Asymmetry in Low-Rank Adapters of Foundation Models

Jiacheng Zhu [1] **Kristjan Greenewald** [2 3] **Kimia Nadjahi** [1] **Haitz Sáez de Ocáriz Borde** [4]
Rickard Brüel Gabrielsson [1] **Leshem Choshen** [1 3] **Marzyeh Ghassemi** [1] **Mikhail Yurochkin** [2 3] **Justin Solomon** [1]

Abstract

Parameter-efficient fine-tuning optimizes large, pre-trained foundation models by updating a subset of parameters; in this class, Low-Rank Adaptation (LoRA) is particularly effective. Inspired by an effort to investigate the different roles of LoRA matrices during fine-tuning, this paper characterizes and leverages unexpected asymmetry in the importance of low-rank adapter matrices. Specifically, when updating the parameter matrices of a neural network by adding a product BA, we observe that the B and A matrices have distinct functions: A extracts features from the input, while B uses these features to create the desired output. Based on this observation, we demonstrate that fine-tuning B is inherently more effective than fine-tuning A, and that a random untrained A should perform nearly as well as a fine-tuned one. Using an information-theoretic lens, we also bound the generalization of low-rank adapters, showing that the parameter savings of exclusively training B improves the bound. We support our conclusions with experiments on RoBERTa, BART-Large, LLaMA-2, and ViTs. The code and data is available at https://github.com/Jiacheng-Zhu-AIML/AsymmetryLoRA

1. Introduction

Foundation models for data-rich modalities such as text and imagery have achieved significant success by pre-training large models on vast amounts of data. While these models are designed to be general-purpose, it is often necessary to *fine-tune* them for downstream tasks. However, the huge size of foundation models can make fine-tuning the entire model impossible, inspiring parameter-efficient fine-tuning (PEFT) methods that selectively update fewer parameters (c.f. Lialin et al., 2023). The effectiveness of PEFT demonstrates that updating even a tiny fraction of the parameters can retain and enrich the capabilities of pretrained models. Indeed, fine-tuning has become a necessary ingredient of modern ML; for example, the PEFT package (HuggingFace, Year) has supported more than 4.4k projects since its creation in November 2022.

Among PEFT methods, low-rank adaptation (LoRA) (Hu et al., 2021) has become increasingly popular, which leverages the assumption that over-parameterized models have a low intrinsic dimension (Aghajanyan et al., 2021). To update a neural network, LoRA trains a subset of the parameters (usually attention) by representing weight matrices as $W_0 + \Delta W$, where W_0 is the fixed weight matrix from the pre-trained model and ΔW is a low-rank update. Compared to full fine-tuning, LoRA considerably reduces the number of trainable parameters and memory requirements and often achieves similar or better performance.

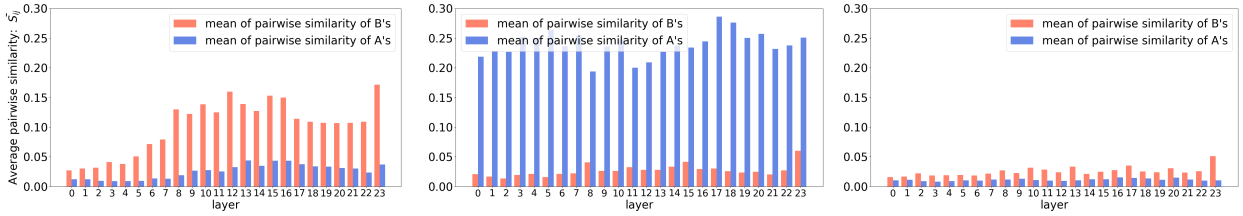
Most LoRA implementations factor $\Delta W = BA$ and optimize for A and B, where A and B have fewer rows and columns (resp.) than ΔW; this approach was proposed by Hu et al. (2021). With this set of variables, the standard LoRA training procedure—where A is initialized to a random matrix and B is initialized to zero—exhibits an interesting asymmetry, which is leveraged in some empirical follow-ups (Zhang et al., 2023a; Kopiczko et al., 2024). In particular, while training B is critical for the performance of LoRA, even a *randomly* initialized A seems to suffice for strong performance. On the other hand, reversing the roles of A and B substantially decreases performance.

Delving into this empirical suggestion from prior work, this paper demonstrates that LoRA's components are inherently asymmetric. In fact, the asymmetry occurs even for linear models (§4.1.1). Indeed, our theoretical (§4) and empirical analysis (§5) suggests that fixing A to a random orthogonal matrix can yield similar performance to full LoRA training, and that this adjustment may even promote generalization. This observation is backed by a comprehensive empirical study, leading to practical suggestions for improving parameter efficiency and generalization of LoRA models. Our contributions are as follows:

- We provide simple theoretical and empirical analysis

[1]MIT CSAIL [2]IBM Research [3]MIT-IBM Watson AI Lab [4]University of Oxford. Correspondence to: Jiacheng Zhu <zjc@mit.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.



(a) Random initialization, same task (b) Fixed initialization, different tasks (c) Random initialization, different tasks

Figure 1. Similarity of learned LoRA matrices A & B across layers of a RoBERTa model fine-tuned with different initialization and data settings. Bs are similar when fine-tuning on the same task (a) and dissimilar when fine-tuning on different tasks (b and c). As are similar when initialized identically (b), even though fine-tuning is done on different tasks, and dissimilar when initialized randomly regardless of the fine-tuning task (a and c). The experiment demonstrates the asymmetric roles of A and B in LoRA.

demonstrating *asymmetry* of training the two adapter matrices, showing that tuning B is more impactful than tuning A. This confirms and builds upon prior empirical observations (Zhang et al., 2023a; Kopiczko et al., 2024).

- We show theoretically and empirically that randomly drawing and freezing A while tuning only B can improve generalization vs. tuning both B and A, in addition to practical gains achieved by $2\times$ parameter reduction.
- We validate our findings through experiments using models including RoBERTa, BART-Large, LLaMA-2, and the vision transformer (ViT), on both text and image datasets.

2. Related Work

Since the introduction of the original LoRA technique (Hu et al., 2021), numerous enhancements have been proposed. For example, quantization can reduce memory usage during training (Gholami et al., 2021; Dettmers et al., 2023; Guo et al., 2024). Also, the number of trainable parameters can be further reduced by adaptively allocating the rank (Zhang et al., 2023b), pruning during training (Benedek & Wolf, 2024), or pruning and quantizing after training (Yadav et al., 2023).

To further reduce the number of trainable LoRA parameters, the idea of reusing (randomly generated) weights or projections (Frankle & Carbin, 2018; Ramanujan et al., 2020) suggests strategies from learning diagonal matrices rescaling randomly-drawn and frozen B, A matrices (VeRA) (Kopiczko et al., 2024), deriving B and A from the SVD decomposition of the pre-trained W_0 and optimizing for a smaller matrix in the resulting basis (SVD-iff) (Han et al., 2023), learning a linear combination of fixed random matrices (NOLA) (Koohpayegani et al., 2023), or fine-tuning using orthogonal matrices (BOFT) (Liu et al., 2024). As echoed in our empirical results, previous methods observe that freezing A in conventional LoRA preserves performance (Zhang et al., 2023a). While nearly *all* recent studies treat the two matrices asymmetrically in their initialization or freezing schemes, there is a lack of formal investigation into this asymmetry in low-rank adaptation.

Zeng & Lee (2023) specifically investigate the expressive power of LoRA, but only focus on linearized networks and linear components. Their analysis does not consider aspects such as the particular distribution of the fine-tuning target data, generalization, or the differing roles of the different matrices. Lastly, we would like to highlight that even before LoRA, the effectiveness of fine-tuning was also explained by leveraging similar ideas related to the intrinsic low dimensionality of large models (Aghajanyan et al., 2021).

3. Preliminaries & Background

Notation. Suppose we are given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ representing a dense multiplication layer of a neural network foundation model. LoRA fine-tunes by updating the weights to $W_0 + \Delta W$, where $\text{rank}(\Delta W) = r \leq \min(d_{\text{out}}, d_{\text{in}})$. In particular, Hu et al. (2021) factor $\Delta W = BA$, where $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$ have restricted rank $\leq r$. During training, W_0 is fixed; LoRA updates (A, B). This yields more efficient updates than full fine-tuning, provided that $r < \frac{d_{\text{in}} d_{\text{out}}}{d_{\text{in}} + d_{\text{out}}}$.

Now using i to index layers of a network, a LoRA update is thus characterized by a set of pre-trained weight matrices $\mathbf{W} \triangleq \{W_i\}_{i=1}^{L}$, a set of pre-trained bias vectors $\mathbf{b} \triangleq \{b_i\}_{i=1}^{L}$, and a set of low-rank trainable weights $\Delta\mathbf{W} \triangleq \{\Delta W_i\}_{i=1}^{L'}$. LoRA may not update all L weight matrices in $\mathbf{W}$, in which case $L' \leq L$.

Motivating example. In Figure 1, we investigate the similarity of learned matrices A and B under three scenarios:

(a) random initialization, A & B trained multiple times on the same task;
(b) fixed initialization, A & B trained multiple times, each time on a different task; and
(c) random initialization, A & B trained multiple times, each time on a different task.

Here, we fine-tune RoBERTa large (Liu et al., 2019) with

LoRA on the tasks from the GLUE benchmark (Wang et al., 2018). Specifically, we fine-tuned *mrpc* with 5 random seeds for (a) and on *mrpc, rte, stsb, and cola* for (b) and (c).

The figure plots similarity of learned A and B matrices across layers in Figure 1, measured by canonical correlation analysis goodness of fit (Ramsay et al., 1984); see Appendix A for motivation.

These plots suggest that B is predominantly responsible for learning, while A is less important. Specifically, when training on the same task with different initializations (scenario (a)), the learned B matrices are similar to each other, while when training on different tasks (scenarios (b) and (c)), they are different. On the contrary, the similarity of learned A matrices is insensitive to training data and is determined by initialization; it is highest in scenario (b) when the initialization is fixed even though training data differs. See Appendix A for additional details of this experiment.

4. Theoretical Analysis

In this section, we analyze the asymmetry in prediction tasks and its effect on generalization. We discuss a general case rather than a specific neural network architecture, considering rank r adaptation of any parameter matrix $W = W_0 + BA$ used multiplicatively on some input-dependent vector, i.e.,

$$\text{layerOutput} = \psi((W_0 + BA) \cdot \phi(\text{layerInput}), \dots) \quad (1)$$

for some differentiable functions ψ, ϕ. Here, ψ may take more arguments depending on layerInput, which may have their own low rank adapted parameter matrices. This generic form encompasses both feedforward and attention layers.

In this setting, A serves to extract r features from $\phi(\text{layerInput})$, which are then used by B to predict some desired output for future layers. We will argue that training B to predict the output is crucial for correct outputs, while using a random A is often sufficient, as B can be optimized to use whatever information is retained in the r-dimensional projection $A \cdot \phi(\text{layerInput})$.

4.1. A, B asymmetry in prediction tasks

If we wish to reduce the effort of training both A and B in (1), in principle either A could be frozen and B tuned or B frozen and A tuned. As shown in §5 and elsewhere, these two options are not empirically equivalent: It is best to freeze A and tune B. In this section, we seek to understand the principle behind this asymmetry by theoretically analyzing the fine-tuning of a class of prediction models. We first build intuition with least-squares linear regression.

4.1.1. Multivariate linear least-squares

As a simple example analogous to a single network layer, we study d_{in}-to-d_{out} least-squares linear regression (in (1), set ϕ, ψ to be identity). Specifically, suppose there is an input $X \in \mathbb{R}^{d_{in}}$, an output $Y \in \mathbb{R}^{d_{out}}$, and a pre-trained linear model

$$y_{pre}(X) = W_0 X + b_0,$$

where $W_0 \in \mathbb{R}^{d_{out} \times d_{in}}$ and $b_0 \in \mathbb{R}^{d_{out}}$. With this model held constant, our goal is regressing (Y_{targ}, X_{targ}) pairs where Y_{targ} is given by:

$$Y_{targ} = W_{targ} X_{targ} + b_{targ}$$

with $W_{targ} = W_0 + \Delta$. Following LoRA, we model the target Δ using a low rank update to the pre-trained W_0, i.e. $W = W_0 + BA$:

$$\hat{y}(x) = (W_0 + BA)x + b,$$

where $B \in \mathbb{R}^{d_{out} \times r}$ and $A \in \mathbb{R}^{r \times d_{in}}$ for some r.

To find an A and B that best matches the output, we optimize the least squares loss on the difference between $\hat{y}$ and Y_{targ}:

$$\mathcal{L}(A, B) = \mathbb{E}_{(Y_{targ}, X_{targ})}[\|Y_{targ} - (W_0 + BA)X_{targ} - b\|_2^2]. \quad (2)$$

Below, we present lemmas on minimizing this loss while freezing either A or B. In both, for simplicity, we set $b = b_{targ}$ and $\mathbb{E}[X_{targ}] = 0$ and defer proofs to Appendix B.

Lemma 4.1 (Freezing A yields regression on projected **features**). *Optimizing $\mathcal{L}(A, B)$ while fixing $A = Q$ with $QQ^\top = I_r$ yields*

$$B^* = \Delta\Sigma Q^\top (Q\Sigma Q^\top)^{-1},$$

where $\Sigma = \text{Cov}[X_{targ}]$, with expected loss

$$\mathcal{L}(Q, B^*) = d_{out}\sigma^2 + \text{Tr}[\Delta\Sigma\Delta^\top] - \text{Tr}[Q\Sigma\Delta^\top\Delta\Sigma Q^\top (Q\Sigma Q^\top)^{-1}].$$

Lemma 4.2 (Freezing B yields regression on projected **outputs**). *Optimizing $\mathcal{L}(A, B)$ while fixing $B = U$ with $U^\top U = I_r$ yields*

$$A^* = U^\top (W_{targ} - W_0),$$

with expected loss

$$\mathcal{L}(A^*, U) = d_{out}\sigma^2 + \text{Tr}[\Delta\Sigma\Delta^\top] - \text{Tr}[U^\top\Delta\Sigma\Delta^\top U],$$

where $\Sigma = \text{Cov}[X_{targ}]$.

Comparing the lemmas above, A^* is simply the U projection of the targeted change in weight matrix $\Delta = W_{targ} - W_0$. Unlike B^*, the optimal choice of A^* does not consider the input data distribution captured by Σ.

Intuitively, if the goal of adaptation is to approximate some desired output, then projecting away the majority (since $r \ll d_{out}$) of the output is undesirable. In contrast, projecting away a portion of the input feature space will be less damaging, if the information X_{targ} contains about Y_{targ} is redundant (c.f., neuron dropout (Srivastava et al., 2014) in neural network training) or if the distribution of X_{targ} tends to be low-rank.

Consider the following extreme example. If $\Sigma = FF^\top$ is at most rank r, e.g. if $F \in d_{in} \times r$, then for each X there exists[1] an $N = F^\dagger X \in \mathbb{R}^r$ such that $X = FN$. Suppose you have tuned a pair A_*, B_*. For any orthonormal $Q \in \mathbb{R}^{r \times d_{in}}$ (e.g. one drawn at random), we can write

$$\begin{aligned} B_*A_*X &= B_*A_*FN \\ &= (B_*A_*F(QF)^{-1})QX, \end{aligned}$$

i.e. regardless of A_*, B_*, for any (random) Q, there is an exactly equivalent LoRA adaptation with $A = Q$ and $B = (B_*A_*F(QF)^{-1})$. In this setting, therefore, randomizing A (to Q) is equally expressive to tuning it (using A_*).

This intuition is also reflected in the typical LoRA initialization. When doing full LoRA (tuning both A, B), A usually is initialized to a random Gaussian matrix, and B is initialized to zero. This procedure—presumably empirically derived by Hu et al. (2021)—intuitively fits our analysis above, since random A yields good random predictive features, in contrast to using a random output prediction basis. Initializing B to zero then starts the optimization at a zero perturbation of the pretrained model.

We validate the above intuition with the following theorem:

Theorem 4.3 (A, B output fit asymmetry)**.** *Consider the settings of Lemmas 4.1 and 4.2, and suppose U, Q are sampled uniformly from their respective Stiefel manifolds. Then, $\mathcal{L}(A^*, U) \geq \mathcal{L}(Q, B^*)$ with high probability as $d/r \to \infty$.*

In other words, the least-squares prediction loss of only fine-tuning B is at least as good as only fine-tuning A.

Intuition on asymmetry gap. Theorem 4.3 is built on the following inequality:

$$\begin{aligned} &\mathrm{Tr}[\Sigma Q^\top (Q\Sigma Q^\top)^{-1} Q\Sigma\Delta^\top\Delta] \\ &\quad \geq \mathrm{Tr}[(Q^\top Q)\Sigma Q^\top (Q\Sigma Q^\top)^{-1} Q\Sigma\Delta^\top\Delta]. \end{aligned} \tag{3}$$

Let us consider an example regime to build intuition on the size of this gap. Following intuition that freezing A is most successful when the information content of the input is redundant (c.f., Aghajanyan et al. (2021)), suppose the distribution of X is low rank, i.e., Σ is of rank r_X. We can then write $\Sigma = U_X S_X U_X^\top$, where $U_X \in \mathbb{R}^{d_{in} \times r_X}$ is orthogonal and $S_X \in \mathbb{R}^{r_X \times r_X}$ is diagonal with nonnegative real entries.

For intuition, set $r_X = r$ and $S_X = \sigma^2 I_r$. We then have

$$\Sigma Q^\top (Q\Sigma Q^\top)^{-1} Q\Sigma\Delta^\top\Delta = \sigma^2 U_X U_X^\top \Delta^\top\Delta,$$

which no longer depends on Q. The expectation of the key inequality gap in (3) then becomes

$$\begin{aligned} &\mathbb{E}_Q \mathrm{Tr}[\Sigma Q^\top (Q\Sigma Q^\top)^{-1} Q\Sigma\Delta^\top\Delta] \\ &- \mathbb{E}_Q \mathrm{Tr}[(Q^\top Q)\Sigma Q^\top (Q\Sigma Q^\top)^{-1} Q\Sigma\Delta^\top\Delta] \\ &= \mathbb{E}_Q \mathrm{Tr}[(I - Q^\top Q)\sigma^2 U_X U_X^\top \Delta^\top\Delta] \\ &\to \left(1 - \frac{r}{d}\right) \mathrm{Tr}[U_X U_X^\top \Delta^\top\Delta] \end{aligned}$$

as d becomes large. In other words, the performance advantage of tuning B over A is large when $d \gg r$, which is the typical regime in practice.

4.1.2. Nonlinear losses and multilayer models

Recalling (1) with an input transformation ϕ and output transformation ψ, consider losses on the output of the form

$$\mathcal{L}(W) = \sum_{i=1}^{n} h(f(\psi(W\phi(x_i)))) - y_i^\top f(\psi(W\phi(x_i))), \tag{4}$$

where f, h are differentiable functions specified by the desired loss, $y_i \in \mathbb{R}^K$, $x_i \in \mathbb{R}^{d_{in}}$, and $W \in \mathbb{R}^{d_{out} \times d_{in}}$. This class contains logistic regression (with y being a one-hot encoded class vector), least-squares regression, and generalized linear regression—including a neural network with cross entropy loss with one layer being tuned.

We next analyze the gradient of this loss. Our argument is stated with one adapted parameter matrix, but it directly applicable to multilayer and transformer networks with multiple matrices being adapted, where ϕ, ψ, and f will in that scenario vary depending on each parameter matrix's position in the network; ϕ, ψ, and f will depend on other parameter matrices and the current value of their adaptations (by definition of gradients). The interpretation will now be that fixing A when adapting a parameter matrix $W^{(\ell)}$ projects the inputs of the corresponding parameter matrix to a lower-dimensional subspace while retaining the ability to fully match the outputs, and fixing B correspondingly projects the parameter matrix's outputs.

For simplicity of notation, the remaining derivation in this section takes ϕ, ψ to be the identity; the extension to general ϕ, ψ is clear. Then, the gradient of (4) is

$$\nabla_W \mathcal{L}(W) = \sum_{i=1}^{n} J_f^\top (Wx_i) \left[\nabla h(f(Wx_i)) - y_i\right] x_i^\top, \tag{5}$$

where J_f is the Jacobian of f. Starting from this formula, below we incorporate (1) by taking $W = W_0 + BA$.

[1] Here $F^\dagger$ denotes pseudoinverse.

Freezing A. Freezing $A = Q$ yields

$$\nabla_B \mathcal{L}(BQ + W_0) = \sum_{i=1}^{n} J_f^\top((BQ + W_0)x_i)\left[\nabla h(f((W_0 + BQ)x_i)) - y_i\right](Qx_i)^\top. \tag{6}$$

Like the least-squares case, the input data is projected by Q but the output y_i is unaffected.

Freezing B. Freezing $B = U$ yields

$$\nabla_A \mathcal{L}(UA + W_0) = U^\top \sum_{i=1}^{n} J_f^\top((UA + W_0)x_i)\left[\nabla h(f((W_0 + UA)x_i)) - y_i\right]x_i^\top. \tag{7}$$

Here, the coefficient of $x_i^\top$ can be thought of as the output fit term. It includes the Jacobian of f since f is applied between the weights and the output. Compared to (5) and (6), in (7) this output fit term is projected by U. If f is (near) linear, then this projection will be (approximately) data-independent, highlighting the loss of output information when freezing B.

Hence, in this more general setting, the different roles of A and B are still apparent, and we expect an asymmetry in being able to fit the output.

Example: Logistic regression. For multiclass logistic regression, we have a training dataset $\{(x_i, c_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^d$ (features) and $c_i \in \{1, \dots, K\}$ (label). Denote by $y_i \in \mathbb{R}^K$ the vector with $y_{c_i} = 1$ and $y_k = 0$ for $k \neq c_i$. The log likelihood is the cross-entropy error

$$\mathcal{L}(w_1, \dots, w_K) = \sum_{i=1}^{n}\sum_{k=1}^{K} y_i \ln(p_{i,k}), \tag{8}$$

where $p_{i,k} = \frac{\exp(w_k^\top x_i)}{\sum_{l=1}^K \exp(w_l^\top x_i)}$ and $w_k \in \mathbb{R}^d$. Let $W \in \mathbb{R}^{K\times d}$ whose k-th row is w_k. Then, (8) becomes

$$\mathcal{L}(W) = \sum_{i=1}^{n} \ln(\mathbf{1}^\top e^{Wx_i}) - y_i^\top W x_i,$$

where $\mathbf{1}$ is the column vector of size K with all elements equal to 1; note $y_i^\top \mathbf{1} = 1$ due to the one-hot structure. This loss can be put in the form (4) by setting $f(z) = z$ and $h(z) = \ln(\mathbf{1}^\top e^z)$. For freezing, we then have

$$\nabla_A \mathcal{L}(UA) = U^\top \sum_{i=1}^{n}(y_i - p_i(UA))x_i^\top \qquad \text{and}$$

$$\nabla_B \mathcal{L}(BQ) = \sum_{i=1}^{n}(y_i - p_i(BQ))(Qx_i)^\top,$$

where $p_i(W) = \frac{e^{Wx_i}}{\mathbf{1}^\top e^{Wx_i}} \in \mathbb{R}^K$. Freezing $B = U$, as in least-squares, implies that each output y_i is projected as $U^\top y_i$, implying that, at best, the model can hope to only learn outputs in the small random subspace U. In contrast, freezing $A = Q$ is equivalent to logistic regression on the full output with features projected by Q: $\{(Qx_i, y_i)\}_{i=1}^n$.

4.2. Advantages of tuning only B over BA together

In the previous section, we established that fine-tuning B alone is typically superior to fine-tuning A alone. It remains, however, to motivate fine-tuning B alone over fine-tuning both A and B together. In this section, we show that the reduced amount of adapted parameters by (roughly) half provides computational gains and improvements in information-theoretic generalization bounds.

4.2.1. Number of parameters

The key benefit of LoRA is parameter efficiency, which saves memory during training, storage and communication (Lialin et al., 2023). Fine-tuning B alone as opposed to both A and B reduces the number of parameters by a factor of $\frac{d_{out}}{d_{out}+d_{in}}$, which equals 0.5 when $d_{in} = d_{out}$.

4.2.2. Generalization bounds

Consider a learning task, where the training examples lie in $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$; here, $\mathcal{X}$ denotes the feature space and $\mathcal{Y}$ is the label space. Suppose one observes a training set $S_n \triangleq (Z_1, \dots, Z_n) \in \mathcal{Z}^n$, with n i.i.d. training examples from unknown distribution μ. Denote by $\mu^{\otimes n} = \mu \times \cdots \times \mu$ the distribution of S_n. The objective of the learner is to find a predictor $f : \mathcal{X} \to \mathcal{Y}$ that maps features to their labels. We assume each predictor is parameterized by $w \in \mathcal{W}$ (e.g., if f is a neural network, w denotes its weights). Denote by $\mathcal{A} : \mathcal{Z}^n \to \mathcal{W}$ the learning algorithm which selects a predictor given S_n. $\mathcal{A}$ is, in general, a probabilistic mapping, and we denote by $P_{W|S_n}$ the distribution of its output W given input S_n. If $\ell : \mathcal{W} \times \mathcal{Z} \to \mathbb{R}_+$ is a loss, we define:

$$\text{Population risk:} \quad \mathcal{R}_\mu(w) \triangleq \mathbb{E}_{Z\sim\mu}[\ell(w, Z)]$$

$$\text{Empirical risk:} \quad \widehat{\mathcal{R}}_n(w) \triangleq \frac{1}{n}\sum_{i=1}^{n} \ell(w, Z_i).$$

The generalization error of $\mathcal{A}$ is

$$\text{gen}(\mu, \mathcal{A}) \triangleq \mathbb{E}_{(W,S_n)\sim P_{W|S_n}\times\mu^{\otimes n}}\left[\mathcal{R}_\mu(W) - \widehat{\mathcal{R}}_n(W)\right].$$

We bound this generalization error using the information-theoretic generalization framework of Xu & Raginsky (2017). Consider the following incarnations of fine-tuning algorithms, corresponding to classic LoRA (tuning both A, B matrices), tuning only B, and tuning only A:

Definition 4.4 (Fine-tuning algorithms)**.** Let $\mathbf{W} = \{W_i\}_{i=1}^L$ be the L parameter matrices of a pretrained model.

Let $\mathcal{I} \subseteq \{1, \ldots, L\}$ be a specified subset of the parameter matrices to be fine-tuned. Given a fine-tuning training set S_n, let r be a chosen rank and suppose each tuned parameter is quantized to q bits. We define the following algorithmic frameworks (other details can be arbitrary) for choosing an adaptation $\mathbf{\Delta W} = \{\Delta_i\}_{i\in\mathcal{I}}$, yielding a fine-tuned $W_{tuned} = \{W_{tuned,i}\}_{i=1}^{L}$ with $W_{tuned,i} = W_i + \Delta_i$ for $i \in \mathcal{I}$ and $W_{tuned,i} = W_i$ otherwise:

- $\mathcal{A}_{BA}$: For each $i \in \mathcal{I}$, constrain $\Delta_i = B_i A_i$ and optimize $\{B_i, A_i\}_{i\in\mathcal{I}}$ to fit the data S_n.
- $\mathcal{A}_B$: For each $i \in \mathcal{I}$, sample $Q_i \in \mathbb{R}^{r\times d_{in}^{(i)}}$ at random, constrain $\Delta_i = B_i Q_i$, and optimize $\{B_i\}_{i\in\mathcal{I}}$ to fit the data S_n.
- $\mathcal{A}_A$: For each $i \in \mathcal{I}$, sample $U_i \in \mathbb{R}^{d_{out}^{(i)}\times r}$ at random, constrain $\Delta_i = U_i A_i$, and optimize $\{A_i\}_{i\in\mathcal{I}}$ to fit the data S_n.

We have the following lemma, proved in Appendix C:

Lemma 4.5 (Generalization bounds on adapting A and/or B)**.** *Consider the algorithms of Definition 4.4. Assume that $\ell^{\mathbf{W},\mathbf{b}}(\Delta\mathbf{W}, \widetilde{Z})$ is σ-sub-Gaussian[2] under $(\Delta\mathbf{W}, \widetilde{Z}) \sim P_{\Delta\mathbf{W}|\mathbf{W},\mathbf{b}} \times \mu$. Then,*

$$|\mathrm{gen}(\mu, \mathcal{A}_{BA})| \leq \sqrt{\frac{2rq\sigma^2 \ln 2}{n} \sum_{i\in\mathcal{I}} (d_{in}^{(i)} + d_{out}^{(i)})},$$

$$|\mathrm{gen}(\mu, \mathcal{A}_{B})| \leq \sqrt{\frac{2rq\sigma^2 \ln 2}{n} \sum_{i\in\mathcal{I}} d_{out}^{(i)}},$$

$$|\mathrm{gen}(\mu, \mathcal{A}_{A})| \leq \sqrt{\frac{2rq\sigma^2 \ln 2}{n} \sum_{i\in\mathcal{I}} d_{in}^{(i)}}.$$

This generalization bound increases with the number of parameters being tuned, which is an increasing function of r and the dimensions of the parameter matrices. Importantly, since tuning just one factor (A or B) involves tuning fewer parameters than A and B together, the generalization bound is correspondingly smaller. In the case where the $d_{in}^{(i)} = d_{out}^{(i)}$, the bound for tuning one factor only is a factor of $\sqrt{2}$ smaller than the bound for tuning both factors, implying that the rank r for $\mathcal{A}_B$ could be doubled and have a generalization bound matching that of $\mathcal{A}_{BA}$.

4.3. Discussion of theoretical analysis

The previous two sections establish two conclusions: (1) Tuning A has limited importance when trying to match a desired output; and (2) Tuning one factor instead of two reduces the number of parameters for the same r, while improving generalization bounds and potentially providing memory benefits.

[2]Bounded losses are sub-Gaussian.

Given a fixed parameter count and generalization budget, therefore, we can use a larger $r = r_B$ when fine-tuning B alone than the r_{BA} that would be used on standard LoRA fine-tuning both A and B. This addition provides more expressive power for the same number of parameters without loss of generalization bounds. Hence, when matching parameter or generalization budget, we expect that fine-tuning a rank-r_B B typically improves performance over fine-tuning a rank-r_{BA} BA LoRA adaptation.

5. Experiments

We investigate the asymmetry of low-rank adaptation methods with RoBERTa (Liu et al., 2019), BART (Lewis et al., 2020), Llama-2 (Touvron et al., 2023), and Vistion Transformer (Dosovitskiy et al., 2020). We evaluate the performance of fine-turning strategies on natural language understanding (GLUE (Wang et al., 2018), MMLU (Hendrycks et al., 2020)), natural language generation (XSum (Narayan et al., 2018) and CNN/DailyMail (Chen et al., 2016)), and multi-domain image classification (Gulrajani & Lopez-Paz, 2020).

We implement all algorithms using PyTorch starting from the publicly-available Huggingface Transformers code base (Wolf et al., 2019). The conventional LoRA method applies a scaling coefficient α/r to ΔW. Following LoRA (Hu et al., 2021), we fix $\alpha = 2r$ to be twice the rank. Throughout our experiments, we use $\hat{A}$ to indicate matrix A is being updated during fine-tuning and use subscripts $\{$*rand,* 0*, km*$\}$ to indicate that the matrix is initialized as a random orthonormal matrix, zero matrix, and the random uniform initialization used in the original LoRA, respectively. Note that a properly normalized $d \times r$ random matrix with independent entries will have close to orthonormal columns when $d \gg r$ (see e.g. Theorem 4.6.1 of Vershynin (2020)), implying that the random orthonormal and random uniform initializations should be essentially equivalent.

We compare to the following methods:

1. **Full fine-tuning (FT):** The most straightforward adaptation method, which initializes model parameters with the pre-trained weights and updates the whole model with gradient back-propagation.
2. **Linear Probing (LP) (Kumar et al., 2022):** A simple yet effective method that updates the last linear layer.
3. **IA^3 (Liu et al., 2022):** Injects learned vectors in the attention and feedforward modules.
4. **LoRA:** (Hu et al., 2021) Fine-tunes both A and B matrices of an additive BA adaptation as introduced in previous sections, with a separate adaptation for each query/key/value parameter matrix.
5. **AdaLora:** (Zhang et al., 2023b) A variant of LoRA that adaptively changes the rank for each layer.

Table 1. Different adaptation methods on the GLUE benchmark. We report the overall (matched and mismatched) accuracy for MNLI, Matthew's correlation coefficient for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. Higher is better for all metrics.

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
LoRA ($r=8$)	0.8%	$90.3_{\pm.07}$	$95.6_{\pm0.36}$	$90.3_{\pm0.85}$	$64.4_{\pm1.8}$	$94.0_{\pm0.29}$	$84.1_{\pm0.96}$	$91.5_{\pm0.16}$	87.2
AdaLoRA	2.5%	$90.4_{\pm.37}$	$95.9_{\pm.13}$	$90.1_{\pm.54}$	$67.5_{\pm1.3}$	$94.7_{\pm.22}$	$85.4_{\pm.20}$	$91.3_{\pm1.0}$	87.9
$(IA)^3$	0.7%	$90.0_{\pm.21}$	$95.4_{\pm.17}$	$83.7_{\pm.13}$	$57.6_{\pm.67}$	$93.7_{\pm.07}$	$70.3_{\pm1.5}$	$87.0_{\pm0.4}$	82.5
LoRA-FA	0.3%	$90.3_{\pm.06}$	$95.6_{\pm.17}$	$90.6_{\pm.32}$	$67.3_{\pm2.3}$	$93.4_{\pm.61}$	$82.4_{\pm1.4}$	$91.2_{\pm.29}$	87.3
$\hat{\mathbf{B}}_0 A_{rand}$ ($r=8$)	0.3%	$90.1_{\pm.19}$	$\underline{95.8}_{\pm.29}$	$89.7_{\pm.13}$	$\mathbf{\underline{67.5}}_{\pm\mathbf{1.2}}$	$\underline{94.0}_{\pm.27}$	$82.8_{\pm1.5}$	$\mathbf{\underline{91.9}}_{\pm\mathbf{.26}}$	87.4
$\hat{\mathbf{B}}_0 A_{rand}$ ($r=16$)	0.8%	$90.1_{\pm.20}$	$\mathbf{\underline{96.1}}_{\pm\mathbf{.18}}$	$\mathbf{\underline{90.7}}_{\pm\mathbf{.90}}$	$66.1_{\pm2.6}$	$\mathbf{\underline{94.4}}_{\pm\mathbf{.10}}$	$\mathbf{\underline{84.1}}_{\pm\mathbf{.96}}$	$91.2_{\pm.42}$	87.5
$B_{rand}\hat{\mathbf{A}}_0$ ($r=8$)	0.3%	$\mathbf{\underline{90.3}}_{\pm\mathbf{.18}}$	$95.5_{\pm.66}$	$89.3_{\pm.09}$	$58.7_{\pm2.5}$	$93.8_{\pm.21}$	$77.1_{\pm1.3}$	$90.7_{\pm.31}$	84.2
$B_{rand}\hat{\mathbf{A}}_0$ ($r=16$)	0.8%	$89.9_{\pm.19}$	$\underline{95.6}_{\pm.64}$	$90.2_{\pm0.23}$	$60.3_{\pm3.3}$	$93.9_{\pm0.25}$	$80.4_{\pm0.21}$	$90.9_{\pm0.13}$	85.9

Table 2. Different initialization of classic LoRA, setting either A or B to be zeros. Note that the trained result is not sensitive to different initializations, with performance differences tending to be smaller than the standard error.

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
$\hat{\mathbf{B}}_0\hat{\mathbf{A}}_V$	0.8%	$\mathbf{90.4}_{\pm\mathbf{0.11}}$	$95.9_{\pm0.16}$	$90.7_{\pm0.84}$	$64.0_{\pm0.50}$	$94.4_{\pm0.16}$	$84.1_{\pm0.15}$	$91.8_{\pm00.15}$	87.3
$\hat{\mathbf{B}}_0\hat{\mathbf{A}}_{rand}$	0.8%	$\mathbf{90.4}_{\pm\mathbf{0.15}}$	$96.0_{\pm0.11}$	$91.5_{\pm1.1}$	$64.1_{\pm0.67}$	$94.5_{\pm0.11}$	$\mathbf{85.6}_{\pm\mathbf{0.96}}$	$\mathbf{92.0}_{\pm\mathbf{0.31}}$	87.7
$\hat{\mathbf{B}}_U\hat{\mathbf{A}}_0$	0.8%	$90.3_{\pm0.07}$	$\mathbf{96.1}_{\pm\mathbf{.18}}$	$\mathbf{91.7}_{\pm\mathbf{0.33}}$	$64.9_{\pm1.5}$	$\mathbf{94.7}_{\pm\mathbf{0.33}}$	$84.8_{\pm0.96}$	$91.9_{\pm0.19}$	87.8
$\hat{\mathbf{B}}_{rand}\hat{\mathbf{A}}_0$	0.8%	$90.3_{\pm0.27}$	$96.0_{\pm.26}$	$90.8_{\pm0.51}$	$\mathbf{66.0}_{\pm\mathbf{1.01}}$	$94.5_{\pm0.38}$	$83.6_{\pm1.5}$	$\mathbf{92.0}_{\pm\mathbf{0.18}}$	87.8

5.1. Natural Language Understanding

We use the General Language Understanding Evaluation (GLUE, Wang et al., 2018) to evaluate the fine-tuning performance of different fine-tuning strategies. The GLUE benchmark contains a wide variety of tasks including question-answering, textual similarity, and sentiment analysis. We applied fine-tuning methods to the RoBERTa (large) model (Liu et al., 2019), which has 355M parameters. To enable a fair comparison, we initialize the weights for all tasks with the original pretrained RoBERTa weights.

In Table 1 (see the appendix for an expanded table), we compare different freezing & initialization strategies with LoRA and other baselines. We underline to indicate that performance is better than conventional LoRA also we use **bold** to denote the best performance when freezing one of the matrices. First, we can see a clear trend where solely updating the B matrix outperforms just learning the A matrix. In addition, when doubling the rank to match the trainable parameters, $\hat{\mathbf{B}}_0 A_{orth}$ consistently outperforms conventional LoRA. This confirms our hypothesis in §4.3 that any loss in expressive power by not tuning A can be made up for by the larger intrinsic rank of B at no additional parameter cost. In fact, its performance statistically matches that of AdaLoRA, which uses over 3 times the parameters (incurring the associated memory and training costs).

To assess the effects of different initialization methods for low-rank adaptation, we investigate different initialization methods thoroughly in Table 2. We can see that the best results always come from orthogonal initialization, which further supports our conclusions in §4.

Table 3. R-1/2/L (%) on text summarization with BART-large on XSum and CNN/DailyMail (* Here we report numbers from (Zhang et al., 2023b)).

Method	# Param.	XSum	CNN/DailyMail
Full FT*	100 %	45.49 / 22.33 / 37.26	44.16 / 21.28 / 40.90
LoRA (r=2)*	0.26 %	42.81 / 19.68 / 34.73	43.68 / 20.63 / 40.71
$\hat{\mathbf{B}}_0 A_{rand,r=16}$	0.44 %	**42.91 / 19.61 / 34.64**	**43.65 / 20.62 / 40.72**
$B_{rand}\hat{\mathbf{A}}_{0,r=16}$	0.44 %	42.37 / 19.30 / 34.29	43.38 / 20.36 / 40.48
$\hat{\mathbf{B}}_0\hat{\mathbf{A}}_{rand,r=8}$	0.44 %	43.78 / **20.47 / 35.53**	43.96 / 20.94 / 41.00
$\hat{\mathbf{B}}_{rand}\hat{\mathbf{A}}_{0,r=8}$	0.44 %	**43.80** / 20.39 / 35.48	**44.07 / 21.08 / 41.19**

5.2. Natural Language Generation

To investigate the asymmetry of low-rank fine-tuning in natural language generation (NLG), we fine-tune a BART-large model (Lewis et al., 2020) and evaluate model performance on the XSum (Narayan et al., 2018) and CNN/DailyMail (Chen et al., 2016) datasets. Following Zhang et al. (2023b), we apply low-rank adaptation to every query/key/value matrix and report ROUGE 1/2/L scores (R-1/2/L, (Lin, 2004)). We fine-tune models for 15 epochs. We select the beam length as 8 and batch size as 48 for XSum, and the beam length as 4, batch size as 48 for CNN/DailyMail. More details of the configurations are

Table 4. DomainBed results (mean accuracy and standard deviation in %). ID and OOD denote in-domain and out-of-domain test error, respectively. For OOD we report the average performance across different environments.

Method	# Param.	VLCS		PACS		OfficeHome	
		(ID)	(OOD)	(ID)	(OOD)	(ID)	(OOD)
LoRA $_{r=8}$	0.46%	$73.51_{\pm 0.62}$	$56.43_{\pm 1.96}$	$94.94_{\pm 0.56}$	$\mathbf{75.58_{\pm 0.92}}$	$78.54_{\pm 1.49}$	$74.46_{\pm 0.40}$
LP	0.00%	$75.58_{\pm 1.66}$	$71.70_{\pm 1.04}$	$81.62_{\pm 0.34}$	$61.73_{\pm 1.25}$	$58.38_{\pm 0.76}$	$68.59_{\pm 0.22}$
Full Fine-tuning	100%	$76.21_{\pm 1.95}$	$64.87_{\pm 6.44}$	$\mathbf{98.15_{\pm 0.56}}$	$74.90_{\pm 2.43}$	$\mathbf{80.67_{\pm 1.22}}$	$63.23_{\pm 0.64}$
$\hat{\mathbf{B}}A_{rand,r=8}$	0.29%	$77.40_{\pm 2.30}$	$\mathbf{75.81_{\pm 1.65}}$	$92.45_{\pm 2.68}$	$72.55_{\pm 1.03}$	$77.66_{\pm 0.89}$	$77.72_{\pm 0.32}$
$\hat{\mathbf{B}}A_{rand,r=16}$	0.46%	$\mathbf{79.10_{\pm 1.41}}$	$75.40_{\pm 1.24}$	$93.52_{\pm 0.20}$	$73.76_{\pm 0.67}$	$77.63_{\pm 0.84}$	$\mathbf{77.85_{\pm 0.33}}$
$B_{rand}\hat{\mathbf{A}}_{r=8}$	0.29%	$76.71_{\pm 0.93}$	$72.50_{\pm 0.89}$	$92.02_{\pm 1.07}$	$66.25_{\pm 0.80}$	$72.36_{\pm 0.69}$	$73.66_{\pm 0.35}$

Table 5. Accuracy (%) on MMLU benchmark.

Method	# Param.	5-shot				
		Hums	STEM	Social	Other	Avg
Llama-2-7B	100%	43.98	34.11	49.08	44.31	43.14
LoRA $_{r=32}$	0.24%	44.59	36.50	51.81	**45.75**	44.76
$\hat{\mathbf{B}}_0 A_{rand,r=32}$	0.12%	**44.17**	**36.00**	**46.88**	45.14	**45.36**
$B_{rand}\hat{\mathbf{A}}_{0,r=32}$	0.12%	44.36	35.93	51.46	46.85	44.51
$\hat{\mathbf{B}}_0 A_{rand,r=64}$	0.12%	**45.10**	**37.65**	**55.08**	51.08	**46.46**

in the Appendix E.

The results are summarized in Table 3. In the first two rows, we observe the asymmetry between the factors since freezing A and only updating B always outperforms only updating A. The last two rows show the results of tuning both matrices with different initializations, showing that the asymmetry is not explained by the initialization strategy.

5.3. Massive Multitask Language Understanding

We fine-tune the pretrained Llama-2-7B model (Touvron et al., 2023) using instruction tuning on the Alpaca dataset (Wang et al., 2023). We assess the asymmetry on the MMLU benchmark (Hendrycks et al., 2020), which consists of 57 distinct language tasks. As shown in Table 5, the asymmetry also exists in larger language models, and updating B consistently outperforms updating A. Moreover, it also outperforms standard LoRA except for "Other" where it matches the performance, reflecting the benefits of being able to increase r without tuning more parameters.

5.4. Vision Transformers and Generalization

We next measure generalization, motivated by the theory in §4.2. In particular, we work with ViTs in image classification tasks using the Domainbed testbed for domain generalization (Gulrajani & Lopez-Paz, 2020). Domainbed contains several datasets, each composed of multiple environments (or domains). Classes in each environment tend to be similar at a high level but differ in terms of style. We fine-tune a pre-trained ViT, originally trained on ImageNet, on the LabelMe, Cartoon, and Clipart environments within the VLCS, PACS, and Office-Home datasets, respectively. We employ different benchmark fine-tuning methods such as full fine-tuning, linear probing, and LoRA, and compare their performance to freezing either A or B in in-domain and out-of-domain generalization. We adhere to the original 80% training and 20% testing splits.

Results are presented in Table 4. In line with our expectations, randomly initializing and freezing matrix A while only updating matrix B generally results in better out-of-domain test accuracy. We report additional generalization results in Appendix H, in which we compare the train set and test set accuracy of the different approaches. We consistently find that fine-tuning a single matrix leads to smaller gaps between these two quantities compared to LoRA, paralleling the corresponding reduction in the generalization bounds of §4.2.

5.5. Ablation study and analysis

We also observe the benefit of computational run time when freezing the A matrix, even when doubling the rank. This is because freezing matrix A means its gradients do not need to be stored or computed, reducing the memory footprint for gradients during the training. We provide additional experimental results on multiple datasets to illustrate the runtime improvement. Specifically, in table (6) we compare the train samples per second of different PEFT methods on multiple fine-tuning tasks.

Table 6. Train samples per second on various datasets

	GLUE RTE	GLUE SST-2
LoRA	$4.71_{\pm 0.03}$	$227.62_{\pm 0.59}$
AdaLoRA	$2.90_{\pm 0.11}$	$88.14_{\pm 0.19}$
$\hat{B}$ (r=8)	$7.29_{\pm 0.16}$	$255.45_{\pm 13.38}$
$\hat{B}$ (r=16)	$6.28_{\pm 0.17}$	$265.80_{\pm 12.13}$

We also conducted a new ablation study to investigate how different fixed A matrices will affect the performance. Specifically, we use three initializations: (1) Columns dependent on each other, (2) Rows dependent on each other, and (3) a Banded matrix with a bandwidth equal to rank. As we can see, the model struggled to learn anything when either the columns and rows of A are correlated. Also, fixing A to be a banded matrix leads to reasonable performance. Such observation further agrees with our theoretical formulation where we require the fixed A to be *orthogonal*.

Table 7. Different fixed A on RTE task

	RTE
$\hat{B}A_{(1)}$	50.9 $\pm$3.13
$\hat{B}A_{(2)}$	52.71 $\pm$3.29
$\hat{B}\ A_{(3)}$	83.51 $\pm$2.18
$\hat{B}\ A_{rand}$ (Ours)	84.1 $\pm$0.83

6. Conclusion

In this paper, we formally identify and investigate asymmetry in the roles of low-rank adapter matrices in LoRA fine-tuning. The A matrices extract features from the input, while the B matrices project these features towards the desired objective. We illustrate differences between the two matrices from both theoretical and empirical perspectives. Our theoretical analysis explains the asymmetry in the fine-tuning of large models and suggests that freezing A as a random orthogonal matrix can improve generalization, a claim we corroborate with experiments across multiple models and datasets. Our work serves as an initial step to unveil the mechanisms of fine-tuning large models, and it provides an understanding that can benefit future research directions, promoting efficiency and interpretability.

Impact Statement

This paper presents work whose goal is to advance machine learning. There are no societal consequences of our work that we feel must be specifically highlighted here.

Acknowledgement

We thank Lingxiao Li, Aritra Guha, and the anonymous reviewers for their valuable feedback and helpful recommendations. The MIT Geometric Data Processing Group acknowledges the generous support of Army Research Office grants W911NF2010168 and W911NF2110293, from the CSAIL Systems that Learn program, from the MIT–IBM Watson AI Laboratory, from the Toyota–CSAIL Joint Research Center, and from an Amazon Research Award.

References

Aghajanyan, A., Gupta, S., and Zettlemoyer, L. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. 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)*. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.568. URL http://dx.doi.org/10.18653/v1/2021.acl-long.568.

Benedek, N. and Wolf, L. Prilora: Pruned and rank-increasing low-rank adaptation. 2024. URL https://api.semanticscholar.org/CorpusID:267068991.

Chen, D., Bolton, J., and Manning, C. D. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2016. doi: 10.18653/v1/p16-1223. URL http://dx.doi.org/10.18653/v1/P16-1223.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *ArXiv*, abs/2305.14314, 2023. URL https://api.semanticscholar.org/CorpusID:258841328.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2018.

Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. A survey of quantization methods for efficient neural network inference, 2021.

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization, 2020.

Guo, H., Greengard, P., Xing, E. P., and Kim, Y. Lq-lora: Low-rank plus quantized matrix decomposition for efficient language model finetuning, 2024.

Han, L., Li, Y., Zhang, H., Milanfar, P., Metaxas, D., and Yang, F. Svdiff: Compact parameter space for diffusion fine-tuning. *arXiv preprint arXiv:2303.11305*, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding, 2020.

Hu, J. E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., and Chen, W. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021. URL https://api.semanticscholar.org/CorpusID:235458009.

HuggingFace. Peft. https://github.com/huggingface/peft, Year.

Koohpayegani, S. A., Navaneet, K., Nooralinejad, P., Kolouri, S., and Pirsiavash, H. Nola: Networks as linear combination of low rank random basis, 2023.

Kopiczko, D. J., Blankevoort, T., and Asano, Y. M. Vera: Vector-based random matrix adaptation, 2024.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.

Kumar, A., Raghunathan, A., Jones, R., Ma, T., and Liang, P. Fine-tuning can distort pretrained features and underperform out-of-distribution, 2022.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.703. URL http://dx.doi.org/10.18653/v1/2020.acl-main.703.

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

Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. A. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35: 1950–1965, 2022.

Liu, W., Qiu, Z., Feng, Y., Xiu, Y., Xue, Y., Yu, L., Feng, H., Liu, Z., Heo, J., Peng, S., Wen, Y., Black, M. J., Weller, A., and Schölkopf, B. Parameter-efficient orthogonal finetuning via butterfly factorization. In *ICLR*, 2024.

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

Narayan, S., Cohen, S. B., and Lapata, M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1206. URL http://dx.doi.org/10.18653/v1/D18-1206.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network? In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. doi: 10.1109/cvpr42600.2020.01191. URL http://dx.doi.org/10.1109/CVPR42600.2020.01191.

Ramsay, J., ten Berge, J., and Styan, G. Matrix correlation. *Psychometrika*, 49(3):403–423, 1984.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023.

Vershynin, R. High-dimensional probability. *University of California, Irvine*, 2020.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, 2018. doi: 10.18653/v1/w18-5446. URL http://dx.doi.org/10.18653/v1/W18-5446.

Wang, Y., Ivison, H., Dasigi, P., Hessel, J., Khot, T., Chandu, K. R., Wadden, D., MacMillan, K., Smith, N. A., Beltagy, I., et al. How far can camels go? exploring the state

of instruction tuning on open resources. *arXiv preprint arXiv:2306.04751*, 2023.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Xu, A. and Raginsky, M. Information-theoretic analysis of generalization capability of learning algorithms. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Yadav, P., Choshen, L., Raffel, C., and Bansal, M. Compeft: Compression for communicating parameter efficient updates via sparsification and quantization. *arXiv preprint arXiv:2311.13171*, 2023.

Zeng, Y. and Lee, K. The expressive power of low-rank adaptation, 2023.

Zhang, L., Zhang, L., Shi, S., Chu, X., and Li, B. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning, 2023a.

Zhang, Q., Chen, M., Bukharin, A., Karampatziakis, N., He, P., Cheng, Y., Chen, W., and Zhao, T. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023b.

A. Similarity Metric in Figure 1

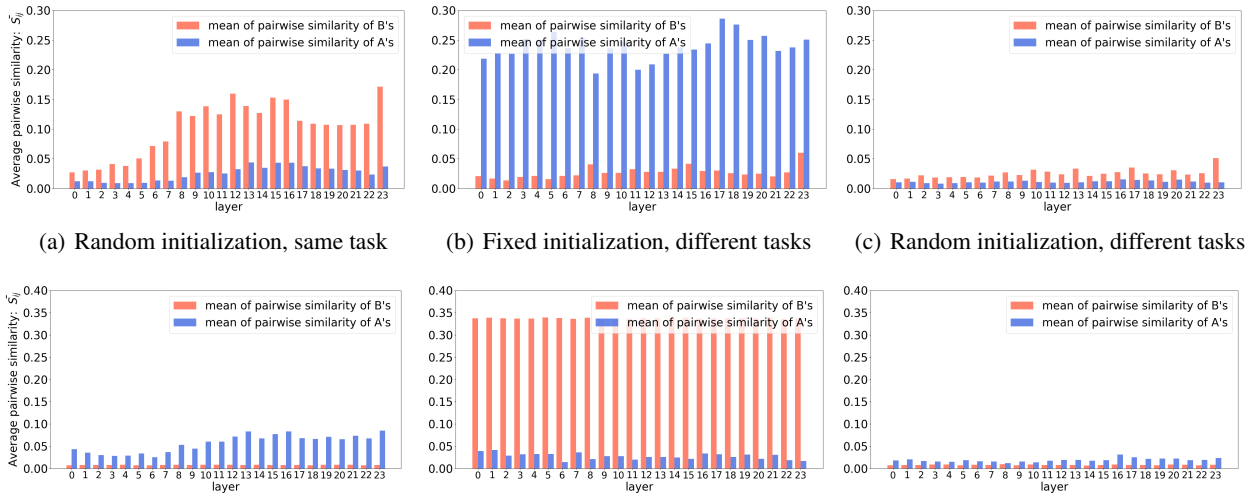
To measure the similarity of learned A and B matrices we adopted a measure that accounts for the invariance of LoRA fine-tuning. Let $\Delta W = BA$ denote the learned LoRA adapter. Since $BA = BCC^{-1}A$ for any invertible matrix $C \in \mathbb{R}^{r\times r}$, we can define $\tilde{B} = BC$ and $\tilde{A} = C^{-1}A$ resulting in the same LoRA adapter $\Delta W = \tilde{B}\tilde{A}$. Thus, to measure the similarity of LoRA matrices we need a metric that is invariant to invertible linear transformations, i.e., dissimilarity$(B, BC) = 0$ for any invertible C. In our experiment, we used Canonical Correlation Analysis goodness of fit (Ramsay et al., 1984), similar to prior work comparing neural network representations (Kornblith et al., 2019). The key idea is to compare orthonormal bases of the matrices, thus making this similarity metric invariant to invertible linear transformations.

More specifically, given two matrices $X \in \mathbb{R}^{n\times r_1}$ and $Y \in \mathbb{R}^{n\times r_2}$, the similarity is computed as follows: $\|U_Y^\top U_X\|_F^2 / \min\{r_1, r_2\}$, where U_X/U_Y is the orthonormal bases for the columns of X/Y. Following a similar method as in Hu et al. (2021), for A we perform SVD and use the right-singular unitary matrices as the bases, and use left-singular unitary matrices for B.

A.1. Reversed Initialization

The initialization of adapter matrices can play an important role in LoRA fine-tuning. To further investigate the effect of initialization on asymmetry, we reverse the initialization compared to conventional LoRA, where A is initialized to zero and B is initialized with random uniform distributions. Overall, we observe that the trend of differences also reverses, which is expected given the significant role of initialization in training deep learning models.

When comparing the similarities of different initialization strategies, we can still draw the same conclusion about the importance of the B matrix. For example, compared with Figure 2(a), the A matrices in Figure 2(d) have a smaller similarity in average. Such difference can also be observed when comparing Figure 2(b) and 2(e).



(a) Random initialization, same task (b) Fixed initialization, different tasks (c) Random initialization, different tasks

(d) Random initialization, same task (e) Fixed initialization, different tasks (f) Random initialization, different tasks

Figure 2. Similarity of learned LoRA matrices A & B across layers of a RoBERTa model fine-tuned with different initialization and data settings. We compare the results from both conventional LoRA initialization (In Figure (a), (b), and (c), A is initialized as random uniform B is initialized as zero) and a reversed initialization (In Figure (d), (e), and (f), A is initialized as zero B is initialized as random uniform.

B. Asymmetry Proofs for Multivariate Least Squares

B.1. Proof of Lemma 4.2

Consider freezing $B = U$ where U is orthogonal ($U^\top U = I_r$) and fine-tuning A. The objective becomes

$$\begin{aligned}
A^* &= \arg\min_A \mathcal{L}(A, U) \\
&= \arg\min_A \mathbb{E}_{(Y_{targ}, X_{targ})} \|Y_{targ} - (W_0 + UA)X_{targ} - b\|_2^2 \\
&= \arg\min_A \mathbb{E} \|(W_{targ}X_{targ} - W_0 X_{targ}) - UAX_{targ}\|_2^2 \\
&= \arg\min_A \mathbb{E} \left\|U^\top((W_{targ} - W_0)X_{targ} + n) - AX_{targ}\right\|_2^2 \\
&= U^\top \Delta.
\end{aligned}$$

Interestingly, note that this solution A^* does not depend on the distribution of X_{targ}, it is simply the projection of the difference between the pretrained W_0 and the target W_{targ}. This is because, intuitively, freezing B is projecting down the *outputs* into r dimensional space, and then optimizing A to match these projected outputs. It can be shown that the expected squared prediction error is

$$\mathcal{L}(A^*, U) = d_{out}\sigma^2 + \mathrm{Tr}[\Delta\Sigma\Delta^\top] - \mathrm{Tr}[U^\top\Delta\Sigma\Delta^\top U],$$

where $\Sigma = \mathrm{Cov}[X_{targ}]$.

B.2. Proof of Lemma 4.1

Consider freezing $A = Q$ where Q is orthogonal ($QQ^\top = I_r$) and fine-tuning B. The objective becomes

$$\begin{aligned}
B^* &= \arg\min_B \mathcal{L}(Q, B) \\
&= \arg\min_B \mathbb{E}_{(Y_{targ}, X_{targ})} \|Y_{targ} - (W_0 + BQ)X_{targ}\|_2^2 \\
&= \arg\min_B \mathbb{E} \|(Y_{targ} - W_0 X_{targ}) - B(QX_{targ})\|_2^2,
\end{aligned}$$

which is simply an ordinary least squares regression problem mapping QX_{targ} to $(Y_{targ} - W_0 X_{targ})$. The solution is known to be

$$B^* = \Delta\Sigma Q^\top (Q\Sigma Q^\top)^{-1}$$

yielding an expected squared prediction error of

$$\mathcal{L}(Q, B^*) = d_{out}\sigma^2 + \mathrm{Tr}[\Delta\Sigma\Delta^\top] - \mathrm{Tr}[Q\Sigma\Delta^\top\Delta\Sigma Q^\top(Q\Sigma Q^\top)^{-1}].$$

Note that the solution is now clearly dependent on the distribution of X_{targ}, and the first two terms of the squared prediction error are the same but the third term is different.

B.3. Proof of Theorem 4.3

Note that since Σ is positive semidefinite, its symmetric square root $\Sigma^{1/2}$ exists and we can simplify the third term in the expression for freezing A using the definition of the Moore-Penrose pseudoinverse $(\cdot)^\dagger$ and trace equalities as

$$\begin{aligned}
III_A &= \mathrm{Tr}[Q\Sigma\Delta^\top\Delta\Sigma Q^\top(Q\Sigma Q^\top)^{-1}] = \mathrm{Tr}[(\Sigma^{1/2}\Delta^\top\Delta\Sigma^{1/2})(\Sigma^{1/2}Q^T(Q\Sigma^{1/2}\Sigma^{1/2}Q^T)^{-1}Q\Sigma^{1/2})] \\
&= \mathrm{Tr}[(\Sigma^{1/2}\Delta^\top\Delta\Sigma^{1/2})((Q\Sigma^{1/2})^\dagger(Q\Sigma^{1/2}))].
\end{aligned}$$

By the properties of the Moore-Penrose pseudoinverse, the matrix $(Q\Sigma^{1/2})^\dagger(Q\Sigma^{1/2})$ is a $d \times d$ orthogonal projection matrix onto the span of the r rows of $Q\Sigma^{1/2}$, i.e. we can write

$$(Q\Sigma^{1/2})^\dagger(Q\Sigma^{1/2}) = Q_\Sigma^T Q_\Sigma$$

for some $r \times d$ orthogonal matrix Q_Σ. But note that as d/r, $\frac{d^2\|\Sigma\|_F^2}{(\mathrm{Tr}(\Sigma))^2} \to \infty$ (Hanson-Wright inequality),

$$\frac{1}{\mathrm{Tr}(\Sigma)}(Q\Sigma^{1/2})(Q\Sigma^{1/2})^T \to_p I_r,$$

where $\to_p$ denotes convergence in probability. In other words, in the limit $\frac{Q\Sigma^{1/2}}{\sqrt{\mathrm{Tr}(\Sigma)}}$ is close to orthogonal with high probability, implying that its transpose approaches its pseudoinverse. Hence

$$\lim_{d/r, \frac{d^2\|\Sigma\|_F^2}{(\mathrm{Tr}(\Sigma))^2}\to\infty} \mathbb{E}[(Q\Sigma^{1/2})^\dagger(Q\Sigma^{1/2})] = \lim_{d/r, \frac{d^2\|\Sigma\|_F^2}{(\mathrm{Tr}(\Sigma))^2}\to\infty} \frac{1}{\mathrm{Tr}(\Sigma)}\mathbb{E}\mathrm{Tr}((Q\Sigma^{1/2})^T(Q\Sigma^{1/2})) = r\frac{\Sigma}{\mathrm{Tr}(\Sigma)}.$$

Hence

$$\mathbb{E}[III_A] \to r\frac{\mathrm{Tr}[\Sigma^2\Delta^\top\Delta]}{\mathrm{Tr}[\Sigma]} = r\frac{\mathrm{Tr}[\Delta\Sigma^2\Delta^\top]}{\mathrm{Tr}[\Sigma]} = r\frac{\|\Delta\Sigma\|_F^2}{\mathrm{Tr}[\Sigma]}. \tag{9}$$

Recall that on the other hand that

$$\mathbb{E}[III_B] \to \frac{r}{d}\mathrm{Tr}[\Delta\Sigma\Delta^\top].$$

Recall that we have assumed that the smallest nonzero eigenvalue of $\Delta\Sigma\Delta^\top$ is $\geq \mathrm{Tr}[\Delta\Sigma\Delta^\top]/d$. Then revisiting (9) above,

$$r\frac{\|\Delta\Sigma\|_F^2}{\mathrm{Tr}[\Sigma]} \geq \frac{r}{d}\frac{\mathrm{Tr}[\Sigma]\mathrm{Tr}[\Delta\Sigma\Delta^\top]}{\mathrm{Tr}[\Sigma]} \to \mathbb{E}[III_B] \tag{10}$$

and the asymmetry is established.

Hence $\lim_{d/r, \frac{d^2\|\Sigma\|_F^2}{(\mathrm{Tr}(\Sigma))^2}\to\infty} \mathbb{E}[III_A] \geq \lim_{d/r\to\infty} \mathbb{E}[III_B]$, implying that freezing A to a random orthogonal matrix achieves lower mean squared error loss than freezing B.

C. Proof of Lemma 4.5: Generalization Bounds

We use the following bound on the generalization error is from (Xu & Raginsky, 2017), specialized to our setting and notation.

Theorem C.1 (specialized from (Xu & Raginsky, 2017))**.** *Denote by $\mathcal{A}$ a LoRA-based fine-tuning algorithm, which outputs $\Delta\mathbf{W}$ given S_n. Assume that $\ell^{\mathbf{W},\mathbf{b}}(\Delta\mathbf{W}, \tilde{Z})$ is σ-sub-Gaussian under $(\Delta\mathbf{W}, \tilde{Z}) \sim P_{\Delta\mathbf{W}|\mathbf{W},\mathbf{b}} \times \mu$. Then,*

$$|\mathrm{gen}(\mu, \mathcal{A})| \leq \sqrt{\frac{2\sigma^2}{n}\mathsf{I}(\Delta\mathbf{W}; S_n|\mathcal{A}, \mathbf{W})}. \tag{11}$$

We consider the case of tuning B only first. Applying the above theorem, note that here

$$\begin{aligned}\mathsf{I}(\Delta\mathbf{W}; S_n|\mathcal{A}_B, \mathbf{W}) &= \mathsf{I}(\{B_iQ_i\}_{i\in\mathcal{I}}; S_n|\mathcal{A}_B, \mathbf{W}) \\ &= \mathsf{I}(\{B_i\}_{i\in\mathcal{I}}; S_n|\mathcal{A}_B, \mathbf{W}),\end{aligned}$$

where we have used the data processing inequality (DPI), noting that the Q_i are here considered orthogonal fixed constant matrices as they are not trained, hence the mapping from B_i to B_iQ_i is invertible.

We can now bound this expression as

$$\begin{aligned}\mathsf{I}(\{B_i\}_{i\in\mathcal{I}}; S_n|\mathcal{A}_B, \mathbf{W}) &\leq H(\{B_i\}_{i\in\mathcal{I}}) \\ &\leq qr\sum_{i\in\mathcal{I}} d_{out}^{(i)},\end{aligned}$$

where we have noted that mutual information is upper bounded by discrete entropy, and entropy in turn is upper bounded by the uniform distribution over its possible support set (q bits in each of $r\sum_{i\in\mathcal{I}} d_{out}^{(i)}$ dimensions). The bounds for the other two algorithms are similar.

Table 8. Hyper-parameter setup for GLUE tasks.

Dataset	learning rate	batch size	# epochs	γ	t_i	Δ_T	t_f
MNLI	5×10^{-4}	48	25	0.1	6000	100	50000
SST-2	5×10^{-4}	48	25	0.1	6000	100	50000
MRPC	5×10^{-4}	48	15	0.1	5000	100	85000
CoLA	5×10^{-4}	48	15	0.1	5000	100	85000
QNLI	5×10^{-4}	48	15	0.1	5000	100	85000
RTE	5×10^{-4}	48	15	0.1	5000	100	85000
STS-B	5×10^{-4}	48	15	0.1	5000	100	85000

D. Natural Language Understanding Training Details

E. Text Generation Training Details

The configuration of our experiments on text generation is listed in Table 10.

Table 9. Hyper-parameter setup for summarization tasks.

Dataset	learning rate	batch size	# epochs	γ	t_i	Δ_T	t_f
XSum	5×10^{-4}	48	25	0.1	6000	100	50000
CNN/DailyMail	5×10^{-4}	48	15	0.1	5000	100	85000

F. Llama-2 Training Details

Table 10. Hyper-parameter setup for summarization tasks.

Dataset	learning rate	batch size	# epochs	γ	t_i	Δ_T	t_f
Alpaca	5×10^{-4}	48	25	0.1	6000	100	50000

G. Additional Language Results

See Table 11 for additional results.

Table 11. Different adaptation methods on the GLUE benchmark. We report the overall (matched and mismatched) accuracy for MNLI, Matthew's correlation coefficient for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. Higher is better for all metrics.

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
LoRA ($r = 8$)	0.8M	$90.3_{\pm.07}$	$95.6_{\pm0.36}$	$90.3_{\pm0.85}$	$64.4_{\pm1.8}$	$94.0_{\pm0.29}$	$84.1_{\pm0.96}$	$91.5_{\pm0.16}$	87.2
AdaLoRA	2.5%	$90.4_{\pm.37}$	$95.9_{\pm.13}$	$90.1_{\pm.54}$	$67.5_{\pm1.3}$	$94.7_{\pm.22}$	$85.4_{\pm.20}$	$91.3_{\pm1.0}$	87.9
$(IA)^3$	0.7%	$90.0_{\pm.21}$	$95.4_{\pm.17}$	$83.7_{\pm.13}$	$57.6_{\pm.67}$	$93.7_{\pm.07}$	$70.3_{\pm1.5}$	$87.0_{\pm0.4}$	82.5
$\hat{B}_0 A_V$ ($r = 8$)	0.3M	$90.1_{\pm.09}$	$95.5_{\pm.01}$	$90.8_{\pm.24}$	$63.8_{\pm4.2}$	$94.2_{\pm.11}$	$83.3_{\pm1.7}$	$91.3_{\pm.24}$	87.0
$\hat{B}_0 A_{rand}$ ($r = 8$)	0.3M	$90.1_{\pm.19}$	$95.8_{\pm.29}$	$89.7_{\pm.13}$	$67.5_{\pm1.2}$	$94.0_{\pm.27}$	$82.8_{\pm1.5}$	$91.9_{\pm.26}$	87.4
$\hat{B}_0 A_{km}$ ($r = 8$)	0.3M	$90.1_{\pm.17}$	$95.6_{\pm.17}$	$90.6_{\pm.32}$	$67.3_{\pm2.3}$	$93.4_{\pm.61}$	$82.4_{\pm1.4}$	$91.2_{\pm.29}$	87.2
$B_U \hat{A}_0$ ($r = 8$)	0.3M	$89.3_{\pm.18}$	$95.4_{\pm0.13}$	$88.8_{\pm0.70}$	$59.1_{\pm0.48}$	$93.8_{\pm0.15}$	$77.5_{\pm2.7}$	$90.7_{\pm.27}$	94.9
$B_{rand} \hat{A}_0$ ($r = 8$)	0.3M	$90.3_{\pm.18}$	$95.5_{\pm.66}$	$89.3_{\pm.09}$	$58.7_{\pm2.5}$	$93.8_{\pm.21}$	$77.1_{\pm1.3}$	$90.7_{\pm.31}$	85.1
$B_{km} \hat{A}_0$ ($r = 8$)	0.3M	$34.5_{\pm1.6}$	$95.2_{\pm.34}$	$89.3_{\pm.11}$	$0.0_{\pm0.0}$	$93.0_{\pm.38}$	$47.3_{\pm.0}$	$91.2_{\pm.24}$	64.4
$\hat{B}_0 A_V$ ($r = 16$)	0.8M	$90.2_{\pm.17}$	$95.8_{\pm.20}$	$90.1_{\pm.56}$	$67.8_{\pm.49}$	$94.5_{\pm.07}$	$82.8_{\pm.42}$	$91.6_{\pm.21}$	87.5
$\hat{B}_0 A_{rand}$ ($r = 16$)	0.8M	$90.1_{\pm.20}$	$96.1_{\pm.18}$	$90.7_{\pm.90}$	$66.1_{\pm2.6}$	$94.4_{\pm.10}$	$84.1_{\pm.96}$	$91.2_{\pm.42}$	87.5
$\hat{B}_0 A_{km}$ ($r = 16$)	0.8M	$90.3_{\pm.06}$	$95.6_{\pm.01}$	$91.1_{\pm.32}$	$65.2_{\pm2.1}$	$94.5_{\pm.02}$	$81.7_{\pm1.8}$	$91.2_{\pm.39}$	87.1
$B_U \hat{A}_0$ ($r = 16$)	0.8M	$90.3_{\pm.07}$	$95.4_{\pm.57}$	$90.4_{\pm1.1}$	$60.7_{\pm.14}$	$94.1_{\pm.30}$	$80.1_{\pm1.2}$	$90.8_{\pm.29}$	86.0
$B_{rand} \hat{A}_0$ ($r = 16$)	0.8M	$89.9_{\pm.19}$	$95.6_{\pm.64}$	$90.2_{\pm0.23}$	$60.3_{\pm3.3}$	$93.9_{\pm0.25}$	$80.4_{\pm0.21}$	$90.9_{\pm0.13}$	85.9
$B_{km} \hat{A}_0$ ($r = 16$)	0.8M	$89.2_{\pm.03}$	$95.2_{\pm.29}$	$90.6_{\pm0.65}$	$40.4_{\pm35.}$	$93.1_{\pm0.23}$	$70.3_{\pm0.19}$	$91.4_{\pm0.26}$	81.5
$\hat{B}_0 \hat{A}_V$ ($r = 8$)	0.8M	$90.4_{\pm.11}$	$95.9_{\pm0.18}$	$90.7_{\pm0.84}$	$64.0_{\pm0.50}$	$94.4_{\pm0.16}$	$84.1_{\pm0.15}$	$91.8_{\pm00.15}$	87.3
$\hat{B}_0 \hat{A}_{rand}$ ($r = 8$)	0.8M	$90.4_{\pm.15}$	$96.0_{\pm.63}$	$91.5_{\pm1.1}$	$64.1_{\pm0.67}$	$94.5_{\pm0.11}$	$85.6_{\pm0.96}$	$92.0_{\pm0.31}$	87.7
$\hat{B}_0 \hat{A}_{km}$ ($r = 8$)	0.8M	$90.3_{\pm.07}$	$95.6_{\pm0.36}$	$90.3_{\pm0.85}$	$64.4_{\pm1.8}$	$94.0_{\pm0.29}$	$84.1_{\pm0.96}$	$91.5_{\pm0.16}$	87.2
$\hat{B}_U \hat{A}_0$ ($r = 8$)	0.8M	$90.3_{\pm.11}$	$96.1_{\pm.18}$	$91.7_{\pm0.33}$	$64.9_{\pm1.5}$	$94.7_{\pm0.33}$	$84.8_{\pm0.96}$	$91.9_{\pm0.19}$	87.8
$\hat{B}_{rand} \hat{A}_0$ ($r = 8$)	0.8M	$90.3_{\pm.27}$	$96.0_{\pm.26}$	$90.8_{\pm0.51}$	$66.0_{\pm1.01}$	$94.5_{\pm0.38}$	$83.6_{\pm1.5}$	$92.0_{\pm0.18}$	87.6
$\hat{B}_{km} \hat{A}_0$ ($r = 8$)	0.8M	$35.5_{\pm1.6}$	$95.6_{\pm.65}$	$90.0_{\pm0.46}$	$21.3_{\pm36.}$	$93.8_{\pm0.01}$	$57.4_{\pm0.17}$	$91.6_{\pm0.43}$	69.3

H. Additional Vision Transformers and Generalization Results

Table 12 displays a more fine-grained version of Table 4 in the main text, and presents results for each out-of-distribution environment independently, in which it is easier to appreciate the benefits of only updating B in terms of out-of-domain performance. Additional results for TerraIncognita, as well as generalization results, can be found in Table 13 and Table 14, respectively. TerraIncognita seems to be a particularly challenging dataset to which low-rank adapters struggle to fit; the most effective method, in this case, appears to be full fine-tuning. In terms of generalization, we can observe that fine-tuning only a single adapter matrix generally results in a lower difference between training set and test set accuracy compared to standard LoRA for all datasets.

Table 12. DomainBed results (mean accuracy and standard deviation in %). ID and OOD denote in-domain and out-of-domain generalization, respectively.

Method	# Trainable Parameters (% full ViT params)	VLCS				PACS				OfficeHome			
		Caltech101 (OOD)	LabelMe (ID)	SUN09 (OOD)	VOC2007 (OOD)	Art (OOD)	Cartoon (ID)	Photo (OOD)	Sketch (OOD)	Art (OOD)	Clipart (ID)	Product (OOD)	Photo (OOD)
$\hat{B}A_{rand}$ ($r = 8$)	0.16M-0.2M (0.18-0.29%)	**93.19**$_{\pm2.27}$	$77.40_{\pm2.30}$	**61.52**$_{\pm1.50}$	$72.72_{\pm1.18}$	$81.22_{\pm1.40}$	$92.45_{\pm2.68}$	$96.07_{\pm0.86}$	$40.37_{\pm0.83}$	$73.59_{\pm0.59}$	$77.66_{\pm0.89}$	$78.02_{\pm0.14}$	$81.55_{\pm0.24}$
$\hat{B}A_{rand}$ ($r = 16$)	0.3M-0.4M (0.36-0.46%)	$91.57_{\pm0.81}$	**79.10**$_{\pm1.41}$	$60.97_{\pm2.44}$	**73.66**$_{\pm0.46}$	**84.36**$_{\pm0.54}$	$93.52_{\pm0.20}$	**97.07**$_{\pm0.47}$	$39.87_{\pm0.99}$	**73.64**$_{\pm0.40}$	$77.63_{\pm0.84}$	**78.07**$_{\pm0.22}$	**81.85**$_{\pm0.36}$
$B_{rand}\hat{A}$ ($r = 8$)	0.16M-0.2M (0.18-0.29%)	$87.18_{\pm0.77}$	$76.71_{\pm0.93}$	$59.89_{\pm1.79}$	$70.44_{\pm0.10}$	$77.05_{\pm0.74}$	$92.02_{\pm1.07}$	$92.06_{\pm0.34}$	$29.65_{\pm1.31}$	$68.36_{\pm0.28}$	$72.36_{\pm0.69}$	$74.00_{\pm0.31}$	$78.63_{\pm0.45}$
$B_{rand}\hat{A}$ ($r = 16$)	0.3M-0.4M (0.36-0.46%)	$89.28_{\pm2.51}$	$78.03_{\pm1.23}$	$60.44_{\pm1.84}$	$70.81_{\pm0.36}$	$81.43_{\pm0.92}$	$93.87_{\pm0.73}$	$95.63_{\pm0.13}$	$35.02_{\pm0.86}$	$71.64_{\pm0.24}$	$73.77_{\pm1.13}$	$75.46_{\pm0.25}$	$80.31_{\pm0.39}$
LoRA ($r = 8$)	0.3M-0.4M (0.35-0.46%)	$44.59_{\pm1.96}$	$73.51_{\pm0.62}$	$60.44_{\pm2.86}$	$64.26_{\pm1.07}$	$81.41_{\pm0.70}$	**94.94**$_{\pm0.56}$	$95.43_{\pm0.54}$	$49.90_{\pm1.51}$	$70.44_{\pm0.46}$	$78.54_{\pm1.49}$	$73.99_{\pm0.64}$	$78.95_{\pm0.10}$
Linear Probing	0.004M (0.00%)	$90.65_{\pm2.51}$	$75.58_{\pm1.66}$	$53.74_{\pm0.27}$	$70.71_{\pm0.35}$	$67.66_{\pm0.63}$	$81.62_{\pm0.34}$	$88.80_{\pm1.43}$	$28.72_{\pm1.70}$	$64.56_{\pm0.23}$	$58.38_{\pm0.76}$	$66.97_{\pm0.43}$	$74.23_{\pm.001}$
Full FT	86.4M (100%)	$70.57_{\pm15.13}$	$76.21_{\pm1.95}$	$57.14_{\pm1.46}$	$66.90_{\pm2.72}$	$75.52_{\pm2.89}$	$98.15_{\pm0.56}$	$89.54_{\pm1.88}$	**59.63**$_{\pm2.53}$	$58.38_{\pm0.64}$	**80.67**$_{\pm1.22}$	$63.05_{\pm0.85}$	$68.27_{\pm0.43}$

Table 13. TerraIncognita results (mean accuracy and standard deviation in %). All methods were trained for 20,000 steps.

Method	# Trainable Parameters (% full ViT params)	TerraIncognita L100 (OOD)	TerraIncognita L38 (ID)	TerraIncognita L43 (OOD)	TerraIncognita L46 (OOD)
$\hat{B}A_{rand}$ $(r=8)$	0.16M-0.2M (0.18-0.29%)	$16.59_{\pm 2.59}$	$79.88_{\pm 0.45}$	$6.46_{\pm 1.25}$	$10.96_{\pm 0.52}$
$\hat{B}A_{rand}$ $(r=16)$	0.3M-0.4M (0.36-0.46%)	$14.14_{\pm 1.45}$	$80.48_{\pm 0.99}$	$7.74_{\pm 0.26}$	$11.09_{\pm 0.76}$
$B_{rand}\hat{A}$ $(r=8)$	0.16M-0.2M (0.18-0.29%)	$12.82_{\pm 0.84}$	$78.65_{\pm 0.57}$	$3.42_{\pm 0.81}$	$7.24_{\pm 1.36}$
$B_{rand}\hat{A}$ $(r=16)$	0.3M-0.4M (0.36-0.46%)	$17.58_{\pm 1.01}$	$78.89_{\pm 0.55}$	$8.41_{\pm 1.88}$	$7.62_{\pm 0.56}$
LoRA $(r=8)$	0.3M-0.4M (0.35-0.46%)	$\mathbf{41.36_{\pm 2.94}}$	$87.33_{\pm .13}$	$13.48_{\pm 2.19}$	$7.76_{\pm 1.69}$
Linear Probing	0.004M (0.00%)	$13.82_{\pm .20}$	$69.82_{\pm 0.36}$	$10.06_{\pm .45}$	$13.90_{\pm .49}$
Full FT	86.4M (100%)	$38.33_{\pm 6.50}$	$\mathbf{95.05_{\pm .31}}$	$\mathbf{14.18_{\pm 2.33}}$	$\mathbf{19.50_{\pm 1.53}}$

Table 14. Generalization results (train set - test set accuracy in %) for DomainBed.

Method	# Trainable Parameters (% full ViT params)	VLCS Caltech101 (OOD)	VLCS LabelMe (ID)	VLCS SUN09 (OOD)	VLCS VOC2007 (OOD)	PACS Art (OOD)	PACS Cartoon (ID)	PACS Photo (OOD)	PACS Sketch (OOD)	OfficeHome Art (OOD)	OfficeHome Clipart (ID)	OfficeHome Product (OOD)	OfficeHome Photo (OOD)	TerraIncognita L100 (OOD)	TerraIncognita L38 (ID)	TerraIncognita L43 (OOD)	TerraIncognita L46 (OOD)
$\hat{B}A_{rand}$ $(r=8)$	0.2M-M (0.29-0.%)	$-1.72_{\pm 2.24}$	$11.82_{\pm 1.21}$	$28.09_{\pm 2.04}$	$16.98_{\pm 0.74}$	$15.82_{\pm 0.68}$	$3.83_{\pm 0.70}$	$0.83_{\pm 0.30}$	$57.34_{\pm 0.89}$	$15.94_{\pm 0.28}$	$11.87_{\pm 1.14}$	$11.51_{\pm 0.47}$	$7.97_{\pm 0.56}$	$64.20_{\pm 2.58}$	$0.91_{\pm 0.43}$	$74.33_{\pm 1.26}$	$69.82_{\pm 0.53}$
$\hat{B}A_{rand}$ $(r=16)$	0.3M-0.4M (0.36-0.46%)	$-2.48_{\pm 0.69}$	$9.99_{\pm 1.44}$	$28.11_{\pm 2.74}$	$15.43_{\pm 0.70}$	$12.92_{\pm 0.87}$	$3.76_{\pm 0.40}$	$0.22_{\pm 0.67}$	$57.42_{\pm 0.62}$	$16.22_{\pm 0.93}$	$12.25_{\pm 1.23}$	$11.81_{\pm 0.34}$	$8.19_{\pm 0.87}$	$66.62_{\pm 1.54}$	$0.28_{\pm 1.18}$	$73.02_{\pm 0.24}$	$69.67_{\pm 0.56}$
$B_{rand}\hat{A}$ $(r=8)$	0.2M-M (0.29-0.%)	$0.19_{\pm 0.86}$	$10.66_{\pm 0.86}$	$27.48_{\pm 1.86}$	$16.93_{\pm 0.19}$	$19.79_{\pm 0.66}$	$4.81_{\pm 0.99}$	$4.78_{\pm 0.29}$	$67.19_{\pm 1.34}$	$17.73_{\pm 0.30}$	$13.73_{\pm 0.86}$	$12.08_{\pm 0.42}$	$7.45_{\pm 0.65}$	$65.86_{\pm 0.64}$	$0.04_{\pm 0.60}$	$75.27_{\pm 0.50}$	$71.45_{\pm 1.17}$
$B_{rand}\hat{A}$ $(r=16)$	0.3M-0.4M (0.36-0.46%)	$-1.50_{\pm 2.88}$	$9.75_{\pm 0.85}$	$27.34_{\pm 2.07}$	$16.97_{\pm 0.61}$	$15.89_{\pm 0.96}$	$3.44_{\pm 0.54}$	$1.69_{\pm 0.30}$	$62.30_{\pm 0.83}$	$15.20_{\pm 0.53}$	$13.07_{\pm 1.30}$	$11.38_{\pm 0.38}$	$6.53_{\pm 0.64}$	$62.17_{\pm 1.41}$	$0.86_{\pm 0.96}$	$71.34_{\pm 1.91}$	$72.13_{\pm 0.15}$
LoRA $(r=8)$	0.3M-0.4M (0.35-0.46%)	$52.94_{\pm 1.48}$	$24.03_{\pm 0.16}$	$37.10_{\pm 3.25}$	$33.28_{\pm 1.64}$	$18.23_{\pm 0.74}$	$4.70_{\pm 0.57}$	$4.22_{\pm 0.43}$	$49.74_{\pm 1.44}$	$26.07_{\pm 0.39}$	$17.97_{\pm 1.80}$	$22.53_{\pm 0.63}$	$17.57_{\pm 0.23}$	$47.53_{\pm 2.80}$	$1.56_{\pm 0.24}$	$75.41_{\pm 2.29}$	$81.12_{\pm 1.73}$
Linear Probing	0.004M (0.00%)	$-12.03_{\pm 2.11}$	$3.04_{\pm 1.38}$	$24.88_{\pm 0.47}$	$7.91_{\pm 0.79}$	$17.18_{\pm 0.13}$	$3.22_{\pm 0.40}$	$-3.96_{\pm 1.90}$	$56.13_{\pm 1.33}$	$6.02_{\pm 0.21}$	$12.20_{\pm 1.03}$	$3.61_{\pm 0.51}$	$-3.65_{\pm 0.19}$	$55.17_{\pm 0.28}$	$-0.82_{\pm 0.31}$	$58.94_{\pm 0.52}$	$55.10_{\pm 0.52}$
Full FT	86.4M (100%)	$29.03_{\pm 15.27}$	$23.40_{\pm 2.05}$	$42.47_{\pm 1.83}$	$32.70_{\pm 2.27}$	$24.41_{\pm 2.94}$	$1.78_{\pm 0.54}$	$10.38_{\pm 1.90}$	$40.30_{\pm 2.49}$	$40.23_{\pm 0.48}$	$17.94_{\pm 1.36}$	$35.56_{\pm 1.02}$	$30.35_{\pm 0.53}$	$59.84_{\pm 6.53}$	$3.12_{\pm 0.26}$	$83.99_{\pm 2.31}$	$78.67_{\pm 1.47}$