Parameter-Efficient Fine-Tuning with Discrete Fourier Transform

Ziqi Gao [1 2 *] **Qichao Wang** [3 *] **Aochuan Chen** [1 *] **Zijing Liu** [4] **Bingzhe Wu** [5] **Liang Chen** [3] **Jia Li** [1 2]

Abstract

Low-rank adaptation (LoRA) has recently gained much interest in fine-tuning foundation models. It effectively reduces the number of trainable parameters by incorporating low-rank matrices A and B to represent the weight change, i.e., $\Delta W = BA$. Despite LoRA's progress, it faces storage challenges when handling extensive customization adaptations or larger base models. In this work, we aim to further compress trainable parameters by enjoying the powerful expressiveness of the Fourier transform. Specifically, we introduce FourierFT, which treats ΔW as a matrix in the spatial domain and learns only a small fraction of its spectral coefficients. With the trained spectral coefficients, we implement the inverse discrete Fourier transform to recover ΔW. Empirically, our FourierFT method shows comparable or better performance with fewer parameters than LoRA on various tasks, including natural language understanding, natural language generation, instruction tuning, and image classification. For example, when performing instruction tuning on the LLaMA2-7B model, FourierFT surpasses LoRA with only 0.064M trainable parameters, compared to LoRA's 33.5M. Our code is released at `https://github.com/Chaos96/fourierft`.

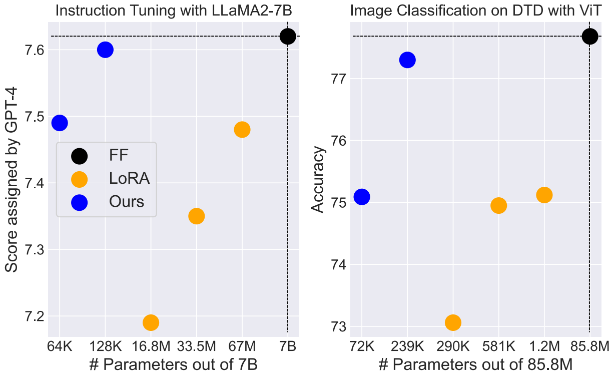


Figure 1. Summary of the performance (**y-axis**) of fine-tuning methods with different numbers (**x-axis**) of trainable parameters on NLP (left) and CV (right) tasks. The left side shows the instruction tuning task, where the LLaMA2-7B model is fine-tuned with Alpaca and evaluated by GPT-4. The right side shows the image classification task, where the Vision Transformer (ViT) is fine-tuned and tested on the DTD dataset. *Black circles* (●) represent the Full Fine-tuning (FF) method. *Orange circles* (●) represent LoRA method with $r = \{32, 64, 128\}$ (left) and $r = \{8, 16, 32\}$ (right). *Blue circles* (●) represent our proposed method with $n = \{1000, 2000\}$ (left) and $n = \{3000, 10000\}$ (right).

1. Introduction

Large foundation models (LFMs) have demonstrated exceptional performance on tasks of multiple domains, including natural language processing (NLP) (Liu et al., 2019; He et al., 2020; Radford et al., 2019; Brown et al., 2020; Li et al., 2022) and computer vision (CV) (Liu et al., 2023a;b; Singh et al., 2022; Rombach et al., 2022). Owing to their impressive capabilities, fine-tuning LFMs for a wide range of downstream tasks has become prevalent (Wang et al., 2022; Taori et al., 2023; Qiu et al., 2020). Under the full fine-tuning paradigm, the new model adapted to each customized task typically contains as many parameters as the original model (Qiu et al., 2020; Raffel et al., 2020; Chen et al., 2024; Gao et al., 2024). As models grow larger and customization needs expand, the demand for storing fine-tuned checkpoints rises, resulting in both costly storage and memory consumption.

As a popular way to address this issue, LoRA (Hu et al., 2021) represents the weight change with two low-rank matrices A and B, i.e., $W_0+\Delta W = W_0+BA$. Despite LoRA's superb performance, its large size of trainable parameters still brings high IT infrastructure consumption, which affects both ends of public communities and individual users. For the former, an intuitive example is that a LoRA adapter (fine-tuned weights) for a specific style of the stable diffusion model (Rombach et al., 2022) requires about 40MB of memory. This necessitates the LFM communities (e.g., Civi-

---

[*] Equal contribution [1]Hong Kong University of Science and Technology (Guangzhou) [2]Hong Kong University of Science and Technology [3]Sun Yat-sen University [4]International Digital Economy Academy [5]AI Lab, Tencent. Correspondence to: Jia Li <jialee@ust.hk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

tai (Civitai, 2024)) to bear high storage and bandwidth costs to cater to a large user base. For the latter, fewer parameters mean direct RAM savings when loading fine-tuned weights in mobile APPs, enabling sufficient customization for individual users (Zhou et al., 2022). To this end, we naturally ask the question: *How can we aggressively compress trainable parameters even further for fine-tuning LFMs?*

Previous works have demonstrated the powerful expressiveness of Fourier basis in data compression, where extremely sparse spectral information can be used to recover high-fidelity data (e.g., 1D signal vectors (Zwartjes & Gisolf, 2007; Duarte & Baraniuk, 2013; Rudelson & Vershynin, 2008) and 2D image matrices (Vlaardingerbroek & Boer, 2013; Song et al., 2021; Shi et al., 2014)). More importantly, when dealing with more general (non-image) matrices that lack strong spatial semantics and are not frequency-sparse, Fourier transform can still handle recovery effectively (Chen & Chi, 2013; Yang & Xie, 2016). Motivated by this, we investigate the potential for updating the weight change ΔW with its sparse spectral coefficients for fine-tuning LFMs.

In this paper, we aim to aggressively reduce the number of trainable parameters for fine-tuning LFMs. To this end, we propose *FourierFT* (Fourier Transform for Fine-Tuning), which treats the weight change ΔW as a matrix in the spatial domain, and learns its sparse spectral coefficients. Specifically, we first randomly select n spectral entries that are shared across all layers. For each layer, FourierFT learns n spectral coefficients located at these n selected entries and then directly applies inverse discrete Fourier transform to compute the updated ΔW. Therefore, fine-tuning a pre-trained model with L_t layers only requires storing $2n$ entry parameters and nL_t coefficient parameters for FourierFT.

Empirically, we compare our method with state-of-the-art LoRA variants and other parameter-efficient fine-tuning methods on various tasks including (1) natural language understanding (on the GLUE benchmark), (2) natural language generation (on the E2E benchmark), (3) instruction tuning (with LLaMA-family models), and (4) image classification (with vision transformers). FourierFT can always achieve comparable or even better performance than LoRA, with about 6.0%, 9.4%, 0.2% and 9.2% of LoRA's trainable parameters for these 4 tasks, respectively. For example in Figure 1, on the instruction tuning task, our FourierFT method outperforms LoRA with only 64K trainable parameters. Moreover, it achieves a comparable score to Full Fine-tuning with only 128K parameters.

2. Related Works

Parameter-Efficient Fine-Tuning. With the rapid expansion of large foundation models (LFM), it has become challenging and important to efficiently adapt them for specific tasks. To this end, numerous methods for parameter-efficient fine-tuning (PEFT) are proposed, demonstrating impressive capabilities in both efficiency and accuracy. Existing PEFT methods are broadly partitioned into two categories: non-weight-based and weight-based methods.

Non-weight-based methods do not optimize pre-trained LFMs at the weight level. Instead, they achieve fine-tunings by introducing additional modules or optimizing prompts and prefixes. Adapter tuning (He et al., 2021; Rebuffi et al., 2017; Pfeiffer et al., 2020; Houlsby et al., 2019; Rücklé et al., 2020; Lin et al., 2020) aims to introduce light-weighted neural modules, called adapters, between pre-trained layers of the base model. These methods keep the pre-trained weights frozen and efficiently fine-tune the adapters for customized tasks. Prompt tuning (Brown et al., 2020; Lester et al., 2021; Gao et al., 2020; Diao et al., 2022) and prefix tuning (Li & Liang, 2021) insert additional prompts or prefix tokens to the layers of the base model. **Weight-based methods**, represented by LoRA (Hu et al., 2021), introduce and then update weight changes that can be merged with the original weights to avoid inference latency. LoRA's innovation lies in the multiplication of low-rank matrices to approximate weight changes. Building upon this, AdaLoRA (Zhang et al., 2023) extends the LoRA method by distributing the parameter budget across weight matrices with importance scores. Additionally, Q-LoRA (Dettmers et al., 2023) proposes to back-propagate gradients upon LoRA through a quantized pre-trained model with 4-bit NormalFloat.

Here, we focus on **weight-based methods** and achieve huge parameter reduction with the powerful expressiveness of Fourier basis, rather than following the low-rank structure.

Sparse Fourier Transform in Deep Learning. Sparse Fourier transform (SFT) has flourished in various fields of deep learning (DL). The SFT technique mainly involves using sparse spectral coefficients of significant (Xu et al., 2020; Ehrlich & Davis, 2019; Gueguen et al., 2018; Tang et al., 2022) or even random (Lin et al., 2014; Rawat et al., 2019; Herrmann, 2010) spectral entries, for representation learning. One important application of this technique is matrix recovery. Patel et al. (2011) designs a gradient-based compressed sensing method to recover images with their sparse Fourier information. Shechtman et al. (2014) proposes an efficient phase retrieval method that improves data recovery using sparse Fourier coefficients. Importantly, previous works (Chen & Chi, 2013; Yang & Xie, 2016; Gao et al., 2022) show that even when the original data is not frequency-sparse, SFT can effectively recover the data with extremely few parameters. Although previous works lack studies on the recovery for the weight matrices of DL models with SFT, the aforementioned methods provide potential support for this work.

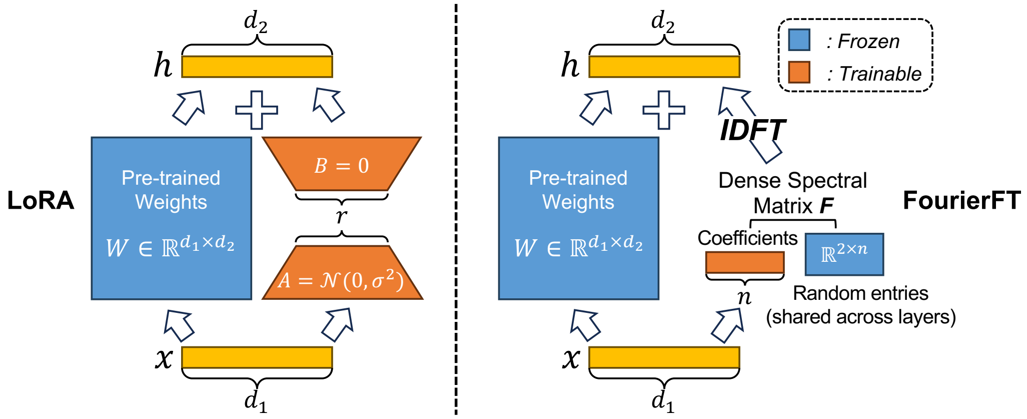

LoRA
d_2
h
Pre-trained Weights
$W \in \mathbb{R}^{d_1 \times d_2}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
x
d_1
d_2
h
: Frozen
: Trainable
IDFT
Dense Spectral Matrix F
Pre-trained Weights
$W \in \mathbb{R}^{d_1 \times d_2}$
FourierFT
Coefficients
$\mathbb{R}^{2 \times n}$
n
Random entries (shared across layers)
x
d_1

Figure 2. **Overview of LoRA (left) and our FourierFT (right) method.** In LoRA, only low-rank (r) matrices A and B are trained. The weight change is represented by their multiplication, i.e., $\Delta W = BA$. For each pre-trained weight W, the theoretical number of trainable parameters in LoRA is $r \times (d_1 + d_2)$. In FourierFT, we first randomly generate the spectral entry matrix $\mathbb{R}^{2 \times n}$, which is shared across all layers to reduce parameter storage requirements. The complete spectral matrix is formed by a trainable coefficient vector $\mathbb{R}^n$ located at selected entries and 0s at the remaining entries. We obtain the weight change ΔW by directly performing inverse discrete Fourier transform (IDFT) on the updated spectral matrix. For all L adapted layers, FourierFT needs to store $n \times (2 + L)$ parameters.

3. Method

We present FourierFT (depicted in Figure 2), a parameter-efficient fine-tuning method based on discrete Fourier transform. FourierFT follows the principle of only learning the change in the pre-trained weight, as proposed by LoRA (Hu et al., 2021). However, unlike LoRA, FourierFT does not adopt the low-rank structure but learns a set of spectral coefficients of Fourier basis. Specifically, we randomly initialize the spectral entry matrix, which is frozen and shared across all layers. We make the spectral coefficients located at selected entries trainable, which jointly form the spectral matrix. Lastly, we apply the inverse discrete Fourier transform to the spectral matrix, yielding its spatial-domain counterpart as the updated weight change.

3.1. Forward Pass

We follow the paradigm of only learning weight changes, as adopted by LoRA-based methods (Hu et al., 2021; Dettmers et al., 2023; Zhang et al., 2023). This can avoid inference latency by merging the pre-trained weight and its change. Formally, we define each pre-trained weight matrix as $W_0 \in \mathbb{R}^{d_1 \times d_2}$, and the weight change for fine-tuning as $\Delta W \in \mathbb{R}^{d_1 \times d_2}$. LoRA aims to parameterize ΔW in the form of low-rank decomposition in the forward pass:

$$h = W_0 x + \Delta W x = W_0 x + BAx, \tag{1}$$

where $B \in \mathbb{R}^{d_1 \times r}$ and $A \in \mathbb{R}^{r \times d_2}$ with the rank $r \ll \min(d_1, d_2)$ are trainable matrices.

The advantage of FourierFT is that the orthogonal and expressive Fourier basis enables recovery of informative weight changes. This promisingly suggests achieving comparable performance to LoRA with significantly fewer parameters. We first randomly initialize the entry matrix $E \in \mathbb{R}^{2 \times n}$ containing discrete 2D spectral entries. Then we randomly initialize the coefficients $c \in \mathbb{R}^n$ with a normal Gaussian distribution. The proposed forward pass is:

$$F = \textsc{ToDense}(E, c) \tag{2}$$

$$S_{p,q} = \sum_{j=0}^{d_1-1} \sum_{k=0}^{d_2-1} F_{j,k} e^{i2\pi\left(\frac{p}{d_1}j + \frac{q}{d_2}k\right)} \tag{3}$$

$$\begin{aligned} h &= W_0 x + \Delta W x \\ &= W_0 x + \alpha \Re(S) x. \end{aligned} \tag{4}$$

Specifically, $\textsc{ToDense}$ in Eq. 2 represents to construct the **spectral matrix** $F \in \mathbb{R}^{d_1 \times d_2}$, i.e., $F_{j,k} = c_l$ (resp. 0), if $j = E_{0,l}$ & $k = E_{1,l}$ (resp. else). Eq. 3 computes the **spatio matrix** S via the inverse discrete Fourier transform, where i represents the imaginary unit. Finally, in Eq. 4, we take the real part of the complex matrix S (denoted as $\Re(S)$) and scale it by α. Kindly note that all layers involve training various c vectors, while sharing the matrix E and value α.

The pseudocode for FourierFT is shown as Algorithm 1, adhering to the PyTorch style.

Initialization for the Entry Matrix E. Previous works lack studies on the importance of the spectral entries in the weight change. Thus, we fill this gap by introducing adjustable frequency bias, causing the entries to be more likely sampled in this area. In addition to randomly sampling entries in the full $d_1 \times d_2$-sized spectral matrix (i.e., no bias), we also implement entry sampling with a bias towards a favored central frequency, e.g., low, middle, or

Algorithm 1 PyTorch-style pseudocode for **FourierFT**.

```
class FourierFT(nn.Module):
    def __init__(
    self,
    n: int = 100, # number of trainable parameters
    alpha: float = 300.0, # scaling
    d1: int = 4096, # input dimension
    d2: int = 4096, # output dimension
    base_layer: nn.Module # pre-trained layer
    )
        # definitions
        self.d1 = d1
        self.d2 = d2
        self.n = n
        self.alpha = alpha
        self.base_layer = base_layer
        # entry initialization (no frequency bias)
        self.E = torch.randperm(d1 * d2)[:n]
        self.E = torch.stack([self.E // self.d2,
        self.E % self.d2], dim=0)
        # spectral coefficient initialization
        self.c = nn.Parameter(torch.randn(n), \\
            requires_grad=True)

    def forward(self, x: torch.Tensor):
        # get dense spectral matrix (Eq.2)
        F = torch.zeros(self.d1, self.d2)
        F[self.E[0, :], self.E[1, :]] = self.c
        # compute Delta_W (Eq.3)
        Delta_W = torch.fft.ifft2(F).real * self.alpha
        # merge (Eq.4)
        h = self.base_layer(x)
        h += torch.einsum('ijk,kl->ijl', x, Delta_W)
        return h
```

high frequencies. Formally, we apply the Gaussian band-pass filter (Gonzales & Wintz, 1987) to model the sampling probability for the entry $(u, v), 0 \le u \le d_1 - 1, 0 \le v \le d_2 - 1$:

$$p(u, v) = \exp\left(-\left(\frac{\mathcal{D}^2 - f_c^2}{\mathcal{D}\mathcal{W}}\right)^2\right), \tag{5}$$

where $\mathcal{D}$ represents the distance from the point (u, v) to the origin (center of the matrix), f_c is the favored central frequency, and $\mathcal{W}$ represents the bandwidth. In Figure 3, we visualize the sampling probability map of a 768×768-sized spectral matrix with different f_c and $\mathcal{W} = 200$.

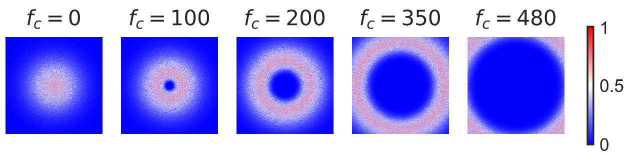


Figure 3. Visualization of entry sampling probability at different favored central frequencies f_c.

Kindly note that unless specially stated, FourierFT is set by default to the entry initialization with no frequency bias.

Table 1. Theoretical number of trainable parameters and storage requirements for fine-tuning. For both LoRA and FourierFT methods, only the query and value layers are tuned within the transformer architectures. The configurations that are exactly chosen in the 'Experiments' Section are highlighted .

Base Models	**LoRA**			**FourierFT**		
	r	# Trainable Parameters	Required Bytes	n	# Trainable Parameters	Required Bytes
RoBERTa Base	4	147K	574KB	200	4.8K	18.8KB
	8	295K	1.13MB	1000	24K	94KB
RoBERTa Large	4	393K	1.5MB	200	9.6K	36.5KB
	8	786K	3MB	1000	48K	183KB
GPT-2 Medium	4	350K	1.34MB	500	24K	94KB
	8	786K	3MB	1000	48K	188KB
GPT-2 Large	4	737K	2.81MB	500	36K	141KB
	8	1.47M	5.74MB	1000	72K	282KB
LLaMA-2 7B	16	8.39M	32.8MB	1000	64K	250KB
	64	33.5M	131.1MB	2000	128K	500KB
LLaMA-2 13B	16	13.1M	51.2MB	1000	80K	312KB
	64	52.4M	204.8MB	2000	160K	625KB
ViT Base	8	295K	1.13MB	3000	72K	281KB
	16	590K	2.25MB	10000	239K	934KB
ViT Large	8	786K	2.93MB	3000	144K	563KB
	16	1.57M	6MB	10000	480K	1.83MB

3.2. Parameter Summary

We summarize the number of trainable parameters for LoRA and FourierFT in Table 1. LoRA relies on a pair of trainable matrices A and B for each layer. Let the number of layers for fine-tuning be L_t. The total number of parameters in LoRA is determined by the rank r and the dimension of weights $d = d_1 = d_2$: $|\Theta|_{LoRA} = 2 \times d \times L_t \times r$. For Fourier, the total number takes the form: $|\Theta|_{FourierFT} = n \times L_t$. As an intuitive example, the RoBERTa Base model contains 12 transformer blocks with $d = 768$, resulting in $L_t = 24$ layers when we only fine-tune the query and value ones. Therefore, we have $|\Theta|_{LoRA} = 294,912$ for $r = 8$, and $|\Theta|_{FourierFT} = 24,000$ for $n = 1000$. In Table 1, we highlight the configurations where LoRA and our method achieve matched performance in subsequent experiments. We note that the advantage of parameter efficiency in FourierFT becomes more pronounced as the model's scale (depth and width) increases (e.g., RoBERTa Base → RoBERTa Large). This could be because $|\Theta|_{LoRA}$ has an explicit linear relationship with width d, unlike $|\Theta|_{FourierFT}$.

4. Experiments

In this section, we evaluate FourierFT in the domains of natural language processing (NLP) and computer vision (CV). For NLP, we implement FourierFT for fine-tuning **(1)** RoBERTa (Base & Large) on natural language understanding (GLUE, (Wang et al., 2018)), **(2)** GPT-2 (Medium & Large) on natural language generation (E2E, (Novikova et al., 2017)) and **(3)** LLaMA-family models (7B & 13B) on instruction tuning. For CV, we apply FourierFT to fine-tune the **(4)** vision transformers (Base & Large) on image classification. Finally, we conduct ablation studies to analyze the effect of frequency bias, the parameter scalability, and the

Table 2. Performance of various fine-tuning methods with RoBERTa Base ($\mathrm{RoB}_{\mathrm{base}}$) and RoBERTa Large ($\mathrm{RoB}_{\mathrm{large}}$) models on 6 datasets of the GLUE benchmark. We report the Matthew's correlation coefficient (MCC) for CoLA, Pearson correlation coefficient (PCC) for STS-B and accuracy (Acc.) for all the remaining tasks. We report the median result of 5 runs, each using different random seeds. The best results for each dataset are shown in **bold**. Higher is better for all metrics in 6 datasets.

Model & Method	# Trainable Parameters	**SST-2** (Acc.)	**MRPC** (Acc.)	**CoLA** (MCC)	**QNLI** (Acc.)	**RTE** (Acc.)	**STS-B** (PCC)	**Avg.**
$\mathrm{RoB}_{\mathrm{base}}$(FF)	125M	94.8	90.2	**63.6**	92.8	78.7	91.2	85.2
$\mathrm{RoB}_{\mathrm{base}}$(BitFit)	0.1M	93.7	**92.7**	62	91.8	**81.5**	90.8	**85.4**
$\mathrm{RoB}_{\mathrm{base}}$($\mathrm{Adpt}^{\mathrm{D}}$)	0.3M	$94.2_{\pm 0.1}$	$88.5_{\pm 1.1}$	$60.8_{\pm 0.4}$	$93.1_{\pm 0.1}$	$71.5_{\pm 2.7}$	$89.7_{\pm 0.3}$	83.0
$\mathrm{RoB}_{\mathrm{base}}$($\mathrm{Adpt}^{\mathrm{D}}$)	0.9M	$94.7_{\pm 0.3}$	$88.4_{\pm 0.1}$	$62.6_{\pm 0.9}$	$93.0_{\pm 0.2}$	$75.9_{\pm 2.2}$	$90.3_{\pm 0.1}$	84.2
$\mathrm{RoB}_{\mathrm{base}}$(LoRA)	0.3M	**95.1**$_{\pm 0.2}$	$89.7_{\pm 0.7}$	$63.4_{\pm 1.2}$	**93.3**$_{\pm 0.3}$	$78.4_{\pm 0.8}$	**91.5**$_{\pm 0.2}$	85.2
$\mathrm{RoB}_{\mathrm{base}}$(AdaLoRA)	0.3M	$94.5_{\pm 0.2}$	$88.7_{\pm 0.5}$	$62.0_{\pm 0.6}$	$93.1_{\pm 0.2}$	$81.0_{\pm 0.6}$	$90.5_{\pm 0.2}$	85.0
$\mathrm{RoB}_{\mathrm{base}}$(DyLoRA)	0.3M	$94.3_{\pm 0.5}$	$89.5_{\pm 0.5}$	$61.1_{\pm 0.3}$	$92.2_{\pm 0.5}$	$78.7_{\pm 0.7}$	$91.1_{\pm 0.6}$	84.5
$\mathrm{RoB}_{\mathrm{base}}$(**FourierFT**)	0.024M	$94.2_{\pm 0.3}$	$90.0_{\pm 0.8}$	**63.8**$_{\pm 1.6}$	$92.2_{\pm 0.1}$	$79.1_{\pm 0.5}$	$90.8_{\pm 0.2}$	85.0
$\mathrm{RoB}_{\mathrm{large}}$(FF)	356M	96.4	**90.9**	68	94.7	86.6	**92.4**	**88.2**
$\mathrm{RoB}_{\mathrm{large}}$($\mathrm{Adpt}^{\mathrm{P}}$)	3M	$96.1_{\pm 0.3}$	$90.2_{\pm 0.7}$	**68.3**$_{\pm 1.0}$	**94.8**$_{\pm 0.2}$	$83.8_{\pm 2.9}$	$92.1_{\pm 0.7}$	87.6
$\mathrm{RoB}_{\mathrm{large}}$($\mathrm{Adpt}^{\mathrm{P}}$)	0.8M	**96.6**$_{\pm 0.2}$	$89.7_{\pm 1.2}$	$67.8_{\pm 2.5}$	**94.8**$_{\pm 0.3}$	$80.1_{\pm 2.9}$	$91.9_{\pm 0.4}$	86.8
$\mathrm{RoB}_{\mathrm{large}}$($\mathrm{Adpt}^{\mathrm{H}}$)	6M	$96.2_{\pm 0.3}$	$88.7_{\pm 2.9}$	$66.5_{\pm 4.4}$	$94.7_{\pm 0.2}$	$83.4_{\pm 1.1}$	$91.0_{\pm 1.7}$	86.8
$\mathrm{RoB}_{\mathrm{large}}$($\mathrm{Adpt}^{\mathrm{H}}$)	0.8M	$96.3_{\pm 0.5}$	$87.7_{\pm 1.7}$	$66.3_{\pm 2.0}$	$94.7_{\pm 0.2}$	$72.9_{\pm 2.9}$	$91.5_{\pm 0.5}$	84.9
$\mathrm{RoB}_{\mathrm{large}}$(LoRA)	0.8M	$96.2_{\pm 0.5}$	$90.2_{\pm 1.0}$	$68.2_{\pm 1.9}$	**94.8**$_{\pm 0.3}$	$85.2_{\pm 1.1}$	$92.3_{\pm 0.5}$	87.8
$\mathrm{RoB}_{\mathrm{large}}$(**FourierFT**)	0.048M	$96.0_{\pm 0.2}$	**90.9**$_{\pm 0.3}$	$67.1_{\pm 1.4}$	$94.4_{\pm 0.4}$	**87.4**$_{\pm 1.6}$	$91.9_{\pm 0.4}$	88.0

expressiveness of the Fourier basis.

Baselines. We compare our FourierFT method with popular parameter-efficient fine-tuning (PEFT) methods. To ensure a comprehensive and fair comparison, we prioritize replicating the setups used in previous works and reusing their reported results. Involved baselines are:

- **Full Fine-tuning (FF)** - During fine-tuning, the base model is initialized with pre-trained weights and biases, and all parameters will undergo gradient updates.
- **Bitfit** (Zaken et al., 2021) - Only the bias vectors are fine-tuned while all other parameters are frozen.
- **Adapter tuning** - This research line was first investigated by Houlsby et al. (2019), which proposes the **Adapter**$^{\mathbf{H}}$ method. **Adapter**$^{\mathbf{H}}$ inserts two-layer adapters between the self-attention and the FNN modules, followed by a subsequent residual connection. We compare it with three additional variants of it. **Adapter**$^{\mathbf{L}}$ (Lin et al., 2020) is more parameter-efficient, with adapter layers applied only after the MLP modules and subsequent to a LayerNorm. **Adapter**$^{\mathbf{P}}$ (Pfeiffer et al., 2020) implements the adapter layers after the feed-forward layer. This design was chosen through a grid search including all settings related to the adapter's position, number, *ect*. **Adapter**$^{\mathbf{D}}$ (Rücklé et al., 2020) further enhances the parameter efficiency by dropping adapter layers that are not activated.
- **LoRA** (Hu et al., 2021) - LoRA is the state-of-the-art method for PEFT. It parameterizes incremental weight updates using trainable low-rank matrices.
- **DyLoRA** (Valipour et al., 2022) - This method trains dynamic search-free LoRA models for the best rank choice.
- **AdaLoRA** (Zhang et al., 2023) - This method proposes the SVD-based fine-tuning and prunes redundant singular values with the importance-aware rank allocation.

4.1. Natural Language Understanding

Models and Datasets. We evaluate our method on the GLUE benchmark (General Language Understanding Evaluation (Wang et al., 2018)), which consists of a wide range of natural language understanding (NLU) tasks, including single-sentence classification tasks, similarity and paraphrase tasks and natural language inference tasks. We fine-tune the pre-trained RoBERTa Base and Large foundation models (Liu et al., 2019) for evaluation.

Implementation Details. For both models, FourierFT is allowed to have 1000 out of 768^2 (RoBERTa Base) and 1024^2 (RoBERTa Large) trainable spectral coefficients in each layer, i.e., $n = 1000$. We randomly sample the spectral entries with no frequency bias, which is shared[1] across all 24 (Base) and 48 (Large) layers. For all 6 datasets in GLUE, we tune the hyperparameters of the learning rates and the scaling values. We follow the experimental setup applied in Hu et al. (2021), which involves fine-tuning only the query and value weights in each transformer block and

---

[1] We use the value 2024 as the `seed` for all layers.

Table 3. Results from GPT-2 Medium and Large models on the E2E benchmark. We present the result from the final epoch. For all metrics, higher values indicate better performance. * indicates that the results are taken from prior works. Best results are shown in **bold**.

Model	Method	# Trainable Parameters	BLEU	NIST	METEOR	ROUGE-L	CIDEr
GPT-2 Medium	FT*	354.92M	68.2	8.62	46.2	71.0	2.47
	Adpt^{L}*	0.37M	66.3	8.41	45.0	69.8	2.40
	Adpt^{L}*	11.09M	68.9	8.71	46.1	71.3	2.47
	Adpt^{H}*	11.09M	67.3$_{\pm.6}$	8.5$_{\pm.07}$	46.0$_{\pm.2}$	70.7$_{\pm.2}$	2.44$_{\pm.01}$
	LoRA	0.35M	68.9$_{\pm.3}$	8.76$_{\pm.06}$	46.6$_{\pm.1}$	71.5$_{\pm.1}$	**2.53**$_{\pm.03}$
	FourierFT	0.048M	**69.1**$_{\pm.1}$	**8.82** $_{\pm.05}$	**47.0** $_{\pm.3}$	**71.8** $_{\pm.1}$	2.51$_{\pm.02}$
GPT-2 Large	FT*	774.03M	68.5	8.78	46.0	69.9	2.45
	Adpt^{L}*	0.88M	69.1$_{\pm.1}$	8.68$_{\pm.03}$	46.3$_{\pm.0}$	71.4$_{\pm.2}$	2.49$_{\pm.0}$
	Adpt^{L}*	23.00M	68.9$_{\pm.3}$	8.70$_{\pm.04}$	46.1$_{\pm.1}$	71.3$_{\pm.2}$	2.45$_{\pm.02}$
	LoRA	0.77M	70.1$_{\pm.3}$	8.83$_{\pm.02}$	46.8$_{\pm.2}$	**72.0**$_{\pm.3}$	2.47$_{\pm.02}$
	FourierFT	0.072M	**70.2**$_{\pm.2}$	**8.90**$_{\pm.02}$	**47.0**$_{\pm.2}$	71.8$_{\pm.1}$	**2.50** $_{\pm.02}$

Table 4. The average scores on MT-Bench and Vicuna assessed by GPT-4. † indicates updating the layers other than `lm_head`. Higher score is better.

Model	Method	# Trainable Parameters	MT-Bench	Vicuna
LLaMA1-7B	LoRA†	159.9M	5.05$_{\pm.3}$	**6.85**$_{\pm.4}$
	LoRA	33.5M	4.99$_{\pm.3}$	6.81$_{\pm.3}$
	FourierFT	0.064M	**5.09**$_{\pm.6}$	**6.85**$_{\pm.8}$
LLaMA1-13B	LoRA†	250.3M	**5.28**$_{\pm.6}$	7.02$_{\pm.3}$
	LoRA	52.4M	5.21$_{\pm.4}$	6.97$_{\pm.4}$
	FourierFT	0.08M	5.23$_{\pm.3}$	**7.14**$_{\pm.5}$
LLaMA2-7B	LoRA†	159.9M	5.19$_{\pm.1}$	7.38$_{\pm.3}$
	LoRA	33.5M	**5.20**$_{\pm.3}$	7.35$_{\pm.6}$
	FourierFT	0.064M	5.18$_{\pm.3}$	**7.49**$_{\pm.4}$
LLaMA2-13B	LoRA†	250.3M	5.78$_{\pm.2}$	7.89$_{\pm.5}$
	LoRA	52.4M	5.80$_{\pm.2}$	7.89$_{\pm.6}$
	FourierFT	0.08M	**5.82**$_{\pm.3}$	**7.92**$_{\pm.5}$

fully fine-tuning the classification head. We provide the hyperparameters in Table 9 in Appendix.

Results. Results are summarized in Table 2. Following Hu et al. (2021), Zhang et al. (2023) and Valipour et al. (2022), we specify the number of trainable parameters for the fine-tuned layers excluding the classification head. We report the median of 5 random seed results, where the best epoch is selected for each run. In general, FourierFT achieves better or on-par performance compared with baseline methods with significantly fewer trainable parameters. Notably, FourierFT outperforms all baselines including fully fine-tuning the RoBERTa Base on CoLA and the RoBERTa Large on RTE. As mentioned in Section 3.2, the parameter count of LoRA is dependent on both the width and depth of models, resulting in a larger count growth (LoRA: $0.8\text{M}/0.3\text{M} \approx 2.7$; ours: $0.048\text{M}/0.024\text{M} = 2$) compared to FourierFT. Nevertheless, FourierFT still performs comparably to LoRA, demonstrating the potential scalability of our method when facing even larger models.

4.2. Natural Language Generation

Models and Datasets. We evaluate the performance of FourierFT on the E2E natural language generation (NLG) task (Novikova et al., 2017). We fine-tune the GPT-2 (Radford et al., 2019) Medium (354M) and Large (774M) models, which are both decoder-only and have 24 and 36 transformer blocks, respectively. The E2E benchmark contains roughly 42,000 training, 4,600 validation and 4,600 test samples from the restaurant domain.

Implementation Details. We report prior results for baselines other than LoRA. For both LoRA and our method, we fine-tune the GPT-2 Medium and Large models with a linear learning rate scheduler for 5 epochs, where we tune the batch size and learning rate. We report the average results over 3 runs, where the last epoch is selected for each run. We provide the hyperparameters in Table 10 in Appendix.

Results. We show the results in Table 3. We note that FourierFT can achieve the best performance on most metrics. More importantly, FourierFT only requires 13.7% and 9.4% of the parameter counts of LoRA, for the GPT-2 Medium and Large models respectively.

4.3. Instruction Tuning

Models and Datasets. Instruction tuning, as described in (Ouyang et al., 2022; Wei et al., 2021; Mishra et al., 2021), refers to the process of fine-tuning a language model on a collection of paired prompts and responses. We apply LoRA and FourierFT to fine-tune the LLaMA (Touvron et al., 2023a) and LLaMA2 (Touvron et al., 2023b) families. Specifically, we consider the LLaMA-7B, LLaMA-13B, LLaMA2-7B and LLaMA2-13B as base models, which are fine-tuned on the Alpaca dataset (Taori et al., 2023). Alpaca contains 51K instruction-following demonstrations generated from `text-davinci-003` (GPT-3.5) (Wang et al., 2022). For evaluation, we use the fine-tuned models to generate responses for the pre-defined questions, which are from the MT-Bench (Zheng et al., 2023) and Vicuna Eval (Chiang et al., 2023). GPT-4 takes these answers as input and evaluates them with scores within 10.

Implementation Details. For LoRA, we use $r = 64$ and apply two configurations: (1) updating all linear layers except the language modelling head (`lm_head`); (2) updating only the W_Q and W_V matrices. For FourierFT, we only adopt the latter configuration with $n = 1000$. To ensure the

Table 5. Fine-tuning results with ViT Base and Large models on different image classification datasets. We report the accuracy (%) after 10 epochs. Avg. represents the average accuracy of each method on all datasets. The best performance is shown in **bold**.

Model	Method	# Trainable Parameters	**OxfordPets**	**StanfordCars**	**CIFAR10**	**DTD**	**EuroSAT**	**FGVC**	**RESISC45**	**CIFAR100**	**Avg.**
ViT-Base	LP	-	$90.28_{\pm0.43}$	$25.76_{\pm0.28}$	$96.41_{\pm0.02}$	$69.77_{\pm0.67}$	$88.72_{\pm0.13}$	$17.44_{\pm0.43}$	$74.22_{\pm0.10}$	$84.28_{\pm0.11}$	68.36
	FF	85.8M	$93.14_{\pm0.40}$	**79.78**$_{\pm1.15}$	**98.92**$_{\pm0.05}$	**77.68**$_{\pm1.21}$	**99.05**$_{\pm0.09}$	**54.84**$_{\pm1.23}$	**96.13**$_{\pm0.13}$	**92.38**$_{\pm0.13}$	**86.49**
	LoRA	581K	$93.19_{\pm0.36}$	$45.38_{\pm0.41}$	$98.78_{\pm0.05}$	$74.95_{\pm0.40}$	$98.44_{\pm0.15}$	$25.16_{\pm0.16}$	$92.70_{\pm0.18}$	$92.02_{\pm0.12}$	77.58
	FourierFT	72K	**93.21**$_{\pm0.26}$	$46.11_{\pm0.24}$	$98.58_{\pm0.07}$	$75.09_{\pm0.37}$	$98.29_{\pm0.04}$	$27.51_{\pm0.64}$	$91.97_{\pm0.31}$	$91.20_{\pm0.14}$	77.75
	FourierFT	239K	$93.05_{\pm0.34}$	$56.36_{\pm0.66}$	$98.69_{\pm0.08}$	$77.30_{\pm0.61}$	$98.78_{\pm0.11}$	$32.44_{\pm0.99}$	$94.26_{\pm0.20}$	$91.45_{\pm0.18}$	80.29
ViT-Large	LP	-	$91.11_{\pm0.30}$	$37.91_{\pm0.27}$	$97.78_{\pm0.04}$	$73.33_{\pm0.26}$	$92.64_{\pm0.08}$	$24.62_{\pm0.24}$	$82.02_{\pm0.11}$	$84.28_{\pm0.11}$	72.96
	FF	303.3M	$94.43_{\pm0.56}$	**88.90**$_{\pm0.26}$	**99.15**$_{\pm0.05}$	$81.79_{\pm1.01}$	**99.04**$_{\pm0.08}$	**68.25**$_{\pm1.63}$	**96.43**$_{\pm0.07}$	$93.58_{\pm0.19}$	**90.20**
	LoRA	1.57M	$94.82_{\pm0.09}$	$73.25_{\pm0.36}$	$99.13_{\pm0.03}$	$81.79_{\pm0.45}$	$98.63_{\pm0.07}$	$42.32_{\pm0.98}$	$94.71_{\pm0.25}$	**94.87**$_{\pm0.10}$	84.94
	FourierFT	144K	$94.46_{\pm0.28}$	$69.56_{\pm0.30}$	$99.10_{\pm0.04}$	$80.83_{\pm0.43}$	$98.65_{\pm0.09}$	$39.92_{\pm0.68}$	$93.86_{\pm0.14}$	$93.31_{\pm0.09}$	83.71
	FourierFT	480K	**94.84**$_{\pm0.05}$	$79.14_{\pm0.67}$	$99.08_{\pm0.01}$	**81.88**$_{\pm0.50}$	$98.66_{\pm0.03}$	$51.28_{\pm0.68}$	$95.20_{\pm0.07}$	$93.37_{\pm0.11}$	86.68

feasibility of training on a single GPU, we deploy the quantization method in Dettmers et al. (2023) for fine-tuning. We train with both methods for only one epoch, and report the average scores of all answers. We provide the hyperparameter setup in Table 11 in the Appendix.

Results. The results are shown in Table 4. We find that the expressive power of the 13B model is much stronger than that of the 7B model, regardless of which fine-tuning method is used. Moreover, FourierFT closely matches or slightly exceeds LoRA's performance with less than 0.2% of its parameters. We provide practical examples containing questions, answers and reviews in the Appendix D.

4.4. Image Classification

Models and Datasets. We evaluate our method on the image classification task. We adopt the Base and Large versions of the popular CV foundation model, Vision Transformer (ViT) (Dosovitskiy et al., 2020). The ViTs are pretrained on the ImageNet-21K dataset (Ridnik et al., 2021). The datasets for fine-tuning include OxfordPets (37[2]), CIFAR10 (10), DTD (47), EuroSAT (10) and RESISC45 (45) with small label spaces, as well as StanfordCars (196), FGVC (100) and CIFAR100 (100) with large label spaces. Detailed information is provided in Table 8 in the Appendix.

Implementation Details. We include three baselines for evaluation: Full Fine-tuning (FF), Linear Probing (LP, fine-tuning the classification head only), and LoRA. For both LoRA and our method, only the query and value matrices of ViT are updated. We use $r = 16$ for LoRA and $n = \{3000, 10000\}$ for FourierFT. We tune the learning rates and weight decay for all methods, and set the maximum training epoch to 10. We provide the hyperparameters in Table 12 in Appendix.

Results. Table 5 summarizes the results for 8 image classification datasets with the ViT Base and Large models. Both LoRA and FourierFT methods significantly outperform the Linear Probing, demonstrating their effectiveness in the CV domain. Our method obtains matched performance using 12.4% and 9.2% of LoRA's parameter count, with ViT Base and Large models, respectively. Notably, when we increase the parameter count of FourierFT to 41.1% (ViT Base) and 30.6% (ViT Large) of LoRA's, it can outperform LoRA by 3.5% and 2.0% respectively. Moreover, our method can even (slightly) outperform the Full Fine-tuning method on OxfordPets and DTD with the ViT Large model.

4.5. Study

Effect of Frequency Bias. We examine how the performance is affected by the frequency bias, i.e., the central frequency f_c in Eq. 5. We directly apply the optimal hyperparameters searched in Table 2 and fine-tune the RoBERTa Base on the MRPC, STS-B, CoLA and RTE datasets. From Figure 5, we note that the fine-tuning performance of FourierFT without any frequency bias can surpass most cases that are restricted by the central frequency bias. This indicates the universality of our method. Surprisingly, we find that it is always possible to obtain results better than "No bias" by traversing the f_c values. Since this traversal is not efficient, we do not conduct further exploration in this paper. However, we believe that making f_c trainable will be a promising new direction for improving FourierFT.

Parameter Scalability. We explore the relationship between the number of trainable parameters and the performance of LoRA and our method. We use the set of ranks $r = \{1, 2, 4, 6, 8, 15\}$ for LoRA and $n = \{50, 100, 200, 1000, 6144, 12288\}$ for FourierFT on 6 tasks of the GLUE benchmark. For both LoRA and ours, the learning rate, and scaling hyperparameters are tuned. For fairness, we ensure that the number of trials for hyperparam-

---

[2] Numbers in parentheses indicate class counts for each dataset.

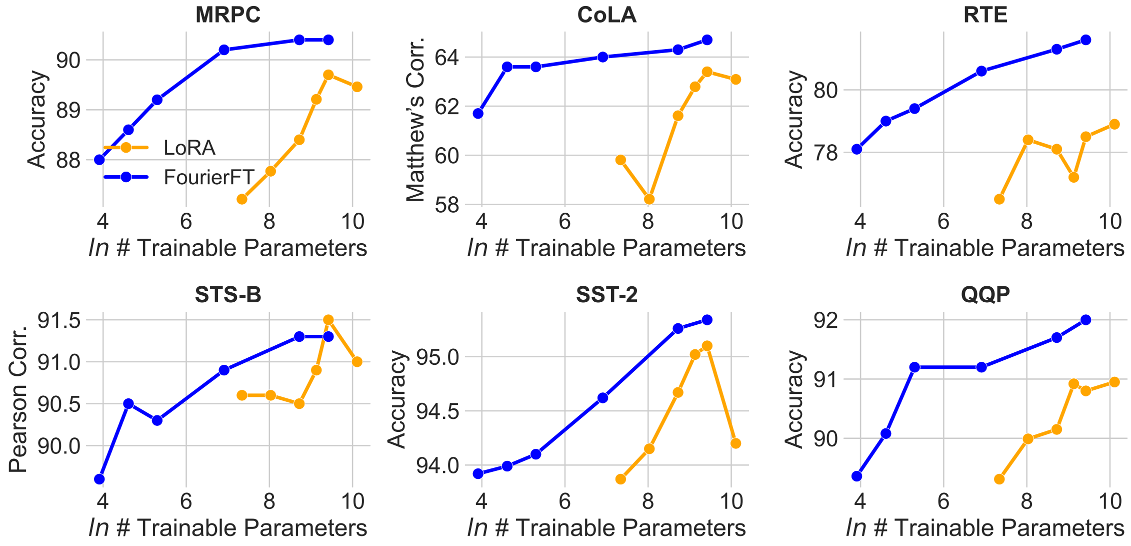


Figure 4. Performance on the GLUE benchmark with RoBERTa Base **vs.** number of trainable parameters (each layer) of LoRA and ours. For all 6 datasets, we apply the setting of $r = \{1, 2, 4, 6, 8, 15\}$ for LoRA and $n = \{50, 100, 200, 1000, 6144, 12288\}$.

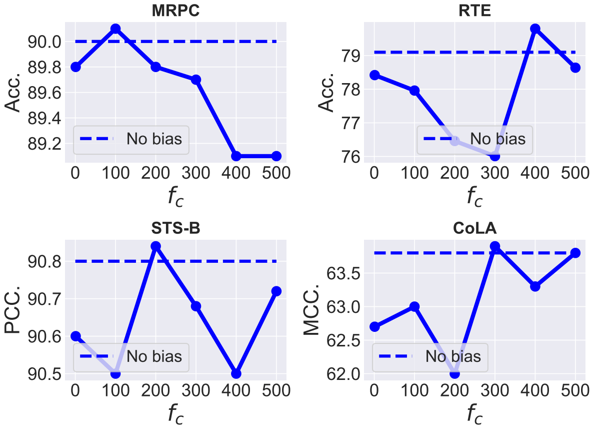


Figure 5. Results on 4 datasets in GLUE with different f_c values.

eter search is 30 for both methods. As shown in Figure 4, our method outperforms LoRA on all 6 datasets. In detail, our method is significantly better than LoRA with the same parameter count, i.e., $\{r = 4, n = 6144\}$ & $\{r = 8, n = 12288\}$. Moreover, we observe that a larger number of parameters does not always bring performance gains for LoRA. On the contrary, the increase of n can consistently improve the accuracy of FourierFT. On most tasks, FourierFT with $n = 50$ can achieve comparable or even better (MRPC, CoLA, RTE) performance than LoRA with $r = 1$. In this case, the parameter count in LoRA is about 31 × that of ours.

Basis Expressiveness. The inverse discrete Fourier transform (IDFT) in Eq. 3 is equivalent to the matrix multiplication (Lu et al., 2021): $S = \mathcal{B}_f F \mathcal{B}_f^\intercal$, where $\mathcal{B}$ is the transformation matrix of IDFT that contains the Fourier basis. To evaluate its expressivity, we replace the Fourier basis with random and orthogonal basis, respectively. Specifically, for $F \in \mathbb{R}^{d_1 \times d_2}$, we initialize random basis $\mathcal{B}_r^1 \in \mathbb{R}^{d_1 \times d_1}$ and $\mathcal{B}_r^2 \in \mathbb{R}^{d_2 \times d_2}$ with the normal Gaussian distribution. Then Eq. 3 becomes $S = \mathcal{B}_r^1 F \mathcal{B}_r^2$. A similar way is used for the orthogonal basis. We compare FourierFT with the random basis (R-B) and orthogonal basis (O-B) on the GLUE benchmark. Table 6 shows the results. We note that the Fourier basis used in our method outperforms the random and orthogonal basis. In addition, the expressive power of the orthogonal basis is much stronger than that of the random basis. The stronger expressive power of the Fourier basis compared to the general orthogonal basis may be attributed to its effective capture of the spectral information of ΔW.

Table 6. Results with three types of basis.

Model	RTE			CoLA		
	Ours	R-B	O-B	**Ours**	R-B	O-B
Base	**79.1**	72.7(↓8.1%)	75.6(↓4.4%)	**63.8**	58.7(↓8.0%)	60.0(↓6.0%)
Large	**87.4**	81.8(↓6.4%)	83.6(↓4.3%)	**67.1**	64.8(↓3.4%)	66.1(↓1.5%)

5. Conclusion

In this paper, we aim to achieve an extremely low storage memory for a single fine-tuning of large foundation models. This will enable the customization of multiple fine-tunings for different domains, tasks, or user preferences. To achieve this, we propose a simple yet powerful fine-tuning method that treats weight changes as spatial-domain matrices and only learns the sparse coefficients in the spectral domain. Compared to the LoRA-style baselines, our approach reduces the number of trainable parameters by more than 10× on a wide range of tasks in the NLP and CV domains.

Impact Statement

This paper presents a work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

The research of Li was supported by NSFC Grant No.62206067, HKUST–HKUST(GZ) 20 for 20 Cross-campus Collaborative Research Scheme C019 and Guangzhou-HKUST(GZ) Joint Funding Scheme 2023A03J0673. The research of Chen was supported by the Shenzhen Science and Technology Program (KJZD20231023094501003), and Tencent AI Lab RBFR2024004.

References

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, N., Li, Y., Tang, J., and Li, J. Graphwiz: An instruction-following language model for graph problems. *arXiv preprint arXiv:2402.16029*, 2024.

Chen, Y. and Chi, Y. Spectral compressed sensing via structured matrix completion. In *International conference on machine learning*, pp. 414–422. PMLR, 2013.

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

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2023.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

Civitai. `https://civitai.com/`, 2024.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

Diao, S., Huang, Z., Xu, R., Li, X., Lin, Y., Zhou, X., and Zhang, T. Black-box prompt learning for pre-trained language models. *arXiv preprint arXiv:2201.08531*, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Duarte, M. F. and Baraniuk, R. G. Spectral compressive sensing. *Applied and Computational Harmonic Analysis*, 35(1):111–129, 2013.

Ehrlich, M. and Davis, L. S. Deep residual learning in the jpeg transform domain. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3484–3493, 2019.

Gao, T., Fisch, A., and Chen, D. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

Gao, Z., Niu, Y., Cheng, J., Tang, J., Xu, T., Zhao, P., Li, L., Tsung, F., and Li, J. Handling missing data via max-entropy regularized graph autoencoder. *arXiv preprint arXiv:2211.16771*, 2022.

Gao, Z., Sun, X., Liu, Z., Li, Y., Cheng, H., and Li, J. Protein multimer structure prediction via prompt learning. *arXiv preprint arXiv:2402.18813*, 2024.

Gonzales, R. C. and Wintz, P. *Digital image processing*. Addison-Wesley Longman Publishing Co., Inc., 1987.

Gueguen, L., Sergeev, A., Kadlec, B., Liu, R., and Yosinski, J. Faster neural networks straight from jpeg. *Advances in Neural Information Processing Systems*, 31, 2018.

He, J., Zhou, C., Ma, X., Berg-Kirkpatrick, T., and Neubig, G. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.

He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

Helber, P., Bischke, B., Dengel, A., and Borth, D. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

Herrmann, F. J. Randomized sampling and sparsity: Getting more information from fewer samples. *Geophysics*, 75 (6):WB173–WB187, 2010.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Krizhevsky, A. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Li, Y., Shen, W., Gao, J., and Wang, Y. Community question answering entity linking via leveraging auxiliary data. *arXiv preprint arXiv:2205.11917*, 2022.

Lin, M., Weng, S., and Zhang, C. On the sample complexity of random fourier features for online learning: How many random fourier features do we need? *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(3):1–19, 2014.

Lin, Z., Madotto, A., and Fung, P. Exploring versatile generative language model via parameter-efficient transfer learning. *arXiv preprint arXiv:2004.03829*, 2020.

Liu, H., Li, C., Li, Y., and Lee, Y. J. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023a.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023b.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Lu, T., Chen, Y.-F., Hechtman, B., Wang, T., and Anderson, J. Large-scale discrete fourier transform on tpus. *IEEE Access*, 9:93422–93432, 2021.

Maji, S., Rahtu, E., Kannala, J., Blaschko, M., and Vedaldi, A. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Mishra, S., Khashabi, D., Baral, C., and Hajishirzi, H. Cross-task generalization via natural language crowdsourcing instructions. *arXiv preprint arXiv:2104.08773*, 2021.

Novikova, J., Dušek, O., and Rieser, V. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*, 2017.

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

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

Patel, V. M., Maleh, R., Gilbert, A. C., and Chellappa, R. Gradient-based image recovery methods from incomplete fourier measurements. *IEEE Transactions on Image Processing*, 21(1):94–105, 2011.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., and Gurevych, I. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10): 1872–1897, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Rawat, A. S., Chen, J., Yu, F. X. X., Suresh, A. T., and Kumar, S. Sampled softmax with random fourier features. *Advances in Neural Information Processing Systems*, 32, 2019.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

Ridnik, T., Ben-Baruch, E., Noy, A., and Zelnik-Manor, L. Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, 2021.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Rücklé, A., Geigle, G., Glockner, M., Beck, T., Pfeiffer, J., Reimers, N., and Gurevych, I. Adapterdrop: On the efficiency of adapters in transformers. *arXiv preprint arXiv:2010.11918*, 2020.

Rudelson, M. and Vershynin, R. On sparse reconstruction from fourier and gaussian measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 61(8): 1025–1045, 2008.

Shechtman, Y., Beck, A., and Eldar, Y. C. Gespar: Efficient phase retrieval of sparse signals. *IEEE transactions on signal processing*, 62(4):928–938, 2014.

Shi, L., Hassanieh, H., Davis, A., Katabi, D., and Durand, F. Light field reconstruction using sparsity in the continuous fourier domain. *ACM Transactions on Graphics (TOG)*, 34(1):1–13, 2014.

Singh, A., Hu, R., Goswami, V., Couairon, G., Galuba, W., Rohrbach, M., and Kiela, D. Flava: A foundational language and vision alignment model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15638–15650, 2022.

Song, Y., Shen, L., Xing, L., and Ermon, S. Solving inverse problems in medical imaging with score-based generative models. *arXiv preprint arXiv:2111.08005*, 2021.

Tang, J., Li, J., Gao, Z., and Li, J. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*, pp. 21076–21089. PMLR, 2022.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Valipour, M., Rezagholizadeh, M., Kobyzev, I., and Ghodsi, A. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*, 2022.

Vlaardingerbroek, M. T. and Boer, J. A. *Magnetic resonance imaging: theory and practice*. Springer Science & Business Media, 2013.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Xu, K., Qin, M., Sun, F., Wang, Y., Chen, Y.-K., and Ren, F. Learning in the frequency domain. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1740–1749, 2020.

Yang, Z. and Xie, L. Exact joint sparse frequency recovery via optimization methods. *IEEE Transactions on Signal Processing*, 64(19):5145–5157, 2016.

Zaken, E. B., Ravfogel, S., and Goldberg, Y. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.

Zhou, Z., Wei, X., Zhang, J., and Sun, G. {PetS}: A unified framework for {Parameter-Efficient} transformers serving. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, pp. 489–504, 2022.

Zwartjes, P. and Gisolf, A. Fourier reconstruction with sparse inversion. *Geophysical prospecting*, 55(2):199–221, 2007.

Supplementary of "Parameter-Efficient Fine-Tuning with Discrete Fourier Transform"

A. Details of Datasets

A.1. GLUE Benchmark

The GLUE (Wang et al., 2018) (General Language Understanding Evaluation) benchmark is widely used in the NLP domain. GLUE consists of a set of 8 NLP datasets: MNLI(inference), SST-2 (sentiment analysis), MRPC (paraphrase detection), CoLA (linguistic acceptability), QNLI (inference), QQP (question-answering), RTE (inference), and STS-B (textual similarity). We summarise their statistics in the following table.

Table 7. Task descriptions and dataset statistics of the GLUE benchmark. STS-B belongs to the regression task. All other tasks are single sentence or sentence pair classification tasks.

Corpus	Task	# Train	# Val	# Test	# Labels	Metrics	Domain
				Single-Sentence Tasks			
CoLA	Acceptability	8.55k	1.04k	1.06k	2	Matthews Corr.	misc.
SST-2	Sentiment	67.3k	872	1.82k	2	Accuracy	Movie reviews
				Similarity and Paraphrase Tasks			
MRPC	Paraphrase	3.67	408	1.73k	2	Accuracy/F1	News
STS-B	Sentence similarity	5.75k	1.5k	1.38k	1	Pearson/Spearman Corr.	misc.
QQP	Paraphrase	364k	40.4k	391k	2	Accuracy/F1	Social QA
				Inference Tasks			
MNLI	NLI	393k	19.65k	19.65k	3	Accuracy	misc.
QNLI	QA/NLI	105k	5.46k	5.46k	2	Accuracy	Wikipedia
RTE	NLI	2.49k	277	3k	2	Accuracy	News & Wikipedia

A.2. E2E Benchmark

The E2E (End-to-End) NLG challenge, proposed by (Novikova et al., 2017), is a dataset for evaluating natural language (data-to-text) generation models. The E2E dataset contains about 42,000 training samples, 4,600 validation samples and 4,600 test samples from the restaurant domain. E2E involves evaluation on 5 metrics: BLEU, NIST, METEOR, ROUGE-L, and CIDEr. A more detailed explanation of them is as follows.

- **BLEU** (Bilingual Evaluation Understudy) is a metric to evaluate the quality of machine-generated text by comparing it to one or more human-generated reference texts.
- **NIST** (National Institute of Standards and Technology) is a metric that evaluates the quality of machine-generated text, similar to BLEU. NIST uses a weighted average of n-gram precisions to calculate the final score, whereas BLEU uses a geometric average.
- **METEOR** (Metric for Evaluation of Translation with Explicit ORdering) aligns the words in the machine-generated text with their corresponding words in the reference text, and then calculates a score based on the harmonic mean of precision and recall.
- **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation) measures the longest common sub-sequence (LCS) between the machine-generated summary and the reference summary. It is particularly useful for evaluating summaries that contain paraphrases or rephrased sentences, as it considers the LCS rather than exact word overlap.

- **CIDEr** (Consensus-based Image Description) measures the similarity between the machine-generated captions and the human-generated captions by considering both the n-gram overlap and the consensus among human annotators.

A.3. Alpaca

Alpaca is a newly proposed dataset that contains only the training set. Alpaca contains 51k instructions and demonstrations generated by the `text-davinci-003` model. It can be used to fine-tune language models for specific instructions and improve their ability to follow instructions accurately. A specific example is as follows.

```
{
    "instructions": Transform the following sentence into the passive voice.
    "input": I bought a book.
    "output": A book was bought by me.
}
```

The **`instruction`** describes the target task which should be performed by the model. The **`input`** denotes optional context or input for the task. The **`output`** is the answer to the instruction generated by `text-davinci-003`.

A.4. MT-bench and Vicuna

MT-bench (Zheng et al., 2023) is a recently proposed benchmark containing a series of open-ended questions. These questions can evaluate the instruction-following ability of a language foundation model. MT-bench primarily distinguishes the abilities of many aspects of the models, including writing, roleplay, reasoning, math, coding, extraction, stem, and humanities. A specific example is as follows.

```
{
    "Q1": The vertices of a triangle are at points (0, 0), (-1, 1), and (3, 3).
    What is the area of the triangle?
    "Q2(follow-up)": What is the area of the circle circumscribing the triangle?
    "Solution": Q1. Area is 3. Q2. 5pi.
}
```

Vicuna Eval refers to the benchmark for evaluating LLM alignment with human preferences, which is the predecessor of MT-bench. Vicuna Eval covers the topics of coding, writing, math, counterfactual, fermi, common sense, roleplay, knowledge, and generic. A specific example is as follows.

```
{
    "question": Implement a binary search algorithm to find a specific element
    in a sorted array.
    "category": coding.
}
```

A.5. Image Classification Datasets

The statistics of the selected 8 vision datasets are in Table 8.

Table 8. Details about the vision datasets.

Dataset	#Train	#Val	#Test	#Class	Rescaled resolution
OxfordPets (Parkhi et al., 2012)	3,312	368	3,669	37	224 × 224
StandfordCars (Krause et al., 2013)	7,329	815	8,041	196	
CIFAR10 (Krizhevsky, 2009)	45,000	5,000	10,000	10	
DTD (Cimpoi et al., 2014)	4,060	452	1,128	47	
EuroSAT (Helber et al., 2019)	16,200	5,400	5,400	10	
FGVC (Maji et al., 2013)	3,000	334	3,333	100	
RESISC45 (Cheng et al., 2017)	18,900	6,300	6,300	45	
CIFAR100 (Krizhevsky, 2009)	45,000	5,000	10,000	100	

B. Hyperparamaters

Table 9. Hyperparameter setup of FourierFT for the GLUE benchmark.

Model	Hyperparameter	STS-B	RTE	MRPC	CoLA	SST-2	QNLI
Both	Optimizer	AdamW					
	LR Schedule	Linear					
	Warmup Ratio	0.06					
	Frequency Bias	False					
	n	1000					
	`seeds`	{0, 11111, 22222, 33333, 44444}					
Base	Epochs	60	90	30	100	40	40
	Learning Rate (FourierFT)	9E-2	9E-2	5E-2	1.2E-1	5E-2	1E-2
	Learning Rate (Head)	9E-3	1.1E-2	6E-3	8E-3	6E-3	1E-3
	Max Seq. Len	512	512	512	512	512	512
	Scaling value	84	110	141	49	140	29
	Batch Size	32	32	32	32	32	32
Large	Epochs	30	60	30	80	10	30
	Learning Rate (FourierFT)	7E-2	8E-2	6E-2	4.3E-2	4.3E-2	6E-2
	Learning Rate (Head)	1E-3	5E-3	1E-3	1.1E-2	1E-3	5E-3
	Max Seq. Len	512	512	512	256	128	512
	Scaling Value	121	90	120	120	99	69
	Batch Size	32	32	32	128	32	32

Table 10. Hyperparameter setup of FourierFT on the E2E benchmark.

Hyperparameter	Medium	Large
Optimizer	AdamW	
Learning Rate (FourierFT)	2E-2	5E-2
Learning Rate (Head)	2E-4	1E-4
Batch Size	128	
Weight Decay	0.01	0.03
n	1000	
Scaling value α	300	
Epochs	5	
Label Smooth	0.1	
LR Schedule	Linear	

Table 11. Hyperparameter setup for instruction-tuning of LoRA and FourierFT.

Hyperparameter	LoRA	FourierFT
Optimizer	AdamW	
Warmup Ratio	0.06	
Batch Size	4	
Accumulation Steps	4	
Epochs	1	
n	1000	–
Scaling Value α	300.0	16.0
LR Schedule	Linear	
Learning Rate	3E-2	3E-3

Table 12. Hyperparameter setup for image classification of FourierFT.

Hyperparameter	OxfordPets	StanfordCars	CIFAR10	DTD	EuroSAT	FGVC	RESISC45	CIFAR100
Epochs	10							
Optimizer	AdamW							
LR Schedule	Linear							
n	3000							
α	300.0							
Learning Rate (FourierFT)	3E-1	3E-1	3E-1	3E-1	2E-1	3E-1	3E-1	2E-1
Learning Rate (Head)	1E-3	1E-3	1E-3	1E-3	8E-4	1E-3	1E-3	7E-4
Weight Decay	8E-4	4E-5	9E-5	7E-5	3E-4	7E-5	3E-4	1E-4

C. Additional Experimental Results

C.1. Training Curve

We show the training curves of our method and LoRA to demonstrate that the superior performance of FourierFT is not due to coincidence. In Figure 6, we set $r = 1$ for LoRA and $n = 1536$ for the MRPC task, so that the total number of trainable parameters is equivalent for both methods. It can be seen that FourierFT consistently outperforms LoRA in terms of accuracy, F1 score, and training loss throughout the entire training process.

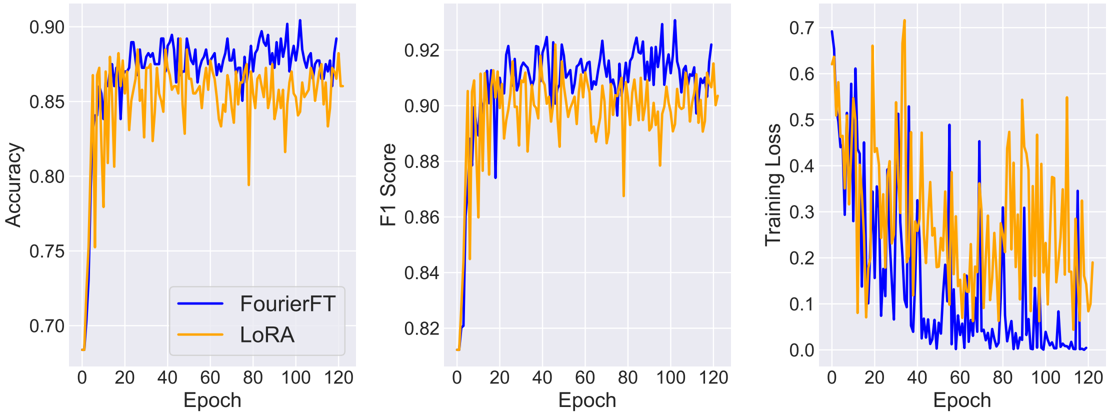


Figure 6. Training process of LoRA and ours. We show the current accuracy, F1 score and training loss of two methods.

C.2. Expressive Ability

To intuitively evaluate the expressive power of our method, we design a simple classification task with a synthetic dataset to simulate a scenario where LoRA encounters performance bottlenecks. Specifically, we specify a 2D center point for each class of data in the 8 classes, and randomly add Gaussian noise based on that point to obtain the 2D coordinates of the input. The dataset visualization is shown on the left of Figure 7. We train a single $64 * 64$ sized hidden layer with LoRA ($r = 1$) and FourierFT ($n = 128$) to fit the synthesized data. In this case, both methods require the same number of trainable parameters. However, the results of the experiments are vastly different. It can be seen that LoRA never reaches 100% accuracy within 2000 epochs, while FourierFT can quickly achieve it (in about 500 epochs). Under certain parameter constraints, LoRA has obvious performance bottlenecks, while FourierFT can easily overcome them.

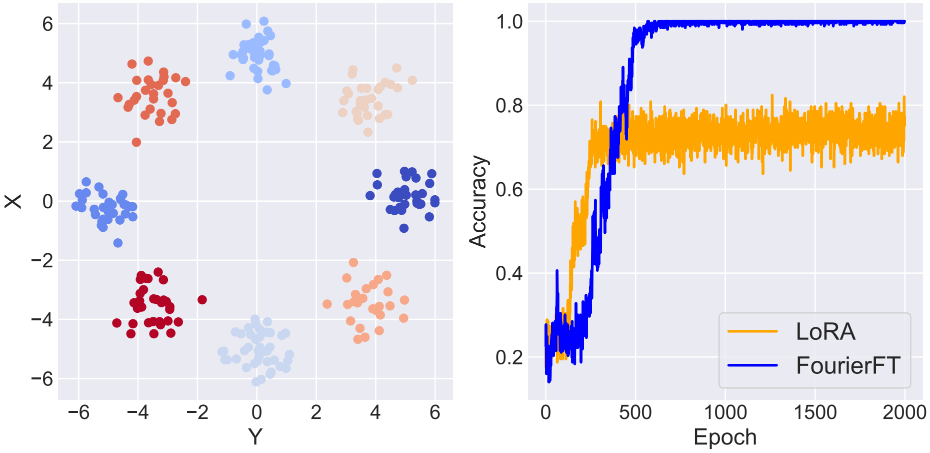
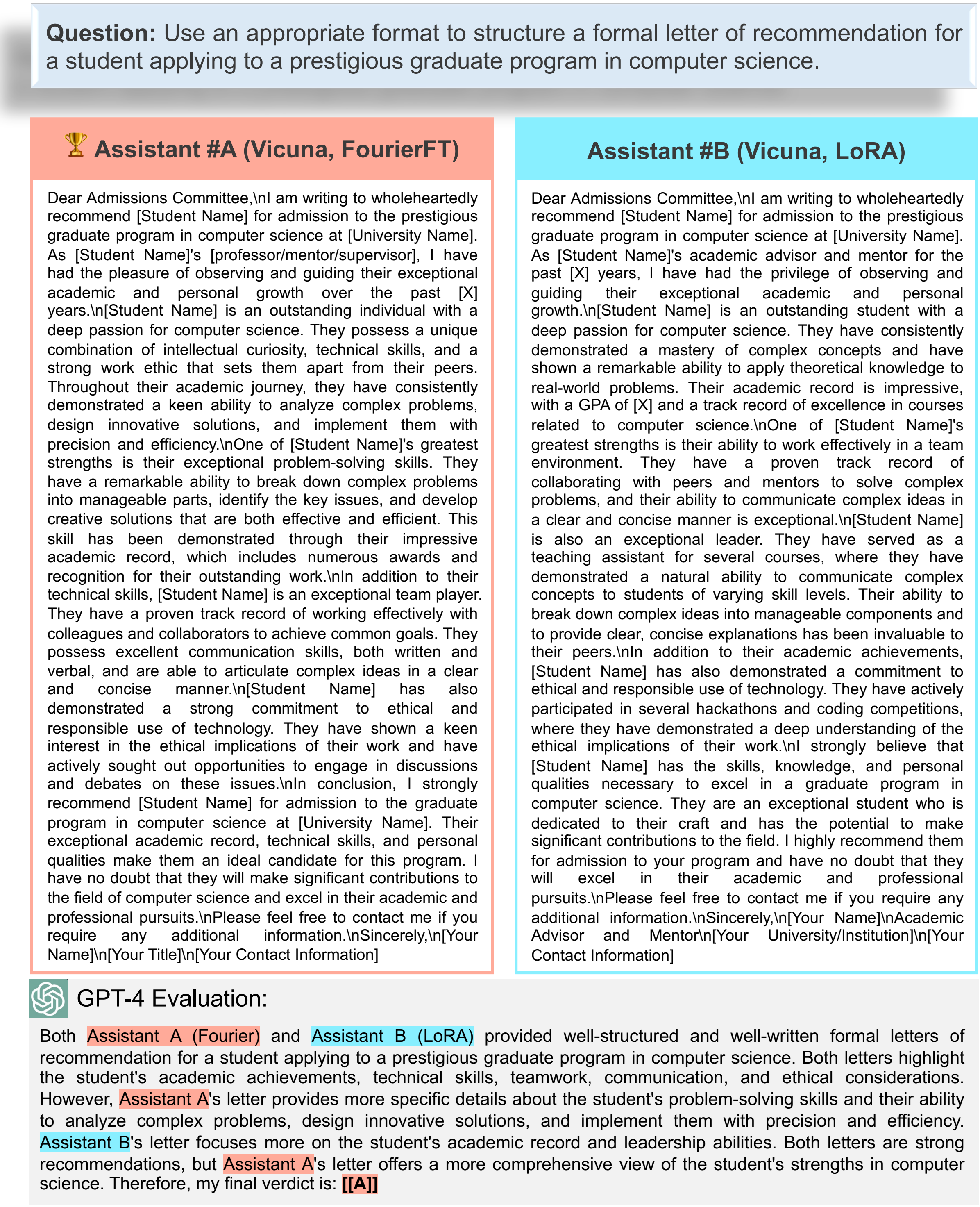


Figure 7. Left: The dataset distribution. Colors represent the classes. The model input is the 2D coordinate of each point. Right: Training process of LoRA and ours.

D. Examples of Instruction Tuning

We provide some practical cases generated by the LLaMA2-7B model that is fine-tuned with LoRA and ours. Each case includes the question, the answer and the review assigned by GPT-4.



Question: Use an appropriate format to structure a formal letter of recommendation for a student applying to a prestigious graduate program in computer science.

🏆 Assistant #A (Vicuna, FourierFT)

Dear Admissions Committee,\nI am writing to wholeheartedly recommend [Student Name] for admission to the prestigious graduate program in computer science at [University Name]. As [Student Name]'s [professor/mentor/supervisor], I have had the pleasure of observing and guiding their exceptional academic and personal growth over the past [X] years.\n[Student Name] is an outstanding individual with a deep passion for computer science. They possess a unique combination of intellectual curiosity, technical skills, and a strong work ethic that sets them apart from their peers. Throughout their academic journey, they have consistently demonstrated a keen ability to analyze complex problems, design innovative solutions, and implement them with precision and efficiency.\nOne of [Student Name]'s greatest strengths is their exceptional problem-solving skills. They have a remarkable ability to break down complex problems into manageable parts, identify the key issues, and develop creative solutions that are both effective and efficient. This skill has been demonstrated through their impressive academic record, which includes numerous awards and recognition for their outstanding work.\nIn addition to their technical skills, [Student Name] is an exceptional team player. They have a proven track record of working effectively with colleagues and collaborators to achieve common goals. They possess excellent communication skills, both written and verbal, and are able to articulate complex ideas in a clear and concise manner.\n[Student Name] has also demonstrated a strong commitment to ethical and responsible use of technology. They have shown a keen interest in the ethical implications of their work and have actively sought out opportunities to engage in discussions and debates on these issues.\nIn conclusion, I strongly recommend [Student Name] for admission to the graduate program in computer science at [University Name]. Their exceptional academic record, technical skills, and personal qualities make them an ideal candidate for this program. I have no doubt that they will make significant contributions to the field of computer science and excel in their academic and professional pursuits.\nPlease feel free to contact me if you require any additional information.\nSincerely,\n[Your Name]\n[Your Title]\n[Your Contact Information]

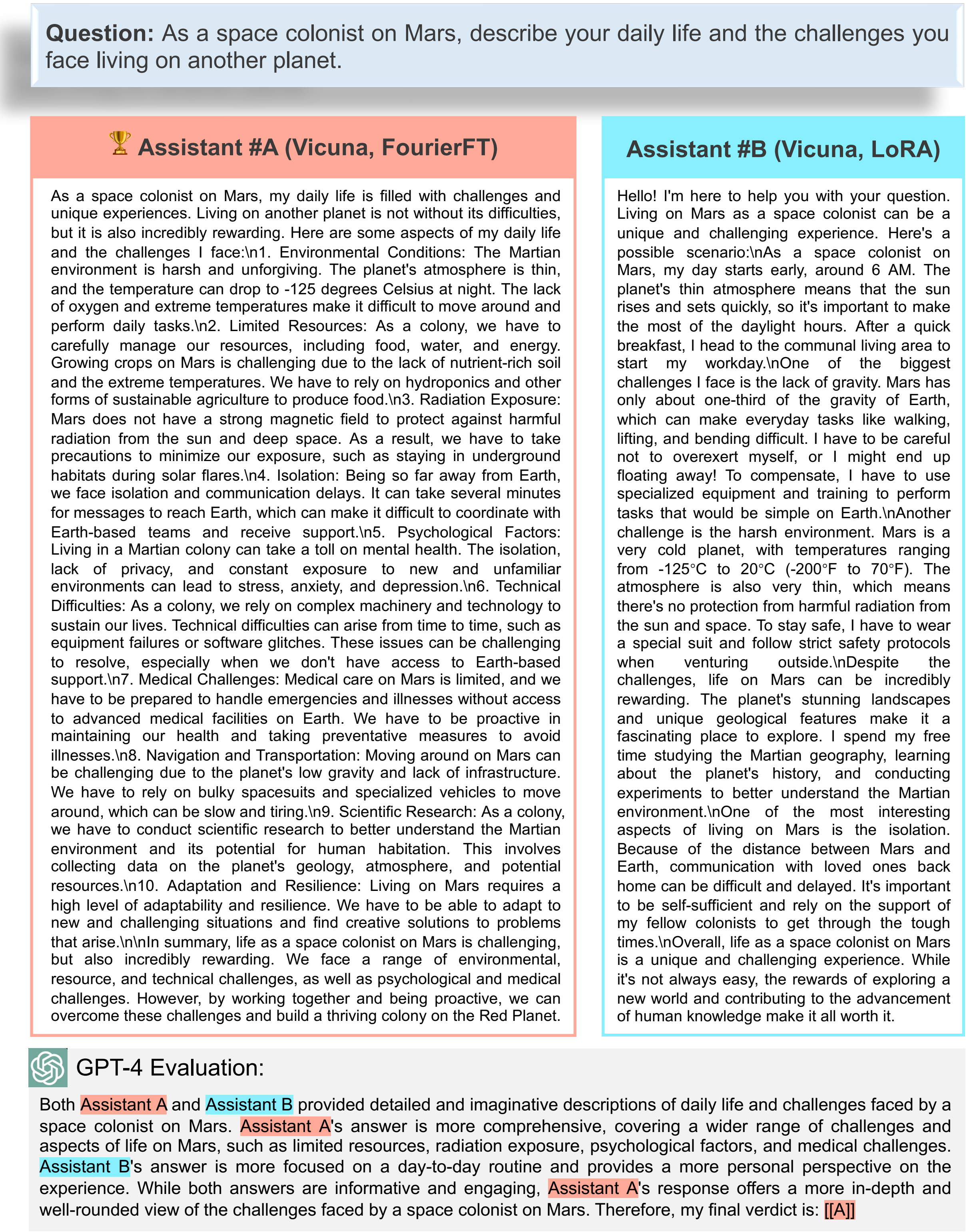
Assistant #B (Vicuna, LoRA)

Dear Admissions Committee,\nI am writing to wholeheartedly recommend [Student Name] for admission to the prestigious graduate program in computer science at [University Name]. As [Student Name]'s academic advisor and mentor for the past [X] years, I have had the privilege of observing and guiding their exceptional academic and personal growth.\n[Student Name] is an outstanding student with a deep passion for computer science. They have consistently demonstrated a mastery of complex concepts and have shown a remarkable ability to apply theoretical knowledge to real-world problems. Their academic record is impressive, with a GPA of [X] and a track record of excellence in courses related to computer science.\nOne of [Student Name]'s greatest strengths is their ability to work effectively in a team environment. They have a proven track record of collaborating with peers and mentors to solve complex problems, and their ability to communicate complex ideas in a clear and concise manner is exceptional.\n[Student Name] is also an exceptional leader. They have served as a teaching assistant for several courses, where they have demonstrated a natural ability to communicate complex concepts to students of varying skill levels. Their ability to break down complex ideas into manageable components and to provide clear, concise explanations has been invaluable to their peers.\nIn addition to their academic achievements, [Student Name] has also demonstrated a commitment to ethical and responsible use of technology. They have actively participated in several hackathons and coding competitions, where they have demonstrated a deep understanding of the ethical implications of their work.\nI strongly believe that [Student Name] has the skills, knowledge, and personal qualities necessary to excel in a graduate program in computer science. They are an exceptional student who is dedicated to their craft and has the potential to make significant contributions to the field. I highly recommend them for admission to your program and have no doubt that they will excel in their academic and professional pursuits.\nPlease feel free to contact me if you require any additional information.\nSincerely,\n[Your Name]\nAcademic Advisor and Mentor\n[Your University/Institution]\n[Your Contact Information]

GPT-4 Evaluation:

Both Assistant A (Fourier) and Assistant B (LoRA) provided well-structured and well-written formal letters of recommendation for a student applying to a prestigious graduate program in computer science. Both letters highlight the student's academic achievements, technical skills, teamwork, communication, and ethical considerations. However, Assistant A's letter provides more specific details about the student's problem-solving skills and their ability to analyze complex problems, design innovative solutions, and implement them with precision and efficiency. Assistant B's letter focuses more on the student's academic record and leadership abilities. Both letters are strong recommendations, but Assistant A's letter offers a more comprehensive view of the student's strengths in computer science. Therefore, my final verdict is: **[[A]]**

Figure 8. Example 1.



Question: As a space colonist on Mars, describe your daily life and the challenges you face living on another planet.

🏆 Assistant #A (Vicuna, FourierFT)

As a space colonist on Mars, my daily life is filled with challenges and unique experiences. Living on another planet is not without its difficulties, but it is also incredibly rewarding. Here are some aspects of my daily life and the challenges I face:\n1. Environmental Conditions: The Martian environment is harsh and unforgiving. The planet's atmosphere is thin, and the temperature can drop to -125 degrees Celsius at night. The lack of oxygen and extreme temperatures make it difficult to move around and perform daily tasks.\n2. Limited Resources: As a colony, we have to carefully manage our resources, including food, water, and energy. Growing crops on Mars is challenging due to the lack of nutrient-rich soil and the extreme temperatures. We have to rely on hydroponics and other forms of sustainable agriculture to produce food.\n3. Radiation Exposure: Mars does not have a strong magnetic field to protect against harmful radiation from the sun and deep space. As a result, we have to take precautions to minimize our exposure, such as staying in underground habitats during solar flares.\n4. Isolation: Being so far away from Earth, we face isolation and communication delays. It can take several minutes for messages to reach Earth, which can make it difficult to coordinate with Earth-based teams and receive support.\n5. Psychological Factors: Living in a Martian colony can take a toll on mental health. The isolation, lack of privacy, and constant exposure to new and unfamiliar environments can lead to stress, anxiety, and depression.\n6. Technical Difficulties: As a colony, we rely on complex machinery and technology to sustain our lives. Technical difficulties can arise from time to time, such as equipment failures or software glitches. These issues can be challenging to resolve, especially when we don't have access to Earth-based support.\n7. Medical Challenges: Medical care on Mars is limited, and we have to be prepared to handle emergencies and illnesses without access to advanced medical facilities on Earth. We have to be proactive in maintaining our health and taking preventative measures to avoid illnesses.\n8. Navigation and Transportation: Moving around on Mars can be challenging due to the planet's low gravity and lack of infrastructure. We have to rely on bulky spacesuits and specialized vehicles to move around, which can be slow and tiring.\n9. Scientific Research: As a colony, we have to conduct scientific research to better understand the Martian environment and its potential for human habitation. This involves collecting data on the planet's geology, atmosphere, and potential resources.\n10. Adaptation and Resilience: Living on Mars requires a high level of adaptability and resilience. We have to be able to adapt to new and challenging situations and find creative solutions to problems that arise.\n\nIn summary, life as a space colonist on Mars is challenging, but also incredibly rewarding. We face a range of environmental, resource, and technical challenges, as well as psychological and medical challenges. However, by working together and being proactive, we can overcome these challenges and build a thriving colony on the Red Planet.

Assistant #B (Vicuna, LoRA)

Hello! I'm here to help you with your question. Living on Mars as a space colonist can be a unique and challenging experience. Here's a possible scenario:\nAs a space colonist on Mars, my day starts early, around 6 AM. The planet's thin atmosphere means that the sun rises and sets quickly, so it's important to make the most of the daylight hours. After a quick breakfast, I head to the communal living area to start my workday.\nOne of the biggest challenges I face is the lack of gravity. Mars has only about one-third of the gravity of Earth, which can make everyday tasks like walking, lifting, and bending difficult. I have to be careful not to overexert myself, or I might end up floating away! To compensate, I have to use specialized equipment and training to perform tasks that would be simple on Earth.\nAnother challenge is the harsh environment. Mars is a very cold planet, with temperatures ranging from -125°C to 20°C (-200°F to 70°F). The atmosphere is also very thin, which means there's no protection from harmful radiation from the sun and space. To stay safe, I have to wear a special suit and follow strict safety protocols when venturing outside.\nDespite the challenges, life on Mars can be incredibly rewarding. The planet's stunning landscapes and unique geological features make it a fascinating place to explore. I spend my free time studying the Martian geography, learning about the planet's history, and conducting experiments to better understand the Martian environment.\nOne of the most interesting aspects of living on Mars is the isolation. Because of the distance between Mars and Earth, communication with loved ones back home can be difficult and delayed. It's important to be self-sufficient and rely on the support of my fellow colonists to get through the tough times.\nOverall, life as a space colonist on Mars is a unique and challenging experience. While it's not always easy, the rewards of exploring a new world and contributing to the advancement of human knowledge make it all worth it.

GPT-4 Evaluation:

Both Assistant A and Assistant B provided detailed and imaginative descriptions of daily life and challenges faced by a space colonist on Mars. Assistant A's answer is more comprehensive, covering a wider range of challenges and aspects of life on Mars, such as limited resources, radiation exposure, psychological factors, and medical challenges. Assistant B's answer is more focused on a day-to-day routine and provides a more personal perspective on the experience. While both answers are informative and engaging, Assistant A's response offers a more in-depth and well-rounded view of the challenges faced by a space colonist on Mars. Therefore, my final verdict is: [[A]]

Figure 9. Example 2.