Learning from the Undesirable: Robust Adaptation of Language Models without Forgetting

Yunhun Nam[1], **Jaehyung Kim**[2], **Jongheon Jeong**[1]
[1]Korea University,
[2]Yonsei University
{yh0326, jonghj}@korea.ac.kr, jaehyungk@yonsei.ac.kr

Abstract

Language models (LMs) are often adapted through supervised fine-tuning (SFT) to specialize their capabilities for downstream tasks. However, in typical scenarios where the fine-tuning data is limited, *e.g.*, compared to pre-training, SFT can lead LMs to overfit, causing them to rely on spurious patterns within the target task or to compromise other broadly useful capabilities as a side effect of narrow specialization. In this paper, we propose *Learning-from-the-Undesirable* (LfU), a simple yet effective regularization scheme for SFT to mitigate overfitting issues when fine-tuning LMs with limited data. Specifically, we aim to regularize the fine-tuning process to favor solutions that are resilient to "undesirable" model updates, *e.g.*, gradient ascent steps that steer the model toward undesirable behaviors. To this end, we propose a novel form of consistency regularization that directly aligns internal representations of the model with those after an undesirable update. By leveraging representation-level data augmentation through undesirable updates, LfU effectively promotes generalization under limited data. Our experiments on diverse LM downstream tasks show that LfU serves as an effective prior that enhances adaptability while preserving pretrained knowledge. For example, our LM from LfU achieves a 16.8% average improvement on math tasks compared to vanilla SFT on the same dataset, where the latter even leads to degraded performance on those tasks. Furthermore, LfU exhibits improved robustness to prompt variations, *e.g.*, yielding a 92.1% lower standard deviation in output performances compared to SFT, highlighting its versatile effects.

Code — https://github.com/yunpal/LfU
Extended version — https://arxiv.org/abs/2511.13052

1 Introduction

Language models (LMs) (Touvron et al. 2023; Mann et al. 2020; OpenAI 2022, 2023; Anthropic 2023; Gemini Team 2023) have recently emerged as strong backbones for various downstream tasks, thanks to their unprecedented capabilities in understanding natural language and encoding vast world knowledge: *e.g.*, in question answering (Lewis et al. 2020), safeguard modeling (Inan et al. 2023), code generation (Roziere et al. 2023), API call generation (Patil et al. 2024) and reward modeling (Rafailov et al. 2023). *Supervised fine-tuning* (SFT) is currently a standard approach for adapting LMs to specific tasks; *i.e.*, by fine-tuning them on curated input-output pairs that reflect the desired behaviors. Ultimately, SFT aims to reliably steer model behavior towards specific tasks while effectively preserving the general knowledge and reasoning capabilities of LMs.

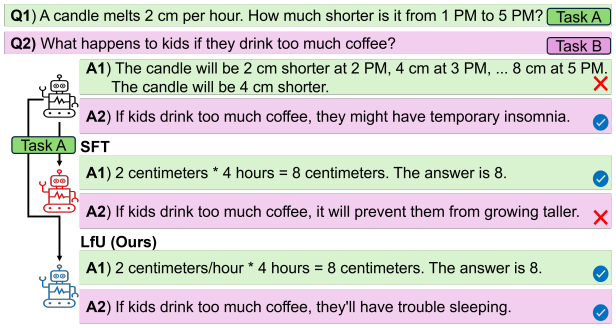
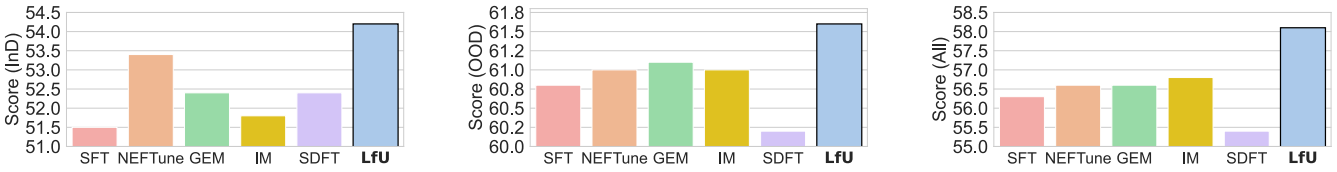


Figure 1: Illustration of forgetting issues in SFT. SFT on Task A causes a model to forget prior knowledge related to Task B. In contrast, LfU (Ours) successfully learns Task A while preserving the prior knowledge about Task B.

In practice, SFT often suffers from overfitting, which not only forgets prior knowledge but also increases vulnerability to adversarial fine-tuning. As shown in figure 1, the model adapted to a downstream task often forgets prior knowledge. Furthermore, they exhibit high sensitivity to prompt variations during inference (Reynolds and McDonell 2021; Zhu et al. 2023), making their behavior unstable across semantically equivalent inputs. In addition, SFT suffers from fragile alignment, as it tends to overfit to superficial refusal patterns rather than learning robust safety behaviors. As demonstrated by (Qi et al. 2024), only a few steps of adversarial fine-tuning are sufficient to undo these behaviors and elicit harmful outputs.

To mitigate these limitations, recent efforts have focused on mitigating overfitting during SFT (Jain et al. 2023; Shi et al. 2024; Li et al. 2025; Yang et al. 2024). For example, Jain et al. (2023) proposes injecting small noise into the input token embedding vectors during the forward pass of training. Other approaches focus on refining the stan-



(a) GSM8k → Math (In-domain) (b) GSM8k → All (Out-of-domain) (c) Alpagasus Dolly 3k → All

Figure 2: Performance comparison between baselines and LfU fine-tuned on Llama-3.1-8B. (a) and (b) show results fine-tuned on GSM8k and evaluated on in-domain and out-of-domain data, respectively. These results indicate that while prior methods struggle to adapt to in-domain examples and lead to only marginal improvements on out-of-domain data, LfU consistently achieves the best performance in both cases. (c) presents results from fine-tuning on the multitask dataset Alpagasus Dolly 3k, with evaluation across all tasks. LfU achieves the best overall performance.

dard cross-entropy objective in SFT. Shi et al. (2024) proposes Instruction Modelling (IM), which applies the loss function not only over the output but also over the instruction. This encourages the model to pay closer attention to the instruction and improves generalization by mitigating overfitting. Li et al. (2025) proposes Game-theoretic Entropy Maximization (GEM), which preserves output diversity by modeling supervised learning as a distribution matching game with entropy regularization. As shown in Figure 2, these methods lead to only marginal improvements on both in-domain (InD) and out-of-domain(OOD) examples. In a complementary direction, Yang et al. (2024) proposes rewriting the outputs of the training data to resemble those of the pre-trained model, thereby better aligning the training data distribution with the original output distribution. However, Figure 2 shows that it requires an instruction-tuned model to begin with, making it incompatible with untuned vanilla models.

Contributions In this paper, we propose a simple yet effective method to mitigate overfitting, coined *Learning-from-the-Undesirable* (LfU). Overfitting often arises in SFT due to limited training data, where the model tends to memorize spurious patterns. A common strategy to alleviate this issue is data augmentation (Wei and Zou 2019; Feder et al. 2023), which improves generalization by exposing the model to diverse variations of inputs. Inspired by this, we leverage the effect of data augmentation at the representation level by introducing undesirable updates to generate diverse internal representations for the same input. The key idea of LfU is to simulate undesirable updates by constructing an auxiliary model and applying a single gradient ascent step that shifts the model in an undesirable direction. To do this, we augment the original model with trainable components, *e.g.*, LoRA parameters or representation steering vectors, and compute the gradient of the SFT objective with respect to these components. Using this gradient, we can get an undesirable model. Then, LfU regularizes the training process by enforcing consistency between the internal representations of the original and undesirable models, thereby encouraging the model to maintain stable internal representations. As shown in Figure 2, our undesirable update strategy provides more effective augmentation and leads to better generalization compared to simple perturbation methods, such as adding Gaussian noise to input embeddings as done in NEFTune (Jain et al. 2023).

Extensive experiments across a wide range of downstream tasks and language models demonstrate the effectiveness of LfU. For example, when fine-tuning Llama-3.1-8B on GSM8k, LfU improves in-domain performance by up to 16.8% over SFT, outperforming all existing baselines in both in-domain and out-of-domain datasets. Moreover, LfU exhibits strong resilience to prompt variations, reducing standard deviation in output performances by 92.1% compared to SFT and improving average accuracy. Furthermore, in a few steps of adversarial fine-tuning scenarios, LfU significantly reduces the Attack Success Rate (ASR), achieving up to a 45.0% lower ASR on harmful datasets over SFT.

2 Related Works

Overfitting issues in SFT It has been observed that SFT tends to overfit limited data, causing forgetting and hallucinations (Kotha, Springer, and Raghunathan 2023; Gekhman et al. 2024; Lin et al. 2024); these problems can cause undesirable shifts during downstream adaptation (Luo et al. 2023; Jiang et al. 2025; Qi et al. 2023). To address this challenge, several studies have explored data augmentation (Wei and Zou 2019; Feder et al. 2023), *e.g.*, including the use of synthetic data (Ba, Mancenido, and Pan 2024). However, due to limited semantic diversity, synthetic data alone often leads to hallucinated or biased outputs and may not offer a fundamental solution (Guo et al. 2023; Shumailov et al. 2024; Rogulsky, Popovic, and Färber 2024; Fang et al. 2024). Another direction develops more principled training methods to better support generalization. For instance, Jain et al. (2023); Yadav and Singh (2023) tackle the challenge by injecting noise into embeddings during training. Shi et al. (2024) proposed to apply the cross-entropy loss to both outputs and instructions, and Li et al. (2025) aim to encourage the output diversity. Another recent study aims to reduce distributional shift by fine-tuning the model on their own generated responses (Yang et al. 2024). In this paper, we aim to enhance generalization of SFT through a new consistency regularization and update dynamics.

Consistency regularization As an alternative to data augmentation, *consistency regularization* (Sajjadi, Javanmardi, and Tasdizen 2016) offers a principled approach to improve

generalization by enforcing stable predictions under small perturbations. It has been widely studied across various areas, including semi-supervised learning (Xie et al. 2020; Sohn et al. 2020; Tang et al. 2023), robust learning (Zhang et al. 2019; Jeong and Shin 2020; Tack et al. 2022), generative modeling (Zhang et al. 2020; Ni et al. 2025), and bias mitigation (Wang et al. 2025). In this paper, LfU enforces internal representation consistency to mitigate overfitting by stabilizing the model against corruptions of itself, which we refer to as *undesirable updates*.

Learning from update dynamics Understanding and leveraging the dynamics of parameter updates has served as a powerful strategy for improving model generalization. One prominent example is meta-learning (Thrun and Pratt 1998; Rajeswaran et al. 2019), which trains models to quickly adapt to unseen tasks rather than optimizing for a single fixed objective. In particular, meta-learning simulates gradient-based updates on diverse tasks during training (Finn, Abbeel, and Levine 2017; Li et al. 2017), enabling the model to learn a parameter initialization that can be efficiently fine-tuned for new tasks. Sharpness-aware training, such as SAM (Foret et al. 2020; Oikonomou and Loizou 2025; Tahmasebi et al. 2024), takes a different approach by performing small gradient ascent steps to locate flatter minima, thereby reducing overfitting and improving robustness. While prior methods leverage update dynamics mainly for in-domain improvements, LfU operates at the representation level to achieve stable and generalizable training.

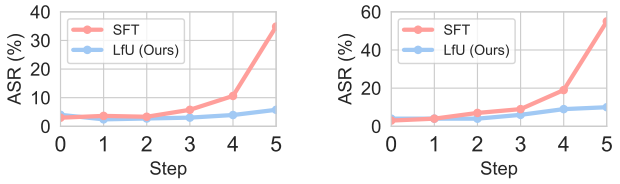
3 Method

We denote the language model by $p_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ represents the model parameters. Given an input $\boldsymbol{x}$, the model generates an output $\boldsymbol{y}$ by sampling each token $\boldsymbol{y}_t \sim p_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$, where t denotes the index of the output token. SFT adapts a language model to downstream tasks by minimizing the negative log-likelihood of the output tokens. To be specific, given a dataset $\mathcal{D}$ of input-output pairs $(\boldsymbol{x}, \boldsymbol{y})$, where an output of length T, the SFT objective is defined as:

$$\ell_{\text{SFT}}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \left[-\frac{1}{T} \sum_{t=1}^{T} \log p_{\boldsymbol{\theta}}(\boldsymbol{y}_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t}) \right]. \quad (1)$$

SFT has been applied to adapt LMs on specific datasets, but it often induces narrow response patterns and degrade generalization to other unrelated tasks. Although subsequent alignment stages, *e.g.*, via RL (Ouyang et al. 2022; Kirk et al. 2023), can help mitigating the issue, it opens up new challenges in specifying rewards and requires significant compute. In this paper, we aim to improve SFT in a way that makes it generalizable, so that the model can preserve its original capability even after specializing to a certain task.

Motivation We are motivated by the observations in Qi et al. (2024) that SFT tends to overfit when applied for safety, *i.e.*, to output short refusal response to harmful instructions. As illustrated in Figure 3, we observe that the behavior of such an SFT-tuned model, say p_{SFT}, can be easily altered by just a few steps of "adversarial" SFT on harmful datasets. To address this, we propose a learning strategy



(a) HEx-PHI (b) PureBad

Figure 3: Attack Success Rate (ASR) on (a) HEx-PHI and (b) PureBad after a few steps of adversarial fine-tuning on BeaverTails (Ji et al. 2023). We first align a Llama-3.1-8B via SFT (or LfU) with the harmless subset of BeaverTails, and then continue fine-tuning the model on the harmful subset (of BeaverTails) using SFT.

that enhances generalization by simulating undesirable behaviors and encouraging the model to remain robust against them. As shown in Figure 3, our approach remains robust even after a few steps of adversarial fine-tuning.[1]

3.1 Approach

We propose *Learning-from-the-Undesirable* (LfU), a regularization method that encourages the model to generalize better by reducing overfitting. The key idea is to simulate undesirable behaviors within the model and guide it to maintain stable internal representations under such conditions.

One step towards the undesirable To encourage stable representations, we simulate a single optimization step that shifts the model parameters toward undesirable directions. To this end, we define an auxiliary model $\boldsymbol{\theta}_{\text{aux}}$, which augments the parameters $\boldsymbol{\theta}$ with additional components. Specifically, the loss is computed with Eq. 1, and we compute its gradient with respect to the $\boldsymbol{\theta}_{\text{aux}}$ to determine the direction of perturbation. A single step of gradient ascent is then applied to update the auxiliary components, as follows:

$$\boldsymbol{\theta}_{\text{aux}} \leftarrow \boldsymbol{\theta}_{\text{aux}} + \alpha \cdot \frac{\nabla_{\boldsymbol{\theta}_{\text{aux}}} \ell_{\text{SFT}}(\boldsymbol{\theta}_{\text{aux}})}{\|\nabla_{\boldsymbol{\theta}_{\text{aux}}} \ell_{\text{SFT}}(\boldsymbol{\theta}_{\text{aux}})\|_2}, \quad (2)$$

where α is a step size controlling the magnitude of the perturbation. This perturbation intentionally induces undesirable behavior.

We next extract internal representations from both the original model $\boldsymbol{\theta}$ and the auxiliary $\boldsymbol{\theta}_{\text{aux}}$. Given the l_{th} layer of an LM, we define the function $M^{(l)}(\cdot; \boldsymbol{\theta})$ to return the internal representations at layer l under parameters $\boldsymbol{\theta}$. Similarly, we obtain the function $M^{(l)}(\cdot; \boldsymbol{\theta}, \boldsymbol{\theta}_{\text{aux}})$ to return the undesirable representations. We formally define the representations of l_{th} layer as follows:

$$\boldsymbol{h}_{l,t} = M^{(l)}(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{\theta}), \quad (3)$$

$$\boldsymbol{h}'_{l,t} = M^{(l)}(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{\theta}, \boldsymbol{\theta}_{\text{aux}}), \quad (4)$$

where $\boldsymbol{h}_{l,t}, \boldsymbol{h}'_{l,t} \in \mathbb{R}^d$ denote the representations from the original and auxiliary models, respectively and d is the dimensionality of the representations.

[1]The detailed experimental setup is provided in Appendix A.

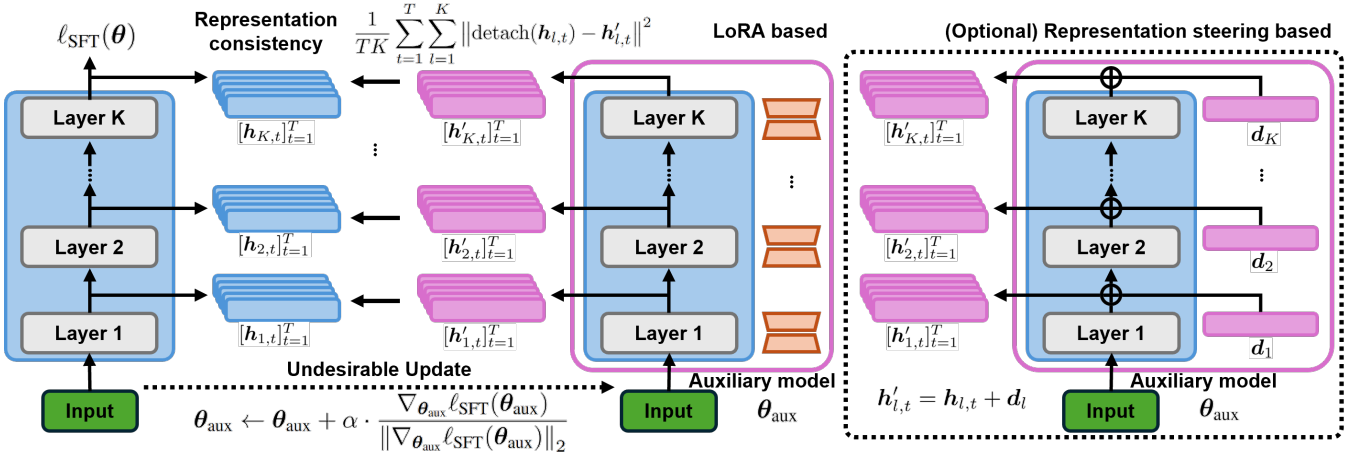


Figure 4: Overview of LfU: LfU promotes stable internal representations by enforcing consistency in internal representations between the original model $\boldsymbol{\theta}$ and auxiliary model $\boldsymbol{\theta}_{\text{aux}}$ that is optimized one step to induce undesirable behaviors. The auxiliary model $\boldsymbol{\theta}_{\text{aux}}$ is constructed by adding additional components to the original parameters $\boldsymbol{\theta}$, using either (1) a LoRA based method, where trainable low-rank matrices are added to each layer or (2) a representation steering based method, where a learnable steering vector is added to the internal representation at each layer. A gradient ascent step is then performed on the additional components by computing the gradient of the SFT objective with respect to $\boldsymbol{\theta}_{\text{aux}}$. The consistency loss is defined as the Mean Squared Error (MSE) between the internal representations of the original and auxiliary models across all layers.

Additional components to make auxiliary model There are two ways to construct $\boldsymbol{\theta}_{\text{aux}}$: the LoRA method and the representation steering method. The LoRA based method leverages Low Rank Adaptation (LoRA) (Hu et al. 2022). We define the model parameters $\boldsymbol{\theta}$ with K layers as $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K]$, where $\boldsymbol{\theta}_l$ denotes the parameters of the l-th layer. Each layer $\boldsymbol{\theta}_l$ is augmented with a trainable low-rank matrix LoRA_l, forming $\boldsymbol{\theta}_{\text{aux}} = [\boldsymbol{\theta}_1 + \text{LoRA}_1, \ldots, \boldsymbol{\theta}_K + \text{LoRA}_K]$. Only the LoRA terms are updated, while $\boldsymbol{\theta}$ remains frozen. The representation steering based method perturbs internal representations instead of parameters. At each layer l, a learnable steering vector $\boldsymbol{d}_l$ is added to the internal representation: $\boldsymbol{h}'_{l,t} = \boldsymbol{h}_{l,t} + \boldsymbol{d}_l$. This shifts internal representations in an undesirable direction without modifying $\boldsymbol{\theta}$, making it a computationally efficient alternative.

One step towards the desirable To encourage stability against undesirable perturbation, we enforce consistency between the model's internal representations before and after applying the perturbation, which was previously defined. We define the consistency regularization as:

$$\ell_{\text{cons.}}(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{aux}}) = \mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \left[\frac{1}{TK} \sum_{t=1}^{T} \sum_{l=1}^{K} \left\| \text{detach}(\boldsymbol{h}_{l,t}) - \boldsymbol{h}'_{l,t} \right\|^2 \right], \quad (5)$$

where $\text{detach}(\cdot)$ indicates that gradients do not flow. This design helps stabilize the internal representation $\boldsymbol{h}_{l,t}$ by encouraging $\boldsymbol{h}'_{l,t}$ to remain close to the fixed target. We incorporate the consistency objective into the overall training loss by combining it with the SFT objective. The resulting objective of LfU is defined as:

$$\ell_{\text{LfU}}(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{aux}}) = \ell_{\text{SFT}}(\boldsymbol{\theta}) + \lambda \cdot \ell_{\text{cons.}}(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{aux}}), \quad (6)$$

where λ is a hyperparameter. Detailed pseudocodes to implement LfU are provided in Appendix D.

4 Experiments

To evaluate the effectiveness of LfU, we conduct extensive experiments covering various adaptation scenarios of language models, including generalization to diverse task categories, applicability to various models, resilience to prompt variations and robustness against adversarial fine-tuning. We compare LfU with baselines (Jain et al. 2023; Li et al. 2025; Shi et al. 2024; Yang et al. 2024) that aim to mitigate the overfitting issues of SFT. Detailed descriptions of these baselines are provided in Appendix B. We begin by introducing the datasets used for evaluation.

4.1 Setup

Datasets To comprehensively evaluate LfU, we adopt diverse datasets for fine-tuning and evaluation. For fine-tuning, we use the multi-task datasets Alpagasus Dolly 3k (Conover et al. 2023) and LIMA (Zhou et al. 2023), and the single-task datasets GSM8k (Cobbe et al. 2021) and ARC-Challenge (Clark et al. 2018). For evaluation, we cover four categories: ***(i)*** Math (GSM8k (Cobbe et al. 2021); MathQA (Amini et al. 2019); ASDiv (Miao, Liang, and Su 2021)), ***(ii)*** Knowledge (MMLU (Hendrycks et al. 2021); PIQA (Bisk et al. 2020); HellaSwag (Zellers et al. 2019); LAMBADA (Paperno et al. 2016)), ***(iii)*** Reasoning (ARC (Clark et al. 2018); CoQA (Reddy, Chen, and Manning 2019)), and ***(iv)*** Helpfulness (ToxiGen (Hartvigsen et al. 2022); TruthfulQA (Lin, Hilton, and Evans 2022)). Each category score is the average performance across tasks within the category, and the rank is computed as the average of the ranks across categories. Further dataset details are provided in Appendix C.

	Math (3)	Knowl. (4)	Reason. (2)	Helpful. (2)	Rank
Llama-3.1-8B					
Vanilla	37.9	72.9	66.1	44.0	–
SFT	51.5	73.3	65.1	44.1	4.8
NEFTune	53.4	73.3	65.3	44.3	3.0
GEM	52.4	73.0	65.3	45.1	3.5
IM	51.8	73.1	**65.6**	44.4	3.5
SDFT	52.4	73.4	65.0	42.1	4.3
LfU	**54.2**	**73.5**	**65.6**	**45.7**	**1.0**
Llama-3.1-8B-Instruct					
Vanilla	60.5	74.0	81.0	69.9	–
SFT	65.7	73.6	80.5	67.6	4.0
NEFTune	66.0	73.6	80.7	67.3	4.3
GEM	65.0	73.8	81.3	67.6	3.8
IM	63.9	**74.1**	**81.5**	68.5	2.5
SDFT	65.5	74.0	79.8	67.4	4.3
LfU	**66.7**	74.0	**81.5**	**69.0**	**1.3**
Llama-2-7B					
Vanilla	17.1	66.4	60.4	41.0	–
SFT	35.9	65.9	60.0	40.5	3.8
NEFTune	36.1	66.2	60.1	40.4	3.3
GEM	35.9	66.4	59.9	40.6	3.3
IM	32.2	66.4	**61.0**	39.7	3.8
SDFT	33.6	65.6	60.0	40.1	5.0
LfU	**37.0**	**66.7**	60.2	**40.9**	**1.3**
Mistral-7B-v0.3					
Vanilla	14.7	72.9	64.9	42.5	–
SFT	48.2	72.8	64.8	43.1	4.5
NEFTune	50.3	72.9	65.4	43.0	3.5
GEM	49.9	73.1	65.6	42.9	3.0
IM	43.0	73.1	65.4	42.3	4.8
SDFT	46.4	**73.5**	**65.9**	42.8	3.0
LfU	**50.5**	**73.5**	65.6	**43.5**	**1.3**

Table 1: Comparison of the performance of language models fine-tuned on GSM8k, evaluated on 11 tasks spanning four categories. Each category score is the average performance across tasks within the category, and the rank is computed as the average of the ranks within each category.

	Math (3)	Knowl. (4)	Reason. (2)	Helpful. (2)	Rank
Llama-3.1-8B					
Vanilla	37.9	72.9	66.1	44.0	–
SFT	37.0	72.8	64.8	50.6	4.3
NEFTune	39.0	72.6	64.2	50.6	4.3
GEM	37.8	72.8	65.1	**50.8**	2.8
IM	38.4	**73.3**	**66.9**	48.6	2.5
SDFT	37.5	73.2	64.7	46.0	4.8
LfU	**43.2**	**73.3**	64.9	**50.8**	**1.5**
Llama-3.1-8B-Instruct					
Vanilla	60.5	74.0	81.0	69.9	–
SFT	56.6	72.9	81.8	69.3	4.8
NEFTune	57.7	72.8	81.8	**69.5**	4.0
GEM	59.0	73.2	**81.9**	68.7	3.5
IM	59.6	**73.6**	**81.9**	68.8	2.5
SDFT	59.9	73.2	81.2	69.4	3.5
LfU	**60.3**	**73.6**	**81.9**	**69.5**	**1.0**
Llama-2-7B					
Vanilla	17.1	66.4	60.4	41.0	–
SFT	24.0	66.9	57.3	**46.1**	3.3
NEFTune	24.3	**67.1**	56.6	45.4	3.3
GEM	24.3	67.0	57.3	45.8	2.8
IM	18.5	66.4	**62.0**	41.2	4.3
SDFT	12.8	66.1	60.0	43.0	4.8
LfU	**24.5**	67.0	57.8	**46.1**	**1.8**
Mistral-7B-v0.3					
Vanilla	14.7	72.9	64.9	42.5	–
SFT	28.9	72.9	65.5	45.4	4.5
NEFTune	32.2	73.0	65.4	47.9	3.8
GEM	18.5	**77.5**	66.1	44.6	3.5
IM	23.4	73.1	**67.1**	44.6	3.3
SDFT	22.3	73.0	65.8	48.5	3.5
LfU	**34.9**	73.2	65.6	**48.6**	**2.0**

Table 2: Comparison of the performance of language models fine-tuned on Alpagasus Dolly 3k, evaluated on 11 tasks spanning four categories. Each category score is the average performance across tasks within the category, and the rank is computed as the average of the ranks within each category.

4.2 Results

Single-task fine-tuning To demonstrate that LfU generalizes across diverse task categories and is applicable to various models, we evaluate it on four different language models: Llama-3.1-8B, Llama-3.1-8B-Instruct, Llama-2-7B, and Mistral-7B-v0.3. These models are fine-tuned on GSM8k and evaluated on 11 tasks covering math, knowledge, reasoning, and helpfulness. As shown in Table 1, LfU achieves the best performance in all task categories on Llama-3.1-8B and records the highest rank. In particular, it improves performance by $+5.2\%$ in math and $+3.4\%$ in helpfulness compared to SFT. For Llama-3.1-8B-Instruct, LfU again achieves the highest rank. While SFT performs worse than the vanilla model in reasoning, LfU surpasses the vanilla model in this category. In the case of Llama-2-7B, LfU also obtains the highest rank among all baselines. In contrast, SFT underperforms the vanilla model on all tasks except math, whereas LfU improves performance on both math and knowledge. Mistral-7B-v0.3 shows a similar trend, with LfU achieving the highest rank. While SFT degrades performance in knowledge and reasoning, LfU enhances results across all task categories, including a $+4.8\%$ improvement in math. LfU also shows a similar trend on ARC-Challenge and performs best on Qwen3 (Qwen Team 2025) when fine-tuned on GSM8k. Detailed results are provided in Appendix E. These results demonstrate that LfU is effective across various models and generalizes well to diverse tasks.

Multi-task fine-tuning We next evaluate LfU in a multi-task setting using Alpagasus Dolly 3k and LIMA. We first present results from training with Alpagasus Dolly 3k. According to the results in Table 2, LfU also demonstrates strong performance in the multi-task setting. In the case of LLaMA-3.1-8B, LfU achieves the highest rank, with a particularly notable improvement of $+16.8\%$ on the math tasks compared to SFT, which even underperforms the vanilla model. For LLaMA-3.1-8B-Instruct, LfU achieves the best performance across all categories and improves the math category score by $+6.5\%$ over SFT. In LLaMA-2-7B, LfU also records the highest rank among all methods. On Mistral-7B-v0.3, LfU improves performance by $+20.8\%$ on

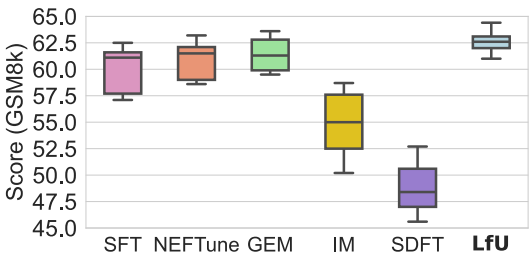


Figure 5: Performance distribution of GSM8k-fine-tuned Llama-3.1-8B models on five prompt variations of GSM8k. We use ChatGPT for generating the variations. LfU achieves the highest average accuracy and the lowest standard deviation, demonstrating strong resilience to prompt variations.

	Math	Knowl.	Reason.	Helpful.	Time/step (ms)
Vanilla	37.9	72.9	66.1	44.0	–
SFT	51.5	73.3	65.1	44.1	1386.9 ± 6.4
LfU (LoRA)	**54.2**	**73.5**	**65.6**	45.7	4142.4 ± 14.2
LfU (RepS)	53.2	73.3	**65.6**	**48.5**	2332.1 ± 9.1

Table 3: Comparison of the performance of Llama-3.1-8B fine-tuned on GSM8k and computation cost measured on a single NVIDIA H100 (80GB) instance, evaluated on 11 tasks spanning four categories: Math (3 tasks), Knowledge (4 tasks), Reasoning (2 tasks), and Helpfulness (2 tasks).

math and $+7.0\%$ on helpfulness compared to SFT, once again achieving the highest rank. These results demonstrate that LfU is effective not only across a wide range of models and task categories, but also in both single-task and multi-task adaptation scenarios. LfU also shows strong performance when fine-tuned with LIMA, and the corresponding results are reported in Appendix E.

Representation steering (RepS) based LfU By default, we consider the LoRA-based design in our experiments for running LfU. In Table 3, we test the representation steering (RepS) based design in comparison to the LoRA version, particulary focusing on its effect in accelerating the LfU training. Specifically, we fine-tune Llama-3.1-8B on GSM8k using the two designs of LfU, and compare their performances across 11 comprehensive tasks, as well as their per-step computation time measured in a single NVIDIA H100 (80GB) instance. The results show that LfU (RepS) is nearly twice as fast as LfU (LoRA), while achieving competitive performance with the LoRA version but with a slight decrease in the in-domain performance.

Robustness to prompt variations When evaluating LMs on various tasks, it is known that performance can be highly sensitive to prompts (Reynolds and McDonell 2021; Zhu et al. 2023). To examine whether LfU is less sensitive to such prompt variation, We evaluate GSM8k-adapted models on the GSM8k using five different prompt variations generated by ChatGPT (OpenAI 2024), with examples listed in Appendix C.4. As shown in Figure 5, LfU consistently yields a 92.1% smaller standard deviation and consistently outperforms other baselines. This indicates that LfU demonstrates resilience to prompt variations.

	HEx-PHI		PureBad		AdvBench	
Method	0-step	5-step	0-step	5-step	0-step	5-step
SFT	2.7	34.8	**3.0**	55.0	0.2	7.3
NEFTune	**2.4**	29.1	**3.0**	37.0	**0.0**	4.8
GEM	3.0	41.2	4.0	60.0	**0.0**	21.7
IM	5.8	60.9	8.0	80.0	1.2	81.9
SDFT	6.1	24.2	12.0	45.0	5.4	10.8
LfU	**2.4**	**5.8**	4.0	**10.0**	**0.0**	**0.6**

Table 4: Attack Success Rate (ASR) on HEx-PHI, PureBad and Advbench after 5 steps of adversarial fine-tuning on BeaverTails. We first align a Llama-3.1-8B via each baseline (or LfU) with the harmless subset of BeaverTails, and then continue fine-tuning the model on the harmful subset (of BeaverTails) using SFT.

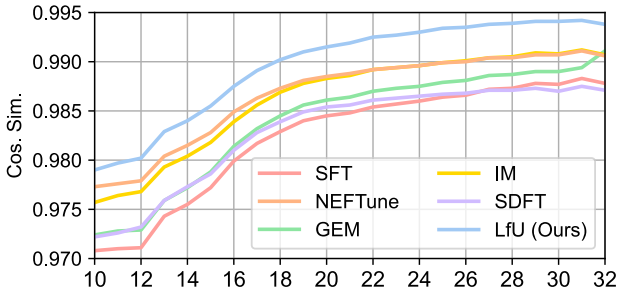


Figure 6: Cosine similarity of internal representations between clean and noisy inputs, using Llama-3.1-8B fine-tuned on GSM8k. Clean inputs correspond to the original training data, while noisy inputs are generated by adding Gaussian noise to their input embeddings. Among all methods, LfU shows the highest similarity across layers.

Robustness to adversarial fine-tuning To empirically demonstrate the vulnerability of safety-aligned models to a few steps of adversarial fine-tuning, we construct safe BeaverTails (Ji et al. 2023) with 5,000 refusal instruction–output pairs, while harmful BeaverTails contains separate instructions paired with harmful outputs (Huang et al. 2024). We first safety-align Llama-3.1-8B using the safe BeaverTails for 3 epochs using each baseline, then apply only 5 steps of adversarial fine-tuning with the harmful BeaverTails via SFT and measure attack success rate (ASR) on PureBad (Qi et al. 2023), HEx-PHI (Qi et al. 2024), and AdvBench (Zou et al. 2023). As shown in Table 4, LfU consistently demonstrates the highest robustness after adversarial training. Notably, compared to the second most robust method, LfU reduces ASR by 18.4% on HEx-PHI and 27.0% on PureBad. Furthermore, it achieves near-zero ASR on AdvBench, highlighting its strong robustness against a few steps of adversarial fine-tuning.

Robustness to input noise To evaluate resilience to input noise, we measure cosine similarity between internal

λ	α	Layer	Math (3)	Knowl. (4)	Reason. (2)	Helpful. (2)
5.0	0.1	All	54.2	73.5	65.6	45.7
0.0	-	-	51.5	73.3	65.1	44.1
(a) Varying hyperparameters						
0.1	0.1	All	48.1	73.2	65.4	44.2
1.0	0.1	All	48.1	73.4	65.4	44.2
10.0	0.1	All	53.4	73.0	65.0	44.4
5.0	0.001	All	52.5	73.1	65.1	44.0
5.0	0.01	All	53.0	73.1	65.2	44.3
5.0	0.5	All	44.5	60.4	54.1	43.4
(b) w/ Layer selection						
5.0	0.1	Early	53.6	73.3	65.5	44.4
5.0	0.1	Middle	53.3	73.3	65.6	44.1
5.0	0.1	Late	53.6	72.6	64.3	45.7

Table 5: Ablation study on Llama-3.1-8B fine-tuned on GSM8k, evaluated across 11 tasks spanning four categories.

representations of clean and noisy inputs across all layers using GSM8k samples. We fine-tune Llama-3.1-8B on GSM8k and add noise to the input embeddings with the same strength as in NEFTune. As shown in Figure 6, LfU maintains higher cosine similarity across layers than all baselines, even surpassing NEFTune despite using the same noise level. This suggests that leveraging corrupted representations offers greater internal stability than using noise.

4.3 Ablation study

In Table 5, we conduct an ablation study to better understand the impact of representation consistency. Specifically, we investigate (a) how varying hyperparameters affect performance and (b) how different layer selections influence the effectiveness of consistency regularization.

Effect of hyperparameters We first analyze the individual effects of hyperparameters of LfU in Table 5(a). As λ increases, we observe an improvement in in-domain performance, *i.e.*, Math, while the performance on out-of-domain tasks slightly decreases. This suggests that there is a trade-off between in-domain alignment and generalization, and the current choice of $\lambda = 5.0$ yields a balance. Regarding α, we observe that an excessive local update, *e.g.*, $\alpha = 0.5$, can severely distort the model, making the resulting undesirable representations less meaningful. However, as long as the model is not overly distorted, such updates can still improve performance in out-of-domain tasks. More results across α and λ values can be found in Appendix F.

Layer selection We also investigate several specific layer selections when computing the proposed consistency loss in Table 5(b), beyond our default design of choosing all layers. We group the Llama-3.1-8B layers into Early (1–11), Middle (12–22), and Late (23–32) stages, and apply the consistency loss to each group individually. All settings provide gains and applying the regularization across all layers yields the best performance. This implies that representation-level regularization benefits from being distributed throughout the entire model, rather than focusing only on a subset of layers.

Alternative loss designs To further demonstrate the effectiveness of representation-level consistency, we compare

Method	Math	Knowl.	Reason.	Helpful.
(a) $\ell_{\text{SFT}}(\boldsymbol{\theta})$	51.5	<u>73.3</u>	65.1	44.1
(b) $\ell_{\text{SFT}}\left(\boldsymbol{\theta} + \alpha \cdot \frac{\nabla_{\boldsymbol{\theta}} \ell_{\text{SFT}}}{\|\nabla_{\boldsymbol{\theta}} \ell_{\text{SFT}}\|_2}\right)$	53.3	<u>73.3</u>	<u>65.5</u>	44.0
(c) $\ell_{\text{SFT}}(\boldsymbol{\theta}) + \lambda \cdot \sum_{t=1}^{T} \sum_{l=1}^{K} \|\boldsymbol{h}_{l,t}\|_1$	53.1	72.2	63.6	43.4
(d) $\ell_{\text{SFT}}(\boldsymbol{\theta}) + \lambda \cdot \sum_{t=1}^{T} \sum_{l=1}^{K} \|\boldsymbol{h}_{l,t}\|_2$	53.6	73.0	<u>65.5</u>	43.3
(e) $\ell_{\text{SFT}}(\boldsymbol{\theta}) + \lambda \cdot \ell_{\text{logit.}}(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{aux}})$	<u>53.9</u>	73.2	65.2	<u>44.9</u>
(f) $\ell_{\text{SFT}}(\boldsymbol{\theta}) + \lambda \cdot \ell_{\text{cons.}}(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{aux}})$	**54.2**	**73.5**	**65.6**	**45.7**

Table 6: Variation of update dynamics: Llama-3.1-8B fine-tuned on GSM8k, evaluated across 11 tasks spanning four categories: Math (3 tasks), Knowledge (4 tasks), Reasoning (2 tasks), and Helpfulness (2 tasks).

LfU with alternative loss designs: (a) the vanilla SFT, (b) directly updating $\boldsymbol{\theta}$ for local updates, which is essentially equivalent to SAM (Foret et al. 2020), (c) feature-level L1 consistency, (d) feature-level L2 consistency and (e) logit-level consistency instead of the representation-level design of LfU, defined as follows:

$$\ell_{\text{logit.}}(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{aux}}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}} \left[\frac{1}{T} \sum_{t=1}^{T} \text{KL}\left(\sigma\left(\text{detach}(\boldsymbol{z}_t)\right) \parallel \sigma(\boldsymbol{z}'_t)\right) \right], \quad (7)$$

where $\boldsymbol{z}_t$ and $\boldsymbol{z}'_t$ denote the logits from the original and undesirable models, respectively, σ denotes the softmax function, and KL is the Kullback–Leibler divergence.

The results are shown in Table 6. We observe that enforcing consistency at representation-level clearly obtains better generalization than directly optimizing the parameters, and simple L1/L2 penalties control only feature magnitudes, failing to preserve the semantic properties of representations. The logit-level consistency also have gains over SFT; but our design consistently outperforms it, suggesting that consistency in internal representations plays a more critical role than consistency in logits. More results on these variations, as well as comparisons with parameter-level averaging approaches including Exponential Moving Average (EMA), are provided in Appendix F, where our representation-level consistency still yields the best performance.

5 Conclusion

In this work, we introduced Learning-from-the-Undesirable (LfU), a method designed to mitigate overfitting in language models by stabilizing internal representations. By simulating undesirable updates and enforcing representation-level consistency, LfU improves generalization by reducing overfitting. Extensive experiments show that LfU consistently outperforms prior fine-tuning methods on in-domain tasks, out-of-domain generalization, resilience to prompt variations, and robustness against adversarial fine-tuning. Furthermore, LfU is broadly compatible with various models and performs well in both single-task and multi-task fine-tuning settings. To reduce computational overhead, we also introduce a lightweight variant based on representation steering. Overall, LfU offers a practical and broadly applicable solution for adapting language models to downstream tasks while preserving their general capabilities.

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University); No. IITP-2025-RS-2025-02304828, Artificial Intelligence Star Fellowship Support Program to Nurture the Best Talents; No. IITP-2025-RS-2024-00436857, Information Technology Research Center (ITRC)), and the Korea Creative Content Agency grant funded by the Ministry of Culture, Sports and Tourism (No. RS-2025-00345025).

References

Amini, A.; Gabriel, S.; Lin, P.; Koncel-Kedziorski, R.; et al. 2019. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms.

Anthropic. 2023. Introducing Claude. https://www.anthropic.com/index/introducing-claude.

Ba, Y.; Mancenido, M. V.; and Pan, R. 2024. Fill in the gaps: Model calibration and generalization with synthetic data.

Bisk, Y.; Zellers, R.; Bras, R. L.; Gao, J.; and Choi, Y. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. In *AAAI*.

Chen, L.; Li, S.; Yan, J.; Wang, H.; Gunaratna, K.; Yadav, V.; Tang, Z.; Srinivasan, V.; Zhou, T.; Huang, H.; et al. 2023. Alpagasus: Training a better alpaca with fewer data.

Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafjord, O. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

Conover, M.; Hayes, M.; Mathur, A.; Xie, J.; Wan, J.; Shah, S.; et al. 2023. Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM.

Fang, X.; Che, S.; Mao, M.; Zhang, H.; et al. 2024. Bias of AI-generated content: an examination of news produced by large language models. *Scientific Reports*, 14(1): 5224.

Feder, A.; Wald, Y.; Shi, C.; Saria, S.; and Blei, D. 2023. Data augmentations for improved (large) language model generalization. *NeurIPS*, 36: 70638–70653.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 1126–1135. PMLR.

Foret, P.; Kleiner, A.; Mobahi, H.; and Neyshabur, B. 2020. Sharpness-aware minimization for efficiently improving generalization.

Gekhman, Z.; Yona, G.; Aharoni, R.; Eyal, M.; Feder, A.; Reichart, R.; and Herzig, J. 2024. Does fine-tuning LLMs on new knowledge encourage hallucinations?

Gemini Team. 2023. Gemini: A Family of Highly Capable Multimodal Models.

Guo, Y.; Shang, G.; Vazirgiannis, M.; and Clavel, C. 2023. The curious decline of linguistic diversity: Training language models on synthetic text.

Hartvigsen, T.; Gabriel, S.; Palangi, H.; Sap, M.; Ray, D.; and Kamar, E. 2022. ToxiGen: A Large-Scale Machine-Generated Dataset for Implicit and Adversarial Hate Speech Detection. In *ACL*.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. *ICLR*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2): 3.

Huang, T.; Hu, S.; Ilhan, F.; Tekin, S. F.; and Liu, L. 2024. Booster: Tackling Harmful Fine-tuning for Large Language Models via Attenuating Harmful Perturbation.

Inan, H.; Upasani, K.; Chi, J.; Rungta, R.; Iyer, K.; Mao, Y.; Tontchev, M.; Hu, Q.; Fuller, B.; Testuggine, D.; et al. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations.

Jain, N.; Chiang, P.-y.; Wen, Y.; Kirchenbauer, J.; Chu, H.-M.; Somepalli, G.; Bartoldson, B. R.; Kailkhura, B.; Schwarzschild, A.; Saha, A.; et al. 2023. Neftune: Noisy embeddings improve instruction finetuning.

Jeong, J.; and Shin, J. 2020. Consistency regularization for certified robustness of smoothed classifiers. *NeurIPS*, 33: 10558–10570.

Ji, J.; Liu, M.; Dai, J.; Pan, X.; Zhang, C.; Bian, C.; Chen, B.; Sun, R.; Wang, Y.; and Yang, Y. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *NeurIPS*, 36: 24678–24704.

Jiang, G.; Jiang, C.; Li, Z.; Xue, S.; Zhou, J.; Song, L.; Lian, D.; and Wei, Y. 2025. Unlocking the power of function vectors for characterizing and mitigating catastrophic forgetting in continual instruction tuning.

Kirk, R.; Mediratta, I.; Nalmpantis, C.; Luketina, J.; Hambro, E.; et al. 2023. Understanding the effects of RLHF on LLM generalisation and diversity.

Kotha, S.; Springer, J. M.; and Raghunathan, A. 2023. Understanding catastrophic forgetting in language models via implicit inference.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *NeurIPS*, 33: 9459–9474.

Li, Z.; Chen, C.; Xu, T.; Qin, Z.; Xiao, J.; Luo, Z.-Q.; and Sun, R. 2025. Preserving diversity in supervised fine-tuning of large language models. In *ICLR*.

Li, Z.; Zhou, F.; Chen, F.; and Li, H. 2017. Meta-sgd: Learning to learn quickly for few-shot learning.

Lin, S.; Hilton, J.; and Evans, O. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In *ACL*, 3214–3252. Dublin, Ireland.

Lin, S.-C.; Gao, L.; Oguz, B.; Xiong, W.; Lin, J.; Yih, W.-t.; and Chen, X. 2024. Flame: Factuality-aware alignment for large language models.

Luo, Y.; Yang, Z.; Meng, F.; Li, Y.; Zhou, J.; and Zhang, Y. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning.

Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; et al. 2020. Language models are few-shot learners.

Miao, S.-Y.; Liang, C.-C.; and Su, K.-Y. 2021. A Diverse Corpus for Evaluating and Developing English Math Word Problem Solvers. arXiv:2106.15772.

Ni, Y.; Wen, S.; Koniusz, P.; and Cherian, A. 2025. Noise Consistency Regularization for Improved Subject-Driven Image Synthesis. In *CVPR Workshop*, 3116–3126.

Oikonomou, D.; and Loizou, N. 2025. Sharpness-Aware Minimization: General Analysis and Improved Rates.

OpenAI. 2022. Introducing ChatGPT. https://openai.com/blog/chatgpt.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

OpenAI. 2024. GPT-4o System Card. arXiv:2410.21276.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *NeurIPS*, 35: 27730–27744.

Paperno, D.; Kruszewski, G.; Lazaridou, A.; Pham, Q. N.; Bernardi, R.; et al. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context.

Patil, S. G.; Zhang, T.; Wang, X.; and Gonzalez, J. E. 2024. Gorilla: Large language model connected with massive apis. *NeurIPS*, 37: 126544–126565.

Qi, X.; Panda, A.; Lyu, K.; Ma, X.; Roy, S.; Beirami, A.; Mittal, P.; and Henderson, P. 2024. Safety alignment should be made more than just a few tokens deep.

Qi, X.; Zeng, Y.; Xie, T.; Chen, P.-Y.; Jia, R.; Mittal, P.; and Henderson, P. 2023. Fine-tuning aligned language models compromises safety, even when users do not intend to!

Qwen Team. 2025. Qwen3 Technical Report.

Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C. D.; Ermon, S.; and Finn, C. 2023. Direct preference optimization: Your language model is secretly a reward model. *NeurIPS*, 36: 53728–53741.

Rajeswaran, A.; Finn, C.; Kakade, S. M.; and Levine, S. 2019. Meta-learning with implicit gradients. *NeurIPS*, 32.

Reddy, S.; Chen, D.; and Manning, C. D. 2019. Coqa: A conversational question answering challenge. *TACL*, 7: 249–266.

Reynolds, L.; and McDonell, K. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *CHI*, 1–7.

Rogulsky, S.; Popovic, N.; and Färber, M. 2024. The Effects of Hallucinations in Synthetic Training Data for Relation Extraction.

Roziere, B.; Gehring, J.; Gloeckle, F.; Sootla, S.; Gat, I.; Tan, X. E.; Adi, Y.; Liu, J.; Sauvestre, R.; Remez, T.; et al. 2023. Code llama: Open foundation models for code.

Sajjadi, M.; Javanmardi, M.; and Tasdizen, T. 2016. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *NeurIPS*, 29.

Shi, Z.; Yang, A.; Wu, B.; Aitchison, L.; Yilmaz, E.; and Lipani, A. 2024. Instruction tuning with loss over instructions. *NeurIPS*, 37: 69176–69205.

Shumailov, I.; Shumaylov, Z.; Zhao, Y.; Papernot, N.; Anderson, R.; and Gal, Y. 2024. AI models collapse when trained on recursively generated data. *Nature*, 631(8022): 755–759.

Sohn, K.; Berthelot, D.; Carlini, N.; Zhang, Z.; Zhang, H.; et al. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *NeurIPS*, 33: 596–608.

Tack, J.; Yu, S.; Jeong, J.; Kim, M.; Hwang, S. J.; and Shin, J. 2022. Consistency regularization for adversarial robustness. In *AAAI*, volume 36, 8414–8422.

Tahmasebi, B.; Soleymani, A.; Bahri, D.; Jegelka, S.; and Jaillet, P. 2024. A universal class of sharpness-aware minimization algorithms.

Tang, L.; Li, K.; He, C.; Zhang, Y.; and Li, X. 2023. Consistency regularization for generalizable source-free domain adaptation. In *ICCV workshop*, 4323–4333.

Thrun, S.; and Pratt, L. 1998. Learning to learn: Introduction and overview. In *Learning to learn*, 3–17. Springer.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models.

Wang, L.; Xu, L.; Yang, X.; Huang, Z.; and Cheng, J. 2025. Debiased distillation for consistency regularization. In *AAAI*, volume 39, 7799–7807.

Wei, J.; and Zou, K. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks.

Xie, Q.; Dai, Z.; Hovy, E.; Luong, T.; and Le, Q. 2020. Unsupervised data augmentation for consistency training. *NeurIPS*, 33: 6256–6268.

Yadav, A. K.; and Singh, A. 2023. SymNoise: Advancing Language Model Fine-tuning with Symmetric Noise.

Yang, Z.; Pang, T.; Feng, H.; Wang, H.; Chen, W.; Zhu, M.; and Liu, Q. 2024. Self-distillation bridges distribution gap in language model fine-tuning.

Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *ACL*.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E.; El Ghaoui, L.; and Jordan, M. 2019. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 7472–7482.

Zhang, H.; Zhang, Z.; Odena, A.; and Lee, H. 2020. Consistency Regularization for Generative Adversarial Networks. In *ICLR*.

Zhou, C.; Liu, P.; Xu, P.; Iyer, S.; Sun, J.; Mao, Y.; Ma, X.; Efrat, A.; Yu, P.; Yu, L.; et al. 2023. Lima: Less is more for alignment. *NeurIPS*, 36: 55006–55021.

Zhu, K.; Wang, J.; Zhou, J.; Wang, Z.; Chen, H.; et al. 2023. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts.

Zou, A.; Wang, Z.; Carlini, N.; Nasr, M.; Kolter, J. Z.; and Fredrikson, M. 2023. Universal and transferable adversarial attacks on aligned language models.