldsymbol{w}}_{t-1}, \boldsymbol{W}_0\}, \mathcal{D}_t), \tag{3}$$

where $\boldsymbol{w}_t$ represents the updated adapter weights at the time t, $\tilde{\boldsymbol{w}}_0$ is assigned as $\boldsymbol{w}_0$ initialized with zero, and $\boldsymbol{W}_0$ denotes the initial weights of the LLMs. The objective function L is the log-likelihood defined as:

$$L_t = \frac{1}{B} \sum_{(x_t^n, y_t^n) \in \mathcal{D}_t} \log p(y_t^n \mid x_t^n; \{\tilde{\boldsymbol{w}}_{t-1}, \boldsymbol{W}_0\}). \tag{4}$$

To compute $\tilde{\boldsymbol{w}}_t$, we introduce a recursive editing operation defined as:

$$\tilde{\boldsymbol{w}}_t, \boldsymbol{u}_t \leftarrow \text{Editing}(\boldsymbol{w}_t, \boldsymbol{u}_{t-1}), \tag{5}$$

where $\boldsymbol{u}_t$ is an adapter assigned to an implicit task $\mathcal{T}_u$ with the same dimensionality as $\boldsymbol{w}_t$. The function Editing$(\cdot)$ operates recursively on $\boldsymbol{w}_t$ and $\boldsymbol{u}_t$, starting with $\boldsymbol{u}_0$ initialized as $\boldsymbol{w}_0$. This allows the implicit task representation to progressively integrate knowledge from past data without maintaining separate adapters for each task, enabling an onCIT process.

Editing for Implicit Task Vector

In the implicit task, we aim to integrate historical knowledge with current information to generalize across multiple tasks. Starting from the zero-initialized adapter $\boldsymbol{w}_0$, the current task vector $\boldsymbol{\tau}_t$ can be defined as $\boldsymbol{\tau}_t = \boldsymbol{w}_t - \boldsymbol{w}_0$, and the task vector for the implicit task as $\boldsymbol{\tau}_t^u = \boldsymbol{u}_t - \boldsymbol{w}_0$. Task vector editing is based on the notion that generalized weights exist in the interpolation regions between weights (Guo et al. 2023; Wortsman et al. 2022; Rame et al. 2022, 2023). Therefore, we compute the implicit task vector by interpolating $\boldsymbol{w}_0$, $\boldsymbol{u}_{t-1}$, and $\boldsymbol{w}_t$ as follows:

$$\boldsymbol{u}_t = \lambda_1 \boldsymbol{w}_0 + \lambda_2 \boldsymbol{u}_{t-1} + \lambda_3 \boldsymbol{w}_t, \tag{6}$$

subject to the constraints $\lambda_1 + \lambda_2 + \lambda_3 = 1$ and $0 \leq \lambda_1, \lambda_2, \lambda_3 \leq 1$. Since $\lambda_1 = 1 - \lambda_2 - \lambda_3$, we can rewrite the equation as $\boldsymbol{u}_t - \boldsymbol{w}_0 = \lambda_2(\boldsymbol{u}_{t-1} - \boldsymbol{w}_0) + \lambda_3(\boldsymbol{w}_t - \boldsymbol{w}_0)$. Expressing this in terms of the task vectors $\boldsymbol{\tau}_t$ and $\boldsymbol{\tau}_t^u$, we have $\boldsymbol{\tau}_t^u = \lambda_2 \boldsymbol{\tau}_{t-1}^u + \lambda_3 \boldsymbol{\tau}_t$. By setting $\lambda_2 = a(1-b)$ and $\lambda_3 = b$, for $0 \leq a, b \leq 1$, the implicit task vector becomes

$$\boldsymbol{\tau}_t^u \leftarrow a\boldsymbol{\tau}_{t-1}^u + b(\boldsymbol{\tau}_t - a\boldsymbol{\tau}_{t-1}^u), \tag{7}$$

$$\boldsymbol{u}_t \leftarrow \boldsymbol{w}_0 + \boldsymbol{\tau}_t^u \tag{8}$$

The Equation (7) is a weighted version of the domain transfer shown in Table 1, where the influence of each task is naturally incorporated. The hyperparameter a controls the influence of the previous implicit task vector $\boldsymbol{\tau}_{t-1}^u$ (see Figure 2 (a)), recovering knowledge of initial weights $\boldsymbol{w}_0$, while b controls the degree to which new information is integrated. A smaller b leads to slower adaptation to rapidly changing

information (see Figure 2 (b)). The term $(\boldsymbol{\tau}_t - a\boldsymbol{\tau}^u_{t-1})$ compares the current task vector $\boldsymbol{\tau}_t$ with the scaled implicit task vector of the previous step $a\boldsymbol{\tau}^u_{t-1}$, transferring novel information that is not already encoded in historical knowledge. By adding this adjusted difference to $a\boldsymbol{\tau}^u_{t-1}$, we prevent the excessive accumulation of redundant task information in the implicit task and effectively mitigate catastrophic forgetting.

Editing for Current Task Vector

We leverage the non-overlapping information from the implicit task adapter to modify the current adapter. However, sufficient diversity among weights is necessary for generalized weights to exist in the interpolation regions between weights (Wortsman et al. 2022; Rame et al. 2022). Paradoxically, the implicit task vector can limit representational diversity, making it essential for the current task vector to remain sufficiently distinct. To achieve this, we encourage the current task vector to incorporate historical knowledge while remaining sufficiently distinct from the implicit task vector. Specifically, we compute the orthogonal projection of $\boldsymbol{\tau}_t$ onto $\boldsymbol{\tau}^u_t$ using task vector arithmetic.

$$\text{oproj}_{\boldsymbol{\tau}^u_t}(\boldsymbol{\tau}_t) = \boldsymbol{\tau}_t - \frac{\boldsymbol{\tau}_t \cdot \boldsymbol{\tau}^u_t}{\boldsymbol{\tau}^u_t \cdot \boldsymbol{\tau}^u_t}\boldsymbol{\tau}^u_t, \tag{9}$$

where $\cdot$ denotes the dot product. This operation computes the component of $\boldsymbol{\tau}_t$ that is orthogonal to $\boldsymbol{\tau}^u_t$, which captures novel information not yet encoded in $\boldsymbol{\tau}^u_t$, such as (Wang et al. 2023). As illustrated in Figure 2(c), the magnitude of this orthogonal component is proportional to the angle between the two vectors, which is their dissimilarity. By leveraging this orthogonal vector, we dynamically adjust the distance between $\boldsymbol{\tau}_t$ and $\boldsymbol{\tau}^u_t$. Finally, we update $\boldsymbol{\tau}_t$ using the following editing operation:

$$\tilde{\boldsymbol{\tau}}_t \leftarrow \boldsymbol{\tau}_t - c \cdot \text{oproj}_{\boldsymbol{\tau}^u_t}(\boldsymbol{\tau}_t), \tag{10}$$

$$\tilde{\boldsymbol{w}}_t \leftarrow \boldsymbol{w}_0 + \tilde{\boldsymbol{\tau}}_t \tag{11}$$

where $0 \leq c \leq 1$ is a hyperparameter that controls the influence of the orthogonal component. This operation corresponds to a weighted version of the unlearning in Table 1, which serves as regularization that prevents $\boldsymbol{\tau}_t$ from deviating too far from $\boldsymbol{\tau}^u_t$ by removing redundant components. As a result, the editing operation effectively balances task diversity and shared information, using the implicit task vector as a support.

Experiments

Experimental Setup

Datasets Extending existing continual learning methods for LLMs (Liang et al. 2025; Zhao et al. 2024; Wang et al. 2023), we evaluated our approach on three benchmarks: the standard CIT benchmark (Qin et al. 2021), the long-sequence benchmark (Razdaibiedina et al. 2023), and the SuperNI benchmark (Wang et al. 2022). The standard CIT benchmark consists of four classification tasks and is evaluated under three different task orders (Order 1, 2, and 3), following the protocol in (Qin et al. 2021). The long-sequence benchmark includes 14 diverse classification tasks and is evaluated in three task orders (Order 4, 5 and 6), as described in (Razdaibiedina et al. 2023). The SuperNI benchmark spans a wide range of task types, including dialogue generation, information extraction, question answering, summarization, and sentiment analysis, evaluated in two task orders (Order 7, 8). Details of the task sequences are provided in the Appendix C.1.

Evaluation Metrics Following conventional continual learning evaluation protocols (Chaudhry et al. 2019), we evaluate performance using four metrics. Let $a_{i,j}$ denote the test performance on the j-th task after training on the i-th task, where accuracy is used for classification tasks and ROUGE-L (Lin 2004) for others. The four metrics are as follows:

(1) Average Performance (AP): the average accuracy across all tasks after completing training on the final task, i.e, $\text{AP} = \frac{1}{I}\sum_{i=1}^{I} a_{I,i}$.

(2) Forgetting Rate (FR): measures how much knowledge of the previous $I-1$ tasks is forgotten after learning the final task, i.e., $\text{FR} = \frac{1}{I-1}\sum_{i=1}^{I-1}\left(\max_{k=1}^{I-1} a_{k,i} - a_{I,i}\right)$.

(3) Forward Transfer (FWT): quantifies how much knowledge learned from previous tasks helps with learning new tasks, i.e, $\text{FWT} = \frac{1}{I}\sum_{i=1}^{I}(a_{i,i} - a_{0,i})$.

(4) Backward Transfer (BWT): evaluates how learning new tasks influences the performance on previously learned tasks, i.e, $\text{BWT} = \frac{1}{I-1}\sum_{i=1}^{I-1}(a_{I,i} - a_{i,i})$.

Comparison Methods We compared OnEDIT with eight CIT baselines for LLMs. LoRA (Hu et al. 2022) learns sequential tasks using a fixed-size adapter without regularization or replay. L2 (Zhang et al. 2023b) adds L2 regularization to constrain adapter drift. IncLoRA (Zhang et al. 2023a) incrementally introduces new adapters per task, while OLoRA (Wang et al. 2023) enforces orthogonality among them. InfLoRA (Liang et al. 2024) reparameterizes pretrained weights with a small number of tunable parameters, showing equivalence to subspace fine-tuning. GainLoRA (Liang et al. 2025) extends a new LoRA branch for each task and integrates branches through gating modules. SAPT (Zhao et al. 2024) aligns PEFT training and selection through a shared attentive learning and selection mechanism. We also include multitask learning (MTL) as an upper bound. Full experimental details are in Appendix C.2.

Implementation Details We adopted two open-source LLMs widely used in prior work: the encoder-decoder T5-large (Raffel et al. 2020) and the decoder-only Llama3 (8B) and Llama3-chat (Touvron et al. 2023; Dubey et al. 2024), the latter incorporating additional instruction tuning. For fairly evaluating performance under the onCIT setting, for methods that do not require task IDs at the inference step (LoRA, L2, GainLoRA, SAPT, and OnEDIT), we excluded all task-descriptive instructions. In this work, we adopted InfLoRA as its update strategy for GainLoRA. For all comparison methods, we followed official implementations and used the hyperparameters reported in their respective papers to ensure consistency with CIT benchmarks. We used the AdamW optimizer (Loshchilov et al. 2017) with $\beta_1 = 0.9$,

Method	Standard CIT Benchmark				Long Sequence Benchmark				SuperNI Benchmark				Task IDs
	AP↑	FR↓	FWT↑	BWT↑	AP↑	FR↓	FWT↑	BWT↑	AP↑	FR↓	FWT↑	BWT↑	
MTL	74.0	-	-	-	45.3	-	-	-	36.9	-	-	-	✓
IncLoRA	49.5	30.4	-17.4	-27.8	18.4	40.3	-52.5	-55.5	18.3	17.1	-6.7	-5.8	
OLoRA	60.1	33.8	-14.6	-25.4	25.2	49.1	-45.5	-45.1	19.8	13.6	-7.8	-6.7	
InfLoRA	52.3	29.7	-16.1	-26.9	26.6	43.8	-49.2	-43.3	20.2	12.7	-6.6	-4.4	
LoRA	48.8	30.2	-15.8	-26.4	17.8	50.5	-59.2	-58.2	18.0	15.6	-10.3	-7.7	-
L2	44.1	34.3	-18.9	-20.8	18.5	52.3	-56.1	-58.5	17.9	17.0	-8.4	-4.8	
GainLoRA	66.4	24.8	-9.5	-20.1	29.8	40.5	-42.9	-39.6	27.6	**12.3**	**-5.4**	-5.6	
SAPT	55.7	29.5	-15.9	-27.6	22.6	45.0	-48.8	-47.3	18.1	16.3	-8.5	-8.2	
OnEDIT	**67.4**	**22.3**	**-6.1**	**-16.9**	**38.4**	**33.1**	**-30.4**	**-30.0**	**28.9**	15.4	-6.4	**-3.5**	

Table 2: Results on the T5-large model across various benchmarks and task sequences. All values are averaged over different task orders within each benchmark for each evaluation metric.

Model	Method	Standard CIT Benchmark			
		AP↑	FR↓	FWT↑	BWT↑
Llama3	LoRA	50.4	29.8	-17.7	-24.8
	OLoRA	62.0	31.8	-12.2	-22.7
	GainLoRA	67.4	25.2	-8.4	-18.0
	SAPT	55.6	28.4	-15.1	-26.3
	OnEDIT	**69.6**	**20.5**	**-7.9**	**-15.4**
Llama3 -chat	LoRA	51.1	28.8	-16.6	-22.2
	OLoRA	62.8	30.1	-13.5	-23.6
	GainLoRA	67.8	25.5	-8.3	-19.6
	SAPT	56.9	27.7	-14.2	-26.5
	OnEDIT	**69.5**	**21.0**	**-7.2**	**-15.1**

Table 3: Results of Llama3 and Llama3-chat models on standard CIT benchmarks. All values are averaged over different task orders within each benchmark for each evaluation metric.

$\beta_2 = 0.999$ and a batch size of 64 and trained models for only one epoch per task. For OnEDIT, we fixed the hyperparameters (a, b, c) to $(0.99, 0.025, 0.15)$ across all experiments, used a learning rate of 0.001, and employed LoRA as the PEFT method for adapter weights.

Results of CIT Benchmarks for LLMs

All reported results are averaged over all task orders within each benchmark. Table 2 summarizes the performance of various CIT methods for LLMs across three benchmarks. OffCIT approaches such as IncLoRA, OLoRA, and InfLoRA consistently outperformed traditional CIT baselines like LoRA and L2, highlighting the advantage of allocating task-specific adapters using task identifiers to retain task-level knowledge and prevent catastrophic forgetting. Among them, OLoRA achieved the best overall performance by enforcing orthogonality between adapters. However, these methods exhibited significant performance degradation when applied to long task sequences or diverse tasks, as in the SuperNI benchmark, indicating that offCIT methods may become unstable and susceptible to forgetting in online settings. Among approaches that do not rely on task IDs, GainLoRA demonstrated relatively strong performance compared to LoRA, L2, and SAPT, which were more vulnerable to instability in online environments. This suggests that GainLoRA effectively balanced the information by accounting for the relative importance of each task-specific LoRA branch. Our proposed method, OnEDIT, achieved the highest average performance across all benchmarks, indicating that it maintains robust performance across varying task lengths and types while preserving historical knowledge and adapting to new tasks. These results suggest that adjusting the information internally through the implicit task vector at each time step is more suitable for onCIT settings than relying on separately allocated adapters for each task.

To observe model-specific performance trends, we evaluated state-of-the-art CIT methods on the standard CIT benchmark using Llama3 and Llama3-chat models, as shown in Table 3. In general, both Llama3 and Llama3-chat exhibited more stable and higher performance among the methods compared to T5-large. Although all methods outperformed the single LoRA baseline, approaches like OLoRA and SAPT still showed performance degradation in online settings, consistent with their behavior in T5-large. Among the methods that do not rely on task IDs, GainLoRA achieved the best performance among existing approaches. In particular, our proposed method, OnEDIT, consistently outperformed all CIT methods on four metrics. This shows that OnEDIT provides robust and consistent performance improvements in onCIT settings, where task IDs are unavailable, and each sample is seen only once, regardless of the model architecture.

Discussion

Analysis of Catastrophic Forgetting

We analyzed the ability to prevent catastrophic forgetting by comparing forward transferability and historical knowledge retention across methods. Figure 3 illustrates the forward transferability for individual tasks (Figure 3(a)) and the retention of knowledge over time (Figures 3(b)–(d)) on the standard CIT benchmark. As shown in Figure 3(a), LoRA and GainLoRA achieved slightly higher or comparable performance to OnEDIT when adapting to each target task. However, Figures 3(b) and 3(c) reveal that both methods experience a rapid decline in performance on previously learned tasks over time, demonstrating susceptibil-

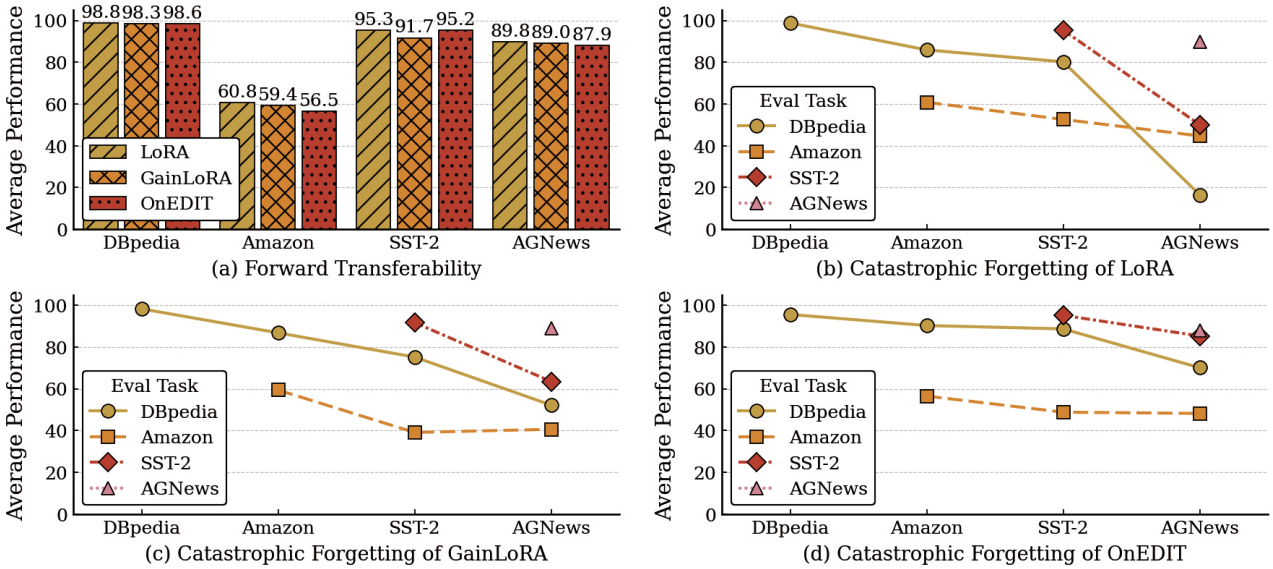

Figure 3: (a) Average performance of each current task after adaptation in Order 1 using the T5-large model. (b)–(d) Average performance of each method along with adaptation time.

Method	Adapter	Forward	Backward	Time (s)
LoRA	1	B	B	39.7
IncLoRA	I	BI	B	45.7
OLoRA	I	BI	$B + Ir^2$	101.6
OnEDIT	2	B	B	42.2

Table 4: Comparison of computation and memory efficiency for each method on Order 1. Adapter is the number of adapters required for I tasks. Forward and Backward indicate the computational load per mini-batch of size B, and Time is the adaptation time for the final task using T5-large.

ity to catastrophic forgetting due to overfitting to the most recent task. In contrast, Figure 3(d) shows that OnEDIT exhibited relatively slight decline during adaptation, indicating effective preservation of prior task performance. These results suggest that implicit task and editing operations play a key role in mitigating forgetting and maintaining long-term knowledge.

Analysis of Computational Efficiency

We compared and analyzed the efficiency of OnEDIT with existing methods. Table 4 summarizes the computational resources required by each approach. IncLoRA and OLoRA required more memory and computation than LoRA, increasing training times. Specifically, IncLoRA maintains multiple adapters, increasing memory usage and computational overhead during the forward pass. OLoRA further introduces an additional loss term to enforce orthogonality among adapters. This orthogonality constraint adds computational complexity proportional to $I \times r^2$ during the backward pass (Wang et al. 2023), where I is the number of tasks and r is the rank of the adapters, resulting in significantly longer training times. In contrast, OnEDIT uses only two adapters, current and implicit, regardless of the number of tasks. Moreover, OnEDIT does not require additional forward or backward passes through the network for each adapter. Instead, it performs simple editing operations on the adapters, such as vector addition and scalar multiplication, which ensure minimal computational overhead.

Conclusion

In this work, we examined the limitations of existing state-of-the-art CIT methods for LLMs under onCIT settings. Although these methods performed well in off-CIT scenarios, they faced significant challenges in real-world environments where task IDs are unavailable and data arrive as a stream. To address this, we focused on onCIT and introduced the concept of an implicit task that balances historical knowledge and novel information without requiring task IDs or access to past data. Based on task vector arithmetic, our proposed method, OnEDIT, mitigates catastrophic forgetting through online editing between the implicit and current task vectors without backpropagation. This enables efficient CIT using only two fixed-size adapters and minimal computation. As a result, OnEDIT consistently outperformed state-of-the-art methods in onCIT settings. These improvements stem from the OnEDIT mechanism that allows LLMs to quickly acquire new information while reducing redundancy within knowledge. A limitation of our approach lies in the requirement for hyperparameter tuning. However, we demonstrated that fixed hyperparameters achieved stable and consistent performance across diverse benchmarks and models. In future work, we aim to implicitly identify these hyperparameters to further improve the applicability and robustness of our method.

Acknowledgments

This work was partly supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2025-00557944) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2025-25443882, Bringing "Her" into the real life: Implementation of the voice AI system enabling sentient conversation).

References

Aljundi, R.; et al. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, 139–154.

Beaulieu, S. L. E.; et al. 2020. Learning to continually learn. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, 992–1001.

Bohao, P.; et al. 2024. Scalable language Model with generalized continual learning. In *Proceedings of the The 12th International Conference on Learning Representations (ICLR)*.

Buzzega, P.; et al. 2020. Dark experience for general continual learning: A strong, simple baseline. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 33: 15920–15930.

Chaudhry, A.; et al. 2019. On tiny episodic memories in continual learning. *arXiv: Learning*.

Chen, W.; et al. 2023. Lifelong language pretraining with distribution-specialized experts. In *Proceedings of the International Conference on Machine Learning (ICML)*, 5383–5395.

Cossu, A.; et al. 2024. Continual pre-training mitigates forgetting in language and vision. *Neural Networks*, 179: 106492.

de Masson d'Autume, C.; et al. 2019. Episodic memory in lifelong language learning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 32.

Dettmers, T.; et al. 2024. Qlora: Efficient finetuning of quantized llms. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 36: 10088–10115.

Dubey, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Geng, B.; et al. 2021. Continual learning for task-oriented dialogue system with iterative network pruning, expanding and masking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 517–523.

Guo, H.; et al. 2023. Stochastic weight averaging revisited. *Applied Sciences*, 13(5): 2935.

He, J.; et al. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Holla, N.; et al. 2020. Meta-learning with sparse experience replay for lifelong language learning. *ArXiv*, abs/2009.04891.

Houlsby, N.; et al. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2790–2799.

Hu, E. J.; et al. 2022. LoRA: Low-Rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ilharco, G.; et al. 2022. Editing models with task arithmetic. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Jang, J.; et al. 2022. Towards continual knowledge learning of language models. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*.

Javed, K.; et al. 2019. Meta-learning representations for continual learning. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 32. Curran Associates, Inc.

Jin, X.; et al. 2022. Lifelong pretraining: Continually adapting language models to emerging corpora. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4764–4780.

Ke, Z.; et al. 2022. Continual training of language models for few-shot learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 10205–10216.

Kirkpatrick, J.; et al. 2017. Overcoming catastrophic forgetting in neural networks. In *Proceedings of the national academy of sciences (PNAS)*, 114(13): 3521–3526.

Li, D.; et al. 2025. Cmt: A memory compression method for continual knowledge learning of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 24413–24421.

Li, G.; et al. 2023. Online noisy continual relation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 13059–13066.

Li, X. L.; and Liang, P. 2021. Prefix-tuning: optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 4582–4597.

Liang, Y.-S.; et al. 2024. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 23638–23647.

Liang, Y.-S.; et al. 2025. Gated integration of low-rank adaptation for continual learning of language models. *ArXiv*, abs/2505.15424.

Lin, C.-Y. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 74–81.

Loshchilov, I.; et al. 2017. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Michieli, U.; et al. 2024. HOP to the next tasks and domains for continual Llearning in NLP. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 14359–14369.

Min, B.; et al. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2): 1–40.

Qin, C.; et al. 2021. LFPT5: A unified framework for lifelong few-shot language learning based on prompt tuning of T5. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Raffel, C.; et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research (JMLR)*, 21(140): 1–67.

Rame, A.; et al. 2022. Diverse weight averaging for out-of-distribution generalization. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 35: 10821–10836.

Rame, A.; et al. 2023. Rewarded soups: Towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 36: 71095–71134.

Razdaibiedina, A.; et al. 2023. Progressive prompts: Continual learning for language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Rebuffi, S.-A.; et al. 2017. ICaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2001–2010.

Rongali, S.; et al. 2020. Continual domain-tuning for pretrained language models. *arXiv preprint arXiv:2004.02288*.

Scialom, T.; et al. 2022. Fine-tuned language models are continual learners. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6107–6122.

Shi, H.; et al. 2024. Continual Learning of Large Language Models: A Comprehensive Survey. *ACM Computing Surveys*.

Smith, J. S.; et al. 2023. Continual diffusion: continual customization of text-to-image diffusion with C-LoRA. *Transactions on Machine Learning Research*.

Song, C.; et al. 2023. ConPET: Continual parameter-efficient tuning for large language models. *ArXiv*, abs/2309.14763.

Sun, F.-K.; et al. 2019. LAMOL: Language modeling for lifelong language learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Touvron, H.; et al. 2023. Llama: open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Wang, X.; et al. 2023. Orthogonal Subspace Learning for Language Model Continual Learning. In *Proceedings of the Association for Computational Linguistics (EMNLP)*, 10658–10671.

Wang, Y.; et al. 2022. Super-naturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 5085–5109.

Wang, Y.; et al. 2024. InsCL: A data-efficient continual learning paradigm for fine-tuning large language models with Instructions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 663–677.

Wang, Z.; et al. 2020. Efficient meta lifelong-learning with limited memory. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 535–548.

Wei, J.; et al. 2021. Finetuned language models are zero-shot learners. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Wortsman, M.; et al. 2022. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the International Conference on Machine Learning (ICML)*, 23965–23998.

Wu, C.; et al. 2024. LLaMA Pro: Progressive LLaMA with block expansion. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 6518–6537.

Yan, Y.; et al. 2023. AF adapter: Continual pretraining for building chinese biomedical language model. In *Proceedings of the IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 953–957.

Yang, Y.; et al. 2024. Recent advances of foundation language models-based continual learning: A Survey. *ACM Computing Surveys*, 57: 1 – 38.

Yao, S.; et al. 2024. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 36: 11809–11822.

Zhang, F.; et al. 2023a. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning. *arXiv preprint arXiv:2308.12043*.

Zhang, Z.; et al. 2023b. CITB: A benchmark for continual instruction tuning. In *Findings of the Association for Computational Linguistics (EMNLP)*, 9443–9455.

Zhao, H.; et al. 2021. Memory-efficient class-incremental learning for image classification. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 33(10): 5966–5977.

Zhao, W.; et al. 2024. SAPT: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, 11641–11661.

Zhao, W. X.; et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zhou, J.; et al. 2023. ChatGPT: Potential, prospects, and limitations. *Frontiers of Information Technology & Electronic Engineering*, 1–6.