Parameter-Efficient Fine-Tuning with Circulant and Diagonal Vectors

Xinyu Ding, **Lexuan Chen**, **Siyu Liao** and **Zhongfeng Wang**

Sun Yat-sen University

dingbai1357718507@gmail.com, chenlx77@mail2.sysu.edu.cn, liaocs2008@gmail.com,
wangzf83@mail.sysu.edu.cn

Abstract

Foundation models have achieved tremendous success in different domains. However, their huge computation and storage complexity make these models difficult to fine-tune and also less applicable in practice. Recent study shows training in Fourier domain can be an effective fine-tuning method in terms of both model performance and number of training parameters. In this work, we propose to further reduce the complexity by the factorization through the product of interleaved circulant and diagonal matrices. In addition, we address the case of non-square fine-tuning weights by partitioning the circulant matrix into blocks. Our method avoids the construction of weight change matrix and utilizes 1D fast Fourier transform (FFT) instead of 2D FFT. Experimental results show that our method achieves similar or better performance across various tasks with much less floating-point operations (FLOPs) and the number of trainable parameters.

1 Introduction

Large foundation models (LFMs) are widely utilized in various fields, including natural language processing [Paaß and Giesselbach, 2023], image recognition and generation [Li *et al.*, 2024a], medical diagnosis [Li *et al.*, 2024b], and autonomous driving [Chen *et al.*, 2024]. [Devlin, 2018] have proposed the bidirectional transformer architecture that understands input data from left to right and right to left. It is trained to predict missing words given input context, and it has served as a foundation model that can be fine-tuned for many downstream tasks. Following the transformer architecture, generative pre-trained transformer (GPT) model by [Radford and Narasimhan, 2018] handles input data from left to right following a sequential prediction order. This mechanism turns out successful in many generation tasks such as text summary, question answering, etc.

Although LFMs learn extensive general knowledge during the pre-training phase, they still require extra adjustments in downstream applications to effectively fullfill the task. Fine-tuning is a typical approach to continue learning on given downstream data and update from pre-trained model parameters. While fine-tuning significantly reduces computational costs compared to training from scratch, existing fine-tuning methods still suffer from the huge complexity of LFMs. As the original model parameters are still kept and maintained during fine-tuning stage, this leaves limited space for the development of fine-tuning methods.

To address the challenge of fine-tuning LFMs, [Hu *et al.*, 2021] have proposed low-rank adaptation (LoRA). This method is an efficient fine-tuning approach designed for LFMs, reducing the number of parameters required during fine-tuning by introducing low-rank matrices. The essential idea is assuming the weight change matrix with low rank structure, expressing it as the product of two low rank matrices and only training these two smaller matrices while keeping the original weights frozen. FourierFT proposed by [Gao *et al.*, 2024a] assumes a sparse structure in fourier domain of the weight matrix updates $\Delta\mathbf{W}$. Although FourierFT reduces the number of training parameters, the computational and storage requirements of the model remain very high, particularly when dealing with LFMs. The two-dimensional Fourier transform used to restore $\Delta\mathbf{W}$ contributes to most of its computation and storage complexity. As a result, the fine-tuning model continues to necessitate high-performance hardware support, including substantial GPU resources and memory, which may be challenging to achieve in practical applications.

[Huhtanen and Perämäki, 2015] has demonstrated that a general complex matrix $\mathbf{X} \in \mathbb{C}^{n \times n}$ can be factorized into the product of multiple circulant matrices and diagonal matrices, with total number of factors not exceeding $2n - 1$. This decomposition method offers several advantages, particularly in terms of computational efficiency and storage optimization. The computation and storage of diagonal matrices can be efficiently managed using vector representations. Moreover, circulant matrices possess a unique structure that allows them to be diagonalized using the fast Fourier transform (FFT), significantly reducing the complexity of matrix operations and accelerating computation speed.

Inspired by previous works, we propose circulant and diagonal vector based fine-tuning (CDVFT), which is also a Fourier domain based method. Our method represents the weight change matrix $\Delta\mathbf{W}$ with the product of interleaved circulant and diagonal matrices. This factorization simplifies the matrix calculation process and reduces storage requirements. Due to the unique properties of matrix product for

circulant and diagonal matrices, the quadratic computation complexity now becomes loglinear. Different from FourierFT based on 2D FFT, our fine-tuning process avoids the restoration of the weight change $\mathbf{\Delta W}$ and only takes 1D FFT operations. However, the product of interleaved circulant and diagonal matrices inherently forms only square matrices, limiting its generality. To address this issue, we propose partitioning the circulant matrix into blocks, enabling more efficient storage management while maintaining the accuracy standards required for large-scale models. As a result, CDVFT can achieve efficient storage and computation at the same time. We summarize our main contributions as following:

- We introduce CDVFT method that represents $\mathbf{\Delta W}$ using the product of interleaved block circulant and diagonal matrices. These matrices have linear storage complexity as each of them can be determined by a single weight vector. In practice, we find only a few matrices and a small number of blocks is sufficient to perform fine-tuning.

- CDVFT avoids the restoration of weight change matrix and has loglinear computation complexity. The circulant matrix vector product can be transformed into 1D FFT, and diagonal matrix vector product is linear in nature. Thus, the overall computation complexity becomes loglinear.

- We evaluate our method on natural language understanding, image classification, and instruction and task adjustment. Experimental results show that our method achieves similar or even better results in terms of moder performance, number of training parameters and FLOPs. For example, for the RoBERTa base model, our method results in $51.81\times$ FLOPs reduction compared to FourierFT and $5.33\times$ trainable parameters saving compared to LoRA, while resulting in similar or even better accuracy.

2 Related Works

Fine-tuning LFMs is a challenging problem due to the large model size and computation requirement. Although training LFMs from scratch is performed on cloud platforms like LLaMA model by [Touvron *et al.*, 2023], fine-tuning is often limited to a specific task and a low-cost computing environment. Besides, fine-tuning runs on a much smaller dataset than the pre-training dataset for LFMs. Thus, fine-tuning process is expected to be cost-effective. The overall complexity should be small and affordable in practice.

Full fine-tuning is a classical approach training and updating all model parameters at the same time. However, it is difficult to perform full fine-tuning on LFMs given the huge computation and storage requirement. [Brown *et al.*, 2020] find LFMs are able to generalize to new tasks with few-shot demonstrations as prompt, thereby saving the effort of training on parameters. [Li and Liang, 2021] argue that adding few-shot demonstrations is bounded by the input length constraint of current LFMs. Instead, they propose the prefix tuning method to train a parameter vector and prepend to input, which is expected to work as prompt in unlimited length.

Updating all model parameters is not desirable in practice, since each task needs to maintain a model. [Houlsby *et al.*, 2019] propose the adapter method, where task dependent parameters are inserted to LFMs. Fine-tuning process only updates those new parameters, thereby each task effectively sharing pre-trained LFM parameters. [Mahabadi *et al.*, 2021] further reduce the task dependent parameters amount by grouping adapters into a hyper network model such that the network can produce task specific parameters on the fly. [Sung *et al.*, 2022] discover backpropagation process through LFMs takes a lot of memory and propose a ladder style adapter design that significantly saves memory consumption. Given that adapters bring in extra inference latency due to their new parameters, [Lei *et al.*, 2023] believe different tasks have different needs for the shared LFM architecture. They decide to learn to skip computations in LFM for different adapters, resulting in a faster inference speed.

It can be noticed that adapter adds task dependent parameters and incurs inference delay. There are also studies working on mergeable adapters so that after fine-tuning they can be merged into LFM architecture without adding inference latency. The essential idea is setting adapter parameters in the same shape as LFM pre-trained parameters, and fine-tuning learns the change of weight parameters, i.e., $\mathbf{\Delta W}$. [Hu *et al.*, 2021] develop LoRA technique that enforces low rank structure into the weight change matrix. Given that LoRA rank can be different for different tasks, [Zhang *et al.*, 2023] decide to learn the rank setting by modifying singular values based on importance score function. Instead of directly learning on $\mathbf{\Delta W}$, [Gao *et al.*, 2024a] propose FourierFT to learn sparse parameters in fourier domain and reconstruct the weight difference using 2D FFT operation. It turns out this method requires much less number of parameters, but its reconstruction needs more memory.

Following the parameter efficient fine-tuning (PEFT) discovery in fourier domain, it is important to look for a method involving matrix and efficient FFT operation. Circulant matrix is related to 1D FFT since circulant matrix vector product can be executed using 1D FFT to accelerate. There are some studies applying circulant matrix to compress neural networks, such as circulant convolution neural network by [Cheng *et al.*, 2015] and circulant long short-memory by [Wang *et al.*, 2018]. However, these works are lack of flexibility on increasing parameter amount and theoretical guarantee on dense matrix approximation. [Huhtanen and Perämäki, 2015] has demonstrated that a general complex matrix $\mathbf{X} \in \mathbb{C}^{n \times n}$ can be expressed as the product of interleaved circulant and diagonal matrices, with the number of factors not exceeding $2n - 1$:

$$\begin{aligned}\mathbf{X} =&\mathbf{A}_{2n-1} \times \mathbf{C}_{2n-2} \times \ldots \times \mathbf{C}_{2j} \times \mathbf{A}_{2j-1} \times \\ &\times \ldots \times \mathbf{A}_3 \times \mathbf{C}_2 \times \mathbf{A}_1 \times \mathbf{x}\end{aligned} \quad (1)$$

where for $j \in \{1, \ldots, n\}$, $\mathbf{A}_{2j-1}$ and $\mathbf{C}_{2j}$ are diagonal and circulant matrices, respectively. Thus, this decomposition theoretically can approximate any dense matrix, and it also enables control on parameter amount by setting number of factors.
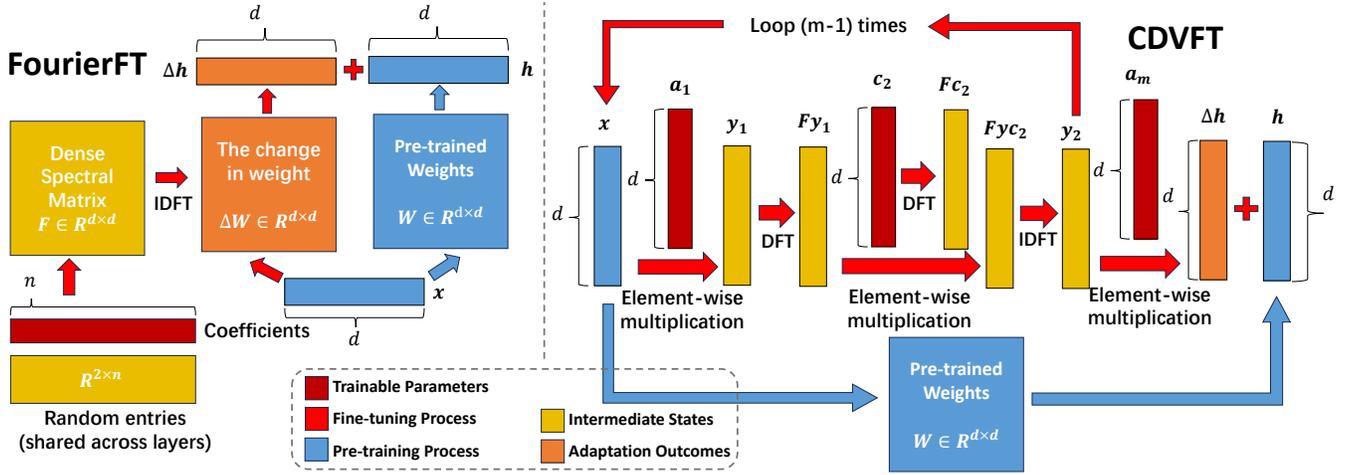
Figure 1: **Overview of FourierFT (left) and our CDVFT (right).** In FourierFT, one coefficient vector $\mathbf{c} \in \mathbb{R}^n$ is trained, and it is used to construct the weight change $\mathbf{\Delta W}$ through 2D FFT operation. In contrast, our CDVFT avoids the construction of $\mathbf{\Delta W}$, where matrix vector products are transformed into vector operations, i.e., element-wise product and 1D FFT, significantly reducing computation complexity and memory requirement. In practice, we find $m = 1$ (no loops required) can effectively fine-tune the model, where there are two diagonal matrices and one circulant matrix.

3 Method

In this section, we introduce circulant and diagonal vector based fine-tuning (CDVFT) method, which is a mergeable adapter design similar to FourierFT. After fine-tuning, our trained circulant and diagonal vectors can be used to build circulant and diagonal matrices, which are further combined to reconstruct $\mathbf{\Delta W}$ and merged into LFMs. However, most importantly, CDVFT does not need to recover $\mathbf{\Delta W}$ during real fine-tuning process, since the reconstruction results in high computation and storage complexity. Instead, our method takes advantage of the fast matrix multiplication algorithm from circulant and diagonal matrices involving 1D FFT and element-wise product to achieve the goal.

The overall computation flow is illustrated in Fig.1. It can be seen that CDVFT only takes vector operations at each step, thereby significantly reducing the computation and storage complexity. Specifically, according to the findings by [Huhtanen and Perämäki, 2015] and the unique properties of circulant and diagonal matrix operations, CDVFT first initializes corresponding vectors to represent these matrices. It then directly performs multiple element-wise multiplication and 1D FFT on input $\mathbf{x}$. Finally, it yields the output $\mathbf{\Delta h}$, which can be added to the output $\mathbf{h}$ from the original weight matrix $\mathbf{W}$.

3.1 Forward Step

Let $\mathbf{x} \in \mathbb{R}^{d \times 1}$ be an input column vector. Assume weight change matrix $\mathbf{\Delta W} \in \mathbb{R}^{d \times d}$ that can be decomposed into $2m - 1$ factors with $m \leq d$. Thus, there are m diagonal matrices and $m-1$ circulant matrices. For $j \in \{1, 2, \ldots, m\}$, each diagonal matrix is defined by a vector $\mathbf{a}_{2j-1} \in \mathbb{R}^{d \times 1}$, and each circulant matrix is defined by a vector $\mathbf{c}_{2j} \in \mathbb{R}^{d \times 1}$.

More specifically, they can be expressed as following:

$$
\begin{aligned}
diag(\mathbf{a}_{2j-1}) &= \begin{bmatrix} \mathbf{a}_{2j-1}^1 & 0 & \ldots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & \mathbf{a}_{2j-1}^d \end{bmatrix}, \\
circ(\mathbf{a}_{2j}) &= \begin{bmatrix} \mathbf{a}_{2j}^1 & \mathbf{a}_{2j}^d & \ldots & \mathbf{a}_{2j}^2 \\ \mathbf{a}_{2j}^2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{a}_{2j}^d \\ \mathbf{a}_{2j}^d & \ldots & \mathbf{a}_{2j}^2 & \mathbf{a}_{2j}^1 \end{bmatrix},
\end{aligned}
\tag{2}
$$

where $diag(\cdot)$ and $circ(\cdot)$ construct a diagonal matrix and circulant matrix, respectively. Therefore, the weight change matrix can be written as:

$$
\begin{aligned}
\mathbf{\Delta W} &= \mathbf{A}_{2m-1} \times \mathbf{C}_{2m-2} \times \mathbf{A}_{2m-3} \times \cdots \times \mathbf{A}_1 \\
&= diag(\mathbf{a}_{2m-1}) \times circ(\mathbf{a}_{2m-2}) \times diag(\mathbf{a}_{2m-3}) \\
&\quad \times \cdots \times diag(\mathbf{a}_1),
\end{aligned}
\tag{3}
$$

where $\times$ is the inner product operation. The end-to-end computation flow then becomes:

$$
\mathbf{h}' = \mathbf{h} + \mathbf{\Delta h} = \mathbf{W} \times \mathbf{x} + \alpha \times \mathbf{\Delta W} \times \mathbf{x},
\tag{4}
$$

where α is a hyper-parameter scalar as in LoRA [Hu *et al.*, 2021], $\mathbf{W} \in \mathbb{R}^{d \times d}$ is the pre-trained weight matrix in given LFM and $\mathbf{h}'$ is the new output after adding our CDVFT adapters. This can also be seen in Fig. 1.

We perform the computation from rightmost to leftmost, thereby avoiding the reconstruction of $\mathbf{\Delta W}$ during fine-tuning process. Let $\mathbf{y} \in \mathbb{R}^{d \times 1}$ represent the intermediate calculation result from matrix vector multiplications. Thus, $\mathbf{y}_{2j-1}$ is the result from diagonal matrix vector multiplication, and $\mathbf{y}_{2j}$ is the result from circulant matrix vector multi-

plication. Note that diagonal matrix vector product is equivalent to element wise product of $\mathbf{a}_{2j-1}$ and input vector:

$$\mathbf{\Delta W} \times \mathbf{x} = \mathbf{A}_{2m-1} \times \ldots \times$$

$$\times \overbrace{\mathbf{C}_{2j} \times \underbrace{\mathbf{A}_{2j-1} \times \ldots \times \mathbf{A}_3 \times \mathbf{C}_2 \times \mathbf{A}_1 \times \mathbf{x}}_{\mathbf{y}_{2j-1}}}^{\mathbf{y}_{2j}}, \quad (5)$$

$$\mathbf{y}_0 = \mathbf{x}, \ \ \mathbf{y}_{2j} = \mathbf{C}_{2j} \times \mathbf{y}_{2j-1}, \quad (6)$$

$$\mathbf{y}_{2j-1} = \mathbf{A}_{2j-1} \times \mathbf{y}_{2j-2} = \mathbf{a}_{2j-1} \odot \mathbf{y}_{2j-2}, \quad (7)$$

where $\odot$ means the element-wise product. The circulant matrix vector product can be transformed into 1D FFT operations:

$$\mathbf{F}_{2j-1} = \mathrm{FFT}(\mathbf{y}_{2j-1}) = \{\sum_{q=0}^{d-1} \mathbf{y}_{2j-1}^q e^{-i2\pi\frac{p}{d}q}\}_{p=0}^{d-1},$$

$$\mathbf{F}_{2j} = \mathrm{FFT}(\mathbf{c}_{2j}) = \{\sum_{q=0}^{d-1} \mathbf{c}_{2j}^q e^{-i2\pi\frac{p}{d}q}\}_{p=0}^{d-1}, \quad (8)$$

$$\hat{\mathbf{F}} = \mathbf{F}_{2j-1} \odot \mathbf{F}_{2j},$$

$$\mathbf{y}_{2j} = \mathrm{IFFT}(\mathbf{F}) = \{\frac{1}{d}\sum_{q=0}^{d-1} \hat{\mathbf{F}}^q e^{i2\pi\frac{p}{d}q}\}_{p=0}^{d-1},$$

where $e^{i2\pi\frac{p}{d}q}$ is the constant term in the Fourier transform, i indicates the imaginary unit, and p is the frequency index of the transform. We use letter $\mathbf{F}$ to indicate vectors in fourier domain. $\mathbf{F}_{2j-1}$ and $\mathbf{F}_{2j}$ represent the Fourier transform results of $\mathbf{y}_{2j-1}$ and the circulant matrix vector $\mathbf{c}_{2j}$, respectively. $\hat{\mathbf{F}}$ is the result of element wise multiplication of $\mathbf{F}_{2j-1}$ and $\mathbf{F}_{2j}$. In consequence, $\mathbf{y}_{2j}$ is the result of inverse fast Fourier transform (IFFT) of $\hat{\mathbf{F}}$.

3.2 Backward Step

Following current deep learning design, we provide the gradient calculation with respect to $\mathbf{a}_{2j-1}$ and $\mathbf{c}_{2j}$ for all j. Denote the objective function (i.e., loss function) as $\mathcal{L}(\cdot)$. The backpropagation follows the chain rule, and we can get:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{a}_{2j-1}} = \frac{\partial\mathcal{L}}{\partial\mathbf{y}_{2j-1}}\frac{\partial\mathbf{y}_{2j-1}}{\partial\mathbf{a}_{2j-1}} = \frac{\partial\mathcal{L}}{\partial\mathbf{y}_{2j-1}} \odot \mathbf{y}_{2j-2}. \quad (9)$$

The backpropagation through the circulant matrix consists of derivatives of one-dimensional Fourier transform [Cheng *et al.*, 2015], which is easier to write with explicit expression of FFT as shown in Eq. (8):

$$\mathbf{F}_y = \mathrm{FFT}(\frac{\partial\mathcal{L}}{\partial\mathbf{y}_{2j}}) = \{\sum_{q=0}^{d-1} \frac{\partial\mathcal{L}}{\partial\mathbf{y}_{2j}}^q e^{-i2\pi\frac{p}{d}q}\}_{p=0}^{d-1},$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{y}_{2j-1}} = \mathrm{IFFT}(\mathrm{FFT}(\hat{\mathbf{c}}_{2j}) \odot \mathbf{F}_y), \quad (10)$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{c}_{2j}} = \mathrm{IFFT}(\mathrm{FFT}(\hat{\mathbf{y}}_{2j-1}) \odot \mathbf{F}_y).$$

Note that $\hat{\mathbf{c}}_{2j}$ and $\hat{\mathbf{y}}_{2j-1}$ are shifted from existing $\mathbf{c}_{2j}$ and $\mathbf{y}_{2j-1}$, respectively. According to [Cheng *et al.*, 2015], the

shift pattern is fixed, i.e., from $(0, 1, \ldots, d-1)$ to $(0, d-1, \ldots, 1)$.

However, we find that this derivation is not efficient due to the need of applying FFT on shifted vectors. When the input is a real vector for FFT, it can be easily proved that the FFT on the shifted vector is equal to the conjugate of the result of FFT on original vector. Therefore, we further improve the backward step as following:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{y}_{2j-1}} = \mathrm{IFFT}(\mathrm{conj}(\mathbf{F}_{2j}) \odot \mathbf{F}_y),$$
$$\frac{\partial\mathcal{L}}{\partial\mathbf{c}_{2j}} = \mathrm{IFFT}(\mathrm{conj}(\mathbf{F}_{2j-1}) \odot \mathbf{F}_y), \quad (11)$$

where the $\mathrm{conj}(\cdot)$ means taking the conjugate of input vector. In this way, we show that the backward step can reuse some results from the forward step. Since FFT operation is the major computation complexity, it can be noticed that forward step takes 2 FFT and 1 IFFT, while backward step takes 2 IFFT and 1 FFT. Overall, both forward and backward steps have similar computation complexity.

3.3 Block Partition

The product of interleaved circulant and diagonal matrices inherently forms only square matrices. As diagonal matrix is always a square matrix, to overcome the limitation, we decide to adopt the block-wise partitioning strategy for circulant matrices [Ding *et al.*, 2017]. This approach ensures compatibility with non-square weight matrices while preserving computational efficiency and storage benefits, maintaining the accuracy requirements of large-scale models.

In essence, this method partitions a non-square matrix into multiple square submatrices of equal size. If a dimension is not evenly divisible by the block size, it is automatically padded through replication. Each resulting square submatrix corresponds to a circulant matrix. Formally, given a matrix $\mathbf{C} \in \mathbb{R}^{n\times n}$, we partition it into blocks of size p, resulting in $q_1 = \lceil d_1/p \rceil$ blocks along the d_1-dimension and $q_2 = \lceil d_2/p \rceil$ blocks along the d_2-dimension. Consequently, the original matrix is decomposed into $q_1 \times q_2$ smaller circulant matrices. During matrix multiplication, block-wise multiplication is applied, where corresponding submatrices are multiplied element-wise. This effectively transforms the non-square matrix multiplication into multiple independent square circulant matrix multiplications, allowing us to fully leverage the computational properties of circulant matrices. The block-wise matrix-vector multiplication can then be formulated as:

$$\mathbf{h} = \mathbf{Cx} = \{\mathbf{h}_i\}_{i=0}^{q_1-1},$$

$$\mathbf{h}_i = \sum_{j=0}^{q_2-1} \mathbf{C}_{i,j}\mathbf{x}_j$$

$$= \sum_{j=0}^{q-1} \mathrm{IFFT}(\mathrm{FFT}(\mathbf{c}_{i,j}) \odot \mathrm{FFT}(\mathbf{x}_j)) \quad (12)$$

$$= \mathrm{IFFT}(\sum_{j=0}^{q-1} \mathrm{FFT}(\mathbf{c}_{i,j}) \odot \mathrm{FFT}(\mathbf{x}_j)),$$

(a) Adapter complexity on RoBERTa-base model.

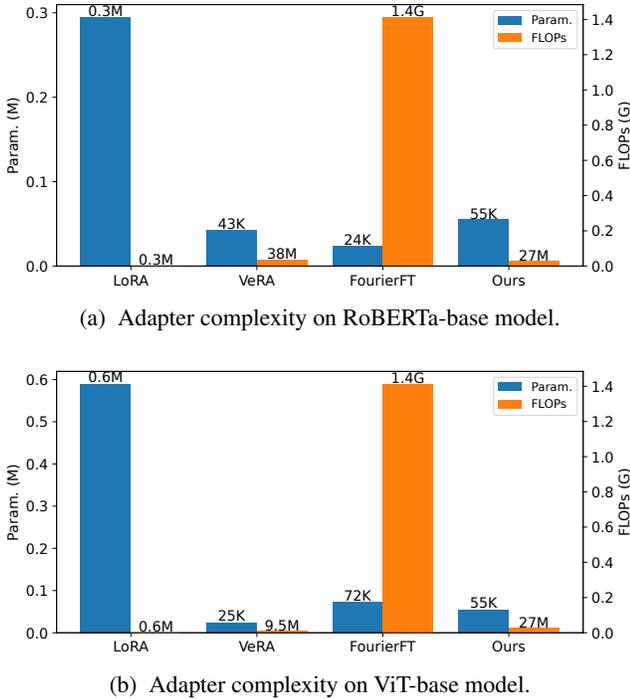(b) Adapter complexity on ViT-base model.

Figure 2: Complexity Analysis of different adapters. FF method is not presented due to its high cost in both parameters and FLOPs. Our proposed block circulant adapters can balance between parameter amount and FLOPs.

Each block matrix $\mathbf{C}_{i,j}$ represents a submatrix of the partitioned matrix, where i and j are integers ranging from 0 to $(q_1 - 1)$ and 0 to $(q_2 - 1)$, respectively. The vector $\mathbf{c}_{i,j} \in \mathbb{R}^{p \times 1}$ corresponds to each small circulant matrix.

It is worth mentioning that this method was specifically proposed to improve the generalization of our method. We only need to enable the partition for the first circulant matrix such that the rest matrices can always be square for maximum efficiency. However, even in scenarios where weight matrices are already square, this method also has the potential to further enhance model performance since there are more learnable parameters after partitioning. In our experiments with RoBERTa and ViT, all fine-tuned weight matrices were inherently square. Therefore, we set $p = d = 768$ to achieve maximum efficiency. For experiments with LLaMA model, there are non-square weight matrices that we need to set different partition sizes to fit the training.

3.4 Complexity Analysis

Assume that the number of layers to be fine-tuned is L_t and $\Delta\mathbf{W} \in \mathbb{R}^{d \times d}$(Priority is primarily given to cases where the weight matrix is square.)

Parameters. The number of parameters Θ to be trained for LoRA is given by $|\Theta|_{\text{LoRA}} = 2 \times d \times L_t \times r$, where $|\cdot|$ means the cardinality. For VeRA, the total number of trainable parameters is $|\Theta|_{\text{VeRA}} = (r + d) \times L_t$. However, in practice VeRA method can take large r to achieve good performance. For FourierFT, let number of spectral coefficients be n, and the total number of trainable parameters is

$|\Theta|_{\text{FourierFT}} = n \times L_t$. For CDVFT(Ours), in the case of square weight matrices, the circulant matrix block size be $p = d$. Assuming that the total number of circulant matrices and diagonal matrices is $2m - 1$, the total number of trainable parameters is $|\Theta|_{\text{CDVFT}} = (2m - 1) \times d \times L_t$ Fig. 2 shows that Fourier domain based method, i.e., FourierFT and ours, require much less number of parameters than LoRA.

FLOPs. Fig.2 also analyzes the computational complexity. It is important to note that computation complexity of FourierFT is independent of its parameter amount n since it always use 2D FFT to reconstruct $\Delta\mathbf{W}$. The complexity of our CDVFT is related to total number of circulant matrices and diagonal matrices and the circulant matrix block size, i.e., $2m - 1$ and p. The computational complexity of CDVFT is smaller than FourierFT. The main reason for the complexity difference is that in FourierFT, the computational complexity of the 2D FFT for computing $\Delta\mathbf{W}$ is $O(d^2 log(d^2)))$. In CDVFT, the complexity brought by element-wise product and 1D FFT is $O(mdlog(d))$, which significantly reduces the computational complexity while keeping similar number of training parameters. It can be seen that across all models, LoRA has the largest parameter amount, and FourierFT has the largest FLOPs. The complexity of the different models listed in Table 3 verifies the analysis.

The case of non-square matrices. When the weight matrices in the fine-tuning layer are non-square, a block-wise partitioning strategy is required to ensure proper fine-tuning. In this case, the number of parameters and FLOPs depend on the block size p of the circulant matrix. A smaller p results in more circulant matrix blocks, increasing the number of parameters required for representation and making the computation more complex. Therefore, in our experiments, we prioritize choosing a larger p. To validate our reasoning, we present results under different values of p in Table 3. Furthermore, our approach achieves significantly lower FLOPs than FourierFT and requires fewer trainable parameters than LoRA, striking an effective balance between computational efficiency and model compactness.

4 Experiments

In this section, we evaluate our CDVFT method across different domains, i.e., natural language understanding (NLU) and computer vision (CV): (1) fine-tune the RoBERTa model [Liu et al., 2019] on the General Language Understanding Evaluation (GLUE) dataset [Wang et al., 2019]; (2) fine-tune the vision transformer model [Dosovitskiy et al., 2021] for various image classification tasks across different domains; (3) fine-tuning the LLaMA2-7b model on the Alpaca and GSM8K datasets.

Our proposed CDVFT is also compared with different adapters: Traditional full fine-tuning (FF) updates all model parameters, achieving high accuracy but incurring significant computational costs. Low-Rank Adaptation (LoRA) [Hu et al., 2021] is a widely adopted fine-tuning technique for large models. It decomposes the weight matrices into low-rank components, significantly reducing the number of trainable parameters while maintaining performance. Vector-based Random Matrix Adaptation (VeRA) [Kopiczko et al., 2024]

Method	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
FF	94.8	90.2	63.6	92.8	78.7	91.2	85.2
LoRA	$95.1_{\pm0.2}$	$89.7_{\pm0.7}$	$63.4_{\pm1.2}$	$93.3_{\pm0.3}$	$78.4_{\pm0.8}$	$91.5_{\pm0.2}$	85.2
VeRA	$94.6_{\pm0.1}$	$89.5_{\pm0.5}$	$65.8_{\pm0.8}$	$91.8_{\pm0.2}$	$78.7_{\pm0.7}$	$90.7_{\pm0.2}$	85.2
FourierFT	$94.2_{\pm0.3}$	$90.0_{\pm0.8}$	$63.8_{\pm1.6}$	$92.2_{\pm0.1}$	$79.1_{\pm0.5}$	$90.8_{\pm0.2}$	85.0
$\text{Ours}_{p=768}$	$94.4_{\pm0.5}$	$90.2_{\pm0.3}$	$64.5_{\pm1.2}$	$92.2_{\pm0.2}$	$78.7_{\pm1.2}$	$90.5_{\pm0.2}$	85.1

Table 1: The performance of LoRA, FourierFT and our CDVFT methods is reported by fine-tuning the RoBERTa base model on 6 datasets of the GLUE benchmark. The experiments report Matthew correlation coefficient (MCC) for CoLA, Pearson correlation coefficient (PCC) for STS-B, and accuracy (Acc.) for all remaining tasks. Following [Gao *et al.*, 2024a], we also report the median result out of 5 runs, each with a different random seed. The best result for each dataset is highlighted in bold. Higher metric value means better model performance for all datasets.

extends LoRA by sharing the low-rank matrices across all layers and inserting two additional vectors after the decomposed matrices, further improving efficiency. LaMDA [Azizi *et al.*, 2024] is another LoRA-based method that freezes the first projection matrix (PMA) during fine-tuning, gradually freezes the second projection matrix (PMB) in the early training stages, and introduces a low-rank square matrix between them. LaMDA++ [Azizi *et al.*, 2024] further extends this approach by assigning different ranks to different fine-tuning layers. FourierFT [Gao *et al.*, 2024b] is a state-of-the-art fine-tuning method that transforms frequency-domain data into trainable weight matrices via the Fourier transform. By training only in the frequency domain, it effectively reduces the number of trainable parameters. Notably, our approach also leverages the Fourier domain, but with lower FLOPs. This is achieved by employing a more efficient FFT operation in our equation (8) rather than the 2D FFT computation used in FourierFT.

4.1 Natural Language Understanding

Models and Datasets. We evaluate CDVFT on the GLUE benchmark dataset, which consists of a diverse range of NLP tasks, each representing a specific type of language understanding task. These tasks include question answering, sentiment analysis, textual entailment, etc. Following the experiment setting as in [Gao *et al.*, 2024a], fine-tuning process runs on following tasks: CoLA, Corpus of Linguistic Acceptability [Warstadt *et al.*, 2019], which determines whether sentences adhere to grammatical rules; SST-2, Stanford Sentiment Treebank [Socher *et al.*, 2013], which classifies the sentiment of sentences as positive or negative; MRPC, Microsoft Research Paraphrase Corpus [Dolan and Brockett, 2005], which assesses whether two sentences convey the same meaning; STS-B, Semantic Textual Similarity Benchmark [Cer *et al.*, 2017], which measures the semantic similarity score between sentence pairs; QNLI, Question Natural Language Inference [Rajpurkar, 2016], which evaluates whether the second sentence correctly answers the question posed by the first; and RTE, Recognizing Textual Entailment [Dagan *et al.*, 2005], which identifies whether there is an entailment relationship between sentence pairs, functioning as a binary classification task. RoBERTa base model [Liu *et al.*, 2019] is a transformer based foundation model, which is widely used in natural language processing. It improves over existing under-trained BERT model [Devlin, 2018] while pre-

Method	FLOPs	Param.	RESISC45	CIFAR100	Avg.
FF	-	-	96.1	92.4	94.2
LoRA	$0.61M$	$0.59M$	92.7	92.0	92.4
VeRA	$9.48M$	$0.02M$	77.0	84.8	80.9
FourierFT	$1.41G$	$0.07M$	92.0	91.2	91.6
$\text{Ours}_{p=768}$	$2.72M$	$0.06M$	92.0	91.1	91.6

Table 2: Fine-tuning results of the ViT Base model on different image classification datasets. The experiments report the accuracy (%) after 10 epochs.

serving the powerful attention mechanism. Thus, it is selected to serve as the foundation model for GLUE dataset.

Implementation Details. Our CDVFT uses a total of 3 factor matrices, i.e., $m = 2$. The block size of the circulant matrix is the same as the matrix size, i.e., $p = d = 768$, and no block is performed. It should be noted that only query and value weights in each transformer block are finetuned, which is also applied to LoRA, VeRA and FourierFT as in [Gao *et al.*, 2024a]. All implementations are in PyTorch [Paszke *et al.*, 2019]. It can be seen that the optimizer is AdamW [Loshchilov, 2017]. For each dataset, there are different learning rates for foundation model language heads, query and value weight matrices. The scaling value is the α as in Eq. (4). The batch size and maximum input sequence length is set the same for all datasets.

Results. Table 1 summarizes fine-tuning results of all methods. The median metric value with standard deviation is reported out of 5 runs of experiments for each fine-tuning method, where each run takes a different random seed. The best performance for each dataset is highlighted in bold. Overall, compared with LoRA and FourierFT, our CDVFT method achieves comparable or even better performance. Besides, according to Fig. 2, our CDVFT results in $5.33\times$ less number of trainable parameters than LoRA and $51.89\times$ less FLOPs than FourierFT while fine-tuning RoBERTa base model on GLUE dataset.

4.2 Image Classification

Models and Datasets. The experiment evaluates the performance of our CDVFT method in image classification tasks, utilizing the Vision Transformer (ViT) by [Dosovitskiy *et al.*, 2021] as the foundation model. Following the setting in [Gao *et al.*, 2024a], we fine-tune on several challenging image clas-

Method	MT-Bench			GSM8K		
	FLOPs	Param.	Score	FLOPs	Param.	Acc.
LoRA	$0.03G$	$33.55M$	5.20	$0.01G$	$28.05M$	36.9
VeRA	$2.29G$	$1.65M$	5.08	-	-	-
FourierFT	$133.14G$	$0.06M$	5.18	-	-	-
LaMDA	-	-	-	$0.06G$	$4.37M$	37.9
LaMDA++	-	-	-	-	$5.12M$	38.2
Ours$_{p=2048}$	$0.06G$	$1.05M$	5.42	$0.22G$	$7.26M$	37.8
Ours$_{p=4096}$	$0.05G$	$0.79M$	5.27	$0.17G$	$4.51M$	37.9

Table 3: Instruction tuning performance of LLaMA2-7B model. Higher accuracy and score value means better tuning performance. Unavailable results are represented with "-". For example, LaMDA++ is lack of rank information resulting in unknown FLOPs.

sification datasets, only two are listed here for observation. RESISC45 [Cheng *et al.*, 2017] provides a diverse range of remote sensing images; CIFAR-100 [Krizhevsky *et al.*, 2009] is classical datasets of tiny images in 100 categories.

Implementation details. We set $m = 2$ and $p = d = 768$ for fine-tuning ViT base model across all these datasets. The ranks of LoRA and VeRA are 16 and 256 respectively. The n of FourierFT is set to 3000. For all method, fine-tuning only runs the query and value weight matrices of ViT, which is the same as in [Gao *et al.*, 2024a]. The learning rate is set differently for fine-tuning ViT heads and query and value weight matrices.

Results. Table 2 presents the performance results on two image classification datasets after fine-tuning the ViT base model. Our CDVFT method demonstrates significant efficiency, requiring $10.7\times$ fewer parameters than LoRA and $51.9\times$ fewer FLOPs than FourierFT, while achieving similar or even better classification accuracy. Additionally, when compared to the latest fine-tuning method, VeRA, our approach uses fewer parameters. Although our method incurs a higher number of FLOPs than VeRA, it is important to note that VeRA's accuracy is relatively lower. To achieve comparable results to our method, VeRA would need to increase its rank, which would result in a corresponding increase in FLOPs.

4.3 Instruction Tuning

Models and datasets. Instruction tuning is a training method that enables models to learn how to perform tasks based on natural language instructions. Its primary goal is to enhance the versatility of models in handling a wide range of tasks, allowing for improved understanding and execution of natural language commands. Compared to traditional supervised learning, instruction tuning emphasizes the model's broad adaptability to task requirements and can be applied in scenarios such as question-answering systems, multi-task learning, and natural language generation.

In this experiment, FourierFT and CDVFT are used to fine-tune LLaMA2-7B on the Alpaca dataset. Specifically, LLaMA2-7B is part of the LLaMA (Large Language Model Meta AI) series developed by Meta, featuring significant improvements in performance and scalability over the first generation. This model generates high-quality text while utilizing fewer computing resources. The Alpaca dataset is developed based on the Stanford Alpaca project and expanded

from instruction data generated by OpenAI's GPT model, focusing on tasks related to natural language understanding and generation. The experiment generates answers to predefined questions from MT-Bench, evaluated using GPT-4. MT-Bench is a benchmark tool specifically designed to assess multi-task language models, quantifying their generalization ability, speed, and accuracy by comparing performance across multiple tasks.

Additionally, the GSM8K dataset is used for task-specific fine-tuning and evaluation of the model. GSM8K consists of approximately 8,000 math problem-solving instances, aimed at improving the model's ability to perform complex reasoning tasks. The dataset includes problem descriptions and detailed solutions, enabling the model to learn how to solve complex mathematical problems.

Implementation details. For FourierFT, the experiments follow previous work and adopt the configuration $n = 1000$. For LoRA, $r = 8$ on the Alpaca dataset and $r = 64$ on the GSM8K dataset. For VeRA and LaMDA, r is 1024 and 32 respectively. For CDVFT, the experiments still set $m = 2$, and the partition size p is set as large as possible to achieve the minimum storage and computational cost. Therefore, p is set to 4096 and 2048.

Following [Gao *et al.*, 2024b], we apply block circulant fine-tuning on query and value weight matrices inside the attention layer of two RoBERTa models and the LLaMA2-7B model fine-tuned on the alpaca dataset. (Except VeRA fine-tune on all layers in MHSA and MLP) Following [Azizi *et al.*, 2024], we fine-tune on the MHSA and FFN layers of LLaMA2-7B model on the GSM8K dataset. The classification head is fully fine-tuned.

We evaluate the fine-tuned model on the Alpaca dataset using MT-Bench [Zheng *et al.*, 2023], with GPT-4 [gpt, 2023] subsequently assigning scores to the model's responses for 80 multi-turn questions on a scale of 10.

Results. Table 3 shows the performance of the fine-tuned LLaMA2-7b model on the Alpaca and GSM8K datasets. It can be noticed that increasing partition size results in decreasing parameter amount and FLOPs. However, the task score or accuracy does not always decrease with larger partition size. This may be caused by the over-parameterization of the large LLaMA model or the strong structure of the proposed method that may serve as a regularization. Compared with other adapters, our method can balance between parameters amount and FLOPs.

5 Conclusion

Motivated by the recent success in Fourier domain based fine-tuning method, this paper proposes the CDVFT method that also learns parameters in Fourier domain. In particular, our method results in both trainable parameters savings and FLOPs reduction when compared with existing methods. The downstream task performance of our fine-tuned model achieves similar performance and sometime even better results across both natural language understanding and computer vision applications. These results effectively demonstrate the promising potential of our method and also the Fourier domain based fine-tuning methods.

Acknowledgments

Contribution Statement

Siyu Liao and Zhongfeng Wang are co-corresponding authors.

References

[Azizi *et al.*, 2024] Seyedarmin Azizi, Souvik Kundu, and Massoud Pedram. Lamda: Large model fine-tuning via spectrally decomposed low-dimensional adaptation, 2024.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[Cer *et al.*, 2017] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[Chen *et al.*, 2024] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[Cheng *et al.*, 2015] Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE international conference on computer vision*, pages 2857–2865, 2015.

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

[Dagan *et al.*, 2005] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer, 2005.

[Devlin, 2018] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Ding *et al.*, 2017] Caiwen Ding, Siyu Liao, Yanzhi Wang, Zhe Li, Ning Liu, Youwei Zhuo, Chao Wang, Xuehai Qian, Yu Bai, Geng Yuan, et al. Circnn: accelerating and compressing deep neural networks using block-circulant weight matrices. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 395–408, 2017.

[Dolan and Brockett, 2005] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*, 2005.

[Dosovitskiy *et al.*, 2021] 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. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[Gao *et al.*, 2024a] Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform, 2024.

[Gao *et al.*, 2024b] Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, 2024.

[gpt, 2023] Gpt-4 technical report. 2023.

[Houlsby *et al.*, 2019] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[Hu *et al.*, 2021] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[Huhtanen and Perämäki, 2015] Marko Huhtanen and Allan Perämäki. Factoring matrices into the product of circulant and diagonal matrices. *Journal of Fourier Analysis and Applications*, 21:1018 – 1033, 2015.

[Kopiczko *et al.*, 2024] Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. Vera: Vector-based random matrix adaptation, 2024.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[Lei *et al.*, 2023] Tao Lei, Junwen Bai, Siddhartha Brahma, Joshua Ainslie, Kenton Lee, Yanqi Zhou, Nan Du, Vincent Zhao, Yuexin Wu, Bo Li, et al. Conditional adapters: Parameter-efficient transfer learning with fast inference. *Advances in Neural Information Processing Systems*, 36:8152–8172, 2023.

[Li and Liang, 2021] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021.

[Li et al., 2024a] Shikai Li, Jianglin Fu, Kaiyuan Liu, Wentao Wang, Kwan-Yee Lin, and Wayne Wu. Cosmicman: A text-to-image foundation model for humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6955–6965, 2024.

[Li et al., 2024b] Xingyu Li, Lu Peng, Yuping Wang, and Weihua Zhang. Open challenges and opportunities in federated foundation models towards biomedical healthcare. *arXiv preprint arXiv:2405.06784*, 2024.

[Liu et al., 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[Loshchilov, 2017] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[Mahabadi et al., 2021] Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. 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 565–576, 2021.

[Paaß and Giesselbach, 2023] Gerhard Paaß and Sven Giesselbach. Knowledge acquired by foundation models. In *Foundation Models for Natural Language Processing: Pre-trained Language Models Integrating Media*, pages 161–185. Springer, 2023.

[Paszke et al., 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[Radford and Narasimhan, 2018] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[Rajpurkar, 2016] P Rajpurkar. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

[Sung et al., 2022] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems*, 35:12991–13005, 2022.

[Touvron et al., 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[Wang et al., 2018] Shuo Wang, Zhe Li, Caiwen Ding, Bo Yuan, Qinru Qiu, Yanzhi Wang, and Yun Liang. C-lstm: Enabling efficient lstm using structured compression techniques on fpgas. In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 11–20, 2018.

[Wang et al., 2019] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019.

[Warstadt et al., 2019] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments, 2019.

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

[Zheng et al., 2023] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.