

PARAMETER-EFFICIENT FINE-TUNING VIA CIRCULAR CONVOLUTION

Anonymous authors

Paper under double-blind review

ABSTRACT

Low-Rank Adaptation (LoRA) has gained popularity for fine-tuning large foundation models, leveraging low-rank matrices \mathbf{A} and \mathbf{B} to represent weight changes (*i.e.*, $\Delta\mathbf{W} = \mathbf{BA}$). This method reduces trainable parameters and mitigates heavy memory consumption associated with full delta matrices by sequentially multiplying \mathbf{A} and \mathbf{B} with the activation. Despite its success, the intrinsic low-rank characteristic may limit its performance. Although several variants have been proposed to address this issue, they often overlook the crucial computational and memory efficiency brought by LoRA. In this paper, we propose Circular Convolution Adaptation (C^3A), which not only achieves high-rank adaptation with enhanced performance but also excels in both computational power and memory utilization. Extensive experiments demonstrate that C^3A consistently outperforms LoRA and its variants across various fine-tuning tasks.

1 INTRODUCTION

In recent years, Large Foundation Models (LFMs) have witnessed a pronounced ascendance in both scholarly and practical realms, attributable to their exceptional efficacy across diverse tasks in natural language processing (NLP) (Brown et al., 2020; Touvron et al., 2023), computer vision (CV) (Radford et al., 2021; Kirillov et al., 2023), and other domains (Li et al., 2024). Distinguished by an extensive parameter count and significant computational requisites, these models have established unprecedented benchmarks in both accuracy and versatility. Nonetheless, their considerable size and intricate structure present formidable obstacles for efficient fine-tuning, especially within resource-constrained environments (Malladi et al., 2023; Zhang et al., 2024b). To mitigate these challenges, parameter-efficient fine-tuning (PEFT) techniques (Mangrulkar et al., 2022), exemplified by Low-Rank Adaptation (LoRA) (Hu et al., 2021), have emerged as highly effective solutions.

LoRA reduces the number of trainable parameters by leveraging low-rank matrices to approximate alterations in weights, thereby facilitating fine-tuning without degrading the model’s efficacy. Specifically, LoRA can be articulated mathematically as follows:

$$\mathbf{W}\mathbf{x} = (\mathbf{W}_0 + \Delta\mathbf{W})\mathbf{x} = \mathbf{W}_0\mathbf{x} + \mathbf{B}(\mathbf{A}\mathbf{x}),$$

where $\mathbf{W}, \mathbf{W}_0, \Delta\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ are weight matrices, $\mathbf{B} \in \mathbb{R}^{d_1 \times r}$, $\mathbf{A} \in \mathbb{R}^{r \times d_2}$ are low-rank matrices formulated to construct $\Delta\mathbf{W}$, and $\mathbf{x} \in \mathbb{R}^{d_2}$ are the activations. The number of trainable parameters is $r(d_1 + d_2)$, thereby motivating the selection of $r \ll \min(d_1, d_2)$

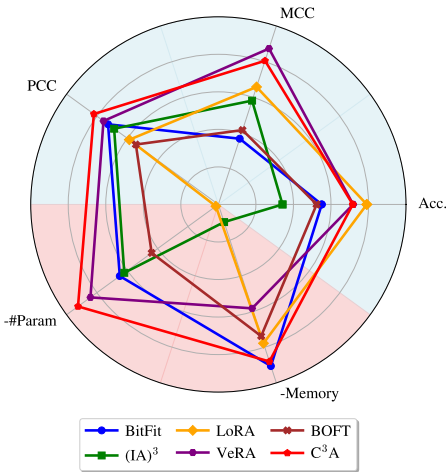


Figure 1: Relative comparison of C^3A and baselines on RoBERTa-Base. The Pearson Correlation Coefficient (PCC) is evaluated on STS-B and the Matthew’s Correlation Coefficient (MCC) on CoLA. Accuracies across SST-2, MRPC, QNLI, and RTE are averaged and reported as Acc. #Param shows the reduced number of learnable parameters compared to LoRA, and -Memory indicates the decrease in peak GPU memory usage during training, also compared to LoRA. The metrics in blue pertain to performance-related values, whereas those shadowed in red correspond to values associated with resource consumption. All metrics are the higher the better. See Table 2 for more statistics.

(e.g., $r = 8$ for $d_1 = d_2 = 1024$) to attain elevated parameter efficiency. Nonetheless, as elaborated by Zeng & Lee (2023), the potential of LoRA to encapsulate a target model is inherently constrained by r . In an effort to reconcile the dichotomy between performance and efficiency, Kopiczko et al. (2023) introduced Vector Random Matrix Adaptation (VeRA). VeRA attains comparable performance with a markedly reduced count of trainable parameters via fixed random-matrix projections. However, despite its minimal parameter count, VeRA demands considerable computational resources and memory capacity due to the extensive nature of the random matrices employed for projection. As depicted in Figure 1, other representative works share the same resource problem. This precipitates the following open research question within the scope of PEFT:

Beyond low parameter counts, how to achieve high-rank adaptation without incurring significant costs of time and memory?

To address this question, we introduce Circular Convolution Adaptation (C^3A), which incorporates the circular convolution operator (Bamieh, 2018). Circular convolution has garnered significant attention in both signal processing (Li et al., 2020) and cryptography (Dworkin et al., 2001) due to its exceptional efficiency and compactness. This operator can be equivalently expressed as multiplication by a circulant matrix, providing rank flexibility that is independent of the number of trainable parameters. Furthermore, by employing the Fast Fourier Transform (FFT), C^3A achieves superior time and memory efficiency compared to the direct multiplication of the circulant matrix (Bamieh, 2018), which makes it competitive with LoRA in terms of efficiency.

In addition, as explicated by Dosovitskiy et al. (2020), dense linear layers exhibit a deficiency of inductive biases, engendering a complex optimization landscape. Consequently, this hampers the effectiveness of transformers in comparison to Convolutional Neural Networks (CNNs) under conditions of limited data availability. Within the framework of a constrained training dataset for the downstream task, we postulate that a robust inductive bias could potentially augment adaptation performance. The circular pattern in C^3A serves precisely as such an inductive bias.

In summary, circular convolution presents a promising solution for circumventing the rank limitations of LoRA at minimal costs. Our contributions can be summarized as follows:

- ❶ We introduce C^3A , a novel approach for PEFT. This method leverages the circular convolution operation and its equivalent circulant matrix to provide a flexible rank, which is free of linear constraint by the number of trainable parameters, for the delta matrix.
- ❷ Leveraging the elegant diagonalization of the circulant matrix, we implement both the forward pass and backpropagation using FFT. With the incorporation of FFT, the computation and memory efficiency of C^3A excels. C^3A strikes a unique balance between performance and efficiency.
- ❸ To offer greater flexibility in controlling the number of trainable parameters, we extend C^3A by incorporating block-circular convolution, which results in block-circulant matrices. This extension allows C^3A to achieve fully customizable parameter counts as well as adaptable rank configurations.
- ❹ We validate C^3A through comprehensive fine-tuning experiments across diverse tasks including natural language understanding, instruction tuning and image classification. Experiments demonstrate C^3A 's outstanding accuracy and memory merits compared to existing state-of-the-art methods.

2 RELATED WORK

2.1 PARAMETER-EFFICIENT FINE-TUNING

Research on PEFT has generally progressed along three main directions. The first direction involves partially updating the pre-trained neural network (e.g., the layer norm (Basu et al., 2024) or the biases (Zaken et al., 2021)). Traditional methods relied on hand-crafted heuristics (Raghu et al., 2019) to identify which parameters are crucial and should be fine-tuned. More advanced approaches employ optimization techniques (Guo et al., 2020; Xu et al., 2021; Fu et al., 2023). For example, Guo et al. (2020) reformulated such a discrete optimization problem into a continuous one by employing Bernoulli masks and the Gumbel-softmax approximation (Jang et al., 2016).

The second direction emerged to maintain the integrity of the pre-trained model while enabling a high degree of parameter sharing through adapter-based methods (He et al., 2021; Rebuffi et al.,

2017; Rücklé et al., 2020; Liu et al., 2022; Lian et al., 2022). These works focus on integrating additional modules, termed adapters, to fit the downstream task, effectively decoupling the pre-trained model parameters from those specific to the downstream task. Prompt Tuning (Brown et al., 2020; Gao et al., 2020; Chen et al., 2023; Zhang et al., 2024a) and Prefix Tuning (Li & Liang, 2021; Jia et al., 2022) also fall into this category, despite ignoring potential semantic meanings.

The final direction is characterized by delta-weight-based methods, such as Low-Rank Adaptation (LoRA) (Hu et al., 2021) and Orthogonal Fine-tuning (OFT) (Qiu et al., 2023). These methods bridge the gap between the pre-trained model and the downstream task by adaptive delta weights, which are stored separately while used in combination with the pre-trained weights. This unique design enables disentanglement of the pretrained and downstream-specific weights. Namely, it achieves parameter sharing and preserves the ability to integrate without additional inference cost. LoRA models the delta-weights by an additive matrix while OFT does it by a multiplicative one. To further improve either parameter efficiency or performance, many variants has been proposed for both of the methods (Kopiczko et al., 2023; Liu et al., 2024; 2023; Yuan et al., 2024; Hayou et al., 2024b; Gao et al., 2024). However, these methods can hardly achieve high parameter efficiency and performance without incurring heavy computation and memory usage.

2.2 CIRCULAR CONVOLUTION

Circular convolution has been extensively studied in signal processing (Rabiner et al., 1978; McGillem & Cooper, 1984; Li et al., 2020) and cryptography (Dworkin et al., 2001; Gong et al., 2024). Owing to its computational advantages, circular convolution has also been explored in machine learning for generating long embeddings of high-dimensional data (Yu et al., 2014) and compressing heavily parameterized layers (Cheng et al., 2015; Ding et al., 2017). Remarkably, it achieves these efficiencies without significant performance degradation, which makes it a promising technique for fine-tuning applications.

Despite its success in small neural networks such as LeNet (Cheng et al., 2015), circular convolution has not demonstrated lossless performance in modern large foundational models (LFMs) or even in their base architecture, the transformer. This limitation may be attributed to the conflict between its high intrinsic bias (*i.e.*, the circulant pattern) and the vast amount of data required for training LFMs. Conversely, when fine-tuning LFMs, it is often impractical to collect as much data as needed for training from scratch. In such scenarios, the intrinsic bias of circular convolution could potentially serve as a regularization mechanism, thereby benefiting the optimization process of fine-tuning.

3 METHOD

In this section, we present C^3A (see an overview in Figure 2), a novel PEFT approach based on the circular convolution. C^3A follows LoRA’s setting of learning an additive linear operation over the original dense linear transformation. However, instead of using low-rank decomposition and the matrix multiplication operator, C^3A resorts to circular convolution as this additive linear operation. Section 3.1 introduces the notations we use. Section 3.2 discusses the circular convolution operator, its equivalent circulant matrix, and its calculation in the frequency domain. Section 3.3 details an efficient method for backpropagation. Section 3.4 describes block-wise convolution for controlling the number of trainable parameters. Finally, Section 3.5 analyzes the computational complexity.

3.1 NOTATIONS

The adapted weight matrix, the original weight matrix, and the delta matrix are denoted by \mathbf{W} , \mathbf{W}_0 , and $\Delta\mathbf{W}$, respectively ($\mathbf{W}, \mathbf{W}_0, \Delta\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$). The activation vector of the previous layer is denoted by $\mathbf{x} \in \mathbb{R}^{d_2}$. The post-transformation vector is \mathbf{z} , where $\mathbf{z} = \mathbf{W}\mathbf{x} \in \mathbb{R}^{d_1}$, and the incremental part is denoted by $\Delta\mathbf{z}$, where $\Delta\mathbf{z} = \Delta\mathbf{W}\mathbf{x} \in \mathbb{R}^{d_1}$. The matrices \mathbf{A} and \mathbf{B} are low-rank matrices introduced by LoRA to represent $\Delta\mathbf{W}$, with r being their rank. r_v specifies the rank of the random projection matrix used in VeRA. The circular convolution kernel of C^3A is denoted by $\Delta\mathbf{w}$ and the circular convolution operator by \star . The loss function is represented by \mathcal{L} . The Fast Fourier Transform and its inverse are denoted by FFT and iFFT, respectively. The Hadamard product is denoted by \circ .

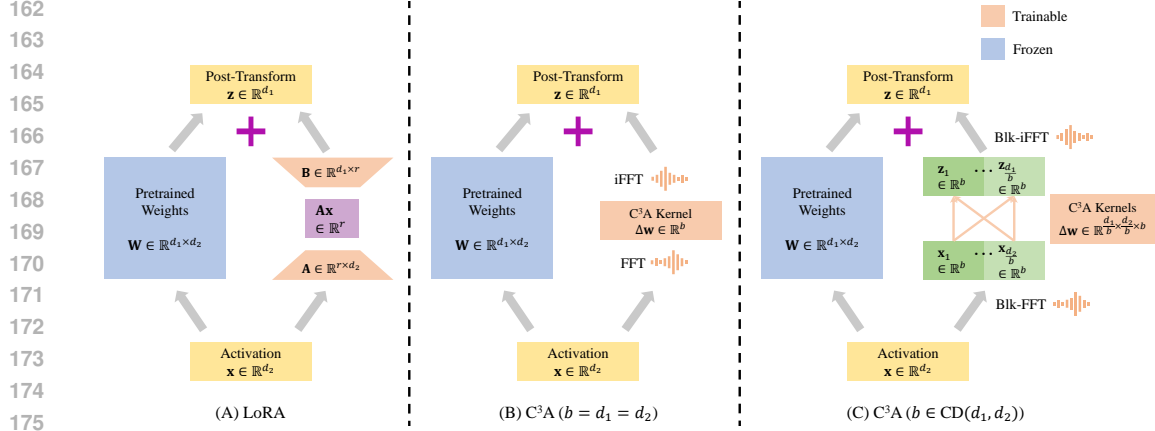


Figure 2: **Overview of LoRA (A) and our C³A (B,C) method.** In LoRA, only low-rank matrices **A** and **B** are trained and the delta weight is represented by their product (*i.e.*, $\Delta\mathbf{W} = \mathbf{B}\mathbf{A}$). The total trainable parameter number is $r(d_1 + d_2)$, which is associated with the rank of the delta weight. In C³A, circular convolution kernels $\Delta\mathbf{w}$ are tuned to adapt to the downstream task and the delta weight is represented by the (block-)circular matrix they construct (*i.e.*, $\Delta\mathbf{W} = \mathcal{C}_{(\text{blk})}(\Delta\mathbf{w})$). The total trainable parameter count is $\frac{d_1 d_2}{b}$, which disentangles with the rank of the delta weight. Here, b is the block size of the block-circular matrix and it should be a common divisor (CD) of d_1 and d_2 .

3.2 CIRCULAR CONVOLUTION

Firstly, for simplicity, we assume $d_1 = d_2 = d$ and $\Delta\mathbf{w} \in \mathbb{R}^d$. The circular convolution operator is defined as $\Delta\mathbf{z} = \Delta\mathbf{w} \star \mathbf{x} = \mathcal{C}(\Delta\mathbf{w})\mathbf{x}$, where $\mathcal{C}(\cdot)$ is a function which takes a vector and outputs the corresponding circulant matrix. Concretely, the first row of $\mathcal{C}(\Delta\mathbf{w})$ is $\Delta\mathbf{w}$ and the following rows are equal to the row above them periodically shifted to the right by one element. In math,

$$\mathcal{C}(\Delta\mathbf{w}) = \begin{bmatrix} \Delta w_1 & \Delta w_2 & \cdots & \Delta w_{d-1} & \Delta w_d \\ \Delta w_d & \Delta w_1 & \cdots & \Delta w_{d-2} & \Delta w_{d-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \Delta w_3 & \Delta w_4 & \cdots & \Delta w_1 & \Delta w_2 \\ \Delta w_2 & \Delta w_3 & \cdots & \Delta w_d & \Delta w_1 \end{bmatrix}.$$

Theoretically, the rank of $\mathcal{C}(\Delta\mathbf{w})$ is given by $d - \text{Deg}(\text{gcd}(f(x), x^d - 1))$ (Ingleton, 1956), where $\text{Deg}(\cdot)$ denotes the degree of a polynomial, $f(x)$ is the polynomial associated with $\Delta\mathbf{w}$ (*i.e.*, $f(x) = \sum_{i=1}^d \Delta w_i x^{i-1}$), and $\text{gcd}(\cdot)$ represents the greatest common divisor. Consequently, the theoretical upper bound on the rank of $\mathcal{C}(\Delta\mathbf{w})$ is d . By learning $\Delta\mathbf{w}$ in the \mathbb{R}^n oracle, C³A automatically achieves dynamic rank selection, which is not linearly constrained by the number of learnable parameters, unlike LoRA.

To achieve high efficiency, enlightened by Ding et al. (2017), we leverage the beautiful circulant structure of $\mathcal{C}(\Delta\mathbf{w})$, which makes it diagonalizable by the Fourier basis (**F**). In math, it can be described as $\mathcal{C}(\Delta\mathbf{w}) = \mathbf{F} \frac{\Lambda}{d} \mathbf{F}^{-1}$ (Golub & Van Loan, 1996), where Λ is its eigenvalues and can be calculated by a Fourier transform of the first row (*i.e.*, $\Lambda = \text{diag}(\mathbf{F}\Delta\mathbf{w})$). Therefore, we can calculate $\Delta\mathbf{w} \star \mathbf{x}$ as

$$\begin{aligned} \Delta\mathbf{w} \star \mathbf{x} &= \mathbf{F} \text{diag}\left(\frac{\mathbf{F}\Delta\mathbf{w}}{d}\right) \mathbf{F}^{-1} \mathbf{x} \\ &= \text{FFT}(\text{FFT}(\Delta\mathbf{w}) \circ \text{iFFT}(\mathbf{x})). \end{aligned} \quad (1)$$

3.3 BACKPROPAGATION

To effectuate backpropagation with optimal efficiency, it is imperative to obtain the analytical derivatives of the loss function \mathcal{L} with respect to $\Delta\mathbf{w}$ and \mathbf{x} . Following the approach outlined in Ding et al. (2017), we aim to explain backpropagation using simpler language. By applying the chain rule, these

derivatives are delineated as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial \Delta \mathbf{z}}{\partial \mathbf{x}} \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{z}}, \quad \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{w}} = \frac{\partial \Delta \mathbf{z}}{\partial \Delta \mathbf{w}} \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{z}}. \quad (2)$$

Given that $\Delta \mathbf{z} = \mathcal{C}(\Delta \mathbf{w})\mathbf{x}$, it logically follows that $\frac{\partial \Delta \mathbf{z}}{\partial \mathbf{x}} = \mathcal{C}(\Delta \mathbf{w})$. Concerning $\frac{\partial \Delta \mathbf{z}}{\partial \Delta \mathbf{w}}$, we observe the commutative property of the circular convolution operation (i.e., $\mathcal{C}(\Delta \mathbf{w})\mathbf{x} = \mathcal{C}(\mathbf{x})\Delta \mathbf{w}$), which implies $\frac{\partial \Delta \mathbf{z}}{\partial \Delta \mathbf{w}} = \mathcal{C}(\mathbf{x})$. Substituting these findings into Equation 2, we derive:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \mathcal{C}(\Delta \mathbf{w}) \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{z}}, \quad \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{w}} = \mathcal{C}(\mathbf{x}) \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{z}}.$$

These expressions can also be interpreted as circular convolutions:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \Delta \mathbf{w} \star \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{z}}, \quad \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{w}} = \mathbf{x} \star \frac{\partial \mathcal{L}}{\partial \Delta \mathbf{z}}.$$

By meticulously executing this derivative computation in accordance with Equation 1, backpropagation can harness the computational efficacy facilitated by the FFT algorithm.

3.4 BLOCK-CIRCULAR CONVOLUTION

Notwithstanding the elegance and efficiency of the circular convolution operator, it is subject to two fundamental limitations stemming from the constraint that the convolution kernel must match the dimensions of the activation vector: ① *It is inapplicable to non-square weight matrices.* ② *The count of learnable parameters remains fixed.* The first restriction hampers its applicability in scenarios such as fine-tuning a LLaMA3-8B model, where the weight matrix dimensions include 4096×1024 . The second constraint diminishes the adaptability of C³A, presenting challenges in addressing complex downstream tasks that necessitate a greater number of learnable parameters. To mitigate these limitations, we employ **block-circular convolution** (Ding et al., 2017). By partitioning the activation vector \mathbf{x} and the post-transformation vector $\Delta \mathbf{z}$ into blocks of identical size, unique convolution kernels can be allocated to each pair of these blocks. Specifically,

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{\frac{d_2}{b}} \end{bmatrix}, \quad \Delta \mathbf{z} = \begin{bmatrix} \Delta \mathbf{z}_1 & \Delta \mathbf{z}_2 & \cdots & \Delta \mathbf{z}_{\frac{d_1}{b}} \end{bmatrix},$$

where b is the block size and b need to be a common divisor of d_1 and d_2 . We will need $\frac{d_1 d_2}{b^2}$ convolution kernels to densely connect these blocks, which can be expressed in math as

$$\Delta \mathbf{z}_i = \sum_{j=1}^{\frac{d_2}{b}} \Delta \mathbf{w}_{ij} \star \mathbf{x}_j, i \in \{1, 2, \dots, \frac{d_1}{b}\}.$$

This calculation can be represented by a block-circular matrix:

$$\Delta \mathbf{z} = \mathcal{C}_{\text{blk}}(\Delta \mathbf{w})\mathbf{x}, \quad \mathcal{C}_{\text{blk}}(\Delta \mathbf{w}) = \begin{bmatrix} \mathcal{C}(\Delta \mathbf{w}_{11}) & \mathcal{C}(\Delta \mathbf{w}_{12}) & \cdots & \mathcal{C}(\Delta \mathbf{w}_{1 \frac{d_2}{b}}) \\ \mathcal{C}(\Delta \mathbf{w}_{21}) & \mathcal{C}(\Delta \mathbf{w}_{22}) & \cdots & \mathcal{C}(\Delta \mathbf{w}_{2 \frac{d_2}{b}}) \\ \cdots & \cdots & \cdots & \cdots \\ \mathcal{C}(\Delta \mathbf{w}_{\frac{d_1}{b} 1}) & \mathcal{C}(\Delta \mathbf{w}_{\frac{d_1}{b} 2}) & \cdots & \mathcal{C}(\Delta \mathbf{w}_{\frac{d_1}{b} \frac{d_2}{b}}) \end{bmatrix}. \quad (3)$$

We refer our readers to Algorithm A1 in Appendix C for a Pytorch implementation. In this context, $\Delta \mathbf{w}_{ij} \in \mathbb{R}^b$, and it follows that $\frac{d_1 d_2}{b^2} b = \frac{d_1 d_2}{b}$ represents the number of learnable parameters. Notably, the parameter b serves as a hyperparameter modulating the quantity of learnable parameters, analogous to the role of r in LoRA. It is imperative to distinguish, however, that whereas r simultaneously governs the rank of the delta matrix and the number of learnable parameters, b exclusively influences the latter. This disentanglement of matrix rank and parameter count facilitates greater adaptability and potentially yields superior outcomes.

3.5 COMPLEXITY ANALYSIS

We compare the time complexity and space complexity of LoRA, VeRA and C³A in Table 1. Detailed analysis follows in this section.

3.5.1 TIME COMPLEXITY

LoRA integrates low-rank matrices \mathbf{A} and \mathbf{B} , which are successively multiplied with the activation vector, resulting in a computational complexity of $\mathcal{O}(r(d_1 + d_2))$. Generally, $r \ll \min(d_1, d_2)$. In contrast, VeRA, despite its high-rank structure and relatively few trainable parameters, suffers from a prohibitive computational complexity of $\mathcal{O}(r_v(d_1 + d_2))$, where r_v can exceed $\max(d_1, d_2)$. Consequently, striking an optimal balance between high rank and computational efficiency remains an elusive task.

On GPUs, the cuFFT backend automatically parallelizes FFT operations along the axes not being transformed, with the degree of parallelism p determined by the available resources. Thanks to the $\mathcal{O}(n \log n)$ complexity of the FFT algorithm used in Equation 1, C³A achieves a time complexity of $\mathcal{O}(\frac{d_1+d_2}{p} \log b + \frac{d_1 d_2}{b})$. The first term is the time complexity for FFT and the second term is for aggregation. In practical scenarios, b is chosen as the greatest common divisor of d_1 and d_2 to achieve a high compression ratio. Given that, C³A is comparable to LoRA in time complexity.

3.5.2 SPACE COMPLEXITY

We analyze the space complexity of LoRA, VeRA, and C³A during training. The differences among these methods primarily arise from the trainable parameters and the auxiliary tensors required for the forward pass and backpropagation. LoRA does not rely on auxiliary tensors, while VeRA necessitates 2 random projection matrices, with a total size of $r_v(d_1 + d_2)$. Since r_v is by no means negligible, the memory usage of VeRA is significantly larger than that of LoRA.

Table 1: Time and space complexity comparison of LoRA, VeRA and C³A. We split the space complexity into Parameter number and Other auxiliary tensors to help better understand the differences. We highlight that in practice, to achieve similar performance, $\frac{\max(d_1, d_2)}{b} \leq r \ll r_v$.

Method	Time	# Param	# Other	# Total
LoRA	$\mathcal{O}(r(d_1 + d_2))$	$r(d_1 + d_2)$	0	$r(d_1 + d_2)$
VeRA	$\mathcal{O}(r_v(d_1 + d_2))$	$r_v + d_1$	$r_v(d_1 + d_2)$	$r_v(d_1 + d_2) + r_v + d_1$
C ³ A	$\mathcal{O}(\frac{d_1+d_2}{p} \log b + \frac{d_1 d_2}{b})$	$\frac{d_1 d_2}{b}$	pb	$\frac{d_1 d_2}{b} + pb$

In terms of C³A, the only additional auxiliary tensor would be of size $pb \leq \min(d_1, d_2)$, which is reserved by the FFT algorithm. By selecting an appropriate b , which is often close to the greatest common divisor of d_1 and d_2 , the space complexity of C³A is optimized. Furthermore, because p scales with the available resources, the algorithm inherently manages dynamic memory consumption without additional effort.

4 EXPERIMENT

We first experiment on a synthetic dataset to show C³A’s superior expressiveness over LoRA. Next, we evaluate C³A in both NLP and CV. For NLP, we show C³A’s effectiveness using the GLUE benchmark with RoBERTa-Base and RoBERTa-Large, and fine-tune the LLaMA family models. For CV, we test classification tasks using Vision Transformers (ViTs) on various datasets. Finally, we perform ablation studies on C³A kernel initialization.

4.1 SYNTHETIC DATA

Settings. We distribute 8 points evenly on a 2D plane as cluster centers and randomly sample 30 points from the 8 corresponding multivariate Gaussian distributions. A 3-layer MLP is then used to classify these point clusters. To compare the expressiveness of 2 types of layers, we replace the middle layer with either a low-rank layer or a circulant layer, ensuring that both layers have the same number of trainable parameters for a fair comparison.

Results. The results are presented in Figure 3. We observe that $\text{LoRA}_{r=1}$ struggles with this simple classification task. In contrast, $\text{C}^3\text{A}_{b=128/2}$, despite using the same number of parameters, achieves a perfect classification, comparable to a standard linear layer. This demonstrates the high expressiveness of C³A given the same parameter budget.

324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377

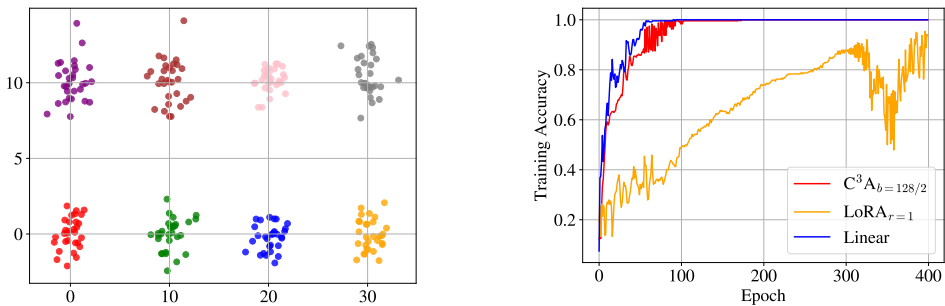


Figure 3: Expressiveness test on synthetic data. The left figure shows the synthetic data used for the experiment, while the right figure illustrates the training accuracy curves of a 3-layer MLP, incorporating C^3A , LoRA, and standard linear layers, respectively.

4.2 NATURAL LANGUAGE UNDERSTANDING

Baselines. We compare our C^3A with several representative PEFT methods, including BitFit (Zaken et al., 2021), $(IA)^3$ (Liu et al., 2022), LoRA (Hu et al., 2021), VeRA (Kopiczko et al., 2023), and BOFT (Liu et al., 2023). BitFit selectively fine-tunes existing parameters, specifically the biases. $(IA)^3$ is the state-of-the-art method that adds additional adapters. LoRA is a widely known PEFT method that employs low-rank decomposition to compress additive delta matrices. VeRA is a recent approach that focuses on further reducing trainable parameters of LoRA while preserving a high rank. BOFT is another innovative method in PEFT research, compressing multiplicative delta matrices using orthogonal decomposition and butterfly factorization.

Table 2: Performance of different PEFT methods on the GLUE benchmark. We fine-tune pre-trained RoBERTa-Base and -Large models on 6 datasets. 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. For each metric, a higher score indicates better performance. “Avg.” denotes the average score of each method across all datasets. The best results for each dataset are highlighted in **bold**. # Trainable parameters does not include the classification head since each method uses a head of the same size. Memory Cost is measured on fixed length (*i.e.*, 256) data with a batchsize of 64.

	Method	# Trainable Parameters	Memory Cost (GB)	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
BASE	Full	124M	17.19	94.01±0.39	87.10 ±0.79	62.00±1.16	92.40±0.28	77.33 ±2.68	90.70 ±0.14	83.92
	BitFit	0.102M	12.60	93.30±0.30	85.80±0.21	59.21±1.74	91.96±0.18	73.07±1.34	90.18±0.17	82.25
	$(IA)^3$	0.111M	19.86	92.98±0.34	85.86±0.59	60.49±1.09	91.56±0.17	69.10±1.18	90.06±0.21	81.67
	LoRA _{r=8}	0.295M	13.75	94.50 ±0.41	85.68±0.74	60.95±1.57	92.54 ±0.20	76.68±1.42	89.76±0.39	83.35
	VeRA _{r=1024}	0.043M	15.51	93.97±0.17	86.23±0.41	62.24±1.91	91.85±0.17	75.74±1.56	90.27±0.25	83.38
	BOFT _{m=2, b=8}	0.166M	14.11	93.23±0.50	84.37±0.54	59.50±1.25	91.69±0.12	74.22±0.84	89.63±0.37	82.11
	$C^3A_{b=768/1}$	0.018M	12.83	93.42±0.26	86.33±0.32	61.83±0.96	91.83±0.04	76.17±1.39	90.46±0.29	83.34
	$C^3A_{b=768/6}$	0.111M	12.72	94.20±0.16	86.67±0.54	62.48 ±1.20	92.32±0.25	77.18±1.41	90.16±0.42	83.84
LARGE	Full	354M	43.40	95.75±0.45	88.35 ±0.64	64.87±1.25	92.40±0.28	84.48±1.14	91.65±0.14	86.25
	BitFit	0.271M	30.65	95.09±0.27	88.10±0.76	65.40±0.76	94.06±0.14	82.60±1.15	91.73±0.20	86.16
	$(IA)^3$	0.295M	48.81	95.32±0.20	87.06±0.57	66.52±1.10	94.18±0.15	84.33±2.38	91.58±0.39	86.50
	LoRA _{r=8}	0.786M	34.12	95.53±0.35	86.12±0.86	65.16±0.76	93.73±0.30	83.75±0.51	91.46±0.21	85.96
	VeRA _{r=256}	0.061M	34.16	95.83 ±0.43	87.72±0.55	63.66±1.45	94.11±0.20	83.03±1.65	91.12±0.37	85.91
	BOFT _{m=2, b=8}	0.442M	34.98	95.76±0.41	88.28±0.33	64.72±2.37	93.89±0.14	82.82±1.40	91.03±0.32	86.08
	$C^3A_{b=1024/1}$	0.049M	31.83	95.78±0.05	88.02±0.62	66.59±1.20	94.22±0.25	82.89±0.67	91.86 ±0.14	86.56
	$C^3A_{b=1024/8}$	0.393M	31.79	95.78±0.15	88.09±0.47	67.18 ±1.92	94.26 ±0.19	84.62 ±1.36	91.81±0.36	86.96

Settings. We evaluate our proposed C^3A on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018), which encompasses a wide range of natural language understanding (NLU) tasks, including single-sentence classification, similarity and paraphrase, and natural language inference. More dataset specifications can be found in Table A1 in Appendix A. To enhance practicality, we split these datasets following the train-validation-test approach. The best-performing model is selected based on validation set performance across the fine-tuning epochs, and the reported performance corresponds to its performance on the test set. For this evaluation, we fine-tune the pre-trained RoBERTa-Base and RoBERTa-Large models (Liu et al., 2019). For the unique

hyperparameters of different baselines, we adopt the values suggested in the original papers (*e.g.*, VeRA’s r and BOFT’s b and m). The number of trainable parameters excludes the classification head, as each method uses one of the same size. The shared hyperparameters (*i.e.*, the learning rate for classification head and for other trainable parameters, separately) are found by hyperparameter search. For the memory cost, to ensure fairness and consistency, we fix the length of input data to 256 tokens and use a batchsize of 64.

Results. The results are presented in Table 2. Overall, $C^3A_{b=768/1}$ and $C^3A_{b=1024/1}$ achieve superior or comparable performance to baseline methods, despite using an exceptionally small number of trainable parameters. As the number of trainable parameters increases, models like $C^3A_{b=768/6}$ and $C^3A_{b=1024/8}$ significantly outperform the baselines. Moreover, compared to (IA)³, LoRA, VeRA, and BOFT, C^3A distinguishes itself with remarkable memory efficiency. The only method demonstrating better memory efficiency is BitFit, which serves as an upper bound since it introduces no new parameters. Additionally, most of the circulant delta matrices identified by C^3A are of full rank, indicating maximal capacity (Zeng & Lee, 2023) and providing a theoretical basis for the outstanding performance.

4.3 INSTRUCTION TUNING

Settings. For a comprehensive comparison, we further conduct instruction tuning on LLaMA families, the prevalent large language models. Specifically, we evaluate C^3A against LoRA and DoRA (Liu et al., 2024), a variant of LoRA sensitive to learning direction. Specifically, we fine-tune LLaMA2-7B/13B (Touvron et al., 2023) and LLaMA3-8B (Dubey et al., 2024) among 7 datasets covering 3 prevalent tasks: ① Arithmetic reasoning on GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2020); ② Functional representation generation on ViGGO (Juraska et al., 2019), and SQL (Zhong et al., 2017); and ③ Commonsense reasoning on BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020) and SIQA (Bisk et al., 2020). For the SQL dataset, we preprocess it by selecting 25% of the data and apply a 4:1 train-test split, resulting in a training set of 16K samples. To ensure a fair comparison, we maintain LoRA parameters with $r = 32$, $\alpha = 32$, and a dropout rate of 0.05, while exploring various learning rates as suggested by (Hu et al., 2021). Please refer to Table A4 in Appendix B for more details.

Results. In Table 3, our principal experimental observations are summarized. The C^3A framework consistently surpasses LoRA within the LLaMA series, with particular efficacy demonstrated in the most recent model, LLaMA3-8B. Noteworthy is the significant enhancement in the efficacy of LLaMA3-8B as a foundational model following the implementation of more sophisticated post-training techniques. This underscores the criticality of optimizing the fine-tuning protocols for this advanced model. It is also remarkable that C^3A achieves such results while employing less than half the parameter count of LoRA. Taken together, the findings robustly underscore the superior efficacy of the C^3A methodology. We refer readers to Appendix D for examples of models after different tuning methods.

Table 3: Comparison of C^3A and LoRA on fine-tuning LLaMA2 and LLaMA3 models in terms of accuracy and trainable parameters. The best results for each dataset are highlighted in **bold**. “Avg.” denotes the average accuracy of each method across all datasets.

Model	Method	# Trainable Parameters	GSM8k	MATH	ViGGO	SQL	BoolQ	PIQA	SIQA	Avg.
LLaMA2-7B	LoRA $_{r=32}$	16.8M	39.57	5.65	96.48	79.66	75.60	85.26	82.09	66.33
	DoRA $_{r=32}$	17.0M	39.05	6.00	96.85	79.66	75.41	85.64	81.93	66.36
	$C^3A_{b=4096/32}$	8.4M	40.18	6.00	97.05	79.28	75.02	85.53	81.62	66.38
LLaMA2-13B	LoRA $_{r=32}$	26.2M	49.02	8.55	97.10	79.97	77.09	87.36	83.21	68.90
	DoRA $_{r=32}$	26.6M	50.02	9.00	97.32	79.66	77.16	87.70	82.60	69.07
	$C^3A_{b=5120/32}$	13.1M	49.66	8.85	97.34	80.12	76.91	87.98	83.05	69.13
LLaMA3-8B	LoRA $_{r=32}$	13.6M	62.80	21.05	96.50	80.61	77.37	89.72	82.19	72.89
	DoRA $_{r=32}$	13.8M	62.95	22.15	96.54	81.22	77.09	90.21	82.44	73.24
	$C^3A_{b=4096/32}$	5.2M	64.22	21.60	96.58	80.73	77.04	90.33	82.60	73.30

4.4 IMAGE CLASSIFICATION

Settings. In this study, we concentrate on the task of image classification leveraging Vision Transformer (ViT) models. Specifically, we employ both the Base and Large variants of this prominent foundational computer vision model, as delineated by (Dosovitskiy et al., 2020). These ViT models undergo pre-training on the expansive ImageNet-21K dataset (Ridnik et al., 2021). During the fine-tuning phase, we use an eclectic array of datasets encompassing Pets (Parkhi et al., 2012), Cars (Krause et al., 2013), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), FGVC (Maji et al., 2013), and RESISC (Cheng et al., 2017). Comprehensive statistics for these datasets are provided in Table A2 in Appendix A.

Table 4: Fine-tuning results with ViT-Base and ViT-Large models on various image classification datasets. The models are fine-tuned for 10 epochs, and the best-performing model, based on validation set accuracy, is selected. The reported accuracy corresponds to the performance on the test set. The best results between LoRA and C³A for each dataset are highlighted in **bold**. “Avg.” denotes the average accuracy of each method across all datasets.

	Method	# Trainable Parameters	Pets	Cars	DTD	EuroSAT	FGVC	RESISC	Avg.
BASE	Head	-	90.28 \pm 0.43	25.76 \pm 0.28	69.77 \pm 0.67	88.72 \pm 0.13	17.44 \pm 0.43	74.22 \pm 0.10	61.03
	Full	85.8M	92.82 \pm 0.54	85.10 \pm 0.21	80.11 \pm 0.56	99.11 \pm 0.07	61.60 \pm 1.00	96.00 \pm 0.23	85.79
	LoRA _{r=16}	0.59M	93.76 \pm 0.44	78.04 \pm 0.33	78.56 \pm 0.62	98.84 \pm 0.08	56.64 \pm 0.55	94.66 \pm 0.17	83.42
	C ³ A _{b=768/12}	0.22M	93.88 \pm 0.22	79.05 \pm 0.35	80.57 \pm 0.53	98.88 \pm 0.07	54.31 \pm 0.79	94.54 \pm 0.23	83.54
LARGE	Head	-	91.11 \pm 0.30	37.91 \pm 0.27	73.33 \pm 0.26	92.64 \pm 0.08	24.62 \pm 0.24	82.02 \pm 0.11	66.94
	Full	303M	94.30 \pm 0.31	88.15 \pm 0.50	80.18 \pm 0.66	99.06 \pm 0.10	67.38 \pm 1.06	96.08 \pm 0.20	87.53
	LoRA _{r=16}	1.57M	94.62 \pm 0.47	86.11 \pm 0.42	80.09 \pm 0.42	98.99 \pm 0.03	63.64 \pm 0.83	95.52 \pm 0.21	86.56
	C ³ A _{b=1024/16}	0.79M	94.48 \pm 0.30	84.94 \pm 0.39	82.62 \pm 0.52	98.75 \pm 0.19	63.80 \pm 0.37	95.94 \pm 0.16	86.69

Results. Table 4 delineates a comprehensive summary of the outcomes derived from six distinct image classification datasets employing the ViT Base and Large models. The LoRA and C³A techniques exhibit significant enhancements in performance relative to Head Tuning, thereby underscoring their efficacy within the realm of image classification. Remarkably, our methodology demonstrates a performance on par with LoRA while necessitating only half of the parameter count.

4.5 INITIALIZATION STUDY

LoRA is known to be sensitive to initialization, primarily due to the distinct roles of its matrices **A** and **B** (Hayou et al., 2024a). In contrast, C³A possesses a simpler structure based on circulant matrices, which may reduce sensitivity to initialization. To investigate this, we focused on the initialization strategies for the convolution kernels that define the circulant matrices in C³A. We conducted experiments comparing four initialization methods: zero initialization, Gaussian initialization, Kaiming uniform, and Xavier uniform. We observe that the variations across different initialization points are mostly within the intrinsic standard deviations, highlighting the robustness of C³A to initialization strategies. Our findings indicate that C³A maintains robust performance across these different initialization strategies, highlighting its resilience to initialization points.

Table 5: Performance of C³A with Different Initialization Strategies. The tasks on CoLA and STS-B were performed using the RoBERTa-Base model, while those on Cars and DTD utilized the ViT-Base model. All other settings are consistent with those in Table 2 and Table 4.

Task	Zero	Gaussian	Kaiming	Xavier	Range
CoLA	60.95 \pm 0.88	61.07 \pm 1.09	60.82 \pm 1.48	62.48 \pm 0.74	1.66
STS-B	90.23 \pm 0.23	90.13 \pm 0.16	90.19 \pm 0.34	90.31 \pm 0.31	0.18
Cars	78.70 \pm 0.60	78.64 \pm 0.67	79.18 \pm 0.37	78.96 \pm 0.25	0.54
DTD	80.82 \pm 0.86	79.58 \pm 0.41	79.76 \pm 1.14	79.95 \pm 0.72	1.24

5 CONCLUSION

In this manuscript, we present C³A, a novel method for Parameter-Efficient Fine-Tuning (PEFT). In contrast to LoRA, which employs low-rank decomposition, C³A leverages circular convolution and its equavelent circulant matrix to represent the delta weight matrix. This methodology aims

486 to independently control the rank of the delta weight matrix and the number of trainable param-
487 eters, facilitating high-rank adaptation while preserving a constrained parameter size. Using the Fast
488 Fourier Transform (FFT) during both the forward and backward propagation phases, C³A attains
489 notable computational and memory efficiency. In short, C³A emerges as a persuasive alternative to
490 LoRA for model fine-tuning.

491 REFERENCES

- 492 Bassam Bamieh. Discovering transforms: A tutorial on circulant matrices, circular convolution, and
493 the discrete fourier transform. *arXiv preprint arXiv:1805.05533*, 2018.
- 494 Samyadeep Basu, Shell Hu, Daniela Massiceti, and Soheil Feizi. Strong baselines for parameter-
495 efficient few-shot fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*,
496 volume 38, pp. 11024–11031, 2024.
- 497 Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical com-
498 monsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*,
499 volume 34, pp. 7432–7439, 2020.
- 500 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,
501 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are
502 few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- 503 Aochuan Chen, Yuguang Yao, Pin-Yu Chen, Yihua Zhang, and Sijia Liu. Understanding and improv-
504 ing visual prompting: A label-mapping perspective. In *Proceedings of the IEEE/CVF Conference*
505 *on Computer Vision and Pattern Recognition*, pp. 19133–19143, 2023.
- 506 Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Bench-
507 mark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.
- 508 Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. An
509 exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings*
510 *of the IEEE international conference on computer vision*, pp. 2857–2865, 2015.
- 511 Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. De-
512 scribing textures in the wild. In *Proceedings of the IEEE conference on computer vision and*
513 *pattern recognition*, pp. 3606–3613, 2014.
- 514 Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina
515 Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint*
516 *arXiv:1905.10044*, 2019.
- 517 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
518 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to
519 solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- 520 Caiwen Ding, Siyu Liao, Yanzhi Wang, Zhe Li, Ning Liu, Youwei Zhuo, Chao Wang, Xuehai
521 Qian, Yu Bai, Geng Yuan, Xiaolong Ma, Yipeng Zhang, Jian Tang, Qinru Qiu, Xue Lin, and
522 Bo Yuan. Circnn: accelerating and compressing deep neural networks using block-circulant
523 weight matrices. In *Proceedings of the 50th Annual IEEE/ACM International Symposium*
524 *on Microarchitecture*, MICRO-50 '17, pp. 395–408, New York, NY, USA, 2017. Association
525 for Computing Machinery. ISBN 9781450349529. doi: 10.1145/3123939.3124552. URL
526 <https://doi.org/10.1145/3123939.3124552>.
- 527 Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas
528 Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An
529 image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint*
530 *arXiv:2010.11929*, 2020.
- 531 Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha
532 Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models.
533 *arXiv preprint arXiv:2407.21783*, 2024.

- 540 Morris Dworkin, Elaine Barker, James Nechvatal, James Foti, Lawrence Bassham, E. Roback, and
541 James Dray. Advanced encryption standard (aes), 2001-11-26 2001.
542
- 543 Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. On
544 the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI conference on*
545 *artificial intelligence*, volume 37, pp. 12799–12807, 2023.
- 546 Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot
547 learners. *arXiv preprint arXiv:2012.15723*, 2020.
548
- 549 Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li.
550 Parameter-efficient fine-tuning with discrete fourier transform. *arXiv preprint arXiv:2405.03003*,
551 2024.
- 552 Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University
553 Press, USA, 1996. ISBN 0801854148.
554
- 555 Yanwei Gong, Xiaolin Chang, Jelena Mišić, Vojislav B Mišić, Jianhua Wang, and Haoran Zhu.
556 Practical solutions in fully homomorphic encryption: a survey analyzing existing acceleration
557 methods. *Cybersecurity*, 7(1):5, 2024.
- 558 Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff prun-
559 ing. *arXiv preprint arXiv:2012.07463*, 2020.
560
- 561 Soufiane Hayou, Nikhil Ghosh, and Bin Yu. The impact of initialization on lora finetuning dynamics.
562 *arXiv preprint arXiv:2406.08447*, 2024a.
- 563 Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models.
564 *arXiv preprint arXiv:2402.12354*, 2024b.
565
- 566 Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a
567 unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*, 2021.
568
- 569 Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset
570 and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected*
571 *Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.
- 572 Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and
573 Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint*
574 *arXiv:2009.03300*, 2020.
- 575 Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,
576 and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint*
577 *arXiv:2106.09685*, 2021.
578
- 579 A. W. Ingleton. The rank of circulant matrices. *Journal of the London Mathe-*
580 *matical Society*, s1-31(4):445–460, 1956. doi: <https://doi.org/10.1112/jlms/s1-31.4.445>.
581 URL [https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/](https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/jlms/s1-31.4.445)
582 [jlms/s1-31.4.445](https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/jlms/s1-31.4.445).
- 583 Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv*
584 *preprint arXiv:1611.01144*, 2016.
585
- 586 Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and
587 Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pp. 709–727.
588 Springer, 2022.
- 589 Juraj Juraska, Kevin K Bowden, and Marilyn Walker. Viggo: A video game corpus for data-to-text
590 generation in open-domain conversation. *arXiv preprint arXiv:1910.12129*, 2019.
591
- 592 Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete
593 Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceed-*
ings of the IEEE/CVF International Conference on Computer Vision, pp. 4015–4026, 2023.

- 594 Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. Vera: Vector-based random
595 matrix adaptation. *arXiv preprint arXiv:2310.11454*, 2023.
- 596
- 597 Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained
598 categorization. In *Proceedings of the IEEE international conference on computer vision work-*
599 *shops*, pp. 554–561, 2013.
- 600
- 601 Changli Li, Hon Keung Kwan, and Xinxin Qin. Revisiting linear convolution, circular convolution
602 and their related methods. *2020 13th International Congress on Image and Signal Processing,*
603 *BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1124–1131, 2020. URL [https://](https://api.semanticscholar.org/CorpusID:227220098)
604 api.semanticscholar.org/CorpusID:227220098.
- 605 Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv*
606 *preprint arXiv:2101.00190*, 2021.
- 607
- 608 Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. Zerog: Investigating cross-dataset
609 zero-shot transferability in graphs. *arXiv preprint arXiv:2402.11235*, 2024.
- 610
- 611 Dongze Lian, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Scaling & shifting your features: A
612 new baseline for efficient model tuning. *Advances in Neural Information Processing Systems*, 35:
613 109–123, 2022.
- 614
- 615 Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and
616 Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context
617 learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022.
- 618
- 619 Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-
620 Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv*
621 *preprint arXiv:2402.09353*, 2024.
- 622
- 623 Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen
624 Liu, Juyeon Heo, Songyou Peng, et al. Parameter-efficient orthogonal finetuning via butterfly
625 factorization. *arXiv preprint arXiv:2311.06243*, 2023.
- 626
- 627 Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike
628 Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining
629 approach. *arXiv preprint arXiv:1907.11692*, 2019.
- 630
- 631 Subhansu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained
632 visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.
- 633
- 634 Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev
635 Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information*
636 *Processing Systems*, 36:53038–53075, 2023.
- 637
- 638 Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin
639 Bossan. Pefit: State-of-the-art parameter-efficient fine-tuning methods. [https://github.](https://github.com/huggingface/peft)
640 [com/huggingface/peft](https://github.com/huggingface/peft), 2022.
- 641
- 642 Clare D. McGillem and George R. Cooper. Continuous and discrete signal and system analysis.
643 1984. URL <https://api.semanticscholar.org/CorpusID:117907785>.
- 644
- 645 Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012*
646 *IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.
- 647
- 648 Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller,
649 and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. *Advances*
650 *in Neural Information Processing Systems*, 36:79320–79362, 2023.
- 651
- 652 L. R. Rabiner, B. Gold, and C. K. Yuen. Theory and application of digital signal processing. *IEEE*
653 *Transactions on Systems, Man, and Cybernetics*, 8(2):146–146, 1978. doi: 10.1109/TSMC.1978.
654 4309918.

- 648 Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal,
649 Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual
650 models from natural language supervision. In *International conference on machine learning*, pp.
651 8748–8763. PMLR, 2021.
- 652 Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse?
653 towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.
- 654 Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with
655 residual adapters. *Advances in neural information processing systems*, 30, 2017.
- 656 Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for
657 the masses. *arXiv preprint arXiv:2104.10972*, 2021.
- 658 Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and
659 Iryna Gurevych. Adapterdrop: On the efficiency of adapters in transformers. *arXiv preprint
arXiv:2010.11918*, 2020.
- 660 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée
661 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and
662 efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- 663 Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman.
664 Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv
665 preprint arXiv:1804.07461*, 2018.
- 666 Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei
667 Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning.
668 *arXiv preprint arXiv:2109.05687*, 2021.
- 669 Felix Yu, Sanjiv Kumar, Yunchao Gong, and Shih-Fu Chang. Circulant binary embedding. In
670 *International conference on machine learning*, pp. 946–954. PMLR, 2014.
- 671 Shen Yuan, Haotian Liu, and Hongteng Xu. Bridging the gap between low-rank and orthogonal
672 adaptation via householder reflection adaptation. *arXiv preprint arXiv:2405.17484*, 2024.
- 673 Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning
674 for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.
- 675 Yuchen Zeng and Kangwook Lee. The expressive power of low-rank adaptation. *arXiv preprint
676 arXiv:2310.17513*, 2023.
- 677 Yihua Zhang, Hongkang Li, Yuguang Yao, Aochuan Chen, Shuai Zhang, Pin-Yu Chen, Meng Wang,
678 and Sijia Liu. Visual prompting reimagined: The power of activation prompts, 2024a. URL
679 <https://openreview.net/forum?id=0b328CMwn1>.
- 680 Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Yinyu Ye, Zhi-Quan Luo, and
681 Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*,
682 2024b.
- 683 Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from
684 natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.
- 685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701

APPENDIX

A DATASET DETAILS

Table A1: Task descriptions and dataset statistics of the GLUE benchmark (Wang et al., 2018).

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

Table A2: Details about the vision datasets.

Dataset	#Train	#Validation	#Test	#Class	Rescaled resolution
Pets (Parkhi et al., 2012)	3,312	368	3,669	37	224×224
Cars (Krause et al., 2013)	7,329	815	8,041	196	
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	
RESISC (Cheng et al., 2017)	18,900	6,300	6,300	45	

B HYPERPARAMETERS

Table A3: Hyperparameter setup of C³A for the GLUE benchmark.

Model	Hyperparameter	SST-2	MRPC	CoLA	QNLI	RTE	STS-B
Both	Optimizer	AdamW					
	LR Schedule	Linear					
	Warmup Ratio	0.06					
	C ³ A Initialization	Xavier Uniform					
	Max Seq. Len	512					
Base	Epochs	40	80	80	40	80	80
	Batch Size	128	128	128	64	64	128
	Learning Rate (C ³ A _{b=768/6})	2E-1	3E-1	2E-1	7E-2	3E-1	2E-1
	Learning Rate (Head)	2E-4	4E-6	3E-2	8E-6	6E-3	4E-2
Large	Epochs	10	80	70	30	60	40
	Batch Size	128	128	128	32	64	128
	Learning Rate (C ³ A _{b=1024/8})	9E-2	3E-1	2E-1	7E-2	5E-2	2E-1
	Learning Rate (Head)	2E-4	5E-6	3E-3	8E-6	3E-3	5E-4

Table A4: Hyperparameter setup of LoRA and C³A for instruction tuning.

Model	Hyperparameter	GSM8k	MATH	ViGGO	SQL	BoolQ	PIQA	SIQA
	Optimizer				AdamW			
	LR Scheduler				Cosine			
	Batch Size				16			
	Warmup Ratio				0.05			
	Dropout				0.05			
	Epoch				3			
LLaMA2-7B	Learning Rate (LoRA)	5E-4	5E-4	5E-4	6E-4	5E-4	4E-4	6E-4
	Learning Rate (DoRA)	4E-4	5E-4	4E-4	6E-4	4E-4	5E-4	5E-4
	Learning Rate (C ³ A)	8E-1	5E-1	5E-1	9E-1	7E-1	4E-1	3E-1
LLaMA2-13B	Learning Rate (LoRA)	5E-4	6E-4	5E-4	6E-4	5E-4	5E-4	4E-4
	Learning Rate (DoRA)	4E-4	6E-4	5E-4	6E-4	4E-4	4E-1	5E-1
	Learning Rate (C ³ A)	6E-1	4E-1	8E-1	1	4E-1	4E-1	8E-1
LLaMA2-8B	Learning Rate (LoRA)	5E-4	5E-4	4E-4	5E-4	4E-4	4E-4	5E-4
	Learning Rate (DoRA)	6E-4	2E-4	5E-4	5E-4	4E-4	4E-4	4E-4
	Learning Rate (C ³ A)	5E-1	3E-1	6E-1	4E-1	3E-1	3E-1	4E-1

Table A5: Hyperparameter setup of C³A for image classification tasks.

Model	Hyperparameter	Pets	Cars	DTD	EuroSAT	FGVC	RESISC
Both	Optimizer				AdamW		
	LR Schedule				None		
	C ³ A Initialization				Xavier Uniform		
	Epochs				10		
	Batch Size				64		
Base	Learning Rate (C ³ A _{b=768/12})	4E-1	4E+0	2E+0	2E+0	7E+0	2E+0
	Learning Rate (Head)	1E-2	1E-2	2E-2	8E-3	1E-2	2E-2
	Weight Decay	3E-4	5E-4	6E-5	2E-5	1E-5	2E-5
Large	Learning Rate (C ³ A _{b=1024/16})	7E-1	4E+0	2E+0	2E+0	4E+0	3E+0
	Learning Rate (Head)	3E-3	8E-3	7E-3	2E-2	1E-1	4E-3
	Weight Decay	4E-3	1E-5	2E-4	5E-4	2E-5	9E-5

C IMPLEMENTATIONS

Algorithm A1 Block-Circular Convolution PyTorch Implementation

```

810
811
812
813
814
815 import torch
816 from torch.autograd import Function
817 from torch.fft import fft, ifft
818
819 class BlockCircularConvolution(Function):
820     @staticmethod
821     def forward(ctx, x, w):
822         m, n, b = w.shape
823         x = x.reshape(*x.shape[:-1], n, b)
824         ctx.save_for_backward(x, w)
825         x = torch.einsum( "...nb,mnb->...mb", ifft(x), fft(w) )
826         x = fft(x).real
827         x = x.reshape(*x.shape[:-2], -1)
828         return x
829
830     @staticmethod
831     def backward(ctx, grad_output):
832         x, w = ctx.saved_tensors
833         m, n, b = w.shape
834         grad_output = grad_output.reshape(*grad_output.shape[:-1], m, b)
835         grad_output_fft = fft(grad_output)
836         x_grad = fft(torch.einsum( "...mb,mnb->...nb", grad_output_fft, ifft(w) )).real
837         x_grad = x_grad.reshape(*x_grad.shape[:-2], -1)
838         w_grad = fft(torch.einsum( "...mb,...nb->mnb", grad_output_fft, ifft(x) )).real
839         return x_grad, w_grad
840
841
842
843
844
845
846
847

```

We present the PyTorch implementation of Block-Circular Convolution in Algorithm A1. Furthermore, due to the inefficiency of directly assigning entries (as shown in Equation 3), we derive an alternative algorithm to compute the $\Delta\mathbf{W}$ more efficiently. Rather than direct assignment, we employ a forward process on the Identity matrix. Mathematically, this can be expressed as

$$\begin{aligned}
 \Delta\mathbf{W} &= \mathcal{C}_{\text{blk}}(\Delta\mathbf{w}) \\
 &= \mathcal{C}_{\text{blk}}(\Delta\mathbf{w}) \cdot \mathbf{I}_{d_2} \\
 &= \mathcal{C}_{\text{blk}}(\Delta\mathbf{w}) \cdot [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{d_2}] \\
 &= [\mathcal{C}_{\text{blk}}(\Delta\mathbf{w})\mathbf{e}_1, \mathcal{C}_{\text{blk}}(\Delta\mathbf{w})\mathbf{e}_2, \dots, \mathcal{C}_{\text{blk}}(\Delta\mathbf{w})\mathbf{e}_{d_2}] \\
 &= [\Delta\mathbf{w} \star \mathbf{e}_1, \Delta\mathbf{w} \star \mathbf{e}_2, \dots, \Delta\mathbf{w} \star \mathbf{e}_{d_2}].
 \end{aligned}$$

Where $\mathbf{I}_{d_2} \in \mathbb{R}^{d_2 \times d_2}$ represents an Identity matrix and \mathbf{e}_i is the i th column of it. In pytorch, we can efficiently compute the iFFT of $\{\mathbf{e}_i\}_{i=1,2,\dots,d_2}$ by a column-wise iFFT of \mathbf{I}_{d_2} . We present the Pytorch implementation in Algorithm A2 as well.

Algorithm A2 Fast Algorithm of Getting $\Delta\mathbf{W}$

```

848 import torch
849 from torch.fft import fft, ifft
850
851 def get_circulant_fast(w):
852     m, n, b = w.shape
853     x = torch.eye(n*b, dtype=w.dtype, device=w.device)
854     x = x.reshape(*x.shape[:-1], n, b)
855     x = torch.einsum( "...nb,mnb->...mb", ifft(x), fft(w) )
856     x = fft(x).real.flatten(start_dim=1).T
857     return x
858
859
860
861
862
863

```

D INSTRUCTION TUNING EXAMPLES

864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917

LLaMA3-8B (MATH)

Find the number of square units bounded by the x -axis, the y -axis and the line $x + y = 6$.
Answer the above question. First think step by step and then answer the final number.

Response**Methods****LoRA****Response**

The line $x + y = 6$ intersects the x -axis at $x = 6$ and the y -axis at $y = 6$.
So, the line divides the square with side length 6 into two triangles.
The area of each triangle is $\frac{1}{2} \times 6 \times 6 = 18$.
So, the area of the square is $2 \times 18 = \boxed{36}$.
The final answer is: 36

C³A

The line $x + y = 6$ intersects the x -axis at $(6, 0)$ and the y -axis at $(0, 6)$.
Thus, the line divides the first quadrant into two triangles, one with base 6 and height 6
and the other with base 6 and height 0.
The area of the first triangle is $(6 \times 6)/2 = 18$ square units, and the area of the second
triangle is $(6 \times 0)/2 = 0$ square units.
Thus, the total area is $18 + 0 = \boxed{18}$ square units.
The final answer is: 18

LLaMA3-8B (GSM8K)

Tim makes a special honey and jam mix. To make 1 jar of the mixture it takes 2 pounds of passion fruit
and 1.5 pounds of jackfruit. It also takes 2 pounds of special honey. The passion fruit costs \$6 per pound,
the jackfruit is \$8 per pound, and the honey is \$10 per pound. He sells each jar for \$50. How much profit
does he make selling 10 jars?

Answer the above question. First think step by step and then answer the final number.

Response**Methods****LoRA****Response**

The passion fruit costs $6 \times 2 = 12$.
The jackfruit costs $8 \times 1.5 = 12$.
The honey costs $10 \times 2 = 20$.
So the total cost is $12 + 12 + 20 = 44$.
So he makes a profit of $50 \times 10 - 44 = 460$.
The final answer is: 460

C³A

The passion fruit costs $6 \times 2 = 12$.
The jackfruit costs $8 \times 1.5 = 12$.
So the fruit costs $12 + 12 = 24$.
The honey costs $10 \times 2 = 20$.
So the total cost per jar is $24 + 20 = 44$.
So he makes a profit of $50 - 44 = 6$ per jar.
So he makes a profit of $6 \times 10 = 60$.
The final answer is: 60