Look Within or Beyond? A Theoretical Comparison Between Parameter-Efficient and Full Fine-Tuning

Anonymous ACL submission

Abstract

Parameter-**E**fficient **F**ine-**T**uning (**PEFT**) has become a popular alternative to **F**ull-Parameter **F**ine-**T**uning (**FFT**), achieving similar performance on many benchmarks with far lower computational and memory costs. Yet, its effectiveness on complex tasks such as reasoning and instruction-following remains unclear. In this work, we provide a theoretical and empirical comparison of PEFT and FFT in terms of representational capacity and robustness. We show that PEFT's solution space is a strict subset of FFT's and derive upper bounds revealing how its restricted parameterization limits expressiveness and increases vulnerability to perturbations. Experiments on 20 datasets and 11 adversarial test sets support these findings, indicating that while PEFT performs well on standard tasks, its weaknesses on complex and adversarial settings call for new directions beyond current PEFT paradigms.The source code is in the anonymous GitHub repository[1].

1 Introduction

Full Parameter **F**ine-**T**uning (**FFT**) has long been the go-to choice for fine-tuning language models (Vaswani et al., 2017; Lewis et al., 2020; Liu et al., 2020b, 2024). However, as large language models (LLMs) continue to grow in size (Zeng et al., 2022; Chiang et al., 2023; Lin et al., 2024), the GPU memory and computational demands for FFT have become increasingly prohibitive. To address these challenges, **P**arameter-**E**fficient **F**ine-**T**uning (**PEFT**) methods have been developed as alternatives that reduce the number of trainable parameters while maintaining performance comparable to FFT (Lialin et al., 2023). The addition-based PEFT methods (e.g., Prefix-Tuning (Li and Liang, 2021), AttentionFusion (Cao et al., 2022)) reduce the number of trainable parameters by updating only newly added parameters while freezing the weights of LLMs. Selection-based methods (e.g., BitFit (Zaken et al., 2022), LT-SFT (Ansell et al., 2022), FAR (Vucetic et al., 2022)) update only a subset of model parameters. Reparameterization-based methods (e.g., LoRA (Hu et al., 2022), KronA (Edalati et al., 2022), S4-model (Chen et al., 2023)) reduce the number of trainable parameters by employing low-rank approximations instead of full weight matrices in learned weight updates.

These PEFT methods mitigate the high computational costs of fine-tuning by updating only a small fraction of model parameters (Han et al., 2024). However, optimization theory indicates that models with greater capacity, often achieved through overparameterization, tend to exhibit stronger representational capabilities and robustness (Kohavi and Wolpert, 1996; Dar et al., 2021; Neal et al., 2018). This implies that restricting the number of trainable parameters during fine-tuning may compromise a model's capacity and robustness. Existing studies have shown that PEFT approaches can perform less effectively than FFT on complex tasks (Raschka, 2023; Artur et al., 2023; Kourosh and Rehaan, 2023). Given the increasing adoption of PEFT in the community, it is crucial to systematically investigate its limitations and broader implications.

Preliminary research reveals that the incremental weight distribution in PEFT is steeper than that of FFT, which exhibits a comparatively flatter profile. According to the sharpness-flatness theory (Foret et al., 2020), flatter parameter distributions are associated with greater robustness of a model, whereas steeper distributions indicate increased sensitivity to perturbations. Furthermore, the greater degrees of freedom in FFT endow it with stronger representational ability than PEFT, with limited capacity. Building on these insights, we theoretically establish that PEFT forms a subset of FFT and derive the upper bound that reveals how the constrained parameter space restricts the model's representational

[1] https://anonymous.4open.science/r/PEFTEval-E2AC

ability and robustness. We validate our theoretical findings through extensive experiments on 20 datasets spanning classification, generation, reasoning, and instruction-following tasks, as well as 11 adversarial test sets. Our main contributions and conclusions are summarized as follows:

- We prove that PEFT is a strict subset of FFT, constituting a low-dimensional, measure-zero manifold within the FFT.
- We establish an explicit upper bound on PEFT's representational capacity, showing that the number of trainable parameters directly limits the model's ability to adapt and learn new knowledge.
- We demonstrate that PEFT's incremental weights suffer from severe feature decay, where a small set of parameters captures most label features and additional parameters yield diminishing returns.
- We show that PEFT's constrained parameter space inherently caps its data-driven gains, resulting in lower marginal improvements from additional training samples compared to FFT.
- We provide both empirical and theoretical evidence that PEFT's incremental weight distribution is steeper and more sharply peaked, making it more susceptible to perturbations than FFT.

2 Related Work

Parameter-Efficient Fine-tuning PEFT adapts a language model (LM) to downstream tasks by freezing most of the weights and only updating a small set of internal or additional parameters, thereby minimizing resource utilization (Liu et al., 2022a; Ding et al., 2023). PEFT paradigms can be classified as addition-based, selection-based, or reparameterization-based methods (Lialin et al., 2023). Addition-based methods introduce and update new parameters while keeping the weights of language models frozen. Examples include Prefix-Tuning (Li and Liang, 2021; Zhang et al., 2023b), AttentionFusion (Cao et al., 2022), and adapters (Hu et al., 2023). Selection-based methods fine-tune a subset of the existing parameters of LMs, such as BitFit (Zaken et al., 2022), LT-SFT (Ansell et al., 2022), and FAR (Vucetic et al., 2022). Reparameterization-based methods use low-rank decomposition to minimize the number of trainable parameters, such as LoRA (Hu et al., 2022), PHM (Karimi Mahabadi et al., 2021), KronA (Edalati et al., 2022), S4-model (Chen et al., 2023), and PERU-LoRA (Jiang et al., 2023b). PEFT reduces memory usage during fine-tuning by decreasing the number of tunable parameters. However, this can potentially diminish the model's representational ability, which may negatively impact its performance and robustness.

Robustness Evaluation It is insufficient to measure the quality of a model solely based on its performance metrics. Existing literature has shown that many high-performance models, including LLMs, are vulnerable to carefully crafted adversarial examples (Wang et al., 2023b; Yuan et al., 2023; Wang et al., 2023a), which can deceive the models into producing arbitrarily incorrect answers by subtly perturbing inputs in ways imperceptible to humans. Artificial general intelligence systems built upon these vulnerable models can be easily misled, leading to significant security issues (Wang et al., 2021a, 2023b; Zhu et al., 2023). Researchers have made several attempts to reveal the robustness of models (Goyal et al., 2023). Wang et al. (2021a) propose Adversarial GLUE (AdvGLUE) to quantitatively and thoroughly explore and evaluate the vulnerabilities of language models under various types of adversarial attacks. Jia and Liang (2017a) and Liu et al. (2020a) provide adversarial SQuAD to evaluate the robustness of reading comprehension systems from different dimensions. In addition to adversarial datasets, integrated attack tools that include word, sentence, and multi-level adversarial attacks play a key role in assessing model robustness (Zeng et al., 2021). OpenAttack (Zeng et al., 2021), TextAttack (Morris et al., 2020), TextFlint (Wang et al., 2021b), and Robustness Gym (Goel et al., 2021) integrate various ad hoc input transformations for different tasks and provide programmable APIs to dynamically test model performance. Wang et al. (2023a) conduct a thorough evaluation of the robustness of ChatGPT from an adversarial and out-of-distribution (Yuan et al., 2023) perspective. In summary, much prior work has focused on evaluating robustness. However, the robustness of PEFT methods remains unexplored.

3 Birds-Eye View of Fine-Tuning

Full Parameter Fine-Tuning refers to updating all parameters in full rank. In full-rank optimization, local critical points are typically saddle points rather than valleys (Dauphin et al., 2014). Saddle points are characterized by having exits in multiple dimensions, which means that models are more

likely to escape from them and converge towards the global optimal solution (Dauphin et al., 2014). Generally, once the dataset and network architecture are specified, the optimization space of the model remains fixed. Subsequent parameter updates by an optimizer explore the details within this frozen space (Li et al., 2018).

We represent the foundational model as a function $f(x, \theta)$, where $\theta \in \Theta = \mathbb{R}^d$, and d denotes the spatial dimension. Given a data D, the purpose of fine-tuning is to identify a parameter θ' that satisfies the following equation:

$$\theta' = \arg\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim D}[\mathcal{L}(f(x;\theta), y)] \tag{1}$$

where $\mathcal{L}$ is the loss function. The model of FFT can be defined as a set of functions:

$$\mathcal{F}_{full} = f(x;\theta) : \theta \in \Theta \tag{2}$$

Its optimization space is $\mathbb{R}^d$.

Parameter-Efficient Fine-Tuning refers to the process of fine-tuning only a subset of parameters Φ (newly introduced or part of the base model) to adapt to downstream tasks. While reducing the number of trainable parameters during fine-tuning can decrease memory usage, it also limits the model's capacity for representation. Compared to FFT, PEFT updates occur in a relatively lower-dimensional space. This implies that once the model encounters a saddle point during fine-tuning, it may be more challenging to escape. However, this does not necessarily indicate that PEFT is more prone to local optima, as high-dimensional space may harbor more saddle points than low-dimensional space.

The model of PEFT can be expressed as $f(x; \theta_0; \Phi)$, where θ_0 is frozen and represents the initial state of the foundational model. For any PEFT method, we assume that the spatial dimension of incremental parameter Φ is k, where $k \ll d$. The model of PEFT can be defined as a set of functions:

$$\mathcal{F}_{peft} = f(x; \theta_0; \Phi) : \Phi \in \mathbb{R}^k \subset \Theta \tag{3}$$

Fundamentally, PEFT can be viewed as a parameter reparameterization mechanism applied to the model, namely:

$$f(x; \theta_0; \Phi) = f(x; \theta_0 + g(\Phi)) \tag{4}$$

where $g : \mathbb{R}^k \to \mathbb{R}^d$ is a mapping from a low-dimensional space to a high-dimensional space, typically with a linear or sparse structure, e.g., LoRA (Hu et al., 2022) and Adapter (Hu et al., 2023).

4 Theoretical Analysis

While empirical results from prior research have shown that the limited parameter space in PEFT constrains the model's performance (Huang et al., 2025; He, 2024; Qiao and Mahdavi, 2024), this section focuses on the theoretical perspective of why FFT is more reliable than PEFT in terms of model representation capacity and robustness. First, we show that the fine-tuning space of PEFT is an embedded submanifold of the parameter space of FFT, and prove that the PEFT fine-tuning subspace is a strict subset of that of FFT (Theorem 1). At the same time, we present an upper bound on the representation capacity of PEFT fine-tuning, which implies that the number of fine-tuning parameters in PEFT directly constrains the maximum extent of model adaptation (Theorem 2). Then, we demonstrate that there exists an optimal number of parameters for PEFT fine-tuning, beyond which increasing the number of tunable parameters yields diminishing returns (Theorem 3). From a data perspective, we show that the limited parameter space of PEFT constrains its ability to benefit from additional data, resulting in smaller marginal gains from sample increases compared to FFT (Theorem 5). From a robustness perspective, our theory indicates that PEFT is more sensitive to perturbations compared to FFT (Theorem 5).

Theorem 1. (Subset PEFT of FFT) *According to equation 4, we define $\theta_\Phi := \theta_0 + g(\Phi) \in \mathbb{R}^d$, where g is a non-surjective function (proof in the Section E.1). The following conclusions can be drawn:*

$$f(x; \theta_0; \Phi) = f(x; \theta_\Phi) \Rightarrow f(x; \theta_0; \Phi) \subset \mathcal{F}_{full} \tag{5}$$

That is to say, $\forall \Phi \in \mathbb{R}^k$, $\exists \theta_\Phi \in \mathbb{R}^d$ such that $f(x; \theta_0; \Phi) = f(x; \theta_\Phi) \subset \mathcal{F}_{full}$

Theorem 2. (PEFT Capacity Upper Bound) *Assume that the upper bounds of the weight norm and activation Lipschitz constant of each layer are M and L respectively, that is, $\|W_{0,k}\| \leq M$, $L_k \leq L$, $\forall k$. At the same time, let $\alpha_k \triangleq L\|\Phi_k\|$, for input $x \in D$, we have (proof in the Section E.2):*

$$\|f(x) - f_0(x)\| \leq \sum_{k=1}^{N} M^{N-k} L^{N-k} \alpha_k \|x\| \tag{6}$$

This theoretical framework demonstrates that the number of fine-tuning parameters in PEFT directly constrains the maximum extent of model adaptation, with this relationship quantified through the term α_k. We observe that the $M^{N-k}L^{N-k}$ term in the formula indicates that PEFT parameters across different layers have unequal effects on the final output. Specifically, parameters closer to the output layer (i.e., with larger k values) have a greater impact, as the exponent $N-k$ becomes smaller.

Theorem 3. (Rule of Diminishing Marginal Benefit) *In PEFT fine-tuning, there exists a critical parameter Φ_c such that when $Rank(\Phi) \geq r_c$ ($Rank(\cdot)$ is the rank function.), further increasing the parameter amount of Φ does not lead to significant improvements in the performance of the optimal solution to the optimization problem. Assume N is the number of training samples, $r \geq r_c$, the following conclusions can be drawn (proof in the Section E.3):*

$$\mathcal{L}_{\mathcal{D}_{test}}(W+\Delta W_r) \leq \mathcal{L}_{\mathcal{D}_{test}}(W+\Delta W^*) + L\sqrt{\epsilon} + O(\sqrt{\frac{r}{N}}) \tag{7}$$

where ΔW^* is the ideal weight update. As r continues to increase, the first term remains unchanged, the second term approaches zero, but the third term (generalization error) increases, leading to a plateau or even a decline in overall performance.

Together, experiments show that once the critical number of parameters is reached, the model performance remains stable even if the parameters continue to increase.

Theorem 4. (PEFT More Sensitive to Disturbances) *Assume the disturbance factor is ϵ, the loss fluctuations induced by the perturbation factor ϵ in PEFT and FFT fine-tuning are as follows (proof in the Section E.4):*

$$\begin{aligned} \Delta\mathcal{L}_{full} &\approx \frac{1}{2}\Delta\theta^T H \Delta\theta \\ \Delta\mathcal{L}_{peft} &\approx \frac{1}{2}\epsilon_{\parallel}^T H_{\Phi}^{+} \epsilon_{\parallel} + \epsilon_{\perp}^T \nabla_{\theta}\mathcal{L} \end{aligned} \tag{8}$$

where H is the Hessian Matrix, H_{Φ}^{+} is the pseudo-inverse of $P^T H P$, $\epsilon_{\parallel}$ is the projection of the perturbation in the PEFT subspace, $\epsilon_{\perp}$ is the component orthogonal to this subspace. We have $\Delta\mathcal{L}_{peft} - \Delta\mathcal{L}_{full} > 0$.

Results across multiple adversarial datasets show that PEFT is more sensitive to perturbations than FFT.

Theorem 5. (Marginal Benefit of Examples) *The limited parameter space of PEFT constrains its data-driven gains, resulting in smaller benefits from increased data compared to FFT(proof in the SectionSection E.5.).*

Models	Mehtods	ViGGO	SQL Generation	GSM8k	AVG
LLaMA2-7B	Vanilla	0.93(2.5)	3.50(3.7)	14.00(4.7)	6.14
	LoRA$^{r=8}$	92.05(1.5)	85.93(1.8)	22.87(3.5)	66.95
	HiFT*	**94.88**	**87.15**	29.85	**70.63**
	FFT	94.86(1.3)	86.6(1.5)	**30.00**(2.9)	70.49
LLaMA2-13B	Vanilla	2.34(4.5)	22.20(5.5)	28.00(3.8)	17.51
	LoRA$^{r=8}$	95.32(2.7)	87.94(3.1)	35.94(2.5)	73.07
	HiFT*	95.66	**90.33**	**48.01**	**78.00**
	FFT	**95.79**(2.6)	89.20(2.5)	47.00(1.9)	77.33
LLaMA2-70B	Vanilla	5.96(2.9)	26.65(3.5)	58.00(3.8)	30.20
	LoRA$^{r=8}$	95.60(1.5)	90.34(2.1)	60.96(2.5)	82.30
	HiFT*	-	-	-	-
	FFT	**97.60**(2.1)	**92.84**(1.9)	**62.00**(2.2)	**84.15**

Table 1: Performance comparison of different fine-tuning methods based on LLaMA2 family, where HiFT and FFT are full-parameter fine-tuning. * indicates results from the original paper. All reported numbers are averaged metrics (standard deviation).

5 Experiments

We conduct comprehensive experiments on both medium-sized masked language models (RoBERTa-base, RoBERTa-large (Liu et al., 2019), and TinyLLaMA (Touvron et al., 2023)) and large autoregressive language models (Mistral-7B (Jiang et al., 2023a), OPT-13B (Zhang et al., 2022), LLaMA2-7B, LLaMA2-13B, and LLaMA2-70B (Touvron et al., 2023)). First, we show that FFT has performance advantages over PEFT in complex tasks such as math reasoning, SQL generation, instruction following, and functional representation. Moreover, FFT performs more robustly on adversarial datasets than PEFT. Further experiments demonstrate that the marginal benefit of FFT for increasing data and parameters is greater than that of PEFT.

Section A reports all used language models and fine-tuning methods. All experiments below use datasets detailed in Section B. All fine-tuning experiments with backpropagation use the Adam Optimizer. Detailed experimental setup and hyperparameters are reported in Section C. Prompts used in the experiment are detailed in Section D.

5.1 Performance Comparison Between FFT and PEFT

We compare the performance of different fine-tuning methods on multiple datasets across various tasks. Table 1 presents the performance comparison of LLaMA2-7B, LLaMA2-13B, and LLaMA2-70B using different fine-tuning methods. Across

Models	Methods	Writing	Roleplay	Reasoning	Math	Extraction	Stem	Humanities	AVG
TinyLLaMA	Vanilla	1.06	2.25	1.17	1.05	1.10	1.50	1.00	1.30
	LoRA	2.80	4.00	1.27	**1.45**	1.05	1.55	2.20	2.05
	Prefix	2.75	3.50	1.20	1.35	1.10	1.45	1.35	1.81
	HiFT	3.50	**4.45**	**2.50**	1.40	1.70	**3.15**	3.20	**2.84**
	FFT	**3.30**	3.85	2.35	1.35	**1.77**	2.80	**3.35**	2.68
Mistral-7B	Vanilla	5.30	3.25	4.55	2.60	**6.55**	**6.20**	4.60	4.72
	LoRA	5.45	4.45	4.60	3.25	5.55	5.50	4.35	4.74
	Prefix	5.35	4.30	4.50	3.15	5.45	5.55	4.40	4.67
	HiFT	**6.45**	5.40	5.45	3.05	5.85	6.05	**6.15**	**5.49**
	FFT	5.55	**4.50**	**5.40**	**3.35**	5.80	4.65	5.50	4.96
LLaMA2-7B	Vanilla	3.05	4.45	2.90	1.75	3.35	5.25	4.50	3.61
	LoRA	6.20	5.60	**4.15**	1.75	4.20	6.30	6.15	4.91
	Prefix	6.35	5.45	3.70	1.40	4.50	6.15	6.20	4.82
	HiFT	**6.70**	**7.15**	3.55	**2.20**	4.55	**6.85**	**7.85**	**5.55**
	FFT	5.50	6.55	3.65	**2.20**	**4.75**	6.55	7.65	5.26

Table 2: Comparison of instruction-following performance on mt-bench with different fine-tuning methods.

Models	SST-2		QQP		MNLI		MNLI-MM		QNLI		RTE		Average	
	Origin	Adv	Origin	Adv	Origin	Adv	Origin	Adv	Origin	Adv	Origin	Adv	Origin	Adv
RoBbase(FFT)	94.80(0.5)	34.62(0.7)	**91.90(0.5)**	39.48(1.5)	87.60(1.5)	**31.48**(1.7)	86.58(0.8)	**30.04**(1.3)	92.80(0.5)	42.12(1.5)	78.70(0.8)	**41.75**(1.9)	88.73	**36.58**
RoBbase(LoRA$^{r=8}$)	**95.10**(1.3)	39.18(1.7)	90.80(2.5)	39.74(2.3)	87.50(1.3)	27.27(1.8)	86.90(0.6)	19.75(0.9)	**93.30**(0.7)	37.16(2.3)	**86.60**(1.2)	22.22(2.5)	**90.03**	30.89
RoBbase(AdaLoRA)	94.49(1.5)	39.18(1.9)	91.54(1.7)	34.61(2.0)	**87.96**(1.7)	19.83(2.0)	**87.38**(1.3)	17.90(0.8)	93.11(1.2)	41.89(1.7)	83.75(1.5)	27.16(2.0)	89.71	30.10
RoBbase(Prefix-Tuning)	94.95(1.8)	**43.91**(2.3)	89.37(0.9)	42.30(1.5)	86.75(2.1)	23.14(1.7)	86.33(0.9)	24.69(1.7)	93.04(1.5)	**47.97**(2.8)	70.75(1.7)	25.92(3.1)	86.87	34.66
RoBbase(IA3)	93.57(2.1)	26.35(1.5)	89.05(1.3)	**43.58**(1.7)	83.97(1.9)	31.40(2.5)	82.54(1.3)	27.20(1.9)	90.73(1.8)	36.48(1.3)	76.17(0.9)	25.92(3.5)	86.01	31.82
RoBbase((BitFit)	93.81(1.5)	19.59(1.9)	87.81(1.8)	29.48(1.6)	84.14(1.7)	30.57(1.5)	84.54(1.4)	22.83(1.5)	91.47(2.3)	37.16(2.5)	78.70(2.3)	29.62(2.7)	86.75	28.21
RoBlarge(FFT)	96.40(1.0)	58.65(0.9)	**92.20**(0.8)	57.44(0.7)	90.20(1.8)	54.54(2.1)	88.50(1.9)	40.24(1.9)	94.70(1.5)	52.64(1.9)	86.60(1.7)	**69.13**(2.8)	91.43	**55.44**
RoBlarge(LoRA$^{r=8}$)	96.20(0.8)	56.08(1.3)	91.60(1.3)	**58.97**(0.9)	**94.90**(1.3)	51.23(1.5)	**90.16**(1.5)	38.88(2.3)	94.90(2.3)	58.10(2.8)	87.40(0.9)	60.49(1.7)	**92.53**	53.96
RoBlarge(AdaLoRA)	**96.44**(1.4)	**60.81**(2.8)	91.95(1.7)	47.43(0.5)	87.79(1.5)	51.23(2.1)	88.20(1.3)	40.74(3.4)	**94.91**(3.4)	**62.16**(3.3)	87.72(2.5)	65.43(3.1)	91.17	54.63
RoBlarge(Prefix-Tuning)	95.87(1.7)	58.78(2.5)	89.98(2.1)	48.71(1.3)	89.99(0.8)	55.02(1.7)	89.79(0.7)	**42.22**(2.8)	94.87(2.8)	60.48(3.5)	87.72(1.9)	66.66(2.0)	91.37	55.31
RoBlarge(IA3)	95.64(2.1)	49.32(2.7)	84.41(1.7)	42.30(1.5)	89.47(1.1)	**55.37**(0.9)	89.39(1.2)	35.18(2.5)	92.76(1.9)	**62.16**(2.8)	86.64(2.3)	58.02(2.3)	89.72	50.39
RoBlarge((BitFit)	95.64(1.8)	51.35(3.3)	89.47(1.9)	**58.97**(0.7)	89.14(2.5)	41.32(1.5)	88.77(1.3)	35.80(3.8)	93.72(3.3)	56.75(2.1)	**87.73**(2.0)	67.9(2.7)	90.75	52.02

Table 3: Performance comparison of different fine-tuning methods based on $\text{RoBERTa}_{\text{base}}$ and $\text{RoBERTa}_{\text{large}}$ on AdvGLUE dataset.

all LLaMA2 models, standard FFT consistently achieves a performance advantage over LoRA fine-tuning. Specifically, FFT-based LLaMA2-7B shows a 2.81% higher performance than LoRA on ViGGO, 0.67% higher on SQL Generation, and 7.31% higher on GSM8k. Similar trends are observed in LLaMA2-13B and LLaMA2-70B. On average, FFT-based LLaMA2-7B outperforms LoRA by 3.54%, LLaMA2-13B outperforms by 4.26%, and LLaMA2-70B outperforms by 1.85%. As model size increases, FFT consistently maintains its performance advantage over LoRA.

Table 2 presents the instruction fine-tuning results of TinyLLaMA, Mistral-7B, and LLaMA2-7B using different fine-tuning methods. FFT-based TinyLLaMA shows performance advantages over PEFT methods, including LoRA, in 6 out of 8 dimensions. Similar trends are observed for LLaMA2-7B and Mistral-7B. Compared to LoRA, FFT-based TinyLLaMA achieves an average performance improvement of 0.57, Mistral-7B by 0.22, and LLaMA2-7B by 0.33. FFT demonstrates a consistent performance advantage over PEFT in the instruction fine-tuning task. Assuming our fine-tuning is effective, according to Theorem 2, we have:

$$f_0(x) \leq f(x) \leq \sum_{k=1}^{N} M^{N-k} L^{N-k} \alpha_k \|x\| + f_0(x) \quad (9)$$

Since the values of N, M, and α_k in FFT are higher than those in PEFT, FFT demonstrates a stronger upper bound than PEFT. When tackling complex tasks, models often need to learn intricate features to make accurate decisions. The FFT offers a more powerful representation space, allowing the model to better capture and utilize these features.

We also observe that LoRA achieves performance advantages over FFT on the original GLUE dataset, as shown in Table 3, while LoRA performs inferiorly to FFT on SQuAD as indicated in Table 4. It's important to emphasize that PEFT methods, including LoRA, do not consistently work well, particularly on complex tasks. Simple tasks with straightforward formats and clear label characteristics may not necessitate a large number of parameters to effectively learn data features; A reduced parameter count can still capture sufficient label-related information. This is where PEFT demonstrates efficacy in certain benchmarks.

Parameter Distribution Comparison Between FFT and PEFT To further explore the differences between FFT and PEFT, Figure 1 shows the

Models	AS		AA		AAE		AAC		ANA		Average		Origin	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
RoBbase(FFT)	**63.70**(0.7)	**71.09**(0.8)	**76.5**(0.5)	**84.37**(0.7)	**58.63**(1.8)	**73.72**(2.3)	**73.83**(0.9)	80.88(1.2)	**77.64**(2.3)	**86.16**(1.3)	**70.06**	**79.24**	**82.31**	**89.93**
RoBbase(Lora$^{r=8}$)	61.15(0.8)	69.74(0.9)	73.9(0.7)	83.30(1.1)	55.77(2.3)	71.52(1.7)	70.31(1.1)	78.92(2.1)	72.53(1.7)	82.98(0.8)	66.73	77.29	77.78	86.97
RoBbase(Lora$^{r=16}$)	60.95(0.5)	68.73(1.3)	74.95(0.6)	83.33(1.7)	56.77(1.9)	71.91(1.3)	73.28(0.5)	**81.18**(1.3)	75.25(1.1)	84.45(1.3)	68.24	77.92	79.00	87.63
RoBbase(Lora$^{r=32}$)	60.50(1.3)	68.13(0.9)	75.60(1.2)	83.90(1.5)	56.44(1.9)	71.60(1.4)	70.95(2.1)	79.03(2.7)	72.58(2.0)	82.34(2.1)	67.21	77.00	78.41	87.37
RoBbase(AdaLoRA)	63.17(1.5)	70.41(0.7)	75.95(2.5)	83.56(2.3)	58.19(3.2)	73.14(2.3)	70.92(1.9)	78.43(1.7)	73.44(1.3)	82.86(1.1)	68.33	77.68	81.24	89.03
RoBbase(Prefix-Tuning)	57.72(2.1)	65.07(1.7)	75.60(1.3)	83.75(1.7)	57.14(2.7)	71.91(1.9)	66.66(1.3)	74.21(2.2)	67.54(1.9)	77.52(2.7)	64.93	74.49	78.97	87.62
RoBbase(IA3)	60.05(0.7)	66.74(1.1)	75.35(0.9)	82.42(2.7)	55.61(1.1)	70.41(1.0)	69.59(2.5)	77.31(2.5)	73.74(1.7)	83.42(1.3)	66.87	76.06	79.27	87.42
RoBbase((BitFit)	57.75(2.0)	64.84(1.3)	73.95(1.7)	80.80(3.8)	57.01(1.7)	71.29(2.9)	69.41(2.0)	76.65(1.4)	72.42(2.3)	81.84(1.5)	66.11	75.08	79.74	87.59
RoBlarge(FFT)	71.63(1.7)	77.67(0.5)	**83.9**(1.1)	**90.69**(1.3)	**63.78**(0.5)	**78.22**(1.1)	75.12(0.3)	81.16(0.7)	79.90(0.9)	87.28(1.3)	**74.87**	**83.00**	**85.71**	**92.39**
RoBlarge(Lora$^{r=8}$)	68.67(2.1)	76.99(0.3)	79.25(0.8)	87.99(2.7)	58.65(1.3)	74.94(0.7)	73.63(0.7)	81.38(1.2)	75.36(1.1)	85.60(2.5)	71.11	81.38	82.34	90.77
RoBlarge(Lora$^{r=16}$)	69.18(1.9)	76.79(0.7)	77.70(1.7)	86.20(1.5)	57.82(1.7)	73.84(1.5)	73.30(0.5)	81.21(0.8)	73.23(0.5)	83.31(2.4)	70.25	80.27	81.75	90.13
RoBlarge(Lora$^{r=32}$)	68.11(2.1)	75.37(1.1)	78.50(1.3)	86.05(1.3)	57.39(2.3)	73.50(2.3)	74.72(0.7)	82.40(0.5)	76.10(0.7)	86.10(2.7)	70.96	80.68	82.01	90.11
RoBlarge(AdaLoRA)	**73.84**(1.5)	**79.39**(1.3)	81.45(1.5)	87.23(1.7)	61.50(2.1)	76.76(1.7)	76.85(0.3)	**83.43**(1.3)	**80.27**(1.5)	**87.41**(1.9)	74.78	82.84	84.91	92.17
RoBlarge(Prefix-Tuning)	71.65(2.0)	78.39(2.4)	80.95(1.3)	88.33(2.2)	59.68(1.5)	75.13(2.1)	76.11(1.1)	82.79(0.4)	75.68(1.2)	84.88(1.7)	72.81	81.90	82.86	91.05
RoBlarge(IA3)	72.41(2.5)	77.48(1.9)	82.25(2.5)	89.00(2.9)	61.27(2.8)	76.46(1.9)	**76.99**(0.5)	83.35(0.9)	80.21(0.7)	87.17(1.6)	74.63	82.69	85.26	92.18
RoBlarge((BitFit)	62.12(2.3)	69.15(2.1)	75.25(1.9)	82.70(1.5)	57.76(3.2)	72.12(2.1)	67.38(0.9)	74.66(1.5)	69.12(1.3)	78.75(2.4)	66.33	75.48	80.60	88.46

Table 4: Performance comparison of different fine-tuning methods based on $\text{RoBERTa}_{\text{base}}$ and $\text{RoBERTa}_{\text{large}}$ on Adversarial SQuAD dataset.

Models	AS		AA		AAE		AAC		ANA		Average		Orign	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
OPT(FFT)	56.56(2.3)	**63.68**(2.5)	69.88(1.7)	77.25(2.7)	**58.64**(1.0)	63.86(1.2)	60.37(2.3)	62.48(0.9)	66.46(0.4)	**70.82**(1.3)	**62.38**	**67.62**	74.13	78.56
OPT(Lorar=8)	54.37(1.7)	61.07(1.7)	**70.06**(0.9)	**77.56**(2.1)	57.78(0.8)	**64.28**(2.3)	58.65(1.5)	62.34(1.3)	**66.76**(1.5)	70.77(1.7)	61.52	67.20	**75.29**	**79.55**
OPT(AdaLoRA)	50.22(1.9)	55.46(1.3)	58.62(3.1)	62.77(1.7)	45.88(1.1)	53.42(2.7)	57.84(1.7)	62.33(1.5)	58.62(2.7)	62.43(2.0)	54.24	59.28	66.72	51.65
OPT(Prefix-Tuning)	58.45(2.3)	61.39(1.5)	67.82(2.5)	71.88(1.3)	50.05(1.4)	56.38(1.5)	60.35(1.9)	62.17(2.7)	61.46(1.5)	64.37(1.5)	59.63	63.24	69.63	72.58
OPT(IA3)	**59.23**(1.5)	62.03(2.0)	68.36(1.5)	70.14(2.3)	56.48(1.9)	62.36(1.4)	**62.64**(1.3)	**66.41**(2.5)	63.62(0.9)	67.42(1.0)	62.07	65.67	71.04	73.54

Table 5: Performance comparison of different fine-tuning methods based on OPT-13B (with 1000 examples) on Adversarial SQuAD dataset.

parameter distribution of various fine-tuning methods. We can observe that the incremental parameters of PEFT (including LoRA, prefix-tuning, BitFit) exhibit a symmetric distribution around zero, with the Standard Deviation value being close to zero. This suggests that the adjustments made by these incremental parameters have a minimal impact on the underlying model. The incremental parameters of FFT display the widest distribution compared to PEFT, indicating a greater capacity for parameter adjustment. This pattern suggests that PEFT methods primarily adapt the model to follow instruction formats rather than incorporating substantial new knowledge. The minimal parameter changes indicate these techniques are effectively repurposing existing knowledge in the base model rather than learning new information during fine-tuning.

5.2 Robustness Between FFT and PEFT

Tables 3 and 4 present the performance comparison of various fine-tuning methods on the adversarial test sets AdvGLUE and Adversarial SQuAD. In AdvGLUE, we observe that $\text{RoBERTa}_{\text{base}}$ and $\text{RoBERTa}_{\text{large}}$ models fine-tuned with LoRA achieve performance advantages over FFT on the original dataset. Specifically, $\text{RoBERTa}_{\text{base}}$ based on LoRA performs 1.3% better than FFT, while $\text{RoBERTa}_{\text{large}}$ shows a 1.1% improvement. Other PEFT methods, such as AdaLoRA, Prefix, and IA3, exhibit performance levels very close to FFT. However, FFT demonstrates an overall average performance advantage over PEFT on AdvGLUE. Specifically, $\text{RoBERTa}_{\text{base}}$ fine-tuned with FFT outperforms LoRA by 5.69%, and $\text{RoBERTa}_{\text{large}}$ outperforms LoRA by 1.48%. On the Adversarial SQuAD dataset, we find that standard FFT consistently outperforms other PEFT methods on both the original dataset and the adversarial test set. We observe similar phenomena on large language models. As shown in Table 5, experiments on the Adversarial SQuAD dataset using OPT-13B show that LoRA achieves the highest performance on the original test set compared to other fine-tuning methods. However, FFT performs better than LoRA on the adversarial test set.

Theorem 4 shows that PEFT is more sensitive than FFT under the same disturbance ϵ. Theorem 1 indicates that PEFT is a subset of FFT. More specifically, PEFT is an embedded submanifold of the FFT parameter space. This means when the disturbance direction is orthogonal to the PEFT space (i.e., $\epsilon_{||} = 0$), the model loses the ability to extract effective features and filter out invalid ones, which significantly impacts its performance. Generally speaking, the disturbance ϵ has components in the orthogonal direction of PEFT space and in the parameter space direction. Since the task feature perception parameters are limited to the PEFT subspace, there is nothing the model can do regarding the orthogonal perturbation features. Due to the completeness and high degree of optimization flexibility in the FFT update parameter space, the model can more effectively learn to extract task-related

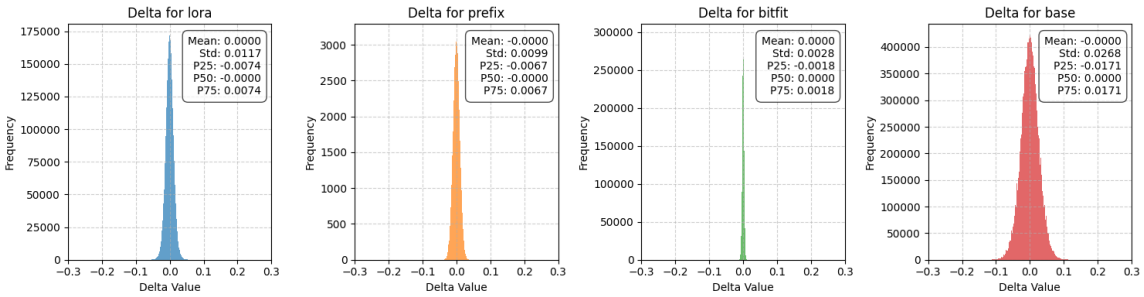

Figure 1: Incremental parameter distributions for different fine-tuning methods. For the prefix, the incremental parameters are prompt embeddings. For LoRA, BitFit, and FFT are the queries of the last layer. The same phenomenon can be observed in other layers for LoRA, BitFit, and FFT. The base model is LLaMA2-7B. The task is instruction tuning on Alpaca.

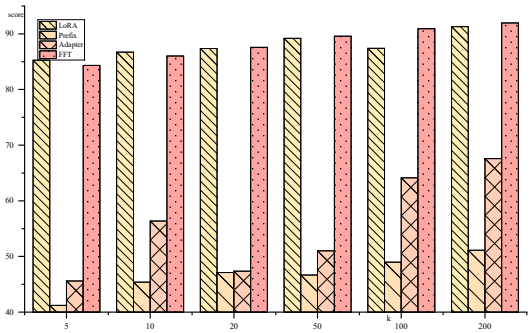

Figure 2: The impact of training samples on the performance of different fine-tuning methods. k denotes the number of training examples per class. The score represents the average performance of different fine-tuning methods on all test set. The base model is LLaMA2-7B.

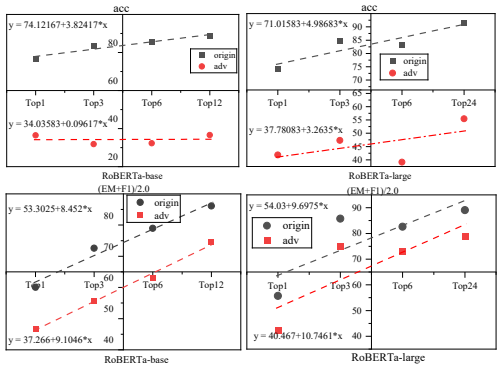

Figure 3: Performance trends of standard fine-tuning with different amounts of fine-tuning parameters on AdvGLUE (up) and Adversarial SQuAD (down). Acc corresponds to the average performance over all test sets.

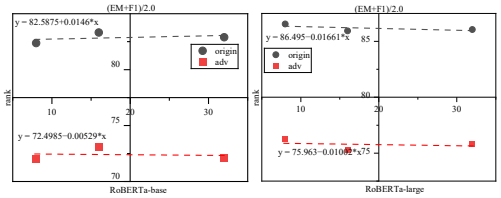

Figure 4: Performance trends of LoRA fine-tuning with different amounts of fine-tuning parameters on Adversarial SQuAD.

features and distinguish between disturbance information. Therefore, from the perspective of user reliability, we believe that FFT is more robust than PEFT.

Relying solely on performance metrics might lead to the misconception that PEFT methods, including LoRA, are viable alternatives to FFT. However, from a robustness perspective, FFT maintains a clear advantage. This prompts us to consider whether reducing the number of trainable parameters could potentially diminish certain aspects of the model's capabilities, such as robustness, making the model more vulnerable, or its representational capacity, hindering its ability to adequately capture data features. Until these potential risks associated with PEFT are fully explored, FFT remains a robust and dependable fine-tuning method.

5.3 Marginal Benefits of Data Increment

Figure 2(detailed results in Table 8 (Section G)) presents the performance variations of different fine-tuning methods across various sample sizes. When $k = 5$ and $k = 50$, FFT outperforms PEFT in two out of five tasks. When $k = 10$ and $k = 20$, FFT outperforms PEFT in only one out of five tasks. However, when $k = 100$ or $k = 200$, FFT outperforms PEFT in four out of five tasks. Figure 3 compares the average performance of different fine-tuning methods on these five benchmarks.

We observe that when $k < 20$, LoRA generally outperforms FFT. When $k = 20$, LoRA and FFT demonstrate similar performance. However, when $k > 20$, FFT begins to outperform PEFT. Based

on these results, we conclude that having sufficient training data is crucial to fully utilize FFT. Under few-shot fine-tuning scenarios, the PEFT method achieves a performance advantage more readily than FFT. Increasing the training data from 5 to 200 samples results in FFT's performance increasing by 9.05%, while LoRA's performance improves by 7.08%. Thus, augmenting the training data significantly enhances the performance advantages of FFT. For large language models, estimating the requisite amount of data for effective fine-tuning can be challenging (Kaplan et al., 2020). According to Theorem 5, when adding the same amount of data at the same time, the marginal benefit ratio of PEFT and FFT is $O(\frac{k}{d}) \ll 1$, where k and d represent the parameter dimensions of FFT of PEFT respectively. The optimization space of FFT is complete, and its parameter updates can encompass all possible feature directions. With sufficient degrees of freedom, the model can always find unoptimized feature dimensions for new data. This ample parameter space allows for the learning of additional features. PEFT restricts optimization to a k-dimensional subspace, with new data being incorporated only through low-dimensional projections (such as the low-rank matrix in LoRA). This limits the capacity of the parameter space, leading to the loss of significant information. Once the low-dimensional space is saturated with features, there is no remaining parameter space to accommodate additional feature information.

5.4 Marginal Benefits of Parameter Increment

Theorem 3 shows that PEFT has diminishing marginal benefits of parameter increments. To investigate the impact of the number of trainable parameters on the performance of FFT and PEFT, we analyze the performance variations when fine-tuning the top-k layers and the performance changes of LoRA at different ranks r, where $k \in \{1, 3, 6, 12, 24\}$ and $r \in \{8, 16, 32\}$. Note that since obtaining models with the same structure but different parameter scales is challenging, we simulate the parameter changes of FFT by fine-tuning the top-k layers.

Figure 3 and Figure 4 (detailed results in Table 9 and Table 10 (Section G)) report the performance changes of $\text{RoBERTa}_{\text{base}}$ (RoBbase) and $\text{RoBERTa}_{\text{large}}$ (RoBlarge) with different numbers of trainable parameters on the AdvGLUE and Adversarial SQuAD datasets, respectively. On AdvGLUE, FFT exhibits the best average performance on both the original dataset and the adversarial test set. A similar trend is observed on Adversarial SQuAD. Standard fine-tuning involves reducing trainable parameters by freezing some layers to lower memory usage, which may potentially impact both model performance and robustness. In the case of LoRA, increasing the rank to add more trainable parameters shows minimal impact on the model's performance and robustness. This indicates that increasing the number of trainable parameters in LoRA fine-tuning does not effectively enhance the model's performance and robustness as it does with FFT. These findings suggest that LoRA fine-tuning may potentially constrain the model's representation capacity.

5.5 Resource Requirements Comparison

Table 7 (Appendix G) reports the comparison of the GPU memory resources required by different fine-tuning methods. We generally believe that FFT requires more resources than PEFT. However, efficient FFT methods represented by HiFT have greatly reduced the resource requirements. For the RoBERTa-base model, HiFT consumes approximately 2.62 GB of GPU memory, slightly less than the 2.63–2.70 GB required by PEFT methods. For the RoBERTa-large model, this advantage becomes more evident, with HiFT using 6.62 GB compared to 6.64–7.13 GB for PEFT methods. The trend is further amplified in the larger LLaMA2-7B model, where HiFT requires only 40.11 GB, lower than the 40.69–43.24 GB consumed by PEFT approaches. These results highlight HiFT's superior memory efficiency, demonstrating that it can maintain even reduce GPU memory consumption compared to PEFT alternatives.

6 Conclusion

In this paper, we compare different fine-tuning methods from the perspective of robustness and performance. We have observed that PEFT can achieve comparable or even superior performance to FFT on certain benchmarks. However, FFT consistently maintains a performance advantage over PEFT in the context of complex tasks that were evaluated. In the adversarial test, PFFT clearly demonstrates a significant performance advantage over PEFT. Therefore, we still recommend using FFT for fine-tuning models to the extent that resources allow.

Limitations

This paper compares the FFT and PEFT fine-tuning methods in terms of robustness and performance, but we believe that there are other potential perspectives yet to be explored, such as generalization. Conducting a more comprehensive evaluation and comparison, as well as exploring the impact of PEFT on the intrinsic mechanisms of the model, remains challenging.

References

Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. 2022. Composable sparse fine-tuning for cross-lingual transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1778–1796.

Niederfahrenhorst Artur, Hakhamaneshi Kourosh, and Ahmad Rehaan. 2023. *Fine-Tuning LLMs: LoRA or Full-Parameter? An in-depth Analysis with Llama-2*.

Avrim Blum. 2007. Machine learning theory. *Carnegie Melon Universit, School of Computer Science*, 26:5.

Jin Cao, Chandana Satya Prakash, and Wael Hamza. 2022. Attention fusion: a light yet efficient late fusion mechanism for task adaptation in nlu. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 857–866.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. SemEval-2019 task 3: EmoContext contextual emotion detection in text. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 39–48, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. 2023. Parameter-efficient fine-tuning design spaces. *arXiv preprint arXiv:2301.01821*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yehuda Dar, Vidya Muthukumar, and Richard G Baraniuk. 2021. A farewell to the bias-variance tradeoff? an overview of the theory of overparameterized machine learning. *arXiv preprint arXiv:2109.02355*.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. 2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems*, 27.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nat. Mac. Intell.*, 5(3):220–235.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. 2022. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*.

Ruili Feng, Kecheng Zheng, Yukun Huang, Deli Zhao, Michael I. Jordan, and Zhengjun Zha. 2022. Rank diminishing in deep neural networks. *ArXiv*, abs/2206.06072.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, volume abs/2010.01412.

Karan Goel, Nazneen Fatema Rajani, Jesse Vig, Zachary Taschdjian, Mohit Bansal, and Christopher Ré. 2021. Robustness gym: Unifying the NLP evaluation landscape. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 42–55. Association for Computational Linguistics.

Shreya Goyal, Sumanth Doddapaneni, Mitesh M Khapra, and Balaraman Ravindran. 2023. A survey of adversarial defenses and robustness in nlp. *ACM Computing Surveys*, 55(14s):1–39.

Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Pengfei He. 2024. Parameter efficient instruction tuning: An empirical study. *arXiv preprint arXiv:2411.16775*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. Llm-adapters: An

adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 5254–5276. Association for Computational Linguistics.

Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. 2025. Hira: Parameter-efficient hadamard high-rank adaptation for large language models. In *The Thirteenth International Conference on Learning Representations*.

Robin Jia and Percy Liang. 2017a. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Robin Jia and Percy Liang. 2017b. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2021–2031. Association for Computational Linguistics.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Weisen Jiang, Baijiong Lin, Han Shi, Yu Zhang, Zhenguo Li, and James T. Kwok. 2023b. Effective and parameter-efficient reusing fine-tuned models. *CoRR*, abs/2310.01886.

Juraj Juraska, Kevin Bowden, and Marilyn Walker. 2019. Viggo: A video game corpus for data-to-text generation in open-domain conversation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 164–172.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035.

Ron Kohavi and David Wolpert. 1996. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, pages 275–283.

Hakhamaneshi Kourosh and Ahmad Rehaan. 2023. *Fine-Tuning Llama-2: A Comprehensive Case Study for Tailoring Models to Unique Applications*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. 2023. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Peiqin Lin, Shaoxiong Ji, Jörg Tiedemann, André FT Martins, and Hinrich Schütze. 2024. Mala-500: Massive language adaptation of large language models. *arXiv preprint arXiv:2401.13303*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022b. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Kai Liu, Xin Liu, An Yang, Jing Liu, Jinsong Su, Sujian Li, and Qiaoqiao She. 2020a. A robust adversarial training approach to machine reading comprehension. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8392–8400.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020b. Multilingual denoising pre-training for neural machine translation. *Trans. Assoc. Comput. Linguistics*, 8:726–742.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yongkang Liu, Yiqun Zhang, Qian Li, Shi Feng, Daling Wang, Yifei Zhang, and Hinrich Schütze. 2024. Hift: A hierarchical full parameter fine-tuning strategy. *arXiv preprint arXiv:2401.15207*.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.

L. Mirsky. 1960. Symmetric gauge functions and unitarily invariant norms. *Quarterly Journal of Mathematics*, 11:50–59.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 119–126. Association for Computational Linguistics.

Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, and Ioannis Mitliagkas. 2018. A modern take on the bias-variance tradeoff in neural networks. *arXiv e-prints*, pages arXiv–1810.

Fuli Qiao and Mehrdad Mahdavi. 2024. Learn more, but bother less: parameter efficient continual learning. *Advances in Neural Information Processing Systems*, 37:97476–97498.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Sebastian Raschka. 2023. *Finetuning LLMs with LoRA and QLoRA: Insights from Hundreds of Experiments*.

Arthur Sard. 1942. The measure of the critical values of differentiable maps.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207.

Danilo Vucetic, Mohammadreza Tayaranian, Maryam Ziaeefard, James J Clark, Brett H Meyer, and Warren J Gross. 2022. Efficient fine-tuning of bert models on the edge. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1838–1842. IEEE.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. 2021a. Adversarial GLUE: A multi-task benchmark for robustness evaluation of language models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, Binxing Jiao, Yue Zhang, and Xing Xie. 2023a. On the robustness

of chatgpt: An adversarial and out-of-distribution perspective. *CoRR*, abs/2302.12095.

Shunxin Wang, Raymond Veldhuis, and Nicola Strisciuglio. 2023b. The robustness of computer vision models against common corruptions: a survey. *arXiv preprint arXiv:2305.06024*.

Xiao Wang, Qin Liu, Tao Gui, Qi Zhang, Yicheng Zou, Xin Zhou, Jiacheng Ye, Yongxin Zhang, Rui Zheng, Zexiong Pang, Qinzhuo Wu, Zhengyan Li, Chong Zhang, Ruotian Ma, Zichu Fei, Ruijian Cai, Jun Zhao, Xingwu Hu, Zhiheng Yan, Yiding Tan, Yuan Hu, Qiyuan Bian, Zhihua Liu, Shan Qin, Bolin Zhu, Xiaoyu Xing, Jinlan Fu, Yue Zhang, Minlong Peng, Xiaoqing Zheng, Yaqian Zhou, Zhongyu Wei, Xipeng Qiu, and Xuanjing Huang. 2021b. Textflint: Unified multilingual robustness evaluation toolkit for natural language processing. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL 2021 - System Demonstrations, Online, August 1-6, 2021*, pages 347–355. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT*, pages 1112–1122.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Lifan Yuan, Yangyi Chen, Ganqu Cui, Hongcheng Gao, Fangyuan Zou, Xingyi Cheng, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023. Revisiting out-of-distribution robustness in NLP: benchmark, analysis, and llms evaluations. *CoRR*, abs/2306.04618.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. 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)*, pages 1–9.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Zixian Ma, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2021. Openattack: An open-source textual adversarial attack toolkit. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL 2021 - System Demonstrations, Online, August 1-6, 2021*, pages 363–371. Association for Computational Linguistics.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023a. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*. Openreview.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: open pre-trained transformer language models. *CoRR*, abs/2205.01068.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Zhenru Zhang, Chuanqi Tan, Haiyang Xu, Chengyu Wang, Jun Huang, and Songfang Huang. 2023b. Towards adaptive prefix tuning for parameter-efficient language model fine-tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1239–1248. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Neil Zhenqiang Gong, Yue Zhang, et al. 2023. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv preprint arXiv:2306.04528*.

A Baselines

Language Models

- **RoBERTa** (Liu et al., 2019) is a transformer-based model built on BERT, designed by Facebook AI. It improves BERT by removing the Next Sentence Prediction objective and training with larger mini-batches and longer sequences. RoBERTa has set new benchmarks in NLP tasks by leveraging more data and training with different hyperparameters, making it a state-of-the-art pre-trained language model.
- **LLaMA** (Touvron et al., 2023) is a family of decoder-only models by Meta, designed to be efficient for a wide range of NLP tasks. LLaMA models are trained with a focus on providing strong performance while maintaining fewer parameters, making them more accessible and computationally efficient compared to larger models like GPT-3. We use 7B, 13B, and 70B versions of LLaMA.
- **TinyLLaMA** is a compact version of LLaMA, optimized for tasks where computational resources are limited. It retains much of the performance of its larger counterpart but is designed to run efficiently on smaller devices.
- **Mistral-7B** (Jiang et al., 2023a) is a dense transformer model with 7 billion parameters, developed by Mistral AI. It offers strong language understanding and generation capabilities while being smaller in scale compared to other large models, balancing efficiency and performance.
- **OPT-13B** (Zhang et al., 2022) is a model developed by Meta, designed to be a large-scale generative language model. With 13 billion parameters, it is optimized for large-scale language tasks and is designed to provide open access to competitive performance, fostering transparency in AI development.

Fine-Tuning strategies

- **BitFit** (Zaken et al., 2022) is a lightweight parameter-efficient fine-tuning method. Unlike traditional fine-tuning approaches that modify all model parameters, BitFit focuses only on fine-tuning the bias terms of the model. This significantly reduces the computational cost and memory requirements while still achieving competitive performance on downstream tasks. It is particularly useful when fine-tuning large models on smaller datasets.
- **AdaLoRA** (Zhang et al., 2023a) is an adaptive parameter-efficient fine-tuning method. It dynamically adjusts the rank of low-rank matrices during fine-tuning, allowing for efficient adaptation to specific tasks. AdaLoRA achieves a balance between efficiency and accuracy by selecting optimal ranks, enabling models to generalize well with minimal computational overhead.
- **Prefix** (Lester et al., 2021) is a fine-tuning method that adds a small number of task-specific parameters (prefix tokens) to the input of the model. This method allows the pre-trained model to adapt to different tasks by simply learning the right combination of these prefix tokens, making it a lightweight and computationally efficient approach for transfer learning.
- **LoRA** (Hu et al., 2022) is an efficient fine-tuning method for large-scale pre-trained language models. Its core idea is to reduce the amount of trainable parameters in the fine-tuning process through low-rank matrix decomposition while maintaining model performance. It is proposed mainly to solve the high cost of traditional full-parameter fine-tuning in terms of computing resources, memory usage, and deployment costs.
- **IA3** (Liu et al., 2022b) is a parameter-efficient fine-tuning method that aims to achieve performance comparable to full parameter fine-tuning of large models with minimal parameter adjustments. The core idea is to dynamically scale the internal activations of the model through learning vectors, so as to adapt to downstream tasks without significantly increasing computing resources.
- **FFT** refers to updating all model parameters for downstream tasks based on pre-trained models to adapt them to specific task requirements. Although parameter-efficient fine-tuning methods have become increasingly popular in recent years, FFT is still the most powerful method for obtaining the strongest baseline in many scenarios.

B Datasets

Adversarial benchmarks include **Adversarial SQuAD** and **AdvGLUE**. **Adversarial SQuAD** is an adversarial question answering dataset, primarily consisting of several sub-datasets:

- **AddSent (AS)** (Jia and Liang, 2017b): A grammatical adversarial test set with 1k instances, in which misleading texts are generated from questions through rules and crowdsourcing.
- **AddAny (AA)** (Jia and Liang, 2017b): Ungrammatical adversarial test set with 1k instances, in which misleading texts are automatically gener-

ated according to question words and common words.

- **AddAnyExtend (AAE)** (Liu et al., 2020a): Extended AddAny with 2.6k instances, which contains not only question words but also high-frequency words, passage words, and random common words.
- **AddAnsCtx (AAC)** (Liu et al., 2020a): Answer context test set with 10k instances, where the misleading texts are answer sentences with the answer tokens removed.
- **AddNegAns (ANA)** (Liu et al., 2020a): Negative expression test set with 5k instances, which uses negative expressions of fake answers as misleading texts.

Adversarial GLUE (Wang et al., 2021a) is a multi-task benchmark for robustness evaluation, mainly including **SST2**, **QQP**, **MNLI**, **MNLI-MM**, **QNLI** and **RTE**.

Performance datasets

- **ViGGO** (Juraska et al., 2019) is an English dataset for data-to-text generation, focusing on video game opinions. The original task involves transforming a functional representation (a collection of attribute-values) into coherent text that includes those attributes. In this paper, we will reverse this process: converting unstructured text into a structured and parsable functional representation. This representation encapsulates the information found in the text and can be used for indexing and other downstream applications.
- **GSM8k** (Cobbe et al., 2021) is a standard academic benchmark for evaluating language models on math reasoning and understanding. The goal of this dataset is to test the reasoning ability of language models.
- **SQL Generation** (Zhong et al., 2017; Yu et al., 2018) is a combination of the WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018). This dataset contains 78,577 examples. Each example consists of a natural language query, corresponding SQL CREATE TABLE statements, and the SQL query corresponding to the natural language question. The goal is to take the natural language query and SQL CREATE TABLE statements as context and produce a SQL output that can query the given SQL tables and produce an output that answers the natural language query.
- **SQUAD** (Socher et al., 2013) is a reading comprehension dataset consisting of questions posed by crowdworkers on a set of Wikipedia articles. SQuAD 1.1 contains over 100,000 question-answer pairs across 536 articles. It serves as a benchmark for evaluating machine reading comprehension systems.
- **SST-2** (Socher et al., 2013) is a sentiment analysis dataset derived from movie reviews. It contains 11,855 sentences, each labeled as either positive or negative. The dataset is parsed with Stanford's parser and includes 215,154 unique phrases, each annotated by three human judges.
- **QQP**[2] consists of pairs of questions from Quora and is used for the task of identifying whether two questions are semantically equivalent. It has been widely used for training and evaluating models on paraphrase detection.
- **MNLI** (Williams et al., 2018) consists of sentence pairs labeled with one of three categories: entailment, contradiction, or neutral. It is designed to evaluate models on natural language inference (NLI) across different genres of text.
- **MNLI-MM** (Williams et al., 2018) includes sentence pairs that are more "matched" in terms of domain and linguistic features, providing a more controlled environment for NLI evaluation.
- **QNLI** (Rajpurkar et al., 2018) is a binary classification task where the goal is to determine if a given passage contains the answer to a question.
- **RTE** (Wang et al., 2018) contains sentence pairs labeled as either entailment or non-entailment. It is used for evaluating models on the task of recognizing whether a hypothesis sentence can be logically inferred from a premise sentence.
- **EmoC** (Chatterjee et al., 2019) focuses on contextual emotion detection in dialogues. It consists of English tweets annotated with four emotion classes: happy, sad, angry, and others. The dataset includes three-turn dialogues to capture emotional nuances in conversational contexts.
- **TREC** (Voorhees and Tice, 2000) is designed for question classification tasks. It contains 5,500 labeled questions in the training set and 500 in the test set, categorized into six coarse classes and 50 fine classes. The average sentence length is 10 words, with a vocabulary size of 8,700.
- **Amazon** (McAuley and Leskovec, 2013) comprises product reviews and metadata from Amazon, including ratings, text, helpfulness votes, descriptions, prices, and images. It spans from May 1996 to July 2014, offering a rich resource

[2] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

for sentiment analysis and recommendation system research.

- **AGNews** (Zhang et al., 2015) is a news classification dataset containing 127,600 training samples and 7,600 test samples across four categories: World, Sports, Business, and Sci/Tech. It is constructed from the AG's corpus of news articles and is commonly used for evaluating text classification models.
- **MT-bench** (Zheng et al., 2024) is a benchmark designed to evaluate the fine-grained abilities of large language models (LLMs) in multi-turn dialogues. It includes 3,388 expert-level pairwise human preferences for model responses to 80 questions, assessing aspects like reasoning, coding, and math. Please refer to MT-bench (Zheng et al., 2024) for more detailed information. Note that we fine-tune language models on Alpaca (Taori et al., 2023), which consists of 51K instruction-following examples generated using text-davinci-003 (GPT-3.5).

C Implementation Details

The performance results of the experiment are based on training with the AdamW optimizer. We selected the optimal hyperparameters, such as batch size and learning rate, through a grid search. For few-shot learning, we take one demonstration per class to form the prompt[3] and append the sample to be predicted at the end of the prompt. We experiment with different numbers of training samples: 5, 10, 20, 50, 100, and 200 samples per class. The results are averaged over 5 random seeds (i.e., 0, 42, 421, 520, and 1228).

For AdvGLUE and Adversarial SQuAD, we train models and select the best model based on the original training and validation sets (i.e., GLUE and SQuAD), and then use the selected model to test on the adversarial test set. Since the test set of Adversarial SQuAD is not open source, we use the adversarial validation set as the test set. We report the key hyperparameter configurations for all experiments in Table 6 (Section F).

D Prompts

We implement prompt-based fine-tuning for few-shot learning tasks. The goal is to predict the correct class given a few examples. We reformulate the task as a language modeling problem. Let M be a language model with vocabulary V, and let $\mathcal{L}$ be a set of label words. The training set $\mathcal{T}$ consists of pairs (s, l), where s is a sequence of tokens from the vocabulary V and l is a label word from the set $\mathcal{L}$. In a sentiment analysis task, for instance, we define a pattern $\mathcal{P}(s, l)$ which associates a text s ='Nice performance' and a label word l ='Positive' as follows:

Review: Nice performance. Sentiment: Positive

For a k-class classification task, we sample one demonstration per class from the training set $\mathcal{T}$, and concatenate them with the text s to be classified to form the prompt $X(s)$:

$$X(s) = \mathcal{P}(s_1, l_1) \oplus \ldots \oplus \mathcal{P}(s_k, l_k) \oplus \mathcal{P}(s, \varepsilon) \quad (10)$$

$\oplus$ denotes the concatenation of the input demonstrations and ε is the empty string.

E Proofs

E.1 Proof of Non-surjective Function

Proof of Theorem 1. Given a continuously differentiable map g: $\mathbb{R}^k \to \mathbb{R}^d$, where $k \ll d$, then g is a non-surjective function.

Define the set of all possible points that can be mapped from $\mathbb{R}^k$ to $\mathbb{R}^d$ via g:

$$Im(g) = \{g(\Phi) \in \mathbb{R}^d | \Phi \in \mathbb{R}^k\} \quad (11)$$

When g is a full rank mapping, we can have:

$$Im(g) = g(\mathbb{R}^k) \subset \mathbb{R}^d \quad (12)$$

The image $Im(g)$ is a k-dimensional embedded submanifold of an d-dimensional space. According to Sard's theorem(Sard, 1942), when $k \ll d$, $Im(g)$, viewed as a subset of $\mathbb{R}^d$, has Lebesgue measure zero in $\mathbb{R}^d$. This shows that the g cannot cover the $\mathbb{R}^d$ and is a non-surjective mapping. This conclusion is further strengthened when g is a non-full rank mapping. □

E.2 PEFT Capacity Upper Bound

Proof of Theorem 2. For input $x \in D$, we have:

$$f(x) = \sigma((W_0 + \Phi)x) \quad (13)$$

where σ is the activation function that typically satisfies the L-Lipschitz condition. For any $u, v \in \mathbb{R}^d$, $\|\sigma(u) - v\| \leq L\|u - v\|$ always holds, where L is the Lipschitz constant of the activation function σ. The incremental difference after fine-tuning is:

$$\begin{aligned} \|f(x) - f_0(x)\| &= \|\sigma(W_0 x + \Phi x) - \sigma(W_0 x)\| \\ &\leq L\|\Phi x\| \leq L\|\Phi\|\|x\| \end{aligned} \quad (14)$$

[3]The templates of prompts are presented in Section D

So for a certain layer we have:

$$\forall x, \|f(x) - f_0(x)\| \leq L\|\Phi\|\|x\| \tag{15}$$

Extending this theory to a network of depth N, the original parameters of each layer, along with the parameters introduced by PEFT, are denoted as follows:

$$\begin{aligned} f_0(x) &= \sigma_N(W_{0,N}\sigma_{N-1}(\cdots\sigma_1(W_{0,1}x))), \\ f(x) &= \sigma_N((W_{0,N} + \Phi_N)\sigma_{N-1}(\cdots\sigma_1((W_{0,1} + \Phi_1)x))) \end{aligned} \tag{16}$$

The activation Lipschitz constant of the kth layer is L_k, and the input-output norm is consistent with $\|\|$. Let $x_0^k = \sigma_k(W_{0,k}x_0^{k-1})$, $x^k = \sigma_k(W_0^k + \Phi_k)x^{k-1}$, where x_0^k denotes the original representation of the foundation model at layer k, and x^k denotes the representation of the fine-tuned model at layer k. The difference in representation between the fine-tuned and foundation models is:

$$\begin{aligned} &\|x^k - x_0^k\| = \|\sigma_k(W_{0,k}x^{(k-1)} + \Phi_k x^{(k-1)}) - \sigma_k(W_{0,k}x_0^{(k-1)})\| \\ &\leq L_k\|W_{0,k}x^{(k-1)} + \Phi_k x^{(k-1)} - W_{0,k}x_0^{(k-1)}\| \\ &\leq L_k\|W_{0,k}\|\|x^{(k-1)} - x_0^{(k-1)}\| + \|\Phi_k\|\|x^{(k-1)}\| \end{aligned} \tag{17}$$

Using recursion we can get the overall output difference:

$$\begin{aligned} \|f(x) - f_0(x)\| \leq \sum_{k=1}^{N}[&L_N\|W_{0,N}\| \cdots L_{k+1}\|W_{0,k+1}\| \\ &\times L_k\|\Phi_k\|\|x\|] \end{aligned} \tag{18}$$

Assume that the upper bounds of the weight norm and activation Lipschitz constant of each layer are M and L respectively, that is, $\|W_{0,k}\| \leq M, \quad L_k \leq L, \quad \forall k$. At the same time, let $\alpha_k \triangleq L\|\Phi_k\|$, we have:

$$\|f(x) - f_0(x)\| \leq \sum_{k=1}^{N} M^{N-k}L^{N-k}\alpha_k\|x\| \tag{19}$$

□

E.3 Rule of Diminishing Marginal Benefit

Proof of Theorem 3. In PEFT fine-tuning, there exists a critical parameter Φ_c such that when $Rank(\Phi) \geq r_c$, further increasing the parameter amount of Φ does not lead to significant improvements in the performance of the optimal solution to the optimization problem.

For any given task, the ideal weight update ΔW^* can be expressed using singular value decomposition as follows:

$$\Delta W^* = U\Sigma V^T = \sum_{i=1}^{r_c} \sigma_i u_i v_i^T \tag{20}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{r_c} \geq 0$ are the singular values, and u_i and v_i are the corresponding left and right singular vectors, respectively. According to the Eckart–Young–Mirsky theorem (Mirsky, 1960), the best rank r approximation is given by:

$$\Delta W_r = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{21}$$

Under the Frobenius norm, the approximation error is:

$$\|\Delta W^* - \Delta W_r\|_F^2 = \sum_{i=r+1}^{r_c} \sigma_i^2 \tag{22}$$

Existing studies have shown that singular values typically decay exponentially(Feng et al., 2022); therefore, we assume $\sigma_i \approx O(e^{-\lambda i})$, where $\lambda > 0$. accordingly, the critical rank r_c satisfies the following:

$$\sum_{i=r_c+1}^{rank(\Phi)} \sigma_i^2 < \epsilon \tag{23}$$

Where ϵ is a positive number related to the generalization error. Since the loss function is typically smooth and continuous, there exists a constant $L > 0$ such that:

$$\begin{aligned} &|\mathcal{L}(W + \Delta W_1) - \mathcal{L}(W + \Delta W_2)| \\ &\qquad \leq L\|\Delta W_1 - \Delta W_2\|_F \end{aligned} \tag{24}$$

Therefore, when $r \geq r_c$, we have:

$$\begin{aligned} &|\mathcal{L}(W + \Delta W^*) - \mathcal{L}(W + \Delta W_r)| \\ &\qquad \leq L\|\Delta W^* - \Delta W_r\|_F \leq L\sqrt(\epsilon) \end{aligned} \tag{25}$$

Statistical learning theory shows that overparameterized models can lead to:

$$\mathcal{L}_{\mathcal{D}_{test}}(W + \Delta W_r) - \mathcal{L}_{\mathcal{D}_{train}}(W + \Delta W_r) \propto \sqrt{\frac{r}{N}} \tag{26}$$

Where N is the number of training samples. Based on the above analysis, when $r \geq r_c$, the following conclusions can be drawn:

$$\begin{aligned} \mathcal{L}_{\mathcal{D}_{test}}(W + \Delta W_r) &\leq \mathcal{L}_{\mathcal{D}_{test}}(W + \Delta W^*) \\ &\quad + L\sqrt{\epsilon} + O(\sqrt{\frac{r}{N}}) \end{aligned} \tag{27}$$

As r continues to increase, the first term remains unchanged, the second term approaches zero, but the third term (generalization error) increases, leading to a plateau or even a decline in overall performance. □

E.4 PEFT More Sensitive to Disturbances

Proof of Theorem 4. Given the perturbed data distribution D',

$$\theta'_{perturbed} = \arg\min_{\theta\in\Theta}\mathbb{E}_{(x,y)\sim D'}[\mathcal{L}(f(x;\theta), y)] \tag{28}$$

The optimization variable in PEFT is Φ, in contrast to FFT, where it is θ. Define the optimization variable $\delta \in \{\Phi, \theta\}$. By Taylor expanding the loss function, we can have:

$$\mathcal{L}(\delta + \Delta\delta) \approx \mathcal{L}(\delta) + \nabla_\delta\mathcal{L}(\delta)^T\Delta\delta + \frac{1}{2}\Delta\delta^T H\Delta\delta \tag{29}$$

where $H = \nabla^2_\delta \mathcal{L}(\delta)$ is Hessian Matrix. Assume the disturbance factor is ϵ, the loss objective is:

$$\mathcal{L}(\delta+\Delta\delta) \approx \mathcal{L}(\delta)+(\nabla_\delta \mathcal{L}(\delta)^T+\epsilon)\Delta\delta+\frac{1}{2}\Delta\delta^T H\Delta\delta \quad (30)$$

Determine the extreme value of the quadratic function with respect to $\Delta\delta$:

$$\nabla_{\Delta\delta}\left[(\nabla_\delta \mathcal{L}+\epsilon)^T\Delta\delta+\frac{1}{2}\Delta\delta^T H\Delta\delta\right]=0 \quad (31)$$

Taking the derivative, we get:

$$\nabla_\delta \mathcal{L}+\epsilon+H\Delta\delta=0 \Rightarrow \Delta\delta=-H^{-1}(\nabla_\delta \mathcal{L}+\epsilon) \quad (32)$$

For FFT, H is a positive definite or semi-definite matrix and invertible, we can have:

$$\Delta\theta_{full}=-H^{-1}(\nabla_\theta \mathcal{L}+\epsilon) \quad (33)$$

Substituting $\Delta\theta_{full}$ into the Taylor expansion of the loss function, and noting that $\Delta\theta_{full}$ is a first-order optimal solution—causing the first-order term to vanish—we obtain:

$$\Delta\mathcal{L}_{\text{full}} \approx \frac{1}{2}\Delta\theta^T H\Delta\theta \quad (34)$$

Substitute $\Delta\theta=-H^{-1}(\nabla_\theta \mathcal{L}+\epsilon)$:

$$\begin{aligned}\Delta\mathcal{L}_{\text{full}} &\approx \frac{1}{2}\left[-H^{-1}(\nabla_\theta \mathcal{L}+\epsilon)\right]^T H\left[-H^{-1}(\nabla_\theta \mathcal{L}+\epsilon)\right]\\ &=\frac{1}{2}(\nabla_\theta \mathcal{L}+\epsilon)^T H^{-1}HH^{-1}(\nabla_\theta \mathcal{L}+\epsilon)\\ &=\frac{1}{2}(\nabla_\theta \mathcal{L}+\epsilon)^T H^{-1}(\nabla_\theta \mathcal{L}+\epsilon)\end{aligned} \quad (35)$$

We only focus on the disturbance term, then:

$$\Delta\mathcal{L}_{full} \approx \frac{1}{2}\epsilon^T H^{-1}\epsilon \quad (36)$$

For PEFT, H is typically a $d \times k$ matrix with $k \ll d$, indicating that H is non-invertible (i.e., not exist H^{-1}). According to Theorem 1, we can represent the parameter update as a projection mapping from $\mathbb{R}^k$ to $\mathbb{R}^d$:

$$\theta=\theta_0+g(\Phi) \quad (37)$$

Here we view g as a linear layer structure, and let $g=P\Phi$. we have:

$$\nabla_\Phi \mathcal{L}=\frac{\partial \mathcal{L}}{\partial\theta}\cdot\frac{\partial\theta}{\partial\Phi}=\nabla_\theta \mathcal{L}\cdot P=P^T\nabla_\theta \mathcal{L} \quad (38)$$

The further Hessian matrix H_Φ is:

$$\begin{aligned}H_\Phi=\nabla^2_\Phi \mathcal{L}&=\frac{\partial^2\mathcal{L}}{\partial\Phi^2}\\ &=\left(\frac{\partial\theta}{\partial\Phi}\right)^T\cdot\frac{\partial^2\mathcal{L}}{\partial\theta^2}\cdot\left(\frac{\partial\theta}{\partial\Phi}\right)=P^T HP\end{aligned} \quad (39)$$

According Equation 18, the parameter update of PEFT is:

$$\Delta\theta_{peft}=-PH^+_\Phi P^T(\nabla_\theta \mathcal{L}+\epsilon) \quad (40)$$

where H^+_Φ is the pseudo-inverse of $P^T HP$ (because it may not be full rank). Since PEFT operates within a restricted low-dimensional space, the perturbation in $\mathbb{R}^d$ can be expressed as:

$$\epsilon=\epsilon_\parallel+\epsilon_\perp \quad (41)$$

where $\epsilon_\parallel$ is the projection of the perturbation in the PEFT subspace, and $\epsilon_\perp$ is the component orthogonal to this subspace. PEFT can only counteract the component of the perturbation within the subspace, $\epsilon_\parallel = PP^T\epsilon$, and cannot counteract the perpendicular component $\epsilon_\perp=(I-PP^T)\epsilon$. The loss variation of PEFT induced by the perturbation is given by:

$$\Delta\mathcal{L}_{peft}\approx\frac{1}{2}\epsilon^T_\parallel H^+_\Phi\epsilon_\parallel+\epsilon^T_\perp\nabla_\theta \mathcal{L} \quad (42)$$

Let $\Delta\mathcal{L}_{\text{peft}}-\Delta\mathcal{L}_{\text{full}}$ have:

$$\Delta\mathcal{L}_{\text{peft}}-\Delta\mathcal{L}_{\text{full}}=\frac{1}{2}\epsilon^T_\parallel H^+_\Phi\epsilon_\parallel+\epsilon^T_\perp\nabla_\theta \mathcal{L}-\frac{1}{2}\epsilon^T H^{-1}\epsilon \quad (43)$$

Expanding $\epsilon=\epsilon_\parallel+\epsilon_\perp$ yields:

$$\begin{aligned}\epsilon^T H^{-1}\epsilon&=(\epsilon_\parallel+\epsilon_\perp)^T H^{-1}(\epsilon_\parallel+\epsilon_\perp)\\ &=\epsilon^T_\parallel H^{-1}\epsilon_\parallel+\epsilon^T_\perp H^{-1}\epsilon_\perp+2\epsilon^T_\parallel H^{-1}\epsilon_\perp\end{aligned} \quad (44)$$

Since $\epsilon_\parallel \perp \epsilon_\perp$, they belong to orthogonal subspaces, then:

$$\epsilon^T_\parallel H^{-1}\epsilon_\perp=0 \quad (45)$$

We have:

$$\begin{aligned}\Delta\mathcal{L}_{\text{peft}}-\Delta\mathcal{L}_{\text{full}}=&\frac{1}{2}(\epsilon^T_\parallel H^+_\Phi\epsilon_\parallel-\epsilon^T_\parallel H^{-1}\epsilon_\parallel)\\ &+\epsilon^T_\perp\nabla_\theta \mathcal{L}-\frac{1}{2}\epsilon^T_\perp H^{-1}\epsilon_\perp\end{aligned} \quad (46)$$

PEFT fine-tunes a model within a low-dimensional subspace, which can be interpreted as a projection from $\mathbb{R}^d$ to $\mathbb{R}^k$. Therefore:

$$\epsilon^T_\parallel H^+_\Phi\epsilon_\parallel\geq\epsilon^T_\parallel H^{-1}\epsilon_\parallel \quad (47)$$

The equality $=$ holds *if and only if* the subspace spanned by P coincides with the subspace containing $\epsilon_\parallel$. Consider the term $\epsilon^T_\perp\nabla_\theta \mathcal{L}-\frac{1}{2}\epsilon^T_\perp H^{-1}\epsilon_\perp$, we have:

$$\begin{aligned}&\epsilon^T_\perp\nabla_\theta \mathcal{L}-\frac{1}{2}\epsilon^T_\perp H^{-1}\epsilon_\perp=\frac{1}{2}\left[2\epsilon^T_\perp\nabla_\theta \mathcal{L}-\epsilon^T_\perp H^{-1}\epsilon_\perp\right]=\\ &\frac{1}{2}[\nabla_\theta \mathcal{L}^T H\nabla_\theta \mathcal{L}-(\epsilon_\perp-H\nabla_\theta \mathcal{L})^T H^{-1}(\epsilon_\perp-H\nabla_\theta \mathcal{L})]\end{aligned} \quad (48)$$

For convenience, π is used to denote $\nabla_\theta \mathcal{L}$, and x to denote $\epsilon_\perp$. Expanding the quadratic term, we can have:

$$\begin{aligned}(x-H\pi)^T H^{-1}(x-H\pi)&=x^T H^{-1}x-2\pi^T H^T x+\pi^T\\ H^T A\pi&=x^T H^{-1}x-2\pi^T Hx+\pi^T H\pi\end{aligned} \quad (49)$$

The whole equation becomes:

$$\begin{aligned}&\frac{1}{2}[\pi^T H\pi-x^T H^{-1}x+2\pi^T Hx-\pi^T H\pi]\\ &\qquad=\frac{1}{2}\left[2\pi^T Hx-x^T H^{-1}x\right]\end{aligned} \quad (50)$$

For convenience, we let $y = H^{1/2}x$, $x = H^{-1/2}y$, the equation becomes $\pi^T H^{1/2} y - \frac{1}{2} y^T H^{-2} y$. This formula is a convex quadratic function of y, and its maximum occurs at:

$$\nabla_y = H^{1/2}\pi - H^{-2}y = 0 \Rightarrow y^* = H^{5/2}\pi \tag{51}$$

Then we can have:

$$\begin{aligned} &\pi^T H^{1/2} y^* = \pi^T H^{1/2} H^{5/2} \pi = \pi^T H^3 \pi \\ &(y^*)^T H^{-2} y^* = \pi^T H^{5/2} H^{-2} H^{5/2} \pi = \pi^T H g \end{aligned} \tag{52}$$

Finally, we have:

$$\pi^T H x - \frac{1}{2} x^T H^{-1} x = \pi^T H^3 \pi - \frac{1}{2}\pi^T H \pi > 0 \tag{53}$$

Since $H \succ 0$, it follows that $H^3 \succ H$, which means:

$$\pi^T H^3 g > \pi^T H \pi \Rightarrow \pi^T H^3 \pi - \frac{1}{2}\pi^T H \pi > 0 \tag{54}$$

Finally, we have:

$$\begin{aligned} &\epsilon_\perp^T \nabla_\theta \mathcal{L} - \frac{1}{2}\epsilon_\perp^T H^{-1} \epsilon_\perp \\ &= \frac{1}{2}[\nabla_\theta \mathcal{L}^T H \nabla_\theta \mathcal{L} \\ &- (\epsilon_\perp - H\nabla_\theta \mathcal{L})^T H^{-1} (\epsilon_\perp - H \nabla_\theta \mathcal{L})] > 0 \end{aligned} \tag{55}$$

That is to say, $\Delta\mathcal{L}_{\text{peft}} - \Delta\mathcal{L}_{\text{full}} > 0$. □

E.5 Proof of Marginal Benefit of Examples

Proof of Theorem 5. According to Equations 1 to 4, the optimal solutions of PEFT and FFT in Empirical Risk Minimization (ERM) are:

$$\begin{aligned} \hat{\theta}_N &= \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(x_i), y_i) \\ \hat{\Phi}_N &= \arg\min_{\Phi \in \mathbb{R}^k} \frac{1}{N} \sum_{i=1}^{N} \ell(f_{\theta_0 + g(\Phi)}(x_i), y_i). \end{aligned} \tag{56}$$

where N represents the number of training sets. The expected risk of ERM can be decomposed into:

$$\begin{aligned} \underbrace{\mathcal{L}(\hat{\theta}_N)}_{\text{expected error}} - \underbrace{\mathcal{L}^*_{\mathcal{H}}}_{\text{approximation error}} &= \underbrace{\mathcal{L}(\hat{\theta}_N) - \mathcal{L}_N(\hat{\theta}_N)}_{\text{generalization error}} + \\ &\underbrace{\mathcal{L}_N(\hat{\theta}_N) - \mathcal{L}^*_{\mathcal{H}}}_{\text{empirical optimality achievement error}} \end{aligned} \tag{57}$$

where $\mathcal{L}(\hat{\theta}_N)$ represents the loss of the model's true distribution, $\mathcal{L}^*_{\mathcal{H}}$ represents the optimal loss of the model in the hypothesis space $\mathcal{H}$ and $\mathcal{L}_N(\hat{\theta}_N)$ represents the loss on the training sample. We make the assumption that the model has already been trained for many steps on the fine-tuning objective, which means $(\mathcal{L}_N(\hat{\theta}_N) - \mathcal{L}^*_{\mathcal{H}}) \to 0$, then:

$$\underbrace{\mathcal{L}(\hat{\theta}_N)}_{\text{expected error}} = \underbrace{\mathcal{L}(\hat{\theta}_N) - \mathcal{L}_N(\hat{\theta}_N)}_{\text{generalization error}} + \underbrace{\mathcal{L}^*_{\mathcal{H}}}_{\text{approximation error}} \tag{58}$$

Rademacher complexity (Blum, 2007) theory gives us:

$$\mathbb{E}\left[\mathcal{L}(\hat{\theta}_N) - \mathcal{L}_N(\hat{\theta}_N)\right] = O\left(\sqrt{\frac{d}{N}}\right). \tag{59}$$

For FFT, the approximation error and generalization error are:

$$\begin{aligned} A_{full} &= \mathcal{L}^*_\Theta \\ E_{full}(N) &= O\left(\sqrt{(d/N)}\right) \end{aligned} \tag{60}$$

Similarly, for PEFT we have:

$$\begin{aligned} A_{peft} &= \mathcal{L}^*_\Phi \\ E_{peft}(N) &= O(\sqrt{\frac{k}{N}}) \end{aligned} \tag{61}$$

Their expected risks are:

$$\begin{aligned} R_{full} &= A_{full} + E_{full}(N) \\ R_{peft} &= A_{peft} + E_{peft}(N) \end{aligned} \tag{62}$$

The decrease in expected risk resulting from increasing the sample size from N to $N+1$ is characterized by:

$$\begin{aligned} \Delta R_{full}(N) &= R_{full}(N) - R_{full}(N+1) \\ \Delta R_{peft}(N) &= R_{peft}(N) - R_{peft}(N+1) \end{aligned} \tag{63}$$

Ignoring constant factors and taking the limit as $N \to N+1$, we obtain:

$$\begin{aligned} &\frac{d}{dN} E_{\text{full}}(N) \approx -\frac{1}{2}\sqrt{\frac{d}{N}} N^{-3/2} = -O(d\, N^{-3/2}) \\ &\frac{d}{dN} E_{\text{peft}}(N) \approx -O(k\, N^{-3/2}). \end{aligned} \tag{64}$$

The approximation errors A_{full} and A_{peft} are independent of N, so we have:

$$\begin{aligned} \Delta R_{\text{full}}(N) \approx -\frac{d}{dN} E_{\text{full}}(N) &= O(d\, N^{-3/2}) \\ \Delta R_{\text{peft}}(N) &= O(k\, N^{-3/2}) \end{aligned} \tag{65}$$

Since $k \ll d$, we can get:

$$\frac{\Delta R_{\text{full}}(N)}{\Delta R_{\text{peft}}(N)} = O\left(\frac{k}{d}\right) \ll 1 \tag{66}$$

That is, for each additional sample, the risk reduction achieved by PEFT is k/d times that of FFT, which is significantly smaller. □

F Hyperparameter

For LLaMa2 family models on ViGGO, GSM8k, and SQL generation tasks, please use the scripts provided in the link: https://github.com/ray-project/ray/tree/master/doc/source/templates/04_finetuning_llms_with_deepspeed. The parameter configuration of LoRA is as follows: rank is 8, LoRA_alpha is 16, Target_modules list is { "q_proj", "v_proj", "k_proj", "o_proj", "gate_proj", "up_proj", "down_proj", "embed_tokens", "lm_head" }. For prefix-tuning, the virtual vocabulary is set to 20 and the rank of AdaLoRA is 8, LoRA_alpha is 32.

G More Experiments

Experiment	Hyperparameters	Values
RoBERTa-base	Total Batch size	64
	Learning rate	$\{1e{-}5, 2e{-}5, 3e{-}5\}$
	warmup	{0.0, 0.02, 0.06}
	Device	8*GTX 1080Ti (11G)
	Weight Decay	0
RoBERTa-large	Total Batch size	32
	Learning rate	$\{1e{-}5, 2e{-}5, 3e{-}5\}$
	warmup	{0.0, 0.02, 0.06}
	Device	8*GTX 1080Ti (11G)
	Weight Decay	0
OPT-13B	Batch size	$\{2, 4, 8\}$
	Learning Rates	$\{1e{-}5, 2e{-}5, 5e{-}5, 8e{-}5\}$
	Device	A100 (80G)
	Weight Decay	0
Mistral-7B	Batch size	$\{2, 4, 8\}$
	Learning Rates	$\{1e{-}5, 2e{-}5, 5e{-}5\}$
	Device	A100 (80G)
	Weight Decay	0
TinyLLaMA	Batch size	$\{2, 4, 8\}$
	Learning Rates	$\{2e{-}5, 5e{-}5, 8e{-}5\}$
	Device	A100 (80G)
	Weight Decay	0
LLaMA2-7B	Batch size	$\{2, 4, 8\}$
	Learning Rates	$\{1e{-}5, 2e{-}5, 5e{-}5, 8e{-}5\}$
	Device	A100 (80G)
	Weight Decay	0
LLaMA2-13B	Batch size	$\{2, 4, 8\}$
	Learning Rates	$\{1e{-}5, 2e{-}5, 5e{-}5, 8e{-}5\}$
	Device	A100 (80G)
	Weight Decay	0
LLaMA2-70B	Batch size	$\{2, 4, 8\}$
	Learning Rates	$\{1e{-}5, 2e{-}5, 5e{-}5, 8e{-}5\}$
	Device	A100 (80G)
	Weight Decay	0

Table 6: The hyperparameter grids used for HiFT experiments.

Methods	LoRA(GB)	IA3(GB)	Prefix(GB)	HiFT(GB)	FFT(GB)
RoBbase	2.63	2.70	2.66	**2.62**	5.67
RoBlarge	6.95	7.13	6.64	**6.62**	15.25
LLaMA2-7B	43.24	43.22	40.69	**40.11**	80+

Table 7: The GPU memory required for different fine-tuning methods when the batch size is 8 and the sentence length is 512.

k	Method	SST-2	EmoC	TREC	Amazon	AGNews
5	LoRA	**95.42**(1.4)	64. 20(0.9)	**88.40**(0.8)	91.80(1.7)	86.60(1.5)
	Prefix	50.96(1.6)	58.56(1.3)	21.36(0.7)	49.36(0.9)	25.78(0.8)
	Adapter	50.92(2.1)	**84.05**(1.5)	18.80(1.2)	49.45(1.5)	24.80(0.9)
	FFT	94.63(1.1)	61.92(0.8)	81.72(0.5)	**95.86**(1.2)	**87.58**(0.7)
10	LoRA	**94.73**(1.0)	63.00(1.5)	**92.80**(0.9)	92.60(1.8)	**90.40**(1.5)
	Prefix	50.8(0.9)	76.98(1.5)	21.20(0.8)	51.42(1.5)	26.44(1.8)
	Adapter	50.92(1.2)	**85.60**(1.9)	41.00(1.2)	52.20(1.3)	52.15(2.2)
	FFT	92.91(0.8)	68.06(1.0)	84.24(1.1)	**96.22**(1.1)	88.64(1.3)
20	LoRA	**95.64**(0.8)	70.80(0.8)	83.60(1.3)	**96.20**(1.3)	**90.60**(2.1)
	Prefix	50.57(0.9)	78.70(1.7)	27.92(1.9)	52.08(1.5)	26.30(1.8)
	Adapter	50.92(1.1)	**85.80**(1.5)	18.80(1.9)	56.40(1.8)	24.80(1.8)
	FFT	95.32(0.4)	69.96(1.2)	**88.08**(1.0)	95.52(1.6)	89.04(1.9)

k	Method	SST-2	EmoC	TREC	Amazon	AGNews
50	LoRA	93.12(1.5)	72.40(1.6)	**94.40**(1.9)	95.40(1.8)	**91.60**(1.5)
	Prefix	50.48(1.8)	76.22(2.1)	28.08(2.5)	50.96(2.5)	27.60(2.7)
	Adapter	50.92(2.0)	**76.80**(1.3)	44.40(2.9)	49.45(1.9)	33.45(2.1)
	FFT	**95.46**(1.6)	74.20(1.5)	91.92(1.7)	**95.82**(1.8)	90.48(1.9)
100	LoRA	92.66(1.4)	**86.60**(1.0)	94.80(1.9)	95.40(2.7)	67.60(2.9)
	Prefix	49.11(1.8)	76.20(1.5)	40.28(2.5)	52.38(2.3)	26.82(2.4)
	Adapter	58.83(1.5)	84.95(1.7)	84.00(2.2)	68.10(3.1)	24.80(2.5)
	FFT	**95.07**(1.0)	76.06(1.3)	**96.20**(1.7)	**96.20**(2.5)	**91.04**(1.9)
200	LoRA	91.29(1.2)	**86.80**(2.1)	93.60(1.5)	95.80(2.7)	90.40(2.1)
	Prefix	48.35(0.9)	81.72(3.2)	45.68(2.3)	52.28(2.3)	27.54(1.7)
	Adapter	50.95(1.5)	85.05(2.8)	88.20(2.8)	49.45(2.8)	81.50(2.4)
	FFT	**95.64**(0.8)	79.90(2.3)	**96.76**(2.1)	**96.12**(3.5)	**91.44**(2.2)

Table 8: The impact of training samples on the performance of different fine-tuning methods. k denotes the number of training examples per class. The base model is LLaMA2-7B.

Models	SST-2		QQP		MNLI		MNLI-MM		QNLI		RTE		Average	
	Orign	Adv	Orign	Adv	Orign	Adv	Orign	Adv	Orign	Adv	Orign	Adv	Origin	Adv
RoBbase(Top1)	87.38	35.13	88.48	**41.02**	73.10	**36.36**	73.09	**31.48**	82.97	41.21	55.59	33.33	76.77	36.42
RoBbase(Top3)	91.85	**37.16**	90.23	39.74	81.87	26.44	81.60	29.62	89.01	34.45	66.06	23.45	83.44	31.81
RoBbase(Top6)	92.77	33.78	90.87	39.74	84.48	31.40	83.67	20.37	90.77	35.13	72.20	33.33	85.79	32.29
RoBbase(FPFT)	**94.80**	34.62	**91.90**	39.48	**87.60**	31.48	**86.58**	30.04	**92.80**	**42.12**	**78.70**	**41.75**	**88.73**	**36.58**
RoBlarge(Top1)	82.91	47.97	87.72	43.58	70.30	36.36	71.05	33.95	81.25	45.94	52.70	43.20	74.32	41.83
RoBlarge(Top3)	95.75	**60.13**	91.60	50.00	88.78	**56.19**	**88.58**	40.12	87.91	40.54	56.31	37.03	84.82	47.34
RoBlarge(Top6)	89.90	33.78	90.46	53.84	86.22	40.49	85.89	33.95	89.18	35.81	58.48	37.03	83.36	39.15
RoBlarge(FFT)	**96.40**	58.65	**92.20**	**57.44**	**90.20**	54.54	88.50	**40.24**	**94.70**	**52.64**	**86.60**	**69.13**	**91.43**	**55.44**

Table 9: The effect of the amount of fine-tuning parameters on model performance under standard fine-tuning on AdvGLUE.

Models	AS		AA		AAE		AAC		ANA		Average		Origin	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
RoBbase(Top1)	36.37	45.22	44.30	53.40	37.19	50.63	43.94	54.36	44.35	57.47	41.23	52.22	53.96	66.18
RoBbase(Top3)	42.69	50.74	54.70	62.88	46.04	59.96	53.39	62.18	56.52	68.28	50.67	60.81	67.62	77.55
RoBbase(Top6)	52.83	61.37	67.65	77.25	50.60	66.07	57.73	65.99	59.13	71.35	57.59	68.41	73.79	84.12
RoBbase(FPFT)	**63.70**	**71.09**	**76.50**	**84.37**	**58.63**	**73.72**	**73.83**	**80.88**	**77.64**	**86.16**	**70.06**	**79.24**	**82.31**	**89.93**
RoBlarge(Top1)	33.39	42.46	39.25	48.87	33.42	46.24	39.30	49.70	39.01	52.18	36.87	47.89	49.34	62.04
RoBlarge(Top3)	62.86	70.41	78.00	85.73	58.36	74.05	**76.71**	**83.76**	75.72	85.45	70.33	79.88	81.59	89.87
RoBlarge(Top6)	62.86	71.35	75.80	84.11	55.90	71.25	70.76	79.97	73.02	84.05	67.67	78.15	77.72	87.53
RoBlarge(FFT)	**71.63**	**77.67**	**83.90**	**90.69**	**63.78**	**78.22**	75.12	81.16	**79.90**	**87.28**	**74.87**	**83.00**	**85.71**	**92.39**

Table 10: The effect of the amount of fine-tuning parameters on model performance under standard fine-tuning on the Adversarial SQuAD dataset.