Tiny Budgets, Big Gains: Parameter Placement Strategy in Parameter Super-Efficient Fine-Tuning

Jinman Zhao[1], Xueyan Zhang[2], Jiaru Li[3], Jingcheng Niu[1,4], Yulan Hu[5], Erxue Min[6], Gerald Penn[1]
[1]Department of Computer Science, University of Toronto, [2]University of Waterloo, [3]CIERA - Northwestern University, [4]UKP Lab, Technical University of Darmstadt, [5]Gaoling School of Artificial Intelligence, Renmin University of China [6]Baidu Inc.
{jzhao,gpenn}@cs.toronto.edu

Abstract

In this work, we propose FoRA-UA, a novel method that, using only 1–5% of the standard LoRA's parameters, achieves state-of-the-art performance across a wide range of tasks. Specifically, we explore scenarios with extremely limited parameter budgets and derive two key insights: (1) fix-sized sparse frequency representations approximate small matrices more accurately; and (2) with a fixed number of trainable parameters, introducing a smaller intermediate representation to approximate larger matrices results in lower construction error. These findings form the foundation of our FoRA-UA method. By inserting a small intermediate parameter set, we achieve greater model compression without sacrificing performance. We evaluate FoRA-UA across diverse tasks, including natural language understanding (NLU), natural language generation (NLG), instruction tuning, and image classification, demonstrating strong generalisation and robustness under extreme compression.[1]

1 Introduction

Ever since the success demonstrated by prior Parameter-Efficient Fine-Tuning (PEFT) work (Houlsby et al., 2019; Hu et al., 2022, *inter alia*), it has become standard practice to apply some form of PEFT technique when fine-tuning large language models (LLMs).

These techniques, including Adapters (Houlsby et al., 2019), Low-Rank Adaption (LoRA; Hu et al., 2022) and Prefix-Tuning (Li and Liang, 2021), allow us to update only a fraction of the original model's weights while achieving comparable domain or task adaptation to full-model fine-tuning. This leads to substantial savings in memory and computational resources. PEFT approaches are thus crucial for the rapid development of new LLMs and LLM-based methods, playing a key role in democratising the field of Artificial Intelligence (AI) and Natural Language Processing (NLP).

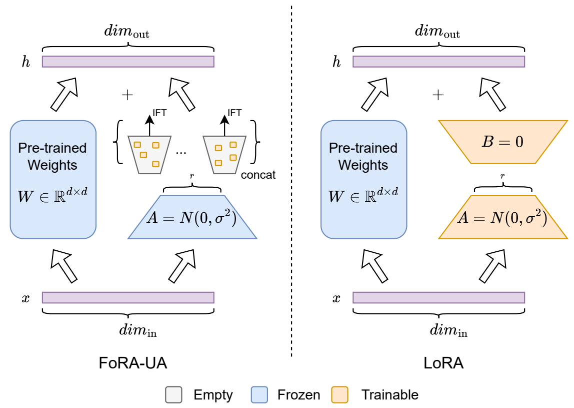


Figure 1: Illustration of the FoRA-UA Architecture. Compared to traditional LoRA approaches, FoRA-UA incorporates a number of smaller matrices for future parameter comparison without sacrificing performance.

Nevertheless, the speed at which LLMs are scaling up is unprecedented and unanticipated. It is now commonplace for a "standard" model to comprise hundreds of billions of parameters. For example, the LLaMA 3.1 (Grattafiori et al., 2024) model contains 405B active parameters, and the flagship Qwen 3 model has 235B (Yang et al., 2025a). Ironically, versions with only a few billion parameters are now referred to as "small language models," even though BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) were originally considered "large language models." Therefore, the current rate of compression in traditional LoRA approaches is not sufficient. Existing models are already reaching record-breaking sizes, and there are very few signs that the trend of scaling is slowing down. There is also a recent push towards mobile computing (Xue et al., 2024; Chen and Li, 2024) which prerequisites memory-efficient LLM inference. Furthermore, effective PEFT methods are key to democratising NLP research, enabling members of the research

[1]Code is available at https://github.com/zhaojinm/FoRA-UA.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 6327–6345
November 4-9, 2025 ©2025 Association for Computational Linguistics

community with limited computational resources to participate in cutting-edge work.

Recent trends in parameter-efficient fine-tuning have pushed the limits of efficiency, with some methods (Kopiczko et al., 2024; Gao et al., 2024; Wu et al., 2024b) using even fewer trainable parameters than the lower bound of LoRA with rank=1. This raises a fundamental question: *how should we allocate a fixed amount of trainable parameters under extremely limited budgets?* To explore this, we begin with FourierFT (Gao et al., 2024) as our baseline, and conduct a series of experiments to investigate how different structural choices affect approximation quality and downstream performance. These experiments lead to three key observations that motivate our method design:

Finding 1: Allocating the fixed amount of trainable parameters in smaller matrices leads to more efficient approximation.

Finding 2: Approximating via a smaller, intermediate representation reduces reconstruction error compared to direct approximation.

Finding 3: Performing IFT on multiple small matrices and combining them leads to equal effectiveness theoretically; and, surprisingly, better downstream task performance in practice.

Therefore, instead of using one large matrix to approximate the entire process, why not use multiple smaller ones? The two aforementioned findings form the foundation of our novel method sparse **Fo**urier l**o**w **Ra**nk with **U**niversal shared **A**daptor (FoRA-UA) method. We employ multiple smaller matrices in parallel to approximate the full matrix. Moreover, many of the weights in the smaller matrices do not need to be initialised and can be left empty.

As a result, FoRA-UA achieve the state-of-the-art space compression without any sacrifice in performance. With only 1–5% of original LoRA's number of parameters, we still reproduce the state-of-the-art performance across a wide range of downstream tasks spanning natural language understanding (NLU), natural language generation (NLG), instruction tuning, and image classification, demonstrating its strong generalization and robustness under extreme compression.

Contribution We make three key contributions.

1. We explore scenarios of LoRA-based PEFT with an extremely limited memory budget (§3). We demonstrate two key findings empirically and theoretically that establish the foundation of FoRA-UA. These novel insights have important implications for the broader PEFT community.
2. We introduce FoRA-UA (§4), a novel PEFT method that uses the fewest parameters while still achieving state-of-the-art performance. To date, FoRA-UA achieves the best trade-off between performance and trainable parameters.
3. We thoroughly evaluate FoRA-UA across a wide range of downstream NLP tasks (§5), and even explore multimodal use cases of LLMs. This comprehensive evaluation demonstrates the effectiveness and robustness of our method, showing that it can bring significant performance gains with only a tiny budget.

2 Related Work

Parameter Efficient Fine-Tuning A foundational PEFT method is Adapter Tuning (Houlsby et al., 2019), which inserts small layers into the model while keeping the pre-trained weights frozen. This enables efficient task adaptation with minimal parameter updates. BitFit (Ben Zaken et al., 2022) takes a more extreme approach by only fine-tuning bias terms, demonstrating that even such minimal modifications can yield competitive results.

LoRA (Hu et al., 2022) further improves efficiency by introducing trainable low-rank matrices to modify model weights. Prompt Tuning (Lester et al., 2021; Liu et al., 2022; Jin et al., 2024) and Prefix-Tuning (Li and Liang, 2021) shift the focus from weight updates to learnable input prompts, allowing models to adapt without modifying their core parameters. Zhong et al. (2025) add a nonlinear layer between the original weight and updated weight. Other efficient methods are commenly used such as LLM embedding/steering (Deng et al., 2025a; Cheng et al., 2025; Gan et al., 2025), distillation (Dong et al., 2024b,c,a), KV cache during inference (Li et al., 2025a), knowledge editing (Deng et al., 2025b), token pruning (Yang et al., 2025c; Liu et al., 2025; Hu et al., 2025), finetune partial layers (Fan et al., 2025a) and chunk-wise gradient pruning(Li et al., 2025b).

LoRA and its Variations After their initial success, several extensions of LoRA (Hu et al., 2022) have been proposed. One notable extension is AdaLoRA (Zhang et al., 2023b) which updates the rank dynamically during fine-tuning based on the importance of the parameters. Another variant, DyLoRA (Valipour et al., 2023), extends LoRA by introducing dynamic matrix updates. Dynamic updating has also been integrated into some work (Liu et al., 2024c; Wang et al., 2024; Ding et al., 2023). LoRA+ (Hayou et al., 2024) proved that to achieve optimal, the learning rates for A and B should be different, with A's learning rate being much smaller than B's. LoRA-FA (Zhang et al., 2023a) freeze A and only fine-tune B. However, unlike our method, LoRA-FA does not share the low-rank adaptation matrices across different linear layers and results in a higher memory cost. Some recent work has integrated Mixture of Experts (MoE) with LoRA (Luo et al., 2024; Dou et al., 2024; Qing et al., 2024), enabling more efficient use of model capacity by activating different subsets of parameters depending on the input. Additionally, some methods (Shen et al., 2024; Ren et al., 2024; Mao et al., 2024; Tian et al., 2024) decompose the adaptation matrix into smaller blocks. There are also other LoRA-based methods (Renduchintala et al., 2024; Liu et al., 2024a; Shi et al., 2024; Zhong and Zhou, 2024; Zhang et al., 2025, *inter alia*) that we do not elaborate on here for brevity, but we refer interested readers to explore them further.

Fourier Transformation in PEFT Fourier transformations (FTs) have recently been explored in PEFT to enhance model generalization and efficiency. Borse et al. (2024) introduces low-rank adaptation in the frequency domain, improving generalization by reducing redundancy and enabling adaptive rank selection. Zeng et al. (2024) leverages Fourier transforms in visual prompt tuning, enhancing robustness across datasets with varying disparities. Unlike LoRA decomposes weights into two low-rank matrices, Gao et al. (2024) directly approximates ΔW using an Inverse Fourier Transformation (IFT).

3 LoRA under Extremely Limited Budget

LoRA (Hu et al., 2022) is a SOTA PEFT method designed to adapt large-scale pre-trained models for downstream tasks while reducing memory and computational overhead. Instead of updating all model parameters, LoRA introduces trainable low-rank matrices to approximate weight updates, effectively reducing the number of learnable parameters.

Formally, given a pre-trained weight matrix $W \in \mathbb{R}^{d\times k}$, LoRA models its update as a low-rank

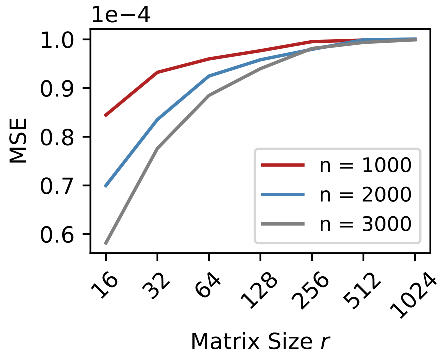


Figure 2: Reconstruction error across different matrix sizes under fixed sparsity.

decomposition,

$$\Delta W = \underline{BA}, \tag{1}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, with $r \ll min(d,k)$ to ensure that the rank of the update remains significantly lower than the original weight matrix. We use underline to denote the *trainable* matrices. The adapted model parameters W' are then represented as:

$$W' = W + \Delta W = W + \underline{BA}. \tag{2}$$

During training, only A and B are updated, while W remains frozen (see the right panel of Figure 1).

Sparse matrices (Ding et al., 2023; Wang et al., 2024) have also been employed in PEFT to further compress the trainable parameter space. Gao et al. (2024) is one of them that uses sparse inverse Fourier transformations to approximate ΔW as:

$$\Delta W = \text{IFT}(\underline{B}) \tag{3}$$

where $B \in \mathbb{R}^{d\times k}$ is a sparse matrix typically with less than 2000 non-zero entries, and IFT stands for the two-dimensional **inverse Fourier transformation**

To better understand how to allocate extremely limited parameter budgets, we designed a series of controlled experiments that reveal several key observations.

3.1 Small Matrices or Large Marices?

Finding 1: *A fixed-size sparse frequency representation (i.e., $B \in \mathbb{R}^{d\times k}$, with n nonzero entries for, e.g., $n = 1000, 2000$) provides more accurate approximations for smaller matrices.*

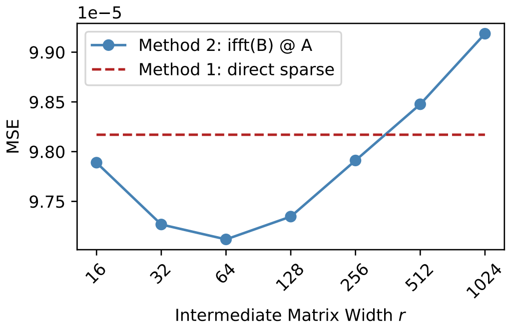


Figure 3: Reconstruction error.

To verify the approximation accuracy of sparse Fourier-domain representations under a fixed budget of non-zero entries, we generated random target matrices $\tilde{B} \in \mathbb{R}^{1024\times r}$ with increasing width $r \in \{16, 32, \dots, 1024\}$, where each entry is sampled from a Gaussian distribution $\mathcal{N}(0; 0.01^2)$.

For each $\tilde{B}$, we initialized a sparse frequency matrix B with exactly $n \in \{1000, 2000, 3000\}$ randomly selected non-zero entries as learnable/trainable parameters. This matrix B is to approximate the Fourier-space representation of the target $\tilde{B}$. Its entries are complex numbers that can be considered as the Fourier coefficients of $\tilde{B}$. The training is performed in the frequency domain directly: the optimal values of the complex coefficient entries are found via gradient descent to minimize the mean squared error (MSE) between $\mathrm{Re}(\mathrm{ifft}(B))$ and the target matrix $\tilde{B}$.

As shown in Figure 2, the reconstruction error increases with the size of the target matrix (i.e., r). This confirms the intuition: larger matrices have higher entry densities, so their Fourier transforms cover wider spectra of frequencies; smaller matrices are the opposite. That is why B better represents smaller $\tilde{B}$ when n is fixed.

3.2 Direct or Indirect Approximation?

Finding 2: *When the number of trainable parameters is fixed, approximating a large matrix via a smaller intermediate representation (e.g., Re($\mathrm{ifft2}(B)) \cdot A$) leads to lower reconstruction error than directly approximating the full matrix.*

We consider the task of reconstructing a structured matrix $W \in \mathbb{R}^{768\times 768}$ under a strict parameter budget: only 1000 trainable entries are allowed. In *Method 1*, we directly approximate W using Equation (3) with 1000 sparse entries in B. In *Method 2*, we instead learn a sparse frequency-domain matrix $B \in \mathbb{C}^{768\times r}$ with the same parameter constraint, and compute $W \approx \mathrm{Re}(\mathrm{ifft2}(B)) \cdot A$, where $A \in \mathbb{R}^{r\times 768}$ is a fixed random projection.

As shown in Figure 3, Method 2 outperforms Method 1 when the intermediate dimension r is small (e.g., $r = 16, 32, 64$). As r increases, the error of Method 2 gradually rises and eventually surpasses the direct method. This supports the hypothesis that, under a fixed parameter budget, it is more effective to allocate capacity toward a compact latent representation than to directly target a large matrix. This trade-off reflects an implicit inductive bias toward low-rank, low-frequency, or compressible structures.

3.3 Split Down Projection

Theorem 3.1. *For any $M \in \mathbb{R}^{a\times b}$, there exist $M_1 \in \mathbb{R}^{a\times b_1}$, $M_2 \in \mathbb{R}^{a\times b_2}$ such that $b_1 + b_2 = b$ and the IFT of M satisfies*

$$\mathrm{IFT}(M) = [\mathrm{IFT}(M_1), \mathrm{IFT}(M_2)]\,, \tag{4}$$

where $[..., ...]$ represents the horizontal concatenation. Similarly, there also exist $M_1' \in \mathbb{R}^{a_1\times b}$, $M_2' \in \mathbb{R}^{a_2\times b}$ such that $a_1 + a_2 = a$ and

$$\mathrm{IFT}(M) = \left[\mathrm{IFT}(M_1'); \mathrm{IFT}(M_2')\right]\,, \tag{5}$$

where $[...; ...]$ represents the vertical concatenation.

Proof. See Appendix G. □

The above theorem shows that further splitting a matrix is theoretically lossless, we also empirically observe in Section 5.7 that such decomposition improve the performance

4 FoRA-UA: Tiny Budget, Big Gains

4.1 Method

Based on the previous findings, our proposed method become:

$$\Delta W = \left[\mathrm{IFT}(\underline{B_1}); ...; \mathrm{IFT}(\underline{B_M})\right] A, \tag{6}$$

Here, $A \in \mathbb{R}^{r\times k}$ is a frozen matrix. $B \in \mathbb{R}^{d\times r}$ is a trainable matrix in the *frequency domain*. This Fourier transformation can improve the training efficiency of our algorithm by allowing it to directly tune the *"spectrum"* encoded in B. We also let the matrix B be **sparse** – during training processes, only the nonzero entries of B need to be updated.

We highlight benefits of our split-down project method:

Computation efficiency: Suppose ΔT is the average timescale to complete an arithmetic operation. Performing the two-dimensional IFT of matrix $B \in \mathbb{R}^{k\times b}$ via a typical fast Fourier transform (FFT) algorithm requires a timescale

$$T_{\text{full}} = [kb\log(kb)]\,\Delta T, \tag{7}$$

while the total time to transform all the B_m's is

$$\begin{aligned} T_{\text{split}} =& [kb_1 \log(kb_1) + \\ & ... + kb_N \log(kb_M)]\Delta T. \end{aligned} \tag{8}$$

One can see that $T_{\text{split}} \leq [kb\log(kb_{\max})]\,\Delta T$, where $b_{\max} = \max(b_1, ..., b_M)$. Hence, T_{split} is shorter than T_{full} because $b_{\max}$ is smaller b.

Flexibility: The method provides the flexibility to optimize each part independently.

4.2 Implementation

The left panel of Figure 1 illustrates the overview of our approach.

We begin by constructing the matrix $A \in \mathbb{R}^{r\times k}$ with its entries randomly drawn from the normal distribution $\mathcal{N}(0, \sigma^2)$. This matrix is frozen during the training steps, meaning the values of its entries remain fixed at all times.

We then initialize a preliminary frequency matrix $B \in \mathbb{R}^{d\times r}$ with entries $B_{i,j}$ set to

$$B_{i,j} = \begin{cases} \mathcal{N}(0,\sigma^2) & \text{if } (i,j) \in S \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

where S contains n randomly selected integer pairs (i, j) representing the indices of B. The smaller spectral matrices $B_m \in \mathbb{R}^{\frac{d}{M}\times n}$ are obtained by equally splitting B into M pieces.

At each training step, we calculate the IFT of B_m matrices using the (inverse) fast Fourier transform algorithm, and obtain the weight ΔW via Equation 6. We then update the non-zero entries of B_m matrices and optimize ΔW.

4.3 Parameter Counting

Let L denote the number of layers, and d the dimension of internal hidden vectors. For LoRA, the number of trainable parameters for a single target module (e.g. *key, value*) is $L \times r \times d \times 2$, where r is the rank of the low-rank matrices. For our approach, let n represent the number of nonzero entries in the matrices B_i, the number of trainable parameters for a single target module for FoRA-UA is $n \times L$. Empirically, n is typically chosen to be less than 2000. Taking RoBERTa-Large as an example, Table 1 shows the parameters required for fine-tuning. We observe that the number of trainable parameters selected by FoRA-UA is significantly smaller than that of LoRA.

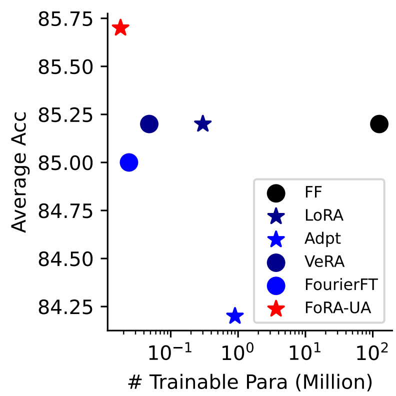


Figure 4: RoBERTa result visualize.

Model	LoRA		FoRA-UA	
	r	#para	n	#para
Rob_large	1	98.3K	200	9.6k
	8	786.4K	500	24k

Table 1: Trainable parameter counting.

5 Results

5.1 Experimental Setup

Benchmark Selection We evaluate our method on a diverse set of tasks spanning multiple domains, including natural language understanding with GLUE (Wang et al., 2018), natural language generation with E2E (Novikova et al., 2017), mathematical reasoning with GSM8k (Cobbe et al., 2021), SingleEq (Koncel-Kedziorski et al., 2015), MultiArith (Roy and Roth, 2015), and SVAMP (Patel et al., 2021), and vision classification with CIFAR100 (Krizhevsky et al., 2009), Food-101 (Bossard et al., 2014), Flowers-102 (Nilsback and Zisserman, 2008) and RESISC45 (Cheng et al., 2017).

Statistics of all benchmarks are listed in Appendix F.

Model Selection We apply our method to a wide variety of model architectures:

1. Encoder-based models such as RoBERTa-base and RoBERTa-large (Liu et al., 2020).

Model	Method	# Trainable Parameters	SST-2 (Acc.)	MRPC (Acc.)	CoLA (MCC)	QNLI (Acc.)	RTE (Acc.)	STS-B (PCC)	Avg.
RoBERTa-base	FF	125M	94.8	90.2	63.6	92.8	78.7	91.2	85.2
	Adpt^{D}	0.3M	$94.2_{\pm0.1}$	$88.5_{\pm1.1}$	$60.8_{\pm0.4}$	$93.1_{\pm0.1}$	$71.5_{\pm2.7}$	$89.7_{\pm0.3}$	83.0
	Adpt^{D}	0.9M	$94.7_{\pm0.3}$	$88.4_{\pm0.7}$	$62.6_{\pm0.6}$	$93.0_{\pm0.2}$	$75.9_{\pm2.2}$	$90.3_{\pm0.4}$	84.2
	LoRA	0.3M	$\mathbf{95.1}_{\pm0.2}$	$89.7_{\pm0.7}$	$63.4_{\pm1.2}$	$\mathbf{93.3}_{\pm0.3}$	$78.4_{\pm0.8}$	$91.5_{\pm0.2}$	85.2
	AdaLoRA	0.3M	$94.5_{\pm0.2}$	$88.7_{\pm0.6}$	$62.0_{\pm0.4}$	$93.1_{\pm0.2}$	$\mathbf{81.0}_{\pm0.6}$	$90.5_{\pm0.2}$	85.0
	VeRA	0.048M	$94.6_{\pm0.1}$	$89.5_{\pm0.5}$	$65.6_{\pm0.8}$	$91.8_{\pm0.2}$	$78.7_{\pm0.7}$	$90.7_{\pm0.2}$	85.2
	FourierFT	0.024M	$94.2_{\pm0.3}$	$90.0_{\pm0.8}$	$63.8_{\pm1.6}$	$92.2_{\pm0.2}$	$79.1_{\pm0.5}$	$90.8_{\pm0.2}$	85.0
	RED	0.02M	$93.9_{\pm0.3}$	$89.2_{\pm1.0}$	$61.0_{\pm3.0}$	$90.7_{\pm0.4}$	$78.0_{\pm2.1}$	$90.4_{\pm0.3}$	83.9
	FoRA-UA	0.018M	$94.7_{\pm0.2}$	$\mathbf{90.8}_{\pm0.8}$	$\mathbf{66.2}_{\pm1.5}$	$91.7_{\pm0.4}$	$79.0_{\pm1.2}$	$\mathbf{91.6}_{\pm0.1}$	**85.7**
RoBERTa-large	FF	356M	96.4	90.9	68.0	94.7	86.6	92.4	**88.2**
	Adpt^{P}	3M	$96.1_{\pm0.3}$	$90.2_{\pm0.7}$	$68.3_{\pm1.0}$	$\mathbf{94.8}_{\pm0.2}$	$83.8_{\pm2.9}$	$92.1_{\pm0.7}$	87.6
	Adpt^{P}	0.8M	$\mathbf{96.6}_{\pm0.2}$	$89.7_{\pm1.2}$	$67.8_{\pm2.5}$	$\mathbf{94.8}_{\pm0.3}$	$80.1_{\pm2.9}$	$91.9_{\pm0.4}$	86.8
	Adpt^{H}	6M	$96.2_{\pm0.3}$	$88.7_{\pm2.9}$	$66.5_{\pm4.4}$	$94.7_{\pm0.2}$	$83.4_{\pm1.1}$	$91.0_{\pm1.7}$	86.8
	Adpt^{H}	0.8M	$96.3_{\pm0.5}$	$87.7_{\pm1.7}$	$66.3_{\pm2.0}$	$94.7_{\pm0.2}$	$72.9_{\pm2.9}$	$91.5_{\pm0.5}$	84.9
	LoRA	0.8M	$96.2_{\pm0.5}$	$90.2_{\pm1.0}$	$68.2_{\pm1.9}$	$\mathbf{94.8}_{\pm0.3}$	$85.2_{\pm1.1}$	$92.3_{\pm0.5}$	87.8
	VeRA	0.061M	$96.1_{\pm0.1}$	$90.9_{\pm0.7}$	$68.0_{\pm0.8}$	$94.4_{\pm0.2}$	$85.9_{\pm0.7}$	$91.7_{\pm0.58}$	87.8
	FourierFT	0.048M	$96.0_{\pm0.2}$	$90.9_{\pm0.9}$	$67.1_{\pm1.4}$	$94.4_{\pm0.4}$	$\mathbf{87.4}_{\pm1.6}$	$91.9_{\pm0.4}$	88.0
	RED	0.05M	$96.0_{\pm0.5}$	$90.3_{\pm1.4}$	$68.1_{\pm1.7}$	$93.5_{\pm0.3}$	$86.2_{\pm1.4}$	$91.3_{\pm0.2}$	87.6
	FoRA-UA	0.024M	$\mathbf{96.6}_{\pm0.3}$	$\mathbf{91.2}_{\pm0.6}$	$\mathbf{69.0}_{\pm1.6}$	$93.9_{\pm0.2}$	$86.9_{\pm1.6}$	$91.9_{\pm0.2}$	**88.3**

Table 2: Performance comparison of different PEFT methods on the GLUE benchmark. Matthew's correlation coefficient is reported for CoLA, Pearson correlation coefficient for STS-B, and accuracy for all other tasks. Results for FF, LoRA, Adapters, AdaLoRA and FourierFT are taken from the Gao et al. (2024), while those for VeRA and Red are sourced from their respective papers (Kopiczko et al., 2024; Wu et al., 2024b).

2. Decoder-based models, including GPT-2 (Radford et al., 2019) and LLaMA 3 (Touvron et al, 2023).

3. Vision Transformers (ViT) (Dosovitskiy et al., 2021).

Baseline Selection We compare our proposed method with the following PEFT baselines: Full fine-tuning (**FF**), **Adapter Tuning** (Houlsby et al., 2019; Pfeiffer et al., 2021; Rücklé et al., 2021), **LoRA** (Hu et al., 2022), **AdaLoRA** (Zhang et al., 2023b), **VeRA** (Kopiczko et al., 2024), **FourierFT** (Gao et al., 2024), and **Red** (Wu et al., 2024b). **DoRA** (Liu et al., 2024a) **LoRA+** (Hayou et al., 2024)[2]

5.2 Natural Language Understanding

The GLUE (Wang et al., 2018) benchmark is a collection of eight diverse datasets designed to evaluate a model's ability to understand and process natural language (Kou et al., 2024b). It includes tasks such as sentiment analysis (SST-2), natural language inference (MNLI, RTE, QNLI), sentence similarity (MRPC, STS-B, QQP), and linguistic acceptability (CoLA).

Implementation We conduct experiments on both RoBERTa-base and RoBERTa-large models. For RoBERTa-base, we set $n = 750$, and for RoBERTa-large, we set $n = 500$. Our experimental setup follows LoRA (Hu et al., 2022), applying weight updates to the query and value matrices, while fully training the classification head. To ensure consistency with prior work like VeRA (Kopiczko et al., 2024) and FourierFT (Gao et al., 2024), we used separate learning rates for the classification head and the adapted layers. We tuned the learning rate, number of epochs, and rank, with the specific hyperparameter choices detailed in Appendix B.

Tasks like MNLI and QQP contain much larger number of data compared to the other six tasks in GLUE. We choose to omit these two, as prior work has done (Kopiczko et al., 2024; Gao et al., 2024). For each dataset, we conduct five independent runs, each using a randomly selected seed (Kou et al., 2025). For each run, we use the best epoch outcome and report the median across the five runs (Kou et al., 2024a).

Results We present the results in Table 2 and Figure 4, where our method achieves the best average performance across all tasks. Notably, our approach requires the fewest trainable parameters among all methods. For RoBERTa-base, our method uses just 6% of the parameters required by LoRA, while for RoBERTa-large, it requires only 3% of LoRA's parameter number.

[2]A more details summarisation about these methods are in Appendix A.

Model	Method	# Trainable Parameters	BLEU	NIST	METEOR	ROUGE-L	CIDEr
GPT2-Medium	FF*	354.92M	65.95	8.52	45.95	69.13	2.35
	Adpt^H*	0.9M	64.31	8.29	44.91	67.72	2.28
	Adpt^P*	0.8M	64.41	8.30	44.74	67.53	2.29
	LoRA	0.4M	66.86	**8.59**	45.94	69.27	**2.41**
	FourierFT	0.048M	64.89	8.38	43.94	67.11	2.20
	RED	0.050M	64.62	8.33	45.14	67.46	2.25
	LoRA	0.098M	64.96	8.40	45.29	68.03	2.32
	VeRA	0.098M	64.50	8.33	45.38	68.77	2.34
	FourierFT	0.098M	64.46	8.33	45.32	67.80	2.28
	FoRA-UA	0.098M	**67.01**	**8.59**	**46.07**	**69.43**	2.40
	FoRA-UA	0.036M	66.15	8.58	45.12	67.70	2.28
GPT2-Large	FT*	774.03M	65.56	8.50	45.40	68.38	2.27
	Adpt^H*	1.8M	65.94	8.46	45.78	68.65	2.23
	Adpt^P*	1.5M	65.53	8.41	45.65	68.46	2.33
	LoRA	0.77M	68.07	8.74	46.20	**69.92**	**2.43**
	VeRA	0.18M	66.72	8.56	46.17	69.30	2.40
	FourierFT	0.07M	66.38	8.55	45.70	68.66	2.33
	RED	0.09M	65.22	8.40	45.59	68.14	2.34
	FoRA-UA	0.05M	67.23	8.66	45.81	68.47	2.40
	FoRA-UA	0.15M	**68.20**	**8.76**	**46.24**	69.65	2.42

Table 3: Performance of different PEFT methods on E2E test set via GPT2-medium and GPT2-large. Results with * are taken from Wu et al. (2024b)

Model	Method	# Trainable Parameters	AddSub	SingleEq	MultiArith	SVAMP
LLaMa2-7B	LoRA	8.4M	80.5	77.2	93.8	42.9
	FoRA-UA	0.1M	75.5	75.6	91.8	42.9
LLaMa3-8B	LoRA	8.4M	89.6	96.7	96.3	75.4
	DoRA	7.1M	90.3	96.1	96	76.8
	LoRA+	6.9M	89.3	96.7	98	68.3
	VeRA	0.4M	88.6	95.7	95.3	71.3
	FoRA-UA	0.1M	85.3	97.0	94.7	72.6

Table 4: Accuracy on math reasoning.

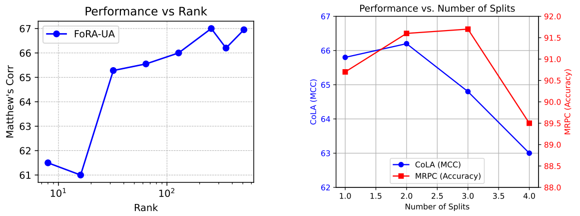
5.3 Natural Language Generation

Implementation We fine-tune the E2E dataset on GPT-2 medium and GPT-2 large models, with detailed hyperparameters provided in Appendix C. Each experiment is conducted three times, and we report the average results. We follow (Li and Liang, 2021) and adopt the script posted by RED (Wu et al., 2024b) and re-ran other PEFT methods for comparison. We tune *key, query, value (i.e. c_attn)* and select the checkpoint with the lowest evaluation loss. For each experiment, we run 3 times and compute the average.

Results Table 3 presents the results on the E2E test set for both GPT2-medium and GPT2-large, demonstrating that our method achieves the smallest parameter size while maintaining competitive performance. For GPT2-medium, our approach outperforms both RED and FourierFT. Additionally, it achieves comparable performance with only 37% of the parameters required by Vera and 9% of those required by Lora. Furthermore, we applied the same number of trainable parameters across different PEFT methods. Specifically, we select $0.098M$ as the number of trainable parameters, which corresponds to the number of parameters Lora needed under the $rank = 1$(smallest) configuration. In this setting, our method surpasses the other three approaches across all five evaluation metrics.

5.4 Reasoning/Instruct Tuning

Implementation We fine-tuned LLaMA2-7B and LLaMA3-8B using the LLM-Adaptor (Hu et al., 2023) framework on the Math-10K dataset. Detailed hyperparameter configurations are provided in Appendix D.



(a) Performance vs. Rank for FoRA-UA on CoLA via RoBERTa-base.

(b) Performance vs. Number of splits.

Figure 5: Comparison of FoRA-UA's effectiveness in terms of rank and number of submatrices(M).

Results Table 4 presents our results across various mathematical benchmarks. Our method achieves performance comparable to LoRA while utilizing only 2% of the parameters that LoRA requires.

5.5 Image Classification

Implementation We use a ViT model pretrained on ImageNet-21K and applied LoRA with a rank of 8 for both ViT-Base and ViT-Large. The only hyperparameter we tuned for our method was the learning rate. Each experiment was run three times, and the results were averaged. For additional hyperparameters, please refer to Appendix E.

Results Table 5 shows that our method achieves performance comparable to full fine-tuning and LoRA in image classification while reducing parameter usage. Specifically, ViT-Base requires only 12.2% of the trainable parameters of LoRA, and ViT-Large utilizes just 6.1% trainable parameters, demonstrating the efficiency of our approach.

5.6 Ablation Study

In this section, we conduct an ablation study to analyze how different factors influence the experimental results. All experiments are repeated five times to ensure statistical reliability. Due to space limitations, we relegate some other studies to Appendix H.

5.6.1 Impact of rank

We conduct these experiments on the CoLA dataset using RoBERTa-base. All experiments were configured with $n = 2000$, and we vary the rank across the following values: $\{16, 32, 64, 128, 256, 350, 512\}$. The results are presented in Figure 5a. From the figure, we observe that performance initially improves as rank increases, with $r = 64$ achieving a Matthew's Correlation of 65.6. However, after $r = 64$, the performance growth slows, indicating diminishing returns. At $r = 256$, performance peaks at 65.55, suggesting an optimal balance between capacity and efficiency. Beyond this point, increasing r further does not lead to improvements. This trend suggests that while higher rank can enhance expressiveness, excessively large ranks may introduce overfitting or optimization instability, leading to degraded generalization.

5.7 Impact of the Number of Splits

To better understand the impact of our split-down projection strategy, we conduct an ablation study on the number of matrix splits used in FoRA-UA on RoBERTa-base and evaluate its effect on downstream performance. As shown in Figure 5b, we observe three key trends. First, introducing a moderate number of splits (e.g., 2 or 3) consistently improves performance over the baseline of a single large matrix. This supports our core motivation that smaller matrices can be more effectively approximated under fixed sparsity constraints. Second, we find that performance degrades sharply when the number of splits becomes too large (e.g., 4), likely because each submatrix receives too few trainable parameters to maintain sufficient representational capacity. Third, the optimal number of splits appears to be task-dependent: while CoLA achieves the highest MCC at 2 splits, MRPC performs best with 3 splits. Please note that the figure does not indicate that our method is unstable. The fluctuations along the y-axis are actually very small; what appears to be a large drop is due to the fact that the difference between the maximum and minimum values is only about 3%.

5.8 Computational Efficiency Analysis

To better understand the computational trade-offs, we compare the FLOPs of different parameter-efficient fine-tuning methods on a single module with hidden size 1024, setting LoRA rank = 16, VeRA rank = 256, and Fora rank = 64. Table 6 summarizes the results. LoRA requires only 16.78M FLOPs, while VeRA incurs a much higher cost of 268M FLOPs due to its larger rank configuration. Our method achieves 67.11M FLOPs, which is higher than LoRA but significantly more efficient than VeRA. This result highlights a favorable balance: although our method introduces additional structural computation compared to LoRA,

Model	Method	# Trainable Parameters	CIFAR100	Food101	Flower102	RESISC45
VIT-B	FF	85.8M	94.53	83.79	98.90	93.07
	LoRA	294.9K	96.21	86.26	100.00	94.81
	FoRA-UA	36K	95.76	83.55	100.00	93.02
VIT-L	FF	303.3M	97.30	88.01	97.06	96.77
	LoRA	786K	96.96	85.69	99.02	96.46
	FoRA-UA	48K	96.64	84.63	100.00	93.49

Table 5: Performance comparison of different PEFT methods on the Image Classification tasks.

Table 6: FLOPs comparison for a single module (hidden size = 1024).

Method	FLOPs
LoRA	16.8 M
VeRA	268.0 M
Ours	67.1 M

the increase remains relatively modest (67.11M vs. 16.78M). At the same time, it avoids the steep overhead of VeRA, which is over **four times** more expensive than ours. Therefore, our approach provides a practical compromise between parameter efficiency and computational cost, offering stronger representational capacity than LoRA without incurring the prohibitive FLOPs of VeRA.

6 Conclusion

We have proposed FoRA-UA, an extremely memory-efficient PEFT method designed to reduce the number of trainable parameters with minimal cost to the performance of other methods. Our experimental results demonstrate that FoRA-UA excels across multiple tasks, including natural language understanding, natural language generation, and image classification while offering superior efficiency compared to existing PEFT methods such as LoRA and VeRA. For instance, with RoBERTa-base, FoRA-UA requires only 6% of the parameters of LoRA, outperforming several current methods.

Limitations

We have demonstrated the effectiveness of our proposed method across a variety of NLP tasks and a representative vision task, we have not yet evaluated its applicability to broader domains such as Code understanding (Wu et al., 2025), VLMs (Yang et al., 2025b; Fan et al., 2025b), RAG (Liu et al., 2024b) or multimodal tasks (Wu et al., 2024a; Li, 2024). Furthermore, although FoRA-UA performs well on standard PEFT benchmarks and model sizes, its scalability to larger-scale model remains an open question. Investigating whether the observed efficiency gains persist in high-capacity, high-throughput settings is an important direction for future work.

References

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, Dublin, Ireland. Association for Computational Linguistics.

Shubhankar Borse, Shreya Kadambi, Nilesh Prasad Pandey, Kartikeya Bhardwaj, Viswanath Ganapathy, Sweta Priyadarshi, Risheek Garrepalli, Rafael Esteves, Munawar Hayat, and Fatih Porikli. 2024. FouRA: Fourier low-rank adaptation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13*, pages 446–461. Springer.

Wei Chen and Zhiyuan Li. 2024. Octopus v2: On-device language model for super agent. *Preprint*, arXiv:2404.01744.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. 2017. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883.

Zifeng Cheng, Zhonghui Wang, Yuchen Fu, Zhiwei Jiang, Yafeng Yin, Cong Wang, and Qing Gu. 2025. Contrastive prompting enhances sentence embeddings in llms through inference-time steering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025*, pages 3475–3487.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias

Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Jingcheng Deng, Zhongtao Jiang, Liang Pang, Liwei Chen, Kun Xu, Zihao Wei, Huawei Shen, and Xueqi Cheng. 2025a. Following the autoregressive nature of llm embeddings via compression and alignment. *Preprint*, arXiv:2502.11401.

Jingcheng Deng, Zihao Wei, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi Cheng. 2025b. Everything is editable: Extend knowledge editing to unstructured data in large language models. In *The Thirteenth International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4133–4145, Singapore. Association for Computational Linguistics.

Junhao Dong, Piotr Koniusz, Junxi Chen, and Yew-Soon Ong. 2024a. Adversarially robust distillation by reducing the student-teacher variance gap. In *European Conference on Computer Vision*, pages 92–111. Springer.

Junhao Dong, Piotr Koniusz, Junxi Chen, Z Jane Wang, and Yew-Soon Ong. 2024b. Robust distillation via untargeted and targeted intermediate adversarial samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28432–28442.

Junhao Dong, Piotr Koniusz, Junxi Chen, Xiaohua Xie, and Yew-Soon Ong. 2024c. Adversarially robust few-shot learning via parameter co-distillation of similarity and class concept learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28535–28544.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. LoRAMoE: Alleviating world knowledge forgetting in large language models via MoE-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1932–1945, Bangkok, Thailand. Association for Computational Linguistics.

Yuchun Fan, Yongyu Mu, YiLin Wang, Lei Huang, Junhao Ruan, Bei Li, Tong Xiao, Shujian Huang, Xiaocheng Feng, and Jingbo Zhu. 2025a. SLAM: Towards efficient multilingual reasoning via selective language alignment. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9499–9515, Abu Dhabi, UAE. Association for Computational Linguistics.

Yuchun Fan, Yilin Wang, Yongyu Mu, Lei Huang, Bei Li, Xiaocheng Feng, Tong Xiao, and Jingbo Zhu. 2025b. Language-specific layer matters: Efficient multilingual enhancement for large vision-language models. *Preprint*, arXiv:2508.18381.

Jinwei Gan, Zifeng Cheng, Zhiwei Jiang, Cong Wang, Yafeng Yin, Xiang Luo, Yuchen Fu, and Qing Gu. 2025. Steering when necessary: Flexible steering large language models with backtracking. *CoRR*, abs/2508.17621.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. 2024. Parameter-efficient fine-tuning with discrete fourier transform. In *Forty-first International Conference on Machine Learning*.

Aaron Grattafiori, Abhimanyu Dubey, and Abhinav Jauhri et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. LoRA+: Efficient low rank adaptation of large models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 17783–17806. PMLR.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

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 *International Conference on Learning Representations*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy 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*, pages 5254–5276, Singapore. Association for Computational Linguistics.

Zhiyuan Hu, Yuliang Liu, Jinman Zhao, Suyuchen Wang, WangYan WangYan, Wei Shen, Qing Gu, Anh Tuan Luu, See-Kiong Ng, Zhiwei Jiang, and Bryan Hooi. 2025. LongRecipe: Recipe for efficient long context generalization in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11857–11870, Vienna, Austria. Association for Computational Linguistics.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. 2024. Disentangling memory and reasoning ability in large language models. *Preprint*, arXiv:2411.13504.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*.

Zhiqiang Kou, Jing Wang, Yuheng Jia, and Xin Geng. 2024a. Inaccurate label distribution learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(10):10237–10249.

Zhiqiang Kou, Jing Wang, Yuheng Jia, Biao Liu, and Xin Geng. 2025. Instance-dependent inaccurate label distribution learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(1):1425–1437.

Zhiqiang Kou, Jing Wang, Jiawei Tang, Yuheng Jia, Boyu Shi, and Xin Geng. 2024b. Exploiting multi-label correlation in label distribution learning. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 4326–4334.

Alex Krizhevsky, Geoffrey Hinton, and 1 others. 2009. Learning multiple layers of features from tiny images.

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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Lei Li. 2024. Cpseg: Finer-grained image semantic segmentation via chain-of-thought language prompting. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 513–522.

Wenhao Li, Yuxin Zhang, Gen Luo, Haiyuan Wan, Ziyang Gong, Fei Chao, and Rongrong Ji. 2025a. Spotlight attention: Towards efficient llm generation via non-linear hashing-based kv cache retrieval. *Preprint*, arXiv:2508.19740.

Wenhao Li, Yuxin Zhang, Gen Luo, Daohai Yu, and Rongrong Ji. 2025b. Training long-context llms efficiently via chunk-wise optimization. *arXiv preprint arXiv:2505.16710*.

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, Online. Association for Computational Linguistics.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024a. Dora: Weight-decomposed low-rank adaptation. In *ICML*.

Wanlong Liu, Junying Chen, Ke Ji, Li Zhou, Wenyu Chen, and Benyou Wang. 2024b. Rag-instruct: Boosting llms with diverse retrieval-augmented instructions. *arXiv preprint arXiv:2501.00353*.

Wanlong Liu, Junxiao Xu, Fei Yu, Yukang Lin, Ke Ji, Wenyu Chen, Yan Xu, Yasheng Wang, Lifeng Shang, and Benyou Wang. 2025. Qfft, question-free fine-tuning for adaptive reasoning. *arXiv preprint arXiv:2506.12860*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach.

Zeyu Liu, Souvik Kundu, Anni Li, Junrui Wan, Lianghao Jiang, and Peter Beerel. 2024c. AFLoRA: Adaptive freezing of low rank adaptation in parameter efficient fine-tuning of large models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–167, Bangkok, Thailand. Association for Computational Linguistics.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *Preprint*, arXiv:2402.12851.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. 2024. DoRA: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11662–11675, Bangkok, Thailand. Association for Computational Linguistics.

Maria-Elena Nilsback and Andrew Zisserman. 2008. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Peijun Qing, Chongyang Gao, Yefan Zhou, Xingjian Diao, Yaoqing Yang, and Soroush Vosoughi. 2024. AlphaLoRA: Assigning LoRA experts based on layer training quality. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20511–20523, Miami, Florida, USA. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. 2024. MELoRA: Mini-ensemble low-rank adapters for parameter-efficient fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3052–3064, Bangkok, Thailand. Association for Computational Linguistics.

Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. 2024. Tied-LoRA: Enhancing parameter efficiency of LoRA with weight tying. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8694–8705, Mexico City, Mexico. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ying Shen, Zhiyang Xu, Qifan Wang, Yu Cheng, Wenpeng Yin, and Lifu Huang. 2024. Multimodal instruction tuning with conditional mixture of LoRA. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 637–648, Bangkok, Thailand. Association for Computational Linguistics.

Shuhua Shi, Shaohan Huang, Minghui Song, Zhoujun Li, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. 2024. ResLoRA: Identity residual mapping in low-rank adaption. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8870–8884, Bangkok, Thailand. Association for Computational Linguistics.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Cheng zhong Xu. 2024. HydraloRA: An asymmetric loRA architecture for efficient fine-tuning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Hugo. Touvron et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287, Dubrovnik, Croatia. Association for Computational Linguistics.

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, Brussels, Belgium. Association for Computational Linguistics.

Haoyu Wang, Tianci Liu, Ruirui Li, Monica Xiao Cheng, Tuo Zhao, and Jing Gao. 2024. RoseLoRA: Row and column-wise sparse low-rank adaptation of pre-trained language model for knowledge editing and fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 996–1008, Miami, Florida, USA. Association for Computational Linguistics.

Jie Wu, Haoling Li, Xin Zhang, Jianwen Luo, Yangyu Huang, Ruihang Chu, Yujiu Yang, and Scarlett Li. 2025. Iterpref: Focal preference learning for code generation via iterative debugging. *Preprint*, arXiv:2503.02783.

Jie Wu, Danni Xu, Wenxuan Liu, Joey Zhou, Yew Ong, Siyuan Hu, Hongyuan Zhu, and Zheng Wang. 2024a. Assess and guide: Multi-modal fake news detection via decision uncertainty. In *Proceedings of the 1st ACM Multimedia Workshop on Multi-Modal Misinformation Governance in the Era of Foundation Models*, MIS '24, page 37–44, New York, NY, USA. Association for Computing Machinery.

Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, Changze Lv, Zixuan Ling, Zhu JianHao, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. 2024b. Advancing parameter efficiency in fine-tuning via representation editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13445–13464, Bangkok, Thailand. Association for Computational Linguistics.

Zhenliang Xue, Yixin Song, Zeyu Mi, Xinrui Zheng, Yubin Xia, and Haibo Chen. 2024. Powerinfer-2: Fast large language model inference on a smartphone. *Preprint*, arXiv:2406.06282.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Sihan Yang, Chenhang Cui, Zihao Zhao, Yiyang Zhou, Weilong Yan, Ying Wei, and Huaxiu Yao. 2025b. Improving alignment in lvlms with debiased self-judgment. *arXiv preprint arXiv:2508.20655*.

Sihan Yang, Runsen Xu, Chenhang Cui, Tai Wang, Dahua Lin, and Jiangmiao Pang. 2025c. Vflowopt: A token pruning framework for lmms with visual information flow-guided optimization. *arXiv preprint arXiv:2508.05211*.

Runjia Zeng, Cheng Han, Qifan Wang, Chunshu Wu, Tong Geng, Lifu Huang, Ying Nian Wu, and Dongfang Liu. 2024. Visual fourier prompt tuning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023a. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *Preprint*, arXiv:2308.03303.

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

Xueyan Zhang, Jinman Zhao, Zhifei Yang, Yibo Zhong, Shuhao Guan, Linbo Cao, and Yining Wang. 2025. UORA: Uniform orthogonal reinitialization adaptation in parameter efficient fine-tuning of large models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11709–11728, Vienna, Austria. Association for Computational Linguistics.

Yibo Zhong, Haoxiang Jiang, Lincan Li, Ryumei Nakada, Tianci Liu, Linjun Zhang, Huaxiu Yao, and Haoyu Wang. 2025. Neat: Nonlinear parameter-efficient adaptation of pre-trained models. *Preprint*, arXiv:2410.01870.

Yibo Zhong and Yao Zhou. 2024. Rethinking low-rank adaptation in vision: Exploring head-level responsiveness across diverse tasks. *Preprint*, arXiv:2404.08894.

A Baseline Method

We compare our proposed method with the following PEFT baselines:

1. Full fine-tuning involves updating all parameters of a model on a specific dataset to adapt it for a target task, offering maximum flexibility but requiring substantial resources.

2. Adapter Tuning insert adapters between modules such as attention/FNN (Houlsby et al., 2019) and feed-forward (Pfeiffer et al., 2021). Rücklé et al. (2021) removes the adapters that are inactive.

3. LoRA (Hu et al., 2022) is the SOTA of PEFT that update weight by $W = W_0 + BA$.

4. AdaLoRA (Zhang et al., 2023b) improves upon standard LoRA by dynamically adjusting the rank allocation for different layers or weight matrices during fine-tuning, focusing resources on the most impactful areas.

5. VeRA (Kopiczko et al., 2024) is a LoRA extended method that freezes low-rank matrices A and B, and optimizes coefficient vectors $\vec{b}$ and $\vec{d}$.

6. FourierFT (Gao et al., 2024) unlike LoRA that decomposes weight into two matrices, it treats weight changes as spatial domain matrices and learns only sparse spectral coefficients.

7. Red (Wu et al., 2024b) directly edits neural network representations via learnable scaling and biasing vectors.

B Hyperparameters for GLUE

Table 7 contains the hyperparameters we use for GLUE.

C Hyperparameters for E2E

Table 8 contains the hyperparameters we use for GLUE.

D Hyperparameters for Instruct Tuning

Table 9 contains the hyperparameters we use for Instruct Tuning.

E Hyperparameters for Image Classification

Table 10 contains the hyperparameter we use for Image Classification benchmarks.

F Statistics of Benchmark

See table 11 for GLUE, Table 12 for E2E, Table 13 for MTBench, Tabel 14 for image datasets.

G Proof of Theorem

G.1 Observations about the Fourier Transform

The two-dimensional Discrete Fourier Transform (DFT) of an array x_{n_1,n_2} is given by

$$X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1} \left(\omega_{N_1}^{k_1 n_1} \sum_{n_2=0}^{N_2-1} \left(\omega_{N_2}^{k_2 n_2} x_{n_1,n_2} \right) \right), \tag{10}$$

where $\omega_{N_{1,2}} = \exp\left(-i2\pi/N_{1,2}\right)$.

Let A be a matrix representing x_{n_1,n_2},

$$A = \begin{bmatrix} x_{0,0} & x_{0,1} & \dots & x_{0,N_2-1} \\ x_{1,0} & x_{1,1} & \dots & x_{1,N_2-1} \\ \dots & \dots & \dots & \dots \\ x_{N_1-1,0} & x_{N_1-1,1} & \dots & x_{N_1-1,N_2-1} \end{bmatrix}, \tag{11}$$

and let $\tilde{A}$ be the DFT of A,

$$\tilde{A} = \begin{bmatrix} X_{0,0} & X_{0,1} & \dots & X_{0,N_2-1} \\ X_{1,0} & X_{1,1} & \dots & X_{1,N_2-1} \\ \dots & \dots & \dots & \dots \\ X_{N_1-1,0} & X_{N_1-1,1} & \dots & X_{N_1-1,N_2-1} \end{bmatrix}. \tag{12}$$

Meanwhile, let F_{N_1} and F_{N_2} be two linear transformations

$$F_{N_1} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_{N_1} & \omega_{N_1}^2 & \dots & \omega_{N_1}^{N_1-1} \\ 1 & \omega_{N_1}^2 & \omega_{N_1}^4 & \dots & \omega_{N_1}^{2(N_1-1)} \\ \dots & \dots & \dots & \dots & \\ 1 & \omega_{N_1}^{N_1-1} & \omega_{N_1}^{2(N_1-1)} & \dots & \omega_{N_1}^{(N_1-1)^2} \end{bmatrix}, \tag{13}$$

and

$$F_{N_2} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_{N_2} & \omega_{N_2}^2 & \dots & \omega_{N_2}^{N_2-1} \\ 1 & \omega_{N_2}^2 & \omega_{N_2}^4 & \dots & \omega_{N_2}^{2(N_2-1)} \\ \dots & \dots & \dots & \dots & \\ 1 & \omega_{N_2}^{N_2-1} & \omega_{N_2}^{2(N_2-1)} & \dots & \omega_{N_2}^{(N_2-1)^2} \end{bmatrix}. \tag{14}$$

Note that F_{N_1} is N_1-by-N_1 with entries given by $(F_{N_1})_{k_1,n_1} = \omega_{N_1}^{n_1 k_1}$, while F_{N_2} is N_2-by-N_2 with entries $(F_{N_2})_{k_2,n_2} = \omega_{N_2}^{k_2 n_2}$. It is not hard to see that

$$\tilde{A} = F_{N_1} A F_{N_2} \tag{15}$$

and

$$A = (N_1 N_2)^{-1} F_{N_1}^{\dagger} \tilde{A} F_{N_2}^{\dagger}, \tag{16}$$

where $(\cdot)^{\dagger}$ represents the Hermitian transpose. This shows that, for any N_1-by-N_2 matrix A, we have its Fourier transform

$$\mathrm{FT}(A) = F_{N_1} A F_{N_2} \tag{17}$$

and

$$\mathrm{IFT}(A) = (N_1 N_2)^{-1} F_{N_1}^{\dagger} A F_{N_2}^{\dagger}, \tag{18}$$

where the F matrices are defined as above.

G.2 Proof of Theorem 3.1

Without loss of generality, let B be N-by-$2M$, let B_1 and B_2 be N-by-M. Suppose $\mathrm{IFT}(B) =$

Model	Hyperparameter	STS-B	RTE	MRPC	CoLA	SST-2	QNLI
Both	Optimizer	AdamW					
	LR Schedule	Linear					
	Warmup Ratio	0.06					
	Frequency Bias	False					
	Seed	$\{42, 43, 44, 45, 46\}$					
Base	Epochs	80	70	60	80	35	45
	Learning Rate (FoRA-UA)	0.02	0.1	0.01	0.03	0.01	0.01
	Learning Rate (Head)	0.006	0.01	0.006	0.006	0.006	0.006
	Scaling Value	5	1	5	4	5	3
	r	256	64	256	256	256	64
	Max Seq. Len	512					
	Batch Size	32					
	n	2×375					
Large	Epochs	40	60	45	60	15	30
	Learning Rate (FoRA-UA)	0.01	0.08	0.05	0.01	0.01	0.06
	Learning Rate (Head)	0.01	0.005	0.005	0.01	0.01	0.005
	Scaling Value	5	2	4	3	5	2
	r	256	64	256	64	256	64
	Max Seq. Len	512					
	Batch Size	32					
	n	2×250					

Table 7: Hyperparameter setup for the GLUE benchmark.

Model	Hyperparameter	VeRA	FourierFT	RED	FoRA-UA
Medium	Learning Rate	0.02	0.08	0.06	0.1
	Scaling Value	-	300	-	15
	r	1024	-	-	64
	n	-	2000	-	3×500
Large	Learning Rate	0.006	0.08	0.02	0.02
	Scaling Value	-	300	-	15
	r	1024	-	-	32
	n	-	2000	-	3×500
Both	Label Smooth	0.0			
	Weight Decay	0.0001			
	Batch Size	8			
	Optimizer	Adam			
	epoch	5			
	Warmup Step	500			
	Learning Rate Schedule	Linear			
	Seed	{42,43,44}			

Table 8: Hyperparameter setup for the E2E benchmark.

$[\mathrm{IFT}(B_1), \mathrm{IFT}(B_2)]$, the matrices should satisfy the relation

$$\frac{1}{2} B F_{2M}^{\dagger} = \left[B_1 F_M^{\dagger}, B_2 F_M^{\dagger} \right]. \tag{19}$$

Hyperparameter	LoRA	FoRA-UA
LR Schedule	Linear	
Warmup Ratio	0.06	
Batch Size	4	
Optimizer	Adam	
Epoch	1	
Rank	64	128
n	-	3×1000
Scaling Value	16	20
Learning Rate	4e-4	3e-2

Table 9: Hyperparameter setup for the Instruct tuning.

Model	Hyperparameter	CIFAR100	Food101	Flowers102	RESISC45
Base	Learning Rate	0.06	0.06	0.07	0.08
	Head Learning Rate	0.002			
	Scaling Value	15			
	r	64			
	n	1×1500			
Large	Learning Rate	0.03	0.02	0.02	0.03
	Head Learning Rate	0.004			
	Scaling Value	15			
	r	64			
	n	1×1000			
Both	Weight Decay	0.0			
	Batch Size	128			
	Optimizer	Adam			
	epoch	10			
	Seed	{42,43,44}			

Table 10: Hyperparameter setup for the Image Classfication benchmark.

Dataset	Train	Valid	Labels
CoLA	8.5K	1K	2
SST-2	67K	872	2
MRPC	3.7K	408	2
STS-B	7K	1.5K	-
QQP	364K	40K	2
MNLI	393K	20K	3
QNLI	108K	5.4K	2
RTE	2.5K	278	2
WNLI	634	71	2

Table 11: Statistics of the GLUE benchmark. Task types: Acceptability (CoLA), Sentiment (SST-2), Paraphrase (MRPC, QQP), Similarity (STS-B), and Natural Language Inference (MNLI, QNLI, RTE, WNLI).

Dataset	Train	Valid	Test
E2E	42,061	4,672	4,693

Table 12: Statistics of the E2E dataset. The dataset consists of structured meaning representations paired with natural language descriptions.

Note that the LHS term $\frac{1}{2}BF_{2M}^{\dagger}$ can be split in two N-by-M matrices. Therefore, we have B_1 and B_2 as

$$B_1 = \left[\frac{1}{2}BF_{2M}^{\dagger}\right]_{\text{columns } 0 \text{ to } M} F_M, \tag{20}$$

$$B_2 = \left[\frac{1}{2}BF_{2M}^{\dagger}\right]_{\text{columns } M+1 \text{ to } 2M} F_M. \tag{21}$$

See Figure 6 as an example.

Category	# Questions
Writing	10
Roleplay	10
STEM	10
Humanities	10
Coding	10
Reasoning	10
Mathematics	10
Extraction	10
Total	80

Table 13: Statistics of the MT-Bench dataset. The dataset consists of 80 multi-turn questions across eight categories.

Dataset	# Classes	Train	Test
CIFAR-100	100	50K	10K
Food-101	101	75.5K	25.3K
Flowers-102	102	2K	6K
RESISC45	45	31.5K	10.5K

Table 14: Statistics of image classification datasets used in our experiments. CIFAR-100 consists of 100 object categories, Food-101 includes various food items, Flowers-102 contains 102 flower species, and RESISC45 is a remote sensing dataset with 45 scene classes.

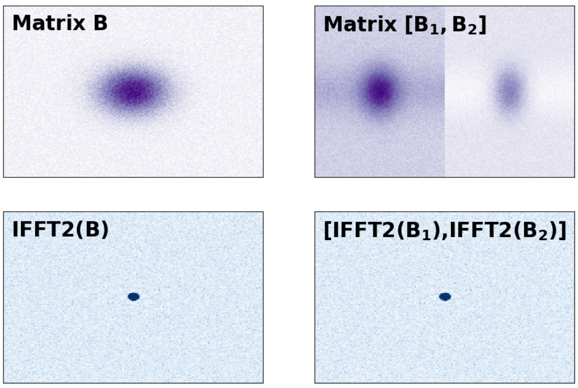


Figure 6: Example of decomposing $\text{IFFT}(B)$ into $[\text{IFFT}(B_1), \text{IFFT}(B_2)]$. The top row shows B and matrix $[B_1, B_2]$. The bottom row, where the colormap shows the absolute values of the IFFT results, confirms that $\text{IFFT}(B) = [\text{IFFT}(B_1), \text{IFFT}(B_2)]$.

Similar proofs also exist for B_1 and B_2 with different sizes and in the case of vertical concatenations.

H More Ablation Studies

H.0.1 Performance *vs.* Trainable Parameters

Figure 8 illustrates the relationship between the number of trainable parameters and model performance on the CoLA dataset. We evaluate FoRA-UA and LoRA, both applied to a RoBERTa-base model. For FoRA-UA, we fix $r = 256$ and vary n across $\{50, 100, 200, 750, 2000, 5000, 10000\}$, while for LoRA, we select rank values $\{1, 2, 4, 6, 8, 15\}$. The x-axis represents the number of trainable parameters on a log scale, and the y-axis reports the corresponding performance. The results indicate that for both FoRA-UA and LoRA, performance improves as the number of trainable parameters increases. However, FoRA-UA consistently outperforms LoRA across all parameter scales, demonstrating its superior efficiency. Notably, FoRA-UA achieves high performance even at relatively small parameter sizes, whereas LoRA requires a larger number of parameters to reach competitive results. This highlights the superior parameter efficiency of FoRA-UA, allowing it to achieve strong results without excessive parameter growth.

H.1 The Importance of A

FourierFT (Gao et al., 2024) simply uses $\text{IFFT}(\Sigma)$ to approximate ΔW. Instead of ours use $\text{IFFT}(\Sigma)$ to approximate the down projection B and then follow the LoRA tradition to learn ΔW. The intuitive behind this is that B is a much smaller matrix than ΔW and a smaller matrix is easier to approximate under the limited trainable parameter condition. To support our statement empirically, we use E2E on GPT2-base as an example. We use the formula $\Delta W = IFT(B)A$ instead of Equation 6 that we use in the experimental part. So the only difference is whether to approximate ΔW directly or approximate B first.

The result is demonstrated in Figure 7. All hypermeter follows Table 8. Note that we pick $n = 2000$ (which is also the #parameter Gao et al. (2024) picked in their experimental part) for FourierFT and $n = 1500$ for FoRA-UA. We can see FoRA-UA consistently achieve better performance across 5 metrics by using 75% #parameters, showing the importance of up projection A.

H.2 Other Transformations

Inverse discrete Fourier transformation of a matrix Σ can also be regarded as matrix multiplications,

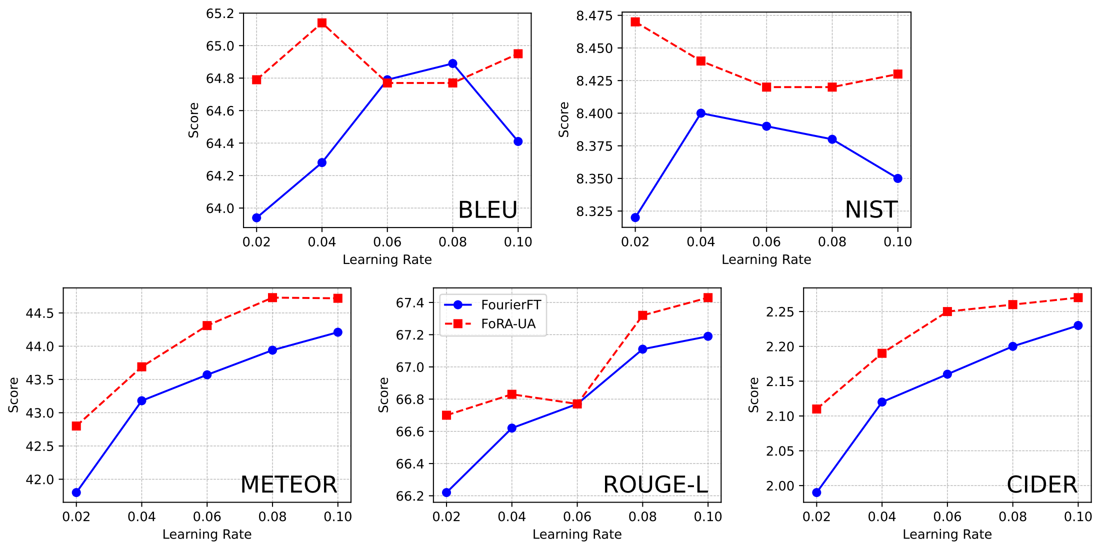


Figure 7: Performance FourierFT and FoRA-UA on E2E, evaluated using 5 different metrics. The solid blue lines with rounded markers represent the scores of FourierFT, while the dashed red lines with squared markers represent the scores of FoRA-UA. FoRA-UA's takes 75% of trainable parameters as FourierFT. This comparison shows that FoRA-UA performs better across all metrics considered, demonstrating the importance of using up projection A.

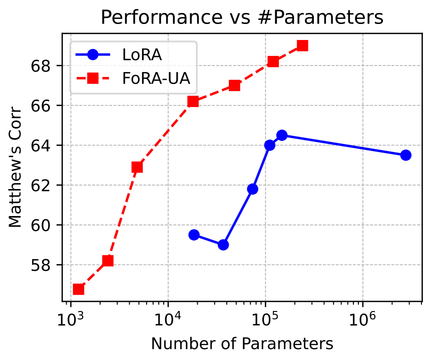


Figure 8: Performance vs. Number of Trainable Parameters for LoRA and FoRA-UA. The x-axis is in the log scale.

$\text{IFFT}(\Sigma) = F_1^\dagger \Sigma F_2^\dagger$. Hence, ΔW can also be written as a product of matrices, $B\Sigma A$, where Σ is trainable, and A and B can take their corresponding values. Results are listed in table 15. We randomly init A and B and try both sparse and dense Σ with a similar number of trainable parameters, ours method archives the best performance and indicates the necessity of the use of inverse Fourier transformations.

Method	MRPC	CoLA
Ours	**91.5**	**67.0**
$B_1\Sigma_1 A_1$	85.8	62.0
$B_2\Sigma_2 A_2$	84.7	60.7

Table 15: Performance comparison of methods on different transformations. The size of B_1 is 768×50 and A_1 is 50×50. Σ_1 is trainable matrix with size 50×50. The size of B_2 is 768×768 and A_2 is 256×768. Σ_2 is trainable *sparse* matrix with size 768×256 and only 2500 nonzero entries.

H.3 Freeze B

We run experiments on MRPC and CoLA with $r = 256$ and $n = 2000$, exploring the impact of freezing different components. As shown in Table 16, freezing A outperforms freezing B, achieving higher scores on both MRPC (91.5) and CoLA (67.0).

Freeze	MRPC	CoLA
A	**91.5**	**67.0**
B	88.5	63.4

Table 16: Performance comparison of methods on freezing A and B on MRPC and CoLA datasets.

H.4 Training Curve

See Figure 9 for Matthew's correlation on CoLA via RoBERTa base. $n = 10000$ for both methods.

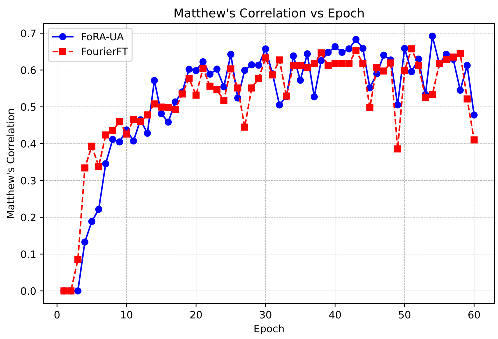


Figure 9: Matthew's correlation comparison between FoRA-UA and FFT.