Artificial Intelligence and Digital Technology

Article

Mitigating Catastrophic Forgetting in Fine-Tuned Large Language Models: An Experimental Study of LoRA and O-LoRA

Xinlan Zhang [1,*]

[1] Chongqing Normal University, Chongqing, China
* Correspondence: Xinlan Zhang, Chongqing Normal University, Chongqing, China

Abstract: Large language models (LLMs) have become a hot topic in AI, and since the GPT series they have achieved remarkable success across many domains. However, directly using a general-purpose model often fails to meet the needs of specific applications, which motivates fine-tuning with domain-specific data. Nevertheless, parameter-efficient fine-tuning (PEFT) methods such as LoRA may perform poorly on certain algorithmic benchmarks, raising concerns about cat-astrophic forgetting. In this paper, we conduct extensive experiments to confirm this phenomenon and investigate O-LoRA as a mitigation strategy. Results show that O-LoRA can effectively alleviate catastrophic forgetting under continual instruction fine-tuning, but its effectiveness can be sensitive to hyperparameters on some datasets. Overall, O-LoRA provides a practical direction for mitigating catastrophic forgetting during continual fine-tuning of LLMs.

Keywords: large language models; catastrophic forgetting; parameter-efficient fine-tuning; LoRA; O-LoRA

Published: 08 February 2026

Copyright: © 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

This work investigates catastrophic forgetting that may arise during continual instruction fine-tuning of large language models (LLMs), with a particular focus on parameter-efficient fine-tuning (PEFT) methods such as LoRA, and explores O-LoRA as a mitigation strategy.

1.1. Background

Since the Turing Test was proposed in the 1950s, researchers have explored how machines can acquire language intelligence. Language is a complex human expression system governed by grammatical rules, and developing AI algorithms that understand and master language remains a major challenge. As an important approach, language modeling has been widely used for language understanding and generation over the past two decades, evolving from statistical language models to neural language models. More recently, pretraining Transformer models on large-scale corpora has led to pretrained language models (PLMs), which show strong capability across a variety of natural language processing (NLP) tasks. As scaling has been found to increase model capacity, researchers further explored scaling laws by increasing parameter counts. Interestingly, once model size exceeds a certain threshold, these scaled-up language models not only improve performance significantly, but also exhibit abilities not seen in smaller models (e.g., BERT), such as in-context learning. To distinguish language models of different scales, the community introduced the term large language models (LLMs), referring to PLMs with very large scale (e.g., tens or hundreds of billions of parameters) trained on massive text data, such as GPT-3, PaLM, Galactica, and LLaMA [1-5]. Recently, both

academia and industry have accelerated research on LLMs; a prominent milestone is ChatGPT, a powerful AI chatbot built on LLMs, which has attracted broad attention. The rapid evolution of LLM technology is reshaping the AI field and may fundamentally change how we develop and use AI algorithms [6].

1.2. Current Status and Motivation

Pretraining lays the foundation for models to learn linguistic capability. By pretraining on large corpora, LLMs obtain basic language understanding and generation abilities [2-3]. However, LLMs are often not well-suited to specific tasks unless they are fine-tuned. Full-parameter fine-tuning initializes from pretrained weights, updates all parameters, and produces a separate instance for each task [7]. As model size grows, updating all parameters and maintaining a separate instance per task becomes impractical. Parameter-efficient fine-tuning (PEFT) was therefore proposed to adapt LLMs to downstream tasks efficiently by training only a small subset of parameters-either a subset of existing parameters or a small set of newly added parameters [8]. Different PEFT methods vary in parameter efficiency, memory efficiency, training speed, final model quality, and inference overhead. In recent years, more than a hundred PEFT papers have been published, and surveys summarize popular approaches, including Adapters, BitFit, Prefix-Tuning, P-Tuning, Prompt-Tuning, and LoRA [9-15].

Despite their efficiency, fine-tuning may induce catastrophic forgetting-a phenomenon in machine learning where a model forgets previously learned knowledge while learning new information [16]. This can limit the generality and scalability of fine-tuned models in real-world applications [17]. Recent studies suggest that even advanced PEFT methods like LoRA can exhibit this behavior under certain conditions [16]. Common mitigation strategies include: (1) experience replay and related methods that store examples from previous tasks and train jointly with the current task, which may raise privacy concerns especially for sensitive data; (2) regularization-based methods that add penalties to the loss to discourage changes to important weights, such as Orthogonal Gradient Descent (OGD), but OGD requires storing gradients of all historical data and is infeasible for LLMs; and (3) methods that dynamically expand model capacity or isolate existing weights to reduce interference, such as Progressive Prompts, although such methods may generalize poorly to unseen tasks [18-20].

These approaches often face two key limitations: they require storing historical data or gradients (costly for large models), and they typically update tasks within a shared vector space that directly affects hidden representations. Motivated by these issues, we focus on O-LoRA, which aims to overcome both limitations [21].

The main contributions of this paper are summarized as follows:

1) We verify that PEFT methods such as LoRA can improve LLM performance on some tasks, but also suffer from catastrophic forgetting during continual fine-tuning.
2) We validate that O-LoRA can alleviate catastrophic forgetting when fine-tuning LLMs, while also observing that its effectiveness can be sensitive to hyperparameter choices.

2. Related Work

This section reviews several parameter-efficient fine-tuning techniques, including Prefix-Tuning, P-Tuning, and Prompt-Tuning. We also describe LoRA, which adapts models via low-rank decomposition and can significantly reduce trainable parameters and GPU memory requirements.

2.1. Prefix-Tuning

Prefix-Tuning is a PEFT method for LLMs that adjusts model behavior by prepending a continuous, learnable sequence of prefix vectors to the input, without

updating the full set of model parameters. The prefix is represented as continuous token embeddings whose influence propagates through all Transformer layers and affects subsequent tokens.

Concretely, one or two prefixes (for encoder-decoder architectures) are inserted before the autoregressive LLM input to form a new input sequence $z = [\text{PREFIX}; x; y]$ or $[\text{PREFIX}; x; \text{PREFIX}'; y]$. Here P_{idx} denotes the prefix index sequence and $|P_{idx}|$ is the prefix length. Unlike discrete prompts drawn from a fixed vocabulary, Prefix-Tuning treats prefix activations as free parameters defined by a matrix P_θ (controlled by parameters θ) with dimension $|P_{idx}| \times \dim(h_i)$. The training objective remains unchanged, but the trainable parameter set changes: the original model parameters ϕ are frozen and only θ is optimized. Each hidden state h_i depends on the learnable prefix matrix$P_\theta$regardless of whether its index lies in the prefix range. To improve optimization stability and performance, Prefix-Tuning uses re-parameterization: instead of directly updating P_θ, it maps a smaller matrix P'_θ through a large MLP_θ to obtain P_θ, and replaces $P_\theta[i,:]$ with $P'_\theta[i, :]$. After training, the re-parameterization parameters can be discarded and only the prefix P_θ is stored [12].

2.2. P-Tuning

P-Tuning aims to improve LLM performance on NLP tasks by inserting a set of continuous, learnable vectors as "virtual tokens" into the input sequence, replacing embeddings that would otherwise correspond to discrete prompt tokens. The continuous virtual tokens do not belong to the original vocabulary; rather, they are optimized as a set of continuous parameters. A template organizes context, targets, and virtual tokens into a structured input format.

During optimization, P-Tuning faces two main challenges: (1) discreteness, since pretrained token embeddings are highly discrete, and (2) dependency among prompt embeddings, since prompt tokens should exhibit correlations rather than evolve independently. To address these issues, P-Tuning uses lightweight neural architectures such as bidirectional LSTMs to encode the continuous prompts, helping them better capture context information and avoid poor local optima [13].

2.3. Prompt-Tuning

Prompt-Tuning is another PEFT approach that learns a set of continuous, optimizable "soft prompts" while keeping the pretrained model frozen [22,23]. Unlike discrete text prompts used in models such as GPT-3, soft prompts are updated end-to-end via backpropagation, and can integrate signals from any number of labeled examples. On large models such as the T5 family, as the number of parameters grows, the performance gap between Prompt-Tuning and full fine-tuning narrows and can become comparable [24]. This makes it possible for a single frozen model to serve multiple downstream tasks, substantially reducing storage and deployment cost.

In implementation, Prompt-Tuning reformulates tasks as text generation and prepends a trainable continuous prompt vector to the input. In T5, the soft prompt is represented as a matrix that is concatenated with the original input embedding matrix as the encoder-decoder input. During training, only the prompt parameters are updated to maximize the likelihood of the output sequence, while the main model parameters remain fixed. Prompt-Tuning can be viewed as a simplified variant of Prefix-Tuning [14].

2.4. LoRA

LoRA (Low-Rank Adaptation) addresses the high storage cost, low computational efficiency, and deployment challenges of adapting large pretrained language models to specific tasks or domains. LoRA keeps the pretrained weights fixed and injects trainable low-rank matrices A and B into each Transformer layer.

Specifically, given a pretrained weight matrix W, LoRA introduces a low-rank update $\Delta W = BA$, where $B \in R^{d \times r}$, $A \in R^{r \times k}$, and r is much smaller than the rank of W (i.e., d and k). During fine-tuning, only A and B are optimized, greatly reducing the number of trainable parameters for downstream tasks and lowering GPU memory usage. Compared with GPT-3 fine-tuned with Adam, LoRA can reduce trainable parameters by up to 10,000× and cut GPU memory requirements by about 3× [15,25].

3. O-LoRA Method

To address catastrophic forgetting, this paper introduces O-LoRA. While keeping pretrained weights fixed, O-LoRA uses a low-rank parameter update mechanism under an orthogonality constraint to limit interference across tasks during learning, thereby reducing catastrophic forgetting.

O-LoRA (Orthogonal Low-Rank Adaptation) is a simple and efficient PEFT method proposed for continual learning with LLMs. In continual learning, a model learns a sequence of tasks, and knowledge from earlier tasks may be overwritten by later tasks. O-LoRA leverages the intrinsic low-rank property of weight updates during fine-tuning and constrains different tasks to be learned in mutually orthogonal low-rank subspaces.

Concretely, for each new task O-LoRA uses new low-rank matrices A and B to form $W_{int} + \Delta W = W_{int} + AB$. The task-specific update computed from A and B is constrained to be orthogonal to the gradient subspace of previous tasks, which reduces interference and mitigates forgetting. Experiments show that O-LoRA outperforms prior state-of-the-art methods on standard continual learning benchmarks, maintains strong average performance even across many tasks, and better preserves the generalization ability of LLMs to unseen tasks. Importantly, it avoids privacy risks by not requiring storage of user data for replay, and does not rely on task identifiers at test time, making it well-aligned with the instruction-tuning setting [21].

4. Experimental Design and Results

4.1. Performance of LoRA and Other PEFT Methods

We conduct instruction fine-tuning on the LLaMA-2-7B model using the Alpaca dataset, and evaluate the resulting models on TruthfulQA, BLiMP Causative, MMLU GlobalFacts, as well as arithmetic tasks. We compare LoRA under different hyperparameter settings with other PEFT methods to analyze performance changes.

Experimental setup: We fine-tune LLaMA-2-7B on Alpaca, which contains 52,000 instruction instances with demonstrations generated by OpenAI text-davinci-003, and is commonly used to improve instruction-following ability. For evaluation, we use representative NLP benchmarks (TruthfulQA, BLiMP Causative, MMLU GlobalFacts) and two arithmetic subtraction tasks of different difficulty (two-digit and four-digit subtraction). We systematically sweep LoRA hyperparameters (rank r and scaling α) and record accuracy on each evaluation task for each configuration.

From Table 1, we confirm that PEFT methods such as LoRA can improve model performance on some datasets. However, on arithmetic (algorithmic) datasets, we observe that under certain hyperparameter settings the accuracy of LoRA fine-tuned models drops substantially. Similar behavior appears for Prefix-Tuning, P-Tuning, and Prompt-Tuning, suggesting that these methods may encounter catastrophic forgetting or instability in continual learning scenarios.

Table 1. Performance of LoRA under different hyperparameter settings and other PEFT methods on multiple evaluation tasks.

Method/Setting	truthfulqa_mcl	truthfulqa_mc2	arithmetic_2ds	arithmetic_4ds	blimp_causativa	global_facts
LLaMA-2-7B	0.2521	0.3897	0.5025	0.3635	0.7580	0.2400
Finetune	0.2754	0.4058	0.5315	0.3720	0.7820	0.3700
LoRA, r=64, α=1	0.2521	0.3740	0.5245	0.3700	0.7680	0.3300
LoRA, r=64, α=16	0.2619	0.4068	0.8275	0.5070	0.7650	0.3700
LoRA, r=64, α=32	0.2619	0.3995	0.6600	0.4620	0.7730	0.3600
LoRA, r=64, α=64	0.2411	0.3695	0.6225	0.0880	0.7540	0.3200
LoRA, r=64, α=128	0.2448	0.4959	0.0000	0.0000	0.4200	0.1800
LoRA,r=128, α=1	0.2546	0.3730	0.5310	0.3695	0.7760	0.3500
LoRA,r=128, α=16	0.2931	0.4387	0.5255	0.3570	0.7710	0.3500
LoRA,r=128, α=32	0.2644	0.4461	0.4100	0.1360	0.7310	0.3900
LoRA,r=128, α=64	0.2166	0.3367	0.0000	0.0000	0.6700	0.2900
LoRA,r=128, α=128	0.2436	0.4949	0.0000	0.0000	0.4280	0.1800
LoRA,r=128, α=256	0.2424	0.5000	0.0000	0.0000	0.4530	0.1800
LoRA,r=256, α=256	0.2387	0.5027	0.0000	0.0000	0.4490	0.1800
Prefix-Tuning	0.2436	0.4858	0.0000	0.0000	0.3910	0.1800
P-Tuning	0.2326	0.4845	0.0000	0.0000	0.4050	0.2300
Prompt-Tuning	0.2411	0.4867	0.0000	0.0000	0.3990	0.2100

4.2. Verifying Catastrophic Forgetting and the Effect of O-LoRA

To further analyze the phenomenon observed in Experiment 4.1-namely catastrophic forgetting during fine-tuning with PEFT methods such as LoRA-we follow the continual learning perspective and evaluation protocol described in "An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning" [26]. We design a sequence of instruction-tuning tasks to examine both learning ability and knowledge retention.

We consider three core instruction-tuning tasks:

1) Text simplification (Simp): simplifying complex text into an easier-to-understand form.
2) Explanation generation (Exp): generating natural language explanations for a given premise, hypothesis, or label to demonstrate deeper understanding and reasoning.

3) Inquisitive question generation (InqQG): generating questions from a long-form answer to test information extraction and reverse reasoning ability.

For evaluation, we use three categories of general-purpose metrics and representative benchmark datasets, summarized in Table 2.

Table 2. Evaluation sets and their elements used in this study.

Set	Elements
Domain Knowledge (DK)	MMLU: STEM; Social; Human; Other
Reasoning (Rs)	BoolQ; PIQA; Winogrande; HellaSwag; MathQA; Mutual
Reading Comprehension (RC)	RACE-high; RACE-middle

Specifically: (1) Domain knowledge is evaluated using the multi-task language understanding benchmark MMLU, measuring knowledge storage and application across STEM, Human, Social, and Other categories; (2) Reasoning ability is tested with widely used commonsense reasoning datasets (HellaSwag, Winogrande, PIQA), along with MathQA for mathematical reasoning and Mutual for dialogue reasoning; (3) Reading comprehension is evaluated on RACE using both middle- and high-difficulty splits to reflect deep understanding and information extraction from complex texts.

Let $E_i (i = 1, 2, 3)$ denote the above evaluation sets. Each set contains multiple datasets or splits (Table 2). For example, E_i corresponds to MMLU and contains STEM, Human, Social, and Other. For each element e in E_i, we denote the evaluation score after the m-th continual learning task as R_m^e. Following, we use the average decrease in scores as a surrogate measure of forgetting, defined as the Forgetting Gradient (FG_i) [17]:

$FG_i = \frac{1}{|E_i|}\sum_{e \in E_i}\frac{1}{N}\sum_{m=1}^{n}\frac{R_0^e - R_m^e}{R_0^e} * 100\%$, where R_0^e is the score of the initial (pre-fine-tuning) LLM, and evaluation is performed using the EleutherAI lm-evaluation-harness framework.

4.2.1. Catastrophic forgetting in LoRA and other PEFT methods

In the data presented in Table 3, R_0^e and R_{-1}^e represent the evaluation scores of the Llama-2-7b model before and after instruction fine-tuning, respectively. R_0^e reflects the model's performance on various evaluation sets in its initial state, while R_{-1}^e corresponds to the results at the end of the fine-tuning process. The forgetting metric FG (Forgetting Gradient) is used to measure the extent to which the LLM (large language model) forgets previously learned knowledge during continuous instruction tuning. The results show that, whether using traditional fine-tuning methods or efficient fine-tuning techniques such as LoRA, the FG values in the three core capabilities of domain knowledge, reasoning, and reading comprehension are all greater than zero. This indicates that the stored commonsense knowledge in the Llama-2-7b model is forgotten during continuous instruction fine-tuning. It is worth noting that all experimental data are based on the unified Llama-2-7b model. Therefore, R_0^e in the experiments represents the model's original performance on different evaluation datasets, providing a benchmark for analyzing and comparing the model's knowledge retention before and after fine-tuning.

Table 3. Forgetting (FG) of LoRA and other PEFT methods under continual fine-tuning on LLaMA-2-7B.

Method/Setting	Domain Knowledge			Reasoning			Reading Comprehension		
	R^e_0	R^e_{-1}	FG	R^e_0	R^e_{-1}	FG	R^e_0	R^e_{-1}	FG
Finetune	42.91	23.08	42.45	60.19	36.69	37.90	39.52	23.44	42.29
LoRA,r=64,α=1	42.91	39.01	30.79	60.19	57.50	20.03	39.52	38.56	19.69
LoRA,r=128,α=1	42.91	24.48	38.87	60.19	35.45	40.46	39.52	21.72	43.90

LoRA,r=64,α=16	42.91	23.11	40.43	60.19	35.42	40.73	39.52	22.68	43.66
LoRA,r=64,α=128	42.91	23.11	42.71	60.19	35.49	40.90	39.52	20.96	44.95
Prefix-Tuning	42.91	25.78	37.15	60.19	37.89	38.54	39.52	23.73	42.05
P-Tuning	42.91	25.27	36.51	60.19	37.89	38.54	39.52	23.73	42.05
Prompt-Tuning	42.91	23.98	38.39	60.19	37.67	37.39	39.52	22.30	41.81

4.2.2. NEFTune combined with LoRA

We additionally test NEFTune, which injects noise into embeddings during instruction fine-tuning, and evaluate whether combining NEFTune with LoRA reduces forgetting [26].

As shown in Table 4, FG remains greater than zero for domain knowledge, reasoning, and reading comprehension even after applying NEFTune+LoRA. This suggests that catastrophic forgetting persists: while the model gains new-task ability, it still forgets some previously acquired knowledge.

Table 4. Forgetting (FG) when combining NEFTune with LoRA.

Method/Setting	**Domain Knowledge**			**Reasoning**			**Reading Comprehension**		
	$\mathbf{R^e_0}$	$\mathbf{R^e_{-1}}$	**FG**	$\mathbf{R^e_0}$	$\mathbf{R^e_{-1}}$	FG	R^e_0	R^e_{-1}	FG
NEFTune,noise =1	42.91	25.18	40.55	60.19	35.60	40.84	39.52	20.48	43.90
NEFTune,noise =5	42.91	23.55	42.40	60.19	35.11	40.94	39.52	21.05	42.69

4.2.3. Does O-LoRA alleviate catastrophic forgetting?

Finally, we evaluate O-LoRA under the same protocol to assess its effectiveness at mitigating catastrophic forgetting in continual learning.

From Table 5, the FG values for domain knowledge and reasoning are still above zero but are noticeably smaller than those of LoRA on the same datasets, and the FG value for reading comprehension can become negative. These results indicate that O-LoRA can mitigate catastrophic forgetting to some extent in continual fine-tuning, addressing challenges faced by LoRA.

Table 5. Forgetting (FG) of O-LoRA under continual fine-tuning on LLaMA-2-7B.

Method/Setting	**Domain Knowledge**			**Reasoning**			**Reading Comprehension**		
	$\mathbf{R^e_0}$	$\mathbf{R^e_{-1}}$	**FG**	$\mathbf{R^e_0}$	$\mathbf{R^e_{-1}}$	FG	R^e_0	R^e_{-1}	FG
OLoRA,r=64,lamda =0.5	42.91	41.43	1.11	60.19	59.74	0.43	39.52	40.67	-1.2
OLoRA,r=128,lamda =0.5	42.91	41.65	0.77	60.19	59.94	0.25	39.52	39.90	-0.24
OLoRA,r =64,α=16, lamda =0.5	42.91	40.64	3.49	60.19	60.17	-0.16	39.52	39.90	-0.88
OLoRA, r=64,α=128, lamda =0.5	42.91	39.76	1.70	60.19	59.24	0.51	39.52	38.47	1.20

4.3. Performance Evaluation of O-LoRA under Hyperparameter Sweeps

To examine whether O-LoRA can alleviate catastrophic forgetting under different configurations while maintaining good task performance, we extend Experiment 4.1 by replacing LoRA with O-LoRA and performing a broad evaluation under various hyperparameter settings. We use the Alpaca dataset (52,000 instruction demonstrations generated by OpenAI text-davinci-003) to instruction-tune LLaMA-2-7B, and evaluate using the EleutherAI lm-evaluation-harness framework.

We evaluate TruthfulQA, BLiMP Causative, and MMLU GlobalFacts, as well as arithmetic subtraction tasks of different difficulty (e.g., two-digit subtraction such as "7 minus 66 = −59" and four-digit subtraction such as "6319 minus 2968 = 3351"). Table 6 summarizes performance across O-LoRA settings.

Table 6. Performance of O-LoRA under different hyperparameter settings on multiple evaluation tasks.

Method/Setting	truthfulqa_mcl	truthfulqa_mc2	arithmetic_2ds	arithmetic_4ds	blimp_causativa	global_facts
LLaMA-2-7B	0.2521	0.3897	0.5025	0.3635	0.7580	0.2400
Finetune	0.2754	0.4058	0.5315	0.3720	0.7820	0.3700
OLoRA,r=64,α=1	0.2546	0.3897	0.5020	0.3660	0.7060	0.2400
OLoRA,r=64,α=16	0.2497	0.3880	0.7750	0.3075	0.6400	0.2500
OLoRA,r=64,α=32	0.2460	0.5063	0.0000	0.0000	0.4450	0.1800
OLoRA,r=64,α=64	0.2424	0.4989	0.0000	0.0000	0.4420	0.1800
OLoRA,r=64,α=128	0.2411	0.4959	0.0000	0.0000	0.4350	0.1800
OLoRA,r=128,α=1	0.2583	0.3887	0.5030	0.3620	0.7470	0.2400
OLoRA,r=128,α=16	0.2228	0.3323	0.5045	0.3470	0.7180	0.3600
OLoRA,r=128,α=32	0.2362	0.3644	0.6575	0.3470	0.7310	0.3300
OLoRA,r=128,α=64	0.2411	0.4943	0.0000	0.0000	0.4090	0.1800
OLoRA,r=128,α=128	0.2338	0.4957	0.0000	0.0000	0.4400	0.1800
0LoRA,r=128,α=256	0.2436	0.4982	0.0000	0.0000	0.4450	0.1800
OLoRA,r=256,α=256	0.2277	0.4894	0.0000	0.0000	0.4500	0.1800

From Table 6, O-LoRA does not always improve monotonically with larger hyperparameters. In some cases, increasing hyperparameters can degrade performance, especially on arithmetic (algorithmic) datasets, where O-LoRA may face challenges similar to those observed for LoRA.

These results suggest that while O-LoRA can help mitigate catastrophic forgetting in certain settings, it is not a universal solution and may require task- or dataset-specific tuning to ensure stable and effective performance. Nonetheless, O-LoRA still shows advantages on some challenging tasks, indicating good applicability across diverse tasks and datasets.

We also compare peak memory usage: LoRA (bfloat16), r=64, alpha=16 uses 30.49 GB; O-LoRA (bfloat16), r=64, alpha=16 uses 36.26 GB.

5. Conclusions and Future Work

Through a series of experiments, we confirm that parameter-efficient fine-tuning techniques such as LoRA can indeed suffer from catastrophic forgetting during continual

fine-tuning of large language models. To address this challenge, we introduce and examine the effectiveness of O-LoRA. Experimental results show that, compared with other PEFT methods, O-LoRA exhibits a positive effect in mitigating catastrophic forgetting. However, when applying O-LoRA to other datasets and task scenarios, its performance can be strongly affected by hyperparameter choices, indicating that it does not achieve optimal performance in all cases. This reveals a key limitation of O-LoRA: sensitivity to hyperparameter settings and lack of universal robustness across fine-tuning environments. Nevertheless, as an innovative approach to addressing catastrophic forgetting in continual fine-tuning of LLMs, O-LoRA remains promising. Future work should focus on further optimizing and improving O-LoRA to reduce its dependence on hyperparameters, better adapting it to diverse datasets and tasks, and ultimately achieving more stable and efficient fine-tuning.

References

1. M. Shanahan, "Talking about large language models," *Communications of the ACM*, vol. 67, no. 2, pp. 68-79, 2024. doi: 10.1145/3624724
2. T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, and D. Amodei, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877-1901, 2020.
3. A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, and N. Fiedel, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1-113, 2023.
4. R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, and R. Stojnic, "Galactica: A large language model for science," *arXiv preprint arXiv:2211.09085*, 2022.
5. H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. A. Lachaux, T. Lacroix, and G. Lample, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.
6. W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, and J. R. Wen, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, vol. 1, no. 2, 2023.
7. N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, and M. Sun, "Parameter-efficient fine-tuning of large-scale pre-trained language models," *Nature machine intelligence*, vol. 5, no. 3, pp. 220-235, 2023. doi: 10.1038/s42256-023-00626-4
8. V. Lialin, V. Deshpande, and A. Rumshisky, "Scaling down to scale up: A guide to parameter-efficient fine-tuning," *arXiv preprint arXiv:2303.15647*, 2023.
9. N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, and M. Sun, "Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models," *arXiv preprint arXiv:2203.06904*, 2022. doi: 10.21203/rs.3.rs-1553541/v1
10. N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, and S. Gelly, "Parameter-efficient transfer learning for NLP," In *International conference on machine learning*, May, 2019, pp. 2790-2799.
11. E. B. Zaken, Y. Goldberg, and S. Ravfogel, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, May, 2022, pp. 1-9.
12. X. L. Li, and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021. doi: 10.18653/v1/2021.acl-long.353
13. X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "GPT understands, too," *AI Open*, vol. 5, pp. 208-215, 2024. doi: 10.1016/j.aiopen.2023.08.012
14. B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021. doi: 10.18653/v1/2021.emnlp-main.243
15. E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *ICLR*, vol. 1, no. 2, p. 3, 2022.
16. H. Li, L. Ding, M. Fang, and D. Tao, "Revisiting catastrophic forgetting in large language model tuning," *arXiv preprint arXiv:2406.04836*, 2024. doi: 10.18653/v1/2024.findings-emnlp.249
17. Y. Luo, Z. Yang, F. Meng, Y. Li, J. Zhou, and Y. Zhang, "An empirical study of catastrophic forgetting in large language models during continual fine-tuning," *IEEE Transactions on Audio, Speech and Language Processing*, 2025. doi: 10.1109/taslpro.2025.3606231
18. D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," *Advances in neural information processing systems*, vol. 32, 2019.
19. M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning," In *International conference on artificial intelligence and statistics*, June, 2020, pp. 3762-3773.
20. A. Razdaibiedina, Y. Mao, R. Hou, M. Khabsa, M. Lewis, and A. Almahairi, "Progressive prompts: Continual learning for language models," *arXiv preprint arXiv:2301.12314*, 2023.

21. X. Wang, T. Chen, Q. Ge, H. Xia, R. Bao, R. Zheng, and X. J. Huang, "Orthogonal subspace learning for language model continual learning," In *Findings of the Association for Computational Linguistics: EMNLP 2023*, December, 2023, pp. 10658-10671.
22. J. Huang, L. Cui, A. Wang, C. Yang, X. Liao, L. Song, and J. Su, "Mitigating catastrophic forgetting in large language models with self-synthesized rehearsal," *arXiv preprint arXiv:2403.01244*, 2024. doi: 10.18653/v1/2024.acl-long.77
23. A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, and S. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," *Advances in neural information processing systems*, vol. 32, 2019.
24. C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1-67, 2020.
25. Y. Zhai, S. Tong, X. Li, M. Cai, Q. Qu, Y. J. Lee, and Y. Ma, "Investigating the catastrophic forgetting in multimodal large language model fine-tuning," In *Conference on Parsimony and Learning*, January, 2024, pp. 202-227.
26. N. Jain, P. Y. Chiang, Y. Wen, J. Kirchenbauer, H. M. Chu, G. Somepalli, and T. Goldstein, "Neftune: Noisy embeddings improve instruction finetuning," *arXiv preprint arXiv:2310.05914*, 2023.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of SOAP and/or the editor(s). SOAP and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.