

LoKI: Low-damage Knowledge Implanting of Large Language Models

Runyu Wang¹, Peng Ping^{2*}, Zhengyu Guo³, Xiaoye Zhang⁴, Quan Shi², Liting Zhou⁵, Tianbo Ji²

¹School of Information Science and Technology, Nantong University

²School of Transportation and Civil Engineering, Nantong University

³South China University of Technology

⁴China Southern Power Grid Company Limited

⁵Dublin City University

2430310032@stmail.ntu.edu.cn, {pingpeng, sq, jitianbo}@ntu.edu.cn, 202264700027@mail.scut.edu.cn, xiaoyz@whu.edu.cn, liting.zhou@dcu.ie

Abstract

Fine-tuning adapts pretrained models for specific tasks but poses the risk of catastrophic forgetting (CF), where critical knowledge from pretraining is overwritten. To address the issue of CF in a general-purpose framework, we propose **Low-damage Knowledge Implanting (LoKI)**, a parameter-efficient fine-tuning (PEFT) technique that utilizes recent mechanistic understanding of how knowledge is stored in transformer architectures. We compare LoKI against state-of-the-art PEFT methods in two real-world fine-tuning scenarios. The results show that LoKI demonstrates significantly better preservation of general capabilities. At the same time, its task-specific performance is comparable to or even surpasses that of full parameter fine-tuning and these PEFT methods across various model architectures. Our work bridges the mechanistic insights of LLMs’ knowledge storage with practical fine-tuning objectives, enabling an effective balance between task-specific adaptation and the retention of general-purpose capabilities.

Code — <https://github.com/Nexround/LoKI>

Extended version — <https://arxiv.org/abs/2505.22120>

Introduction

Transformer-based language models (Vaswani, Shazeer et al. 2017; Petroni, Rocktäschel et al. 2019) accumulate extensive world knowledge during pretraining (Radford and Narasimhan 2018; Brown, Mann et al. 2020), which becomes implicitly embedded in their parameters. Owing to their broad generalization capabilities, they serve as a strong foundation for downstream task adaptation, prompting significant efforts to develop increasingly efficient fine-tuning methods (Prattasha, Chowdhury et al. 2025; Xin, Yang et al. 2025). However, the process of fine-tuning pretrained models for downstream tasks is often accompanied by catastrophic forgetting (CF) (Kotha, Springer et al. 2024; Luo, Yang et al. 2025), a phenomenon where the model loses previously acquired capabilities after fine-tuning. Recent studies have pointed out that conventional fine-tuning approaches (Prattasha, Chowdhury et al. 2025; Xin, Yang

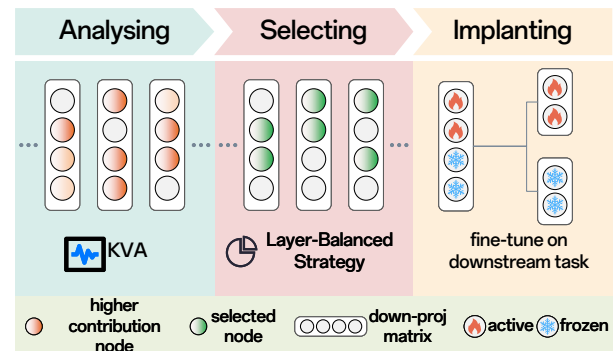


Figure 1: Schematic illustration of the staged fine-tuning process in LoKI.

et al. 2025) typically perform indiscriminate updates across modules within transformer architectures, oblivious to these crucial knowledge-storing weights in general tasks. Such unconstrained optimization may perturb crucial memory traces (Cohen, Geva et al. 2023; Petroni, Rocktäschel et al. 2019), leading to irreversible knowledge loss and degraded generalization performance (Li, Ding et al. 2024; Huang, Cui et al. 2024). Extensive research has investigated the internal structure of large language models (LLMs) (Li, Li et al. 2024; Meng, Sharma et al. 2023a), revealing that model-internal knowledge is both identifiable and editable via targeted interventions such as knowledge localization and rewriting (Geva, Schuster et al. 2021; Meng, Sharma et al. 2023a). Moreover, the inherent redundancy in LLMs has been shown to support techniques like sparsification and parameter pruning (He, Zhou et al. 2022; Frankle and Carbin 2019), enabling more efficient representation without significantly degrading performance (Lasby, Golubeva et al. 2024). However, these findings have yet to be effectively integrated into the field of model fine-tuning. These findings lead us to hypothesize that it is possible to identify low-contributing weights in pretrained models, which can then serve as capacity for injecting new, task-specific knowledge with minimal impact on existing competencies. Based on this assumption, we propose **Low-damage**

*Corresponding author.

Knowledge Implanting, as **LoKI**, a parameter-efficient fine-tuning method that leverages insights from interpretability studies on the internal knowledge storage mechanisms of LLMs. Aiming to mitigate catastrophic forgetting, LoKI consists of three stages: **analyzing**, **selecting**, and **implanting** (see Figure 1). In the **analyzing** stage, we introduce Knowledge Vector Attribution (KVA), a gradient-based attribution technique (Sundararajan, Taly et al. 2017) that evaluates the contribution of each vector in the down-projection matrix W_{down} (see Background section for details) to the model’s pretrained behavior. In the **selecting** stage, motivated by the known interdependence between transformer layers (Sun, Pickett et al. 2025a)—especially in the progressive refinement of knowledge (Geva, Bastings et al. 2023; Dai, Dong et al. 2022a)—we propose the Layer-Balanced Strategy, which ensures that new knowledge aligns with the model’s hierarchical structure. Leveraging KVA results, this strategy enforces an equal quota of trainable parameters per layer by decomposing each W_{down} into two subsets: W_S (active) and $W_{\setminus S}$ (frozen). This yields a layer-balanced trainable subset \mathbb{W}_S . In the **implanting** stage, we freeze all model parameters except \mathbb{W}_S , which is updated via fine-tuning. Compared to existing fine-tuning approaches, LoKI offers several major advantages.

- **Superior balance between CF and task-specific performance.** Fine-tuning LLMs of two representative sizes with LoKI effectively mitigates CF, outperforming state-of-the-art PEFT methods while maintaining strong task performance.
- **Intrinsically parameter-efficient.** LoKI updates only a subset of the model’s original parameters and allows for explicit control over the number of trainable weights.
- **Synergistic with other tuning methods.** In addition to adapting directly for downstream tasks, we demonstrate that LoKI can be combined with existing parameter-efficient tuning techniques such as Low-Rank Adaptation (LoRA) (Hu, Shen et al. 2022), further reducing the number of trainable parameters.

By allocating updates to carefully selected weights, LoKI provides a competitive approach to sustainable LLM customization, enabling models to evolve while preserving their core capabilities. In summary, our main contributions are:

- We introduce LoKI, a fine-tuning framework for LLMs that primarily aims to mitigate CF during adaptation to downstream tasks.
- We propose KVA, a technique to quantify the contribution of individual knowledge vectors (defined in the Background section) to the model’s stored representations. Furthermore, using this method, we uncover a surprising phenomenon: in transformer models, knowledge vectors with both globally high and low contributions tend to be densely located in similar layers.
- Building on an understanding of the hierarchical organization of knowledge in transformers, we develop the Layer-Balanced Strategy for allocating trainable weights. Our experiments show that this strategy is critical to LoKI’s ability to preserve pre-trained capabilities while learning new tasks.

Related Works

Catastrophic Forgetting CF has long been recognized as a critical challenge in neural networks (French 1993; Kemker, McClure et al. 2018a), and recent studies have begun to analyze its manifestation specifically in LLMs (Li, Ding et al. 2024; Kotha, Springer et al. 2024). Representative methods include: Replay-based methods (de Masson d’Autume, Ruder et al. 2019; Lopez-Paz and Ranzato 2017; Chaudhry, Ranzato et al. 2019) interleave data from previous tasks during training to reinforce prior knowledge. Regularization-based methods (Li and Hoiem 2018; Kirkpatrick, Pascanu et al. 2016; DENG, Chen et al. 2021) constrain parameter updates to prevent drift from previously important weights. However, most of the previous methods generally treat knowledge retention as a black box, rarely considering how pretrained LLMs organize and store knowledge internally.

Parameter-Efficient Fine-Tuning Recent advances in PEFT aim to mitigate CF by preserving pretrained knowledge while enabling task adaptation. For example, Zhu, Sun et al. refines a small subset of critical parameters to retain performance on original tasks. However, it requires task-specific parameter selection. Some methods structurally decouple knowledge to resist CF. CorDA (Yang, Li et al. 2024) freezes dominant singular directions, assumed to encode general knowledge, while adapting residual subspaces. LoRASculpt (Liang, Huang et al. 2025) uses magnitude-based masking and conflict-aware regularization to constrain LoRA (Hu, Shen et al. 2022) updates within critical knowledge regions. Orthogonal subspace approaches like O-LoRA (Wang, Chen et al. 2023) and LoRI (Zhang, You et al. 2025) reduce task interference by isolating updates in mutually orthogonal directions, effective in multi-task scenarios but less focused on preserving original capabilities. In contrast, LoKI proposes a one-time knowledge localization strategy, directly quantifying parameter contributions to general knowledge, enabling scalable and effective CF resistance across diverse downstream tasks.

Knowledge Locating and Editing ROME (Meng, Bau et al. 2022) revises individual factual associations by directly modifying the FFN weight vectors. KN (Dai, Dong et al. 2022a) regulates the expression of specific facts by controlling the activation levels of identified neurons. AlphaEdit (Fang, Jiang et al. 2025) further advances this line by applying zero-space projection to the output weight matrix of FFN layers to perform targeted edits. However, despite their effectiveness in editing specific knowledge expressions, these methods have yet to be seamlessly integrated into parameter-efficient fine-tuning pipelines that aim to retain the model’s broad pretrained capabilities. Moreover, compared to existing knowledge localization methods, KVA and the Layer-Balanced Strategy constitute a complete model analysis pipeline in practice. This combination transcends the limitation of focusing solely on specific factual expressions, evolving the underlying ideas into a general-purpose approach for model analysis.

Background

We will begin this section by briefly reviewing transformer architecture and relevant research on knowledge storage in transformer-based language models, which will establish a foundation for our proposed method. Transformer-based language models are built from stacked layers, each consisting of two primary components: a multi-head self-attention (MHSA) mechanism (Vaswani, Shazeer et al. 2017) and a position-wise feed-forward network (FFN). While some studies have explored the role of MHSA in transformers (Voita, Talbot et al. 2019; Clark, Khandelwal et al. 2019; Vig and Belinkov 2019), a growing number of studies focus on the FFN layers for knowledge localization and editing (Geva, Schuster et al. 2021; Geva, Caciularu et al. 2022; Katz, Belinkov et al. 2024). For an input vector $x \in \mathbb{R}^{d_{\text{model}}}$, a typical FFN without bias applies two linear transformations with a non-linearity in between:

$$\text{FFN}(x) = W_{\text{down}} \sigma(W_{\text{up}} x) \quad (1)$$

where $W_{\text{up}} \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$ is the up-projection matrix, $W_{\text{down}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ is the down-projection matrix, and $\sigma(\cdot)$ denotes an element-wise activation function. Recent studies suggest that FFN layers function as linear associative memories, where the W_{up} acts as a collection of keys detecting input patterns, and the W_{down} contains values corresponding to interpretable concepts (Geva, Schuster et al. 2021; Geva, Caciularu et al. 2022). This view has been reinforced by both activation- and gradient-based analyses (Katz, Belinkov et al. 2024). Specifically, each output coordinate y_j in the FFN can be viewed as a knowledge output node, computed as $y_j = v_j^\top \sigma(W_{\text{up}} x)$, where v_j is the j -th row of W_{down} and serves as a knowledge vector. On the other hand, while single-layer FFN reveals this memory behavior, deeper inspection shows a hierarchy: lower FFN layers capture surface-level patterns, whereas upper layers encode higher-level semantics (Geva, Bastings et al. 2023; Dai, Dong et al. 2022a; Tan, Zhang et al. 2024; Katz, Belinkov et al. 2024). The final output distribution of transformer-based language models is gradually constructed in a bottom-up fashion (Sun, Pickett et al. 2025a; Tenney, Das et al. 2019; Wallat, Singh et al. 2020). Such insights have inspired a wave of model editing techniques that target the FFN layers—particularly W_{down} —to insert, update, or erase factual knowledge without extensive retraining (Geva, Caciularu et al. 2022; Dai, Dong et al. 2022b; Meng, Sharma et al. 2023b). Motivated by the success of these approaches, our study focuses on implanting task-specific knowledge through FFNs, which we consider the main site for implanting new knowledge into the model. Importantly, prior research reveals that LLMs exhibit considerable parameter redundancy, especially within FFNs (Kobayashi, Kuribayashi et al. 2024; Sanh, Wolf et al. 2020a; Zhang, Lin et al. 2022; Sanh, Wolf et al. 2020b; Frantar, Alistarh et al. 2023). Numerous studies show that substantial portions of model weights can be pruned or restructured with minimal performance loss on general tasks (Frankle and Carbin 2019; Michel, Levy et al. 2019; Sanh, Wolf et al. 2020b; Frantar, Alistarh et al. 2023). Based on the previous research

mentioned above, we hypothesize that these low-impact parameters in FFN layers can be repurposed to encode new task-specific knowledge without excessively degrading the model’s original capabilities. Our experimental results provide evidence supporting this hypothesis.

Low-damage Knowledge Implanting

We introduce LoKI, a three-stage framework—**analyzing**, **selecting**, and **implanting**—designed to edit transformer models with minimal disruption, thereby mitigating CF (see Figure 1). Below, we briefly outline each stage before providing a detailed explanation:

1. **Analyzing**: Evaluate the contribution of each knowledge vector to general tasks.
2. **Selecting**: Choose trainable knowledge vectors within each FFN based on the analysis results.
3. **Implanting**: Train the chosen vectors to incorporate task-specific knowledge.

Analysing

We start this subsection by introducing **Knowledge Vector Attribution (KVA)**, an attribution method inspired by (Hao, Dong et al. 2021; Dai, Dong et al. 2022a), designed to evaluate the contribution of individual knowledge vectors to the model’s performance. Next, we illustrate the workflow of the analysis stage—in other words, how KVA is utilized within the LoKI framework.

Knowledge Vector Attribution KVA is a computational approach based on Integrated Gradients (IG) (Sundararajan, Taly et al. 2017) that measures the contribution of knowledge vectors to specific output logits.

$$\text{IG}_i(x) = (x_i - x'_i) \int_0^1 \frac{\partial F(x' + \alpha(x - x'))}{\partial x_i} d\alpha, \quad (2)$$

where F is the network function, x' is a baseline (we use $x' = \mathbf{0}$), and $\alpha \in [0, 1]$ is an interpolation coefficient that defines the integration path from baseline to input. IG attributes predictions to input features by integrating gradients along the straight-line path from baseline x' to input x . To trace knowledge flow through all layers of a transformer on input sequence \mathbf{x} , let \mathcal{L} denote the target logit, h_{l-1} represent the hidden state from the previous layer. At layer l , the FFN first computes pre-activations

$$z_l = W_{\text{up}}^{(l)} h_{l-1}, \quad \mathbf{u}_l = \sigma(z_l), \quad \mathbf{z}_{l,j} = \mathbf{u}_{l,j} W_{\text{down},j}^{(l)}. \quad (3)$$

As mentioned above, we treat FFNs as collections of key-value memory pairs. To attribute the contribution of each knowledge vector to the final output of an LLM, we define the layer-wise, path-integrated attribution of node j by

$$\text{Attr}_{l,j}(\mathbf{x}) = \int_0^1 \frac{\partial \mathcal{L}(\alpha \mathbf{z}_{l,j})}{\partial \mathbf{z}_{l,j}} d\alpha, \quad (4)$$

which aggregates the total gradient flowing through knowledge vector j as its contribution is scaled from zero up to its actual activation. We approximate the above integral via

Riemann approximation with m equally-spaced steps (we set the step $m = 7$ for any model, and further discussion on the setting of m can be found in Appendix A):

$$\text{Attr}_{l,j}(\mathbf{x}) \approx \frac{1}{m} \sum_{k=1}^m \frac{\partial \mathcal{L}(\frac{k}{m} \mathbf{z}_{l,j})}{\partial \mathbf{z}_{l,j}}. \quad (5)$$

During each sample, KVA computes and stores $\text{Attr}_{l,j}(\mathbf{x})$ for every layer l and every knowledge output node j . This yields a complete attribution log that can be analyzed post hoc to assess the contribution of each knowledge vector during inference.

Scoring Knowledge Vectors Pretrained LLMs exhibit a wide range of capabilities, including world knowledge, common-sense reasoning, and instruction following. While it is difficult to precisely determine the full scope of an LLM’s internal knowledge, it can be approximated through extensive textual input (Meng, Sharma et al. 2023a). To evaluate the contribution of individual knowledge vectors to general-purpose performance—and to identify those with minimal impact on core capabilities—a critical factor is the choice of dataset for quantitative analysis. To ensure broad domain coverage, we apply KVA to the Massive Multitask Language Understanding (MMLU) benchmark (Kemker, McClure et al. 2018b). MMLU is ideal because:

- it has 57 diverse subjects (STEM, humanities, professional, etc.), which covers a broad spectrum of general-knowledge tasks, ensuring the evaluation score of vectors is not domain-specialized;
- its standardized prompts and evaluation make attributions comparable across tasks;
- high performance on MMLU correlates with real-world language understanding, so protecting original performance is measurable.

This process, executed on an RTX4090 GPU using Llama3.1-8B-Instruct (Dubey et al. 2024), takes an average of 9.69 seconds per sample. In parallel, we apply KVA to the full MMLU dataset. Results show that the final selected nodes—based on the Layer-Balanced Strategy (detailed later)—overlap by **97.57%** between the reduced set and the full dataset. This confirms that our sampling strategy substantially reduces computational cost while preserving high fidelity in attribution outcomes. Notably, KVA needs to be performed only once per model, independent of downstream tasks. That is, this computational overhead is incurred a single time per model. Implementation details and additional discussion are provided in Appendix A.

Selecting

As shown in Figure 2, the evaluation results of KVA reveal significant heterogeneity in the distribution of knowledge output nodes across different layers, regardless of their contribution level. This uneven distribution phenomenon among nodes with high contributions aligns with the findings of Meng et al. (2022; 2023a) and Dai et al. (2022b). Notably, our results uncover a novel phenomenon: both high-

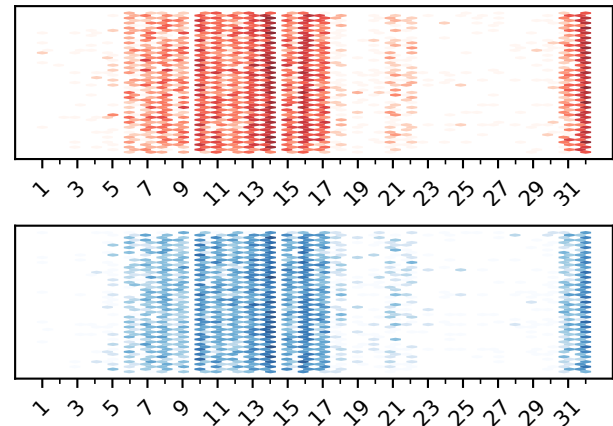


Figure 2: Heatmaps of the top 5% KVA results across all 32 layers of Llama3.1-8B-Instruct. The vertical axis denotes node indices, and the horizontal axis denotes layer indices. The upper (red-tinted) heatmap illustrates the distribution of high-contribution node positions, while the lower (blue-tinted) heatmap illustrates the distribution of low-contribution node positions. Color intensity (log-scale) reflects the density of nodes within each category, with darker colors indicating higher density. Heatmaps for additional models are provided in Appendix B.

and low-contribution nodes are densely concentrated in the same layers. This suggests a potential structural relationship in the placement of knowledge vectors across different attribution levels. We believe this observation may offer new insights into inter-layer knowledge organization in transformer architectures. As our current focus is on leveraging the analysis results for guiding fine-tuning, we leave a deeper investigation of this phenomenon to future work. These findings shed a critical light on fine-tuning strategies: the layer-wise distribution of knowledge is not random but structurally biased. Consequently, naïve parameter-efficient approaches that update only a limited number of layers may inadvertently disrupt the intrinsic knowledge hierarchy of transformers (Sun, Pickett et al. 2025b; Geva, Caciularu et al. 2022). Our experiments show that such imbalanced parameter allocation exacerbates catastrophic forgetting (see the Ablation Studies section). Furthermore, Hase et al. (2023) have found a significant challenge: even when factual knowledge resides in specific layers, such as the mid-layer FFNs, modifying weights in distant layers, especially the earlier ones, can effectively “override” the information flow downstream.

Layer-Balanced Strategy Based on the previous considerations, we propose the Layer-Balanced Strategy, which is a strategy that determines the trainable parameter positions under the constraint of allocating the same number of parameters to each layer. This strategy aims to (a) ensure that newly implanted knowledge conforms to the hierarchical relationship of the transformer model’s knowledge structure, and (b) avoid disproportionately disturbing its inherent knowledge hierarchy. Let the model have L layers, where

only parameters within each layer’s W_{down} matrix are partially trainable, while all other parameters across all layers remain frozen. Each W_{down} in layer l contains D_l knowledge output nodes (with $D_l = D$ assumed identical across layers for simplicity). The hyperparameter $q \in (0, 100)$ governs the percentage of trainable nodes selected from all W_{down} matrices. Given N inference samples, we denote the KVA result for node i in layer l on sample t as:

$$\text{Attr}_{l,i}^{(t)}, \quad i = 1, \dots, D, \quad t = 1, \dots, N$$

The implementation proceeds as follows:

1. **Quota Allocation:** Calculate the total trainable slots:

$$T = \frac{q}{100} \cdot \sum_{l=1}^L D_l = \frac{q}{100} \cdot LD \quad (6)$$

Allocate equal quotas per layer: $k_l = \lfloor \frac{T}{L} \rfloor$, $l = 1, \dots, L$.

2. **Per-Sample Local Selection:** For each sample t and layer l , select k_l nodes with the smallest values:

$$S_l^{(t)} = \text{argsort} \downarrow \left(\hat{\text{Attr}}_{l,i}^{(t)} \right) [1 : k_l] \quad (7)$$

3. **Frequency Aggregation:** Tally selection frequencies across samples:

$$c_{l,i} = \sum_{t=1}^N \mathbb{I}(i \in S_l^{(t)}) \quad (8)$$

4. **Final Selection:** For each layer l , select nodes with the highest frequencies:

$$S_l = \text{argsort} \uparrow (c_{l,i}) [1 : k_l], \quad \mathcal{S} = \bigcup_{l=1}^L S_l \quad (9)$$

Resulting in a balanced set \mathcal{S} with $\sum_{l=1}^L |S_l| \leq T$.

To empirically assess the effectiveness of the Layer-Balanced Strategy, we perform an ablation study comparing it with imbalanced baselines that naïvely select knowledge vectors with the highest or lowest global attribution scores as trainable parameters. Experimental details and results are provided in the Ablation Studies section.

Implanting

Building upon the selected knowledge output nodes $\mathcal{S} = \bigcup_{l=1}^L S_l$, we implement parameter-efficient fine-tuning through strategic decomposition of FFN layers. For each layer l ’s down-projection matrix $W_{\text{down}}^{(l)}$, we partition the parameters into two complementary subspaces:

$$W_{\text{down}}^{(l)} = \begin{bmatrix} W_{S_l} \\ W_{\setminus S_l} \end{bmatrix}, \quad (10)$$

where $W_{S_l} \in \mathbb{R}^{|S_l| \times d_{\text{ff}}}$ contains the weights corresponding to our selected low-contribution knowledge output nodes (from layer l ’s quota S_l), and $W_{\setminus S_l}$ represents the remaining parameters. During training, we:

- Keep $W_{\setminus S_l}$ **frozen** to preserve existing knowledge representations
- Update only W_{S_l} **actively** to implant new knowledge

This decomposition retains the mathematical formulation of the original layer while restricting parameter updates to the chosen subspaces. In addition, it transforms LoKI training into a module-wise process, allowing easy integration with existing training pipelines. For clarity, we refer to this implementation as **LoKI Linear**, and the readers can find its PyTorch code in Appendix C.

Incorporating LoRA It is worth noting that this implementation allows for the incorporation of low-rank decomposition techniques (Hu, Shen et al. 2022; Liu, Wang et al. 2024; Zhang, Chen et al. 2023) into LoKI. Specifically, the trainable weights are parameterized as:

$$W_{S_l} = W_{S_l}^{(0)} + \Delta W_{S_l}, \quad \Delta W_{S_l} = A_l B_l, \quad (11)$$

where $W_{S_l}^{(0)}$ denotes the frozen base weights, and $A_l \in \mathbb{R}^{|S_l| \times r}$, $B_l \in \mathbb{R}^{r \times d_{\text{ff}}}$ are learnable low-rank matrices with rank $r \ll \min(|S_l|, d_{\text{ff}})$. To verify the feasibility of this integration, we experimented with the Experiment section.

Experiments

To evaluate our proposed method, we focus on adapting models to the following two datasets:

1. **ToolACE Function-Calling Dataset** (Liu, Huang et al. 2024): This dataset contains 26,507 distinct APIs across various domains, designed to enhance the function-calling capabilities of LLMs. Consistent with the official model released by the dataset provider, we fine-tuned Llama3.1-8B-Instruct on this dataset. The details of the experiment setup can be found in Appendix D.
2. **LB Reranker Dataset** (Lightblue 2024): This multilingual dataset consists of 2.28 million query-text pairs annotated with fine-grained relevance scores on a 1-7 scale, designed for training NLP-based retrieval models. It has an official full parameter fine-tuning model based on Qwen2.5-0.5B-Instruct (Qwen, Yang et al. 2025); we utilized the same base model on this dataset. The experiment setup for this task can be found in Appendix E.

These publicly available datasets were specifically selected due to their demonstrated practical utility in real-world applications and their ability to simulate realistic fine-tuning demand scenarios. Each experiment assesses the capability of LoKI to resist the CF phenomenon while acquiring task-specific performance. Notably, while our experiments involved only two model types, the proposed method is readily applicable to other model architectures.

Measuring Catastrophic Forgetting To systematically evaluate CF during fine-tuning, we assess the model’s retained general capabilities across six diverse benchmarks: TriviaQA (Joshi, Choi et al. 2017) (world knowledge), GSM8K (Cobbe et al. 2021) (mathematical reasoning), HellaSwag (Zellers, Holtzman et al. 2019) and WinoGrande (Sakaguchi, Bras et al. 2020) (commonsense

Model	Overall Acc (%, \uparrow)	Single Turn Acc		Multi Turn (%, \uparrow)	Hallucination Measurement	
		Non-Live(%, \uparrow)	Live(%, \uparrow)		Relevance(%, \uparrow)	Irrelevance(%, \downarrow)
ToolACE \dagger ($r=16$)	58.32	87.56	76.10	7.62	83.33	88.05
DoRA($r=16$)	58.90	82.04	75.61	16.00	83.33	88.92
PiSSA($r=16$)	53.97	80.56	72.68	5.62	61.11	92.19
LoKI($q=10$)	56.76	80.81	70.46	<u>16.50</u>	83.33	82.21
LoKI($q=20$)	58.16	81.02	73.26	17.75	83.33	85.73
LoKI($q=30$)	58.93	82.71	75.70	16.25	83.33	87.65
LoKI*($q=30, r=32$)	57.16	<u>81.96</u>	<u>71.66</u>	15.75	83.33	<u>84.01</u>

Table 1: Performance comparison on the Berkeley Function Calling Leaderboard V3. ToolACE refers to the official model trained using LoRA. r denotes the rank. Models marked with \dagger indicate that the corresponding performance metrics are sourced directly from the official BFCL documentation. An asterisk (*) denotes models where LoRA was applied during training. **Bold** represents the top performance score in each column; underline represents the runner-up.

Model	#Params	TriviaQA	GSM8K	Hellaswag	WinoGrande	HumanEval	IFEval	Avg(\downarrow)
Llama3.1 (untuned)	–	65.77	84.46	73.85	62.98	68.29	79.76	–
ToolACE($r=16$)	42M	64.63	79.53	46.81	42.78	60.37	72.76	16.11%
DoRA($r=16$)	43M	63.97	82.64	70.78	57.85	65.24	73.47	4.92%
PiSSA($r=16$)	42M	52.50	51.40	37.98	13.18	15.85	55.95	48.93%
LoKI($q=10$)	188M	65.85	84.99	75.14	61.33	68.90	77.54	0.34%
LoKI($q=20$)	376M	65.60	84.31	73.60	60.62	67.68	78.18	0.93%
LoKI($q=30$)	563M	65.49	84.61	71.16	61.09	70.73	77.94	1.23%
LoKI*($q=30, r=32$)	16M	64.30	83.47	70.78	62.51	70.12	78.21	1.26%

Table 2: Benchmark scores of models fine-tuned on ToolACE Function-Calling Dataset. Llama3.1 denotes Llama3.1-8B-Instruct. Avg denotes the average performance degradation percentage of each indicator compared to the original model indicator. We complete all the benchmarks on OpenCompass and report their results directly.

Model	MAP@1 (%)	Recall@1 (%)	NDCG@1 (%)	P@1 (%)
Qwen2.5	-78.1	-78.1	-78.7	-77.5
DoRA($r=8$)	-8.7	-8.7	-11.2	-10.4
PiSSA($r=8$)	-9.6	-9.6	-9.7	-10.2
CorDA($r=8$)	-3.3	-3.1	-2.6	-3.3
LoKI($q=5$)	-3.1	-3.1	-2.2	-1.8
LoKI($q=10$)	-2.3	-2.3	-0.3	-0.5
LoKI($q=20$)	<u>+0.2</u>	<u>+0.2</u>	<u>+0.7</u>	<u>+0.8</u>
LoKI($q=30$)	+1.0	+1.0	+2.1	+2.5

Table 3: Retrieval performance on BEIR benchmark. Results show the percentage difference relative to the full parameter fine-tuning model across standard retrieval metrics, where positive values indicate superior performance. CorDA uses the KPM setting on the NQ-Open dataset.

understanding), HumanEval (Chen et al. 2021) (code generation), and IFEval (Zhou, Lu et al. 2023) (instruction following).

We define the average forgetting score across all benchmarks as: $Avg = \frac{100}{N} \sum_{i=1}^N \frac{S_i^o - S_i^t}{S_i^o}$, where S_i^o and S_i^t denote the performance of the original and fine-tuned models on the i -th benchmark, respectively, and N is the total number of benchmarks. A higher value of Avg indicates a greater degree of forgetting. Further details regarding the evaluation of these benchmarks can be found in Appendix F.

Task 1: ToolACE Function-Calling Dataset Based on the training results on this dataset, we first compare LoKI with existing SOTA methods in terms of downstream task

adaptation. We evaluate fine-tuned models’ performance on the Berkeley Function Calling Leaderboard V3, comparing LoKI against LoRA (model provided by the dataset authors), DoRA (Liu, Wang et al. 2024), and PiSSA (Meng, Wang et al. 2024). As shown in Table 1, LoKI achieves the highest overall accuracy when $q=30$. Under the $q=20$ setting, LoKI exhibits the strongest multi-turn reasoning capability, with a success rate of **17.75%**. Notably, all LoKI variants consistently reduce the Irrelevance metric, suggesting that LoKI may help mitigate hallucination effects introduced during fine-tuning. Table 2 further reveals LoKI’s unique capability in preserving foundational model competencies. Compared to DoRA, LoKI ($q=30$) not only outperforms in fine-tuned performance, but also reduces the average performance degradation percentage by **75%** across six evaluation benchmarks. Relative to the original pretrained model, the degradation is as small as **1.23%**. Notably, as the hyperparameter q increases from 10 to 30—despite a substantial growth in the number of trainable parameters—the rate of performance degradation slows down. Additionally, we experimented to investigate the potential of combining LoRA with LoKI (LoKI*($q=30$)). As anticipated, the number of trainable parameters in the model significantly decreased when integrated with LoRA, showing a reduction of 97.16% compared to LoKI($q=30$). Importantly, this combination did not visibly compromise LoKI’s ability to resist catastrophic forgetting. We believe that this combination holds significant promise with further optimization of training settings.

Task 2: LB Reranker Dataset The progressive performance improvement with increasing trainable parameters

Model	#Params	TriviaQA	GSM8K	HellaSwag	WinoGrande	HumanEval	IFEval	Avg(↓)
Qwen2.5 (untuned)	–	24.37	39.65	30.98	44.20	26.83	33.39	–
LB-Reranker	494M	4.65	2.65	0.00	0.00	0.00	23.20	84.13%
DoRA($r=8$)	4M	19.30	34.01	22.21	44.70	25.22	31.67	12.23%
PiSSA($r=8$)	4M	11.32	6.44	18.54	31.97	6.71	28.85	48.95%
CorDA($r=8$)	4M	4.64	2.43	0.00	0.00	0.00	23.20	84.22%
LoKI($q=5$)	5.1M	20.53	38.36	30.52	40.09	24.39	34.12	6.12%
LoKI($q=10$)	10.4M	20.53	37.98	21.53	47.04	24.39	33.39	8.86%
LoKI($q=20$)	20.9M	19.77	36.77	33.16	47.28	27.44	33.27	1.70%
LoKI($q=30$)	31.3M	20.10	37.60	38.39	43.96	26.83	32.25	0.46%

Table 4: Benchmark scores of models fine-tuned on the LB Reranker Dataset. LB-Reranker refers to the full parameter fine-tuning model released by the dataset provider. Qwen2.5 represents the Qwen2.5-0.5B-Instruct.

(a)		(b)	
Model	Avg(↓)	Model	Avg(↓)
S-H	36.73%	LoKI	8.86%
S-L	11.35%	G-H	39.04%
		G-L	30.48%

Table 5: (a) Performance of two suppression strategies on benchmarks when $q=1$. (b) Performance comparison of different methods when $q=10$. Detailed results are provided in Appendix G.

(from $q=5$ to $q=30$) reveals a clear parameter-performance tradeoff. Even with only $q=20$, LoKI already achieves positive performance gains in four metrics. As shown in Table 4, compared to all baseline methods including DoRA, PiSSA, and CorDA (Yang, Li et al. 2024), LoKI models of all configurations exhibit significantly less performance degradation across all metrics. Interestingly, when $q=30$, LoKI not only achieved the best averaged performance on the BEIR benchmark, but also showed the least degradation on general tasks. We attribute this phenomenon to our empirical learning rate schedule: higher q values used proportionally lower learning rates (e.g., the learning rate for $q=30$ was $0.4\times$ that for $q=10$), which appears to provide a better balance between task adaptation and knowledge preservation. Further exploration of this hyperparameter interplay is discussed in Appendix E.

Ablation Studies

In this section, we evaluate (a) the effect of KVA in quantifying knowledge vectors’ contributions to general task performance, and (b) the necessity of the Layer-Balanced Strategy in LoKI.

Validation of KVA We use Llama3.1-8B-Instruct to validate if KVA can effectively attribute the contribution of knowledge vectors to general tasks. We compare two suppression strategies (i.e., zeroing the corresponding weights) with the base model using the same benchmarks in the Experiments section:

- **S-H/L**: Suppresses the top $q\%$ knowledge vectors that appear most frequently with the **highest/lowest** attribution scores (expected to **significantly/minimally** degrade performance if KVA correctly identifies critical vectors).

As presented in Table 5a, when the top 1% of high-contribution vectors are suppressed, there is a significant performance decline, with an average degradation of 36.73%. In contrast, the performance gap between the two suppression strategies is **25.38%**, which indicates significant differences and supports the accuracy of KVA.

Validation of Layer-Balanced Strategy Next, we test the necessity of the Layer-Balanced Strategy by fine-tuning Qwen2.5-0.5B-Instruct on the LB Rerank Dataset, using two globally imbalanced schemes. Equally, we compare these results with models trained using LoKI on the 6 benchmarks mentioned above.

- **G-H/L**: Globally set the top $q\%$ knowledge vectors across all layers that appear most frequently with the **highest/lowest** attribution scores as trainable parameters.

As shown in Table 5b, both imbalanced strategies fall short of LoKI, despite using the same quota, underscoring the importance of the Layer-Balanced Strategy. Additional BEIR benchmark results are included in Appendix G.

Conclusions

We present LoKI, a parameter-efficient fine-tuning framework that achieves balanced adaptation between downstream task performance and preservation of pre-trained knowledge in large language models. Our key insight stems from systematically analyzing the hierarchical knowledge storage mechanism in transformers and developing a layer-balanced parameter selection strategy guided by integrated gradient attribution. Through experiments on retrieval and tool-use tasks, we demonstrate that LoKI achieves competitive task adaptation while significantly reducing catastrophic forgetting compared to full parameter fine-tuning and prevalent PEFT methods. Our experimental results demonstrate that integrating insights from mechanistic interpretability research with fine-tuning objectives is effective, highlighting the potential of this interdisciplinary direction.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants 52202496, 52442218, and U2433216; The Key Research and Development Project of Nantong City, China (Special Project for Prospective Technology Innovation, No. GZ2024001); and

the Key Laboratory of Target Cognition and Application Technology (2023-CXPT-LC-005).

References

- Brown, T.; Mann, B.; et al. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, 1877–1901.
- Chaudhry, A.; Ranzato, M.; et al. 2019. Efficient Lifelong Learning with A-GEM. In *ICLR 2024*.
- Chen, M.; Tworek, J.; Jun, H.; et al. 2021. Evaluating Large Language Models Trained on Code.
- Clark, K.; Khandelwal, U.; et al. 2019. What Does BERT Look at? An Analysis of BERT’s Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@ACL 2019, Florence, Italy, August 1, 2019*, 276–286.
- Cobbe, K.; Kosaraju, V.; Bavarian, M.; et al. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.
- Cohen, R.; Geva, M.; et al. 2023. Crawling The Internal Knowledge-Base of Language Models. In *Findings of the Association for Computational Linguistics: EAACL 2023*, 1856–1869. Dubrovnik, Croatia.
- Dai, D.; Dong, L.; et al. 2022a. Knowledge Neurons in Pre-trained Transformers. In *ACL*, 8493–8502.
- Dai, D.; Dong, L.; et al. 2022b. Knowledge Neurons in Pre-trained Transformers. In *ACL 2022*, 8493–8502. Dublin, Ireland.
- de Masson d’Autume, C.; Ruder, S.; et al. 2019. Episodic Memory in Lifelong Language Learning. In *NeurIPS 2019*, 13122–13131.
- DENG, D.; Chen, G.; et al. 2021. Flattening Sharpness for Dynamic Gradient Projection Memory Benefits Continual Learning. In *Advances in Neural Information Processing Systems*, volume 34, 18710–18721.
- Dubey, A.; et al. 2024. The Llama 3 Herd of Models. *CoRR*, abs/2407.21783.
- Fang, J.; Jiang, H.; et al. 2025. AlphaEdit: Null-Space Constrained Knowledge Editing for Language Models. In *ICLR 2024*.
- Frankle, J.; and Carbin, M. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *ICLR 2024*.
- Frantar, E.; Alistarh, D.; et al. 2023. SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot. In *ICML 2024*, volume 202 of *Proceedings of Machine Learning Research*, 10323–10337.
- French, R. M. 1993. Catastrophic Interference in Connectionist Networks: Can It Be Predicted, Can It Be Prevented? In *NeurIPS*, 1176–1177.
- Geva, M.; Bastings, J.; et al. 2023. Dissecting Recall of Factual Associations in Auto-Regressive Language Models. In *EMNLP 2023*, 12216–12235. Singapore.
- Geva, M.; Caciularu, A.; et al. 2022. Transformer Feed-Forward Layers Build Predictions by Promoting Concepts in the Vocabulary Space. In *EMNLP 2022*, 30–45. Abu Dhabi, United Arab Emirates.
- Geva, M.; Schuster, R.; et al. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *EMNLP 2021*, 5484–5495. Online and Punta Cana, Dominican Republic.
- Hao, Y.; Dong, L.; et al. 2021. Self-Attention Attribution: Interpreting Information Interactions Inside Transformer. In *AAAI 2021*, 12963–12971.
- Hase, P.; Bansal, M.; et al. 2023. Does Localization Inform Editing? Surprising Differences in Causality-Based Localization vs. Knowledge Editing in Language Models. In *NeurIPS 2023*.
- He, J.; Zhou, C.; et al. 2022. Towards a Unified View of Parameter-Efficient Transfer Learning. In *ICLR 2024*.
- Hu, E. J.; Shen, Y.; et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2): 3.
- Huang, J.; Cui, L.; et al. 2024. Mitigating Catastrophic Forgetting in Large Language Models with Self-Synthesized Rehearsal. In *ACL*, 1416–1428.
- Joshi, M.; Choi, E.; et al. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL*, 1601–1611.
- Katz, S.; Belinkov, Y.; et al. 2024. Backward Lens: Projecting Language Model Gradients into the Vocabulary Space. In *EMNLP 2024*, 2390–2422. Miami, Florida, USA.
- Kemker, R.; McClure, M.; et al. 2018a. Measuring catastrophic forgetting in neural networks. In *AAAI*, 1.
- Kemker, R.; McClure, M.; et al. 2018b. Measuring Catastrophic Forgetting in Neural Networks. In *AAAI 2018*, 3390–3398.
- Kirkpatrick, J.; Pascanu, R.; et al. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796.
- Kobayashi, G.; Kuribayashi, T.; et al. 2024. Analyzing Feed-Forward Blocks in Transformers through the Lens of Attention Maps. In *ICLR 2024*.
- Kotha, S.; Springer, J. M.; et al. 2024. Understanding Catastrophic Forgetting in Language Models via Implicit Inference. In *ICLR 2024*.
- Lasby, M.; Golubeva, A.; et al. 2024. Dynamic Sparse Training with Structured Sparsity. In *ICLR 2024*.
- Li, H.; Ding, L.; et al. 2024. Revisiting Catastrophic Forgetting in Large Language Model Tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 4297–4308. Miami, Florida, USA.
- Li, X.; Li, S.; et al. 2024. PMET: Precise Model Editing in a Transformer. In *AAAI 2024*, 18564–18572.
- Li, Z.; and Hoiem, D. 2018. Learning without Forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12): 2935–2947.
- Liang, J.; Huang, W.; et al. 2025. LoRASculpt: Sculpting LoRA for Harmonizing General and Specialized Knowledge in Multimodal Large Language Models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, 26170–26180.

- Lightblue. 2024. LB Reranker: A multilingual reranker model fine-tuned from Qwen2.5-0.5B-Instruct. <https://github.com/lightblue-tech/lb-reranker>. Accessed: 2025-8-22.
- Liu, S.; Wang, C.; et al. 2024. DoRA: Weight-Decomposed Low-Rank Adaptation. In *ICML 2024*.
- Liu, W.; Huang, X.; et al. 2024. ToolACE: Winning the Points of LLM Function Calling. ArXiv:2409.00920 [cs].
- Lopez-Paz, D.; and Ranzato, M. 2017. Gradient Episodic Memory for Continual Learning. In *NeurIPS 2017*, 6467–6476.
- Luo, Y.; Yang, Z.; et al. 2025. An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning. ArXiv:2308.08747 [cs].
- Meng, F.; Wang, Z.; et al. 2024. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models. In *NeurIPS 2024*.
- Meng, K.; Bau, D.; et al. 2022. Locating and Editing Factual Associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, 17359–17372.
- Meng, K.; Sharma, A. S.; et al. 2023a. Mass-Editing Memory in a Transformer. In *ICLR 2024*.
- Meng, K.; Sharma, A. S.; et al. 2023b. Mass-Editing Memory in a Transformer. ArXiv:2210.07229 [cs].
- Michel, P.; Levy, O.; et al. 2019. Are Sixteen Heads Really Better than One? In *NeurIPS 2019*, 14014–14024.
- Petroni, F.; Rocktäschel, T.; et al. 2019. Language Models as Knowledge Bases? In *EMNLP-IJCNLP 2019*, 2463–2473. Hong Kong, China.
- Prottasha, N. J.; Chowdhury, U. R.; et al. 2025. PEFT A2Z: Parameter-Efficient Fine-Tuning Survey for Large Language and Vision Models. ArXiv:2504.14117 [cs].
- Qwen; Yang, A.; et al. 2025. Qwen2.5 Technical Report. ArXiv:2412.15115 [cs].
- Radford, A.; and Narasimhan, K. 2018. Improving Language Understanding by Generative Pre-Training.
- Sakaguchi, K.; Bras, R. L.; et al. 2020. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. In *AAAI 2020*, 8732–8740.
- Sanh, V.; Wolf, T.; et al. 2020a. Movement Pruning: Adaptive Sparsity by Fine-Tuning. In *NeurIPS 2020*.
- Sanh, V.; Wolf, T.; et al. 2020b. Movement Pruning: Adaptive Sparsity by Fine-Tuning. In *Advances in Neural Information Processing Systems*, volume 33, 20378–20389.
- Sun, Q.; Pickett, M.; et al. 2025a. Transformer Layers as Painters. In *AAAI-25*, 25219–25227.
- Sun, Q.; Pickett, M.; et al. 2025b. Transformer Layers as Painters. ArXiv:2407.09298 [cs].
- Sundararajan, M.; Taly, A.; et al. 2017. Axiomatic Attribution for Deep Networks. In *ICML 2024*, volume 70 of *Proceedings of Machine Learning Research*, 3319–3328.
- Tan, C.; Zhang, G.; et al. 2024. Massive Editing for Large Language Models via Meta Learning. In *ICLR 2024*.
- Tenney, I.; Das, D.; et al. 2019. BERT Rediscovered the Classical NLP Pipeline. In *ACL 2019*, 4593–4601. Florence, Italy.
- Vaswani, A.; Shazeer, N.; et al. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30.
- Vig, J.; and Belinkov, Y. 2019. Analyzing the Structure of Attention in a Transformer Language Model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@ACL 2019, Florence, Italy, August 1, 2019*, 63–76.
- Voita, E.; Talbot, D.; et al. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *ACL 2019*, 5797–5808. Florence, Italy.
- Wallat, J.; Singh, J.; et al. 2020. BERTnesia: Investigating the capture and forgetting of knowledge in BERT. In *BlackboxNLP 2019*, 174–183. Online.
- Wang, X.; Chen, T.; et al. 2023. Orthogonal Subspace Learning for Language Model Continual Learning. In *Findings of EMNLP*, 10658–10671.
- Xin, Y.; Yang, J.; et al. 2025. Parameter-Efficient Fine-Tuning for Pre-Trained Vision Models: A Survey. ArXiv:2402.02242 [cs].
- Yang, Y.; Li, X.; et al. 2024. CorDA: Context-Oriented Decomposition Adaptation of Large Language Models for Task-Aware Parameter-Efficient Fine-tuning. In *NeurIPS 2024*.
- Zellers, R.; Holtzman, A.; et al. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *ACL*, 4791–4800.
- Zhang, J.; You, J.; et al. 2025. LoRI: Reducing Cross-Task Interference in Multi-Task Low-Rank Adaptation. *CoRR*, abs/2504.07448.
- Zhang, Q.; Chen, M.; et al. 2023. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *ICLR 2024*.
- Zhang, Z.; Lin, Y.; et al. 2022. MoEification: Transformer Feed-forward Layers are Mixtures of Experts. In *Findings of ACL*, 877–890.
- Zhou, J.; Lu, T.; et al. 2023. Instruction-Following Evaluation for Large Language Models. ArXiv:2311.07911 [cs].
- Zhu, D.; Sun, Z.; et al. 2024. Model Tailor: Mitigating Catastrophic Forgetting in Multi-modal Large Language Models. In *ICML 2024*.