How Much Adaptation Space Does LoRA Need?

Anonymous authors
Paper under double-blind review

Abstract

Low-rank adaptation (LoRA) and its variants provide memory- and compute-efficient alternatives to full-parameter fine-tuning. Still, it remains unclear how aggressively we can restrict their adaptation space without sacrificing performance and generalization. We study this question through structured extensions of LoRA—Cheap LoRA (cLA), which trains a single low-rank factor while fixing the other in a compute-efficient way, and c^3LA, a chained-circulant variant of cLA. We interpret cLA as a structured form of asymmetric LoRA and, more broadly, as a controlled column-subspace restriction of full fine-tuning. We derive information-theoretic generalization error bounds for these restricted adaptation classes along with their base variants, providing a theoretical perspective on their generalization behavior. Empirically, *we evaluate **11 fine-tuning methods** across **10 pre-trained models and 14 datasets*** using downstream performance, generalization, loss-landscape, and spectral analyses. Across tasks, cLA and c^3LA remain competitive with baselines matched on trainable parameters while additionally reducing training time relative to LoRA by up to 14.4% and peak GPU memory by up to 11%. Our results suggest that substantially restricting the adaptation subspace can preserve effective transfer while reducing fine-tuning cost as an additional benefit.

1 Introduction

Full fine-tuning (FFT) (Devlin et al., 2019) adapts all parameters of a pre-trained model to a downstream task. Still, its memory and computational requirements grow as model size increases. Parameter-efficient fine-tuning (PEFT) reduces these costs by restricting the parameters updated during adaptation. Among these approaches, Low-Rank Adaptation (LoRA) (Hu et al., 2021) parameterizes weight updates as the product of two trainable low-rank factors and often achieves downstream performance comparable to FFT. Its success raises a fundamental question: *which freedoms in the adaptation space are necessary for effective fine-tuning?*

We use *adaptation space* to refer to the family of weight updates permitted by a fine-tuning parameterization. Trainable-parameter count alone does not fully characterize this family; parameterizations with similar budgets can permit updates in different directions and distribute them differently across the pre-trained weights. Existing LoRA variants modify these freedoms in several ways. Asymmetric LoRA (Zhu et al., 2024) fixes one factor, CoLA (Xia et al., 2024) accumulates updates through sequential adapters, and RAC-LoRA (Malinovsky et al., 2024) combines asymmetric adaptation with chaining. Other approaches modify optimization or rank allocation, including LoRA+ (Hayou et al., 2024) and AdaLoRA (Zhang et al., 2023); see (Xu et al., 2026; Han et al., 2024).

These existing variants impose different structural constraints on the adaptation space. However, it is unclear how aggressively the adaptation subspace can be restricted while preserving downstream performance, how these restrictions affect generalization, or how this trade-off depends on the model, task, rank, and other hyperparameters. Existing empirical comparisons provide useful evidence (Xu et al., 2026; Shuttleworth et al., 2025), but they do not isolate the role of the adaptation subspace itself. Geometric diagnostics, such as loss-landscape and intruder-dimension analyses (Li et al., 2018; Huang et al., 2020; Shuttleworth et al., 2025), cannot consistently explain generalization behavior.

Figure 1 illustrates this limitation. Across fine-tuning strategies, the loss-landscape geometry does not necessarily align with downstream generalization. Although FFT has the sharpest minimum and lowest test accuracy, it achieves the smallest empirical generalization error, conflicting with the well-known heuristic that models with sharper minima should generalize worse Li et al. (2018); Huang et al. (2020). Thus, sharpness alone does not consistently explain differences in empirical

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

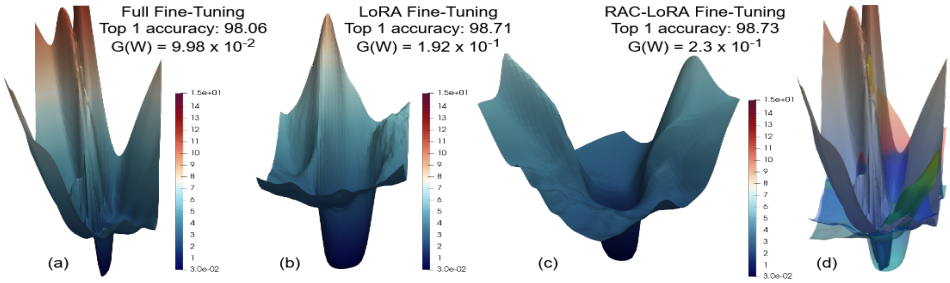


Figure 1: Loss-landscape visualizations for ViT-Base adapted to CIFAR-10 after ImageNet-21K pre-training and ImageNet-1K fine-tuning. In (d), we overlay the surfaces. The labels $\mathcal{G}(\mathbf{W})$ denote empirical estimates of the generalization error; see Definition 1 and Table 15. FFT has the narrowest minimum but the smallest empirical generalization error compared to LoRA and RAC-LoRA. The comparison illustrates that sharper minima do not necessarily correspond to worse generalization, motivating a direct study of the structural restrictions imposed on the adaptation space.

generalization across fine-tuning strategies. This motivates a more controlled question: *how does restricting the adaptation space affect performance and generalization?*

We study this question by introducing *Cheap LoRA (cLA)*, a coordinate-structured instance of asymmetric LoRA. cLA fixes one factor to a column selector and trains the other, restricting each weight update to selected columns of the pre-trained matrix. This construction makes the restriction explicit; rather than learning both low-rank factors, cLA learns an unrestricted update within a prescribed column block. It therefore shares its representable update space with partial connection adaptation (PaCA) (Woo et al., 2025), which directly updates selected columns. This connection allows us to examine column-restricted adaptation in both LoRA-style and direct-update parameterizations.

We further introduce *circulant chain of Cheap LoRA (c^3LA)*, which shifts the selected columns across training stages. By merging each stage's update into the model before moving to the next column block, this construction expands cumulative coverage while keeping the number of simultaneously trainable adapter parameters fixed. Randomized counterparts of our proposed methods allow us to distinguish the effect of the restriction itself from that of a particular column selection.

Our analysis combines theoretical comparisons with a broad empirical evaluation. We derive information-theoretic generalization bounds for the considered fine-tuning parameterizations, making explicit their dependence on quantities including adapter rank, chain length, parameter precision, and sample size; see §3. Under the stated assumptions, the bounds for the column-restricted constructions match those of their corresponding asymmetric parameterizations. These results characterize capacity-related terms in the bounds, rather than establish an ordering of actual generalization errors or downstream performance.

Empirically, we compare 11 fine-tuning methods across 10 pre-trained models and 14 datasets spanning language, vision, code generation, and logical reasoning. We evaluate downstream performance alongside training time, throughput, and peak GPU memory, and examine empirical generalization errors and geometric diagnostics as complementary measurements. Across the evaluated settings, cLA and c^3LA often approach LoRA's downstream performance while reducing training costs, although the trade-off varies across models, tasks, and hyperparameters. Our findings support a qualified conclusion: the full flexibility of two-factor LoRA is not always necessary for effective adaptation, but the structure and scheduling of the permitted updates matter. The resulting perspective positions adaptation-space structure, alongside parameter count, as an important consideration in the design and evaluation of efficient fine-tuning methods.

2 Fine-Tuning Through Structured Adaptation Spaces

While the memory costs of full fine-tuning (FFT) become increasingly impractical as model size and deployment multiplicity grow, PEFT methods restrict adaptation to a low-rank parameterization and reduce trainable parameters. We organize this section around the resulting adaptation spaces, from unrestricted full-parameter adaptation or FFT, to low-rank LoRA adaptation, and finally to structured restrictions of the LoRA space. This perspective motivates our column-restricted extensions, which further constrain where adaptation occurs while retaining LoRA's parameter-efficient structure.

2.1 FULL FINE-TUNING (FFT) AND LORA

Without loss of generality, consider a L-layer fully-connected neural network whose layers are, $\{W^i\}_{i=1}^L$, where $W^i \in \mathbb{R}^{n_i \times m_i}$ are trainable weights. Let $x \in \mathbb{R}^{m_1}$ be the input and $\mathbf{W} = [W^1, ..., W^L]$. The network $f_{\mathbf{W}}(\cdot) : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ is of the form:

$$f_{\mathbf{W}}(x) = \sigma_L(W^L \dots (\sigma_2(W^2 \sigma_1(W^1(x)) \dots)), \tag{1}$$

where $\sigma^i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is a nonlinear activation function for the i^{th} layer. Let $\mathbf{W}_0 = [W_0^1, \cdots, W_0^L]$ denote the pre-trained weights and $N := \{(x_i', y_i')\}_{i=1}^{|N|}$ denotes a downstream training dataset. We define the corresponding loss function $\mathcal{L}(\mathbf{W}) := \frac{1}{|N|} \sum_{i=1}^{|N|} \ell(f_{\mathbf{W}}(x_i), y_i)$, where $\ell(\cdot) : \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}} \to \mathbb{R}$ is a measure of the distance between $f_{\mathbf{W}}(x_i)$ and y_i.

FFT. Given the pre-trained weights $\mathbf{W}_0$, FFT learns layerwise updates $\Delta\hat{\mathbf{W}}$ by solving:

$$\Delta\hat{\mathbf{W}} \approx \text{argmin}_{\Delta\mathbf{W}} \mathcal{L}(\mathbf{W}_0 + \Delta\mathbf{W}) \tag{2}$$

yielding the fine-tuned model $f_{\mathbf{W}_0 + \Delta\hat{\mathbf{W}}}$. FFT can be prohibitively expensive for large language models. In contrast, parameter-efficient fine-tuning (PEFT) trains orders of magnitude fewer parameters while often attaining performance comparable to FFT Houlsby et al. (2019); Xu et al. (2026).

LoRA Hu et al. (2021) is a popular PEFT method that replaces the layer-wise updates ΔW^i with a low-rank representation $B^i A^i$, such that $B^i \in \mathbb{R}^{n_i \times r}$, $A^i \in \mathbb{R}^{r \times m_i}$, $r \ll \min(m_i, n_i)$ for all $i \in [L]$. Denote $\mathbf{W}_0 + \frac{\alpha}{r}\mathbf{BA} := [W_0^1 + \frac{\alpha}{r} B^1 A^1, \cdots, W_0^L + \frac{\alpha}{r} B^L A^L]$, where $\alpha > 0$ is a scaling factor. LoRA initializes $B^i = 0$, $A^i \sim \mathcal{N}(0, 0.02^2)$, and solves:

$$(\hat{\mathbf{B}}, \hat{\mathbf{A}}) \approx \text{argmin}_{\mathbf{B},\mathbf{A}} \mathcal{L}\left(\mathbf{W}_0 + \frac{\alpha}{r}\mathbf{BA}\right) \tag{3}$$

to obtain B^i, A^i for each tuned layer. LoRA may not need to be applied to all layers; some layers can remain frozen. LoRA substantially reduces trainable parameters, saves training time, and the update $\mathbf{BA}$ can be merged into the base weights to avoid additional inference latency. LoRA is compute- and storage-efficient, but yields worse generalization than FFT Shuttleworth et al. (2025).

2.2 EVOLUTION OF LORA

Rather than treating LoRA as a fixed parameterization, recent work has explored how its adaptation space can be modified through asymmetric updates, structured parameter sharing, and adaptive rank allocation. Asymmetric LoRA Zhu et al. (2024) restricts optimization to one factor, while CoLA (Xia et al., 2024) and RAC Malinovsky et al. (2024) introduce additional structure through chaining or parameter coupling; LoRA+ Hayou et al. (2024) modifies the optimization of the two factors; see further details in §A. These approaches do not necessarily demonstrate that the adaptation space itself can be reduced and leave open a more fundamental question—*how far can the adaptation space itself be restricted while retaining useful transfer?* This question motivates our column-restricted constructions, which impose an explicit structural constraint on the LoRA update rather than merely changing how its factors are optimized.

2.3 FROM LORA TO STRUCTURED ADAPTATION

As model dimensionality and adaptation costs continue to grow, a natural question is whether the LoRA adaptation space can be restricted further without sacrificing downstream performance. Empirical evidence that training B while fixing A can be effective Zhu et al. (2024), together with structured chaining approaches Xia et al. (2024); Malinovsky et al. (2024), motivates two simple, interpretable, and efficient variants. Specifically, we restrict each BA update to r selected columns, controlling where adaptation occurs while retaining LoRA's factorization.

(*i*) **Cheap LoRA (cLA).** Building on asymmetric LoRA (Zhu et al., 2024), cLA trains B^i while fixing $A^i = [I_r \mid \mathbf{0}_{r \times (m_i - r)}] \in \mathbb{R}^{r \times m_i}$. For each layer, with $W^i \in \mathbb{R}^{n_i \times m_i}$, and $B^i \in \mathbb{R}^{n_i \times r}$, we have $\Delta W^i = B^i \left[I_r | \mathbf{0}_{r \times (m_i - r)}\right] = \left[B^i | \mathbf{0}_{n_i \times (m_i - r)}\right]$. Let $\mathbf{B}^c$ denote the collection of these layerwise updates. Then cLA solves:

$$\hat{\mathbf{B}}^c \approx \text{argmin}_{\mathbf{B}^c} \mathcal{L}\left(\mathbf{W}_0 + \frac{\alpha}{r}\mathbf{B}^c\right), \tag{4}$$

We consider two instantiations of the fixed factor, deterministic (*cLA*) and random (*random-cLA*), where we randomly permute the columns of A^i on initialization, restricting the adaptation space

from that of a particular column selection. cLA is not just Asym LoRA with fewer parameters; conceptually, it is a variant of Asym LoRA whose update is restricted to a particular coordinate subspace of the pretrained weight space.

(*ii*) **Circulant Chain of Cheap LoRA (c^{3}LA).** Motivated by repeated adapter initializations in CoLA and RAC-LoRA (Xia et al., 2024; Malinovsky et al., 2024), we extend cLA through structured chaining. This method shifts the identity I_r in each matrix $\left[I_r|\mathbf{0}_{r\times(m_i-r)}\right]$ by r columns to the right. That is, starting with $\left[I_r|\mathbf{0}_{r\times(m_i-r)}\right]$, the next chain is $\left[\mathbf{0}_{r\times r}\;\middle|\; I_r \;\middle|\; \mathbf{0}_{r\times(m_i-2r)}\right]$, and so on. For $j \in [k]$, let $\mathbf{B}^{c^3}$ denote c^3LA's update and denote $\mathbf{W}^{(k,B^{c^3})} := \mathbf{W}_0 + \sum_{j=1}^{k} \frac{\alpha}{r}\hat{\mathbf{B}}^{c^3,j}$, and $\mathbf{W}^{(0,B^{c^3})} = \mathbf{W}_0$. Then c^3LA of chain length k solves:

$$\hat{\mathbf{B}}^{c^3,j} \approx \operatorname{argmin}_{\mathbf{B}^{c^3,j}} \mathcal{L}\left(\mathbf{W}^{(j-1,B^{c^3})} + \frac{\alpha}{r}\mathbf{B}^{c^3,j}\right), \tag{5}$$

to obtain the fine-tuned model $f_{\mathbf{W}^{(k,B^{c^3})}}$ for a chain of length k. Given sufficient epochs and chain length, this ensures we can update all elements in each W_0. Proposition 1 formalizes this.

Proposition 1. *Let $k \in \mathbb{N}$ be such that $d_{in} = kr$. Let E be the total number of epochs used in c^3LA fine-tuning. Then by creating a new chain in every $\lfloor \frac{E}{k} \rfloor$ epochs, c^3LA updates each element in W_0.*

The intuition behind c^3LA goes beyond chaining cheap LoRA modules; its structured shifts expand the representational capacity of the learned B matrices. We provide pseudocode of our variants in §C.

Connection with partial connection adaptation (PaCA) Woo et al. (2025) can be viewed as a special case of our adaptation-space restriction framework. With slight abuse of notation, for a pretrained weight matrix $\mathbf{W}$, our general parameterization restricts the update to a prescribed column subspace, with the LoRA update taking the form

$$\mathbf{W}' = \mathbf{W} + \mathbf{B}\mathbf{A}P_{\mathcal{S}},$$

where $P_{\mathcal{S}}$ selects the columns accessible to adaptation. Standard LoRA is recovered when $P_{\mathcal{S}} = I$, while cLA and its variants impose structured restrictions on $P_{\mathcal{S}}$. PaCA corresponds to the same column-restricted construction without the low-rank constraint. In PaCA, the selected columns are directly trainable, which can be represented as the full-rank boundary case of our parameterization. Thus, PaCA is not a separate adaptation principle, but a particular choice of adaptation space and update parameterization within the broader framework.

This perspective also clarifies the computational distinction. PaCA directly updates the selected columns of W and therefore avoids evaluating a LoRA residual, computing Wx rather than $Wx + B(Ax)$. In contrast, cLA retains the parameter-efficient LoRA parameterization while imposing the same type of column-subspace restriction through the trainable factors. The distinction is therefore primarily one of update parameterization and computational cost, rather than adaptation geometry.

The unified view allows results developed for our restricted-LoRA framework to extend naturally to PaCA whenever the corresponding assumptions depend only on the restricted adaptation space. In particular, the generalization analysis in §3.2 applies to PaCA as its full-rank special case, while optimization and convergence results for restricted LoRA provide a common analytical framework for both constructions. Conversely, PaCA's direct-update formulation motivates extensions of our framework that eliminate the LoRA forward-pass overhead; for example, the chain construction described in §B.1 can be incorporated into the corresponding column-restricted direct-update formulation. Empirically, the resulting methods exhibit similar adaptation behavior under comparable column restrictions (Table 4), supporting the view that their common structure is the restricted adaptation space rather than the particular parameterization used to realize it.

3 Theoretical Insights

Adaptation-space restrictions offer a useful lens for understanding how much freedom LoRA-based PEFT methods require for effective fine-tuning. Generalization is central to this inquiry, as successful adaptation must extend beyond fitting the downstream training data. To investigate how these restrictions enter generalization guarantees, in this section we examine generalization-error upper bounds for LoRA-based PEFT methods under an information-theoretic framework Xu & Raginsky (2017); Russo & Zou (2020). Under our layerwise training setup, in which trainable adapter parameters are updated using gradient descent (GD), we first establish capacity-based bounds for LoRA and its

Variant	Chaining?	$\mathcal{G}(\mathbf{W}_0 + \Delta\mathbf{W}) \leq \Phi_{\mathbf{W}_0} +$
LoRA Hu et al. (2021)	×	$\sqrt{\frac{2rq\sigma^2 \ln 2 \sum_{i=1}^{L}(m_i+n_i)}{\lvert N\rvert}}$
LoRA+ Hayou et al. (2024)	×	$\sqrt{\frac{2rq\sigma^2 \ln 2 \sum_{i=1}^{L}(m_i+n_i)}{\lvert N\rvert}}$
Asym-LoRA Zhu et al. (2024)	×	$\sqrt{\frac{2rq\sigma^2 \ln 2 \sum_{i=1}^{L} n_i}{\lvert N\rvert}}$
CoLA Xia et al. (2024)	✓	$\sqrt{\frac{2rq\sigma^2 k \ln 2 \sum_{i=1}^{L}(m_i+n_i)}{\lvert N\rvert}}$
RAC Malinovsky et al. (2024)	✓	$\sqrt{\frac{2rq\sigma^2 k \ln 2 \sum_{i=1}^{L} n_i}{\lvert N\rvert}}$
cLA & PaCA (This paper, Woo et al. (2025))	×	$\sqrt{\frac{2rq\sigma^2 \ln 2 \sum_{i=1}^{L} n_i}{\lvert N\rvert}}$
c^3LA	✓	$\sqrt{\frac{2rq\sigma^2 k \ln 2 \sum_{i=1}^{L} n_i}{\lvert N\rvert}}$

Table 1: **Generalization error upper bounds of LoRA variants.** The expression, $\Phi_{\mathbf{W}_0}$ is in Theorem 1. r is the adapter rank, k chain length, $|N|$ size of fine-tuned dataset, q bitwidth, (m_i, n_i) are the (input,output) dimensions for layer i. The loss, $\mathcal{L}$ is σ-sub-Gaussian (Assumption 6).

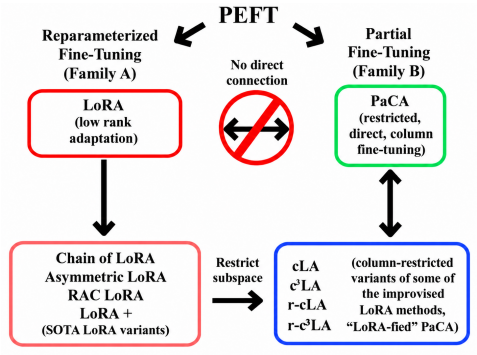


Figure 2: **Evolution of fine-tuning methods.** LoRA reparameterizes updates (**Family A**); PaCA directly fine-tunes randomly selected pretrained columns (**Family B**). Partial adaptation and restricted LoRA can be understood within a common adaptation-space framework.

variants. We then complement these bounds with a comparison between the generalization bounds of the adapted model and its pretrained and update-only components.

3.1 Setup and Generalization Error

Let $\mathcal{X} \times \mathcal{Y}$ be an input space and label space with ν distribution of pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Recall $N = \{(x_i, y_i)\}_{i=1}^{|N|}$ represents the training dataset. We assume each (x_i, y_i) is i.i.d. from ν distribution of $\mathcal{X} \times \mathcal{Y}$. Given a *hypothesis* (model), $f_{\mathbf{W}}(\cdot) : \mathcal{X} \to \mathcal{Y}$ The *true risk* of the hypothesis, $f_{\mathbf{W}}(\cdot)$ is defined as $\hat{\mathcal{L}}_{\text{global}}(\mathbf{W}) := \mathbb{E}_{X,Y\sim\nu}[\ell(f_{\mathbf{W}}(X), Y)]$. With the above setup, we define *generalization error*, which tells us how well the hypothesis, $f_{\mathbf{W}}$, generalizes from the training sample to the underlying population distribution.

Definition 1. (**Generalization Error** Xu & Raginsky (2017)) Generalization error, $\mathcal{G}(\mathbf{W})$, is the difference between a hypothesis's true risk and its empirical risk on the training dataset, i.e., $\mathcal{G}(\mathbf{W}) := \hat{\mathcal{L}}_{\text{global}}(\mathbf{W}) - \mathcal{L}(\mathbf{W})$.

3.2 On the Generalization of Different LoRA Variants

For our analysis, we made general assumptions in §D.1. Based on them, Theorem 1 upper bounds the generalization error of a fine-tuned, L-layer fully connected DNN, parameterized by $\mathbf{W}_0 + \Delta\mathbf{W}$, relative to that of either the pretrained model, $\mathbf{W}_0$, or the update-only model, $\Delta\mathbf{W}$.

Theorem 1. ***(Generalization bounds)*** *Let $f_{\boldsymbol{W}_0+\Delta\boldsymbol{W}}(x)$ be a L-layer fine-tuned DNN. Let the loss function be $L_{\mathcal{L}}$-Lipschitz continuous (Assumption 2), the vector-valued activation function, $\sigma_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$, for each layer i, be L_{σ_i}-Lipschitz continuous (Assumption 3), and the input vectors be bounded (Assumption 1). Then $|\mathcal{G}(\boldsymbol{W}_0 + \Delta\boldsymbol{W}) - \mathcal{G}(\boldsymbol{W}_0)| \leq \Phi_{\Delta\boldsymbol{W}}$, and $|\mathcal{G}(\boldsymbol{W}_0 + \Delta\boldsymbol{W}) - \mathcal{G}(\Delta\boldsymbol{W})| \leq \Phi_{\boldsymbol{W}_0}$. Consequently, $\mathcal{G}(\boldsymbol{W}_0 + \Delta\boldsymbol{W}) \leq \min\{\mathcal{G}(\boldsymbol{W}_0) + \Phi_{\Delta\boldsymbol{W}}, \mathcal{G}(\Delta\boldsymbol{W}) + \Phi_{\boldsymbol{W}_0}\}$.* [1]

Intuition behind Theorem 1. Theorem 1 compares the generalization error of the fine-tuned model to either the pretrained model or the update-only model. The correction terms, $\Phi_{\Delta\mathbf{W}}$ and $\Phi_{\mathbf{W}_0}$, depend on the Lipschitz constants of the loss and layerwise activations, and the spectral norms of $\{W_0^i, \Delta W^i\}_{i\in[L]}$. Their product-difference structure retains every mixed combination of pretrained and update weights introduced throughout the network. When $\sigma_i(0) \neq 0$, additional terms account for how these activation offsets propagate through the remaining layers. We present the complete expressions and proof in §D.1.2.

Consistency of the bounds. The correction terms in Theorem 1 vanish with the corresponding perturbation: $\Phi_{\Delta\mathbf{W}} = 0$ when $\Delta\mathbf{W} = 0$, and $\Phi_{\mathbf{W}_0} = 0$ when $\mathbf{W}_0 = 0$. Thus, Theorem 1 recovers equality when the compared models coincide; see §D.1.4.

[1] The complete expressions for $\Phi_{\Delta\mathbf{W}}$ and $\Phi_{\mathbf{W}_0}$ are given in §D.1.2.

Theorem 1 applied to attention mechanism. Theorem 1 can handle advanced architectures, such as Transformers, whose main working component is attention, by considering inputs $x \in \mathcal{X}$ after embedding. The key step is showing that multi-head attention (MHA) blocks can be expressed as compositions of linear maps and Lipschitz operators. See Theorem 4 in §D.1.5 for the full derivation.

Theorem 1 under special conditions. The generalization upper bound $\mathcal{G}(\mathbf{W}_0 + \Delta\mathbf{W})$ in Theorem 1 contains two terms: (*i*) $\mathcal{G}(\mathbf{W}_0) + \Phi_{\Delta\mathbf{W}}$ and (*ii*) $\mathcal{G}(\Delta\mathbf{W}) + \Phi_{\mathbf{W}_0}$. We can adapt some additional assumptions on loss, quantization bit-width, size of fine-tuning datasets, and layer dimensions (see §D.1.6 for the assumptions) and bound $\mathcal{G}(\mathbf{W}_0)$ and $\mathcal{G}(\Delta\mathbf{W})$ below.

(*i*) **Bounding $\mathcal{G}(\mathbf{W}_0)$.** We use the PAC-Bayes generalization bound for fine-tuning by Li & Zhang (2021); see Theorem 5 in §D.1.6. The loss function, $\mathcal{L}$, is bounded by C_2. Since $\|W_0^{(i)} - W_0^{(i)}\| = 0$, for all $i \in [L]$, in Theorem 5, we obtain $Q_i := 0$. Hence, $\mathcal{G}(\mathbf{W}_0) \leq \epsilon + C_2\sqrt{|N|^{-1}(3\ln|N|\delta^{-1} + 8)}$, holds with probability at least $1 - 2\delta$, where $\epsilon, \delta > 0$, are arbitrarily small numbers. Together with Theorem 1, we arrive at $\mathcal{G}(\mathbf{W}_0 + \Delta\mathbf{W}) \leq \epsilon + C_2\sqrt{|N|^{-1}(3\ln|N|\delta^{-1} + 8)} + \Phi_{\Delta\mathbf{W}}$; we quote this result formally in Theorem 6 in §D.1.6.

(*ii*) **Bounding $\mathcal{G}(\Delta\mathbf{W})$.** For a DNN, let q be the training bitwidth; we use $q = 32$. We assume $\mathcal{L}$ is σ-sub-Gaussian for all $\mathbf{W}$ and use the generalization upper bound of $\mathcal{G}(\Delta\mathbf{W})$ as in Lemma 4.5 of Zhu et al. (2024), which only bounds the generalization error of the fine-tuned update, $\Delta\mathbf{W}$, that is, $\mathcal{G}(\Delta\mathbf{W})$, while keeping the pretrained weights $\mathbf{W}_0$ fixed. In contrast, Theorem 1 is a standalone general result, not an extension. It gives an explicit general bound for the full model $\mathbf{W}_0 + \Delta\mathbf{W}$ and tells how the pretrained backbone and the fine-tuned update interact, which is characterized by $\Phi_{\mathbf{W}_0}$.

Key Takeaway. Theorem 1 relates the generalization error of the full adapted model to that of the update-only model through $\mathcal{G}(\mathbf{W}_0 + \Delta\mathbf{W}) \leq \Phi_{\mathbf{W}_0} + \mathcal{G}(\mathbf{BA})$, where $\mathcal{G}(\mathbf{BA})$ represents the generalization error of different LoRA variants; see Table 1 and §D.1.6. Under standard assumptions, results in Table 1 highlight that our column-restricted LoRA variants have identical generalization error upper bounds as their corresponding base variants, showing that restricting the adaptation space does not impact the theoretical generalization error upper bounds.

4 Empirical Insights

This section empirically evaluates whether our column-restricted methods preserve downstream performance in practice and yield additional computational benefits from their structured adaptation.

Implementation details and models used. Our empirical evaluation encompasses 10 pretrained models: (*i*) DeBERTa v3 Base, (*ii*) DeBERTa v2 XXL, (*iii*) GPT2-small, (*iv*) RoBERTa Base, (*v*) RoBERTa Large, (*vi*) DeepseekCoder-1.3B-base, (*vii*) TinyLlama-1.1B, (*viii*) Llama 3-8B, (*ix*) ViT Base, and (*x*) ViT-Tiny. See Table 5 for a detailed summary of the models and Table 6 in §E.1 for implementation details and reproducibility. We report results from the epoch with the best validation for the quality metric of the task for each model. We report additional ablation studies to justify the choices of hyperparameters in Table 6, such as learning rate, rank, scaling factor, and chain-reset in §E.2, spanning Tables 7-12.

Fine-tuning tasks and datasets. (*i*) *Natural Language Processing (NLP).* We use PAWS, TREC-50, and various GLUE benchmarks, including MRPC, CoLA, STS-B, and RTE for NLP tasks. (*ii*) *Image Classification.* We fine-tuned on Office-Home and CIFAR-10. (*iii*) *Code Generation.* Code generation presents unique challenges; minor deviations can lead to runtime errors or semantic mismatches. There is relatively limited LoRA-focused literature on programming tasks; we evaluate how different LoRA variants adapt to these tasks on Django and report results using Exact Match (EM). (*iv*) *Logical Reasoning.* We use OpenBookQA for elementary science multiple-choice reasoning, FOLIO for natural language reasoning with first-order logic, LogiQA for logical comprehension, and CLUTRR for compositional relational reasoning from text.

4.1 Quality of the Fine-Tuned Models

In Figure 3, we plot the test performance (Y-axis) and training-step speedup (X-axis) relative to LoRA for 10 fine-tuning methods; the closer a data point is to the top right, the better. For LoRA variants that train only one matrix, we double the rank compared to LoRA to more closely match the trainable parameter count. Based on this formalization, we plot the encoder model results in the *top row* and decoder models in the bottom row. The *first column* shows each fine-tuning method,

Table 2: Color and shape key for Figures 3 and 9.

Methods	FFT	LoRA	CoLA	Asym	RAC	LoRA+	**cLA**	**c^3LA**	**r-cLA**	**r-c^3LA**

Encoder Models	**Shape**	**Tasks Averaged Over**	**Encoder Models**	**Shape**	**Tasks Averaged Over**
ViT-Tiny	▲	Office-Home, CIFAR-10	**RoBERTa-Base**	★	MRPC, CoLA
ViT-Base	■	Office-Home, CIFAR-10	**RoBERTa-Large**	✖	MRPC, CoLA
DeBERTa-v2-XXL	◆	MRPC, TREC-50, PAWS	**DeBERTa-v3-Base**	✚	MRPC, TREC-50, PAWS, RTE, STS-B

Decoder Models	**Shape**	**Tasks Averaged Over**	**Decoder Models**	**Shape**	**Tasks Averaged Over**
DeepSeekCoder-1.3B	✖	Django	**TinyLlama**	▲	FOLIO, LogiQA, CLUTRR, OpenBookQA
Llama3-8B	■	OpenBookQA, CLUTRR	**GPT2-Small**	◆	E2E



Figure 3: **Test performance and training-step speedup relative to LoRA for all fine-tuned models, datasets, and methods over five seeds.** The *top row* presents only the encoder models (e.g., ViT, RoBERTa, DeBERTa); the *bottom row* presents the decoder models (e.g., TinyLlaMA, LLaMA-3-8B, GPT2-Small, DeepSeekCoder). In the *first column*, we show the performance of each fine-tuning method relative to LoRA averaged over each of the model's fine-tuning tasks (e.g., performance of cLA on ViT-Tiny averaged over Office-Home and CIFAR-10 is given by ▲); in the *second column*, we show the performance of each fine-tuning method relative to LoRA averaged over all models and tasks (e.g., performance of cLA averaged over ViT-Tiny fine-tuned on Office-Home and CIFAR-10, ViT-Base on Office-Home and CIFAR-10, DeBERTa-v2-XXL on MRPC, TREC-50, PAWS, and so on is given by ●). We define the colors and shapes in Table 2. Some results (such as the average performance of LoRA+ on decoder models for (d)) were not included for visual clarity. See Figure 9 for complete visuals.

and each model averaged over the model's fine-tuning tasks (Figs. 3a and 3c); the *second column* averages each method's performance over all models and tasks (Figs. 3b and 3d).

Figures 3b and 3d show that *substantial restriction to the adaptation space can retain a large fraction of LoRA's downstream performance.* On average, our column-restricted variants not only retain $97.3 - 98.1\%$ of LoRA's test performance, but also, as an ad hoc benefit, reduce training-step time by $6.3 - 7.8\%$ and reduce peak GPU memory by up to 11%.

Furthermore, our column-restricted LoRA variants outperform FFT and LoRA in some settings. For instance, on ViT-Tiny fine-tuned on Office-Home, FFT achieves 61.29 test accuracy compared to cLA's 78.76. cLA achieves 88.32 test accuracy on LLaMA-3-8B fine-tuned on OpenBookQA compared to LoRA's 87.64 while reducing training-step time by 8.3%. In Table 16, we report the exact test performance and training-step times of our experimental study. However, we also noticed a slight drop in performance on decoder models compared to encoder models, suggesting that the bidirectional attention of the encoder models may be more susceptible to reduced adaptation space.

Table 3: **Empirical generalization error**, $\mathcal{G}(\mathbf{W}) \approx \mathbb{E}(\mathcal{L}_{\text{test}}) - \mathcal{L}_{\text{train}}$, normalized to LoRA and averaged over five seeds for selected models and tasks; see Table 15 for all results. Light-gray columns identify our column-restricted variants. Red text indicates a strictly smaller $\mathcal{G}(\mathbf{W})$ than the parent LoRA variants.

Model	Dataset	Baselines		LoRA Variants		Parent Method	Column-Restricted Variants		Parent Method	Column-Restricted Variants	
		FFT	LoRA	CoLA	LoRA+	Asym	cLA	r-cLA	RAC	c^3LA	r-c^3LA
ViT-Tiny	Office-Home	0.099	1.000	0.869	1.052	0.942	1.104	1.073	1.052	1.176	1.203
ViT-Base	Office-Home	0.219	1.000	1.201	1.191	1.079	0.911	0.985	1.213	1.094	0.948
	CIFAR-10	0.048	1.000	0.994	0.134	0.597	1.136	0.916	0.784	0.768	0.895
DeBERTa-v3-Base	TREC-50	0.982	1.000	1.263	1.335	1.388	1.458	1.182	1.422	2.048	1.656
RoBERTa-Large	MRPC	0.400	1.000	0.884	0.013	0.607	1.182	0.614	0.373	1.075	1.103
TinyLlama	LogiQA	1.184	1.000	0.837	0.086	0.737	0.918	0.986	0.893	1.185	1.119
	CLUTRR	2.118	1.000	0.664	1.860	0.668	0.618	0.731	0.655	0.643	0.648
Llama3-8B	OpenBookQA	0.780	1.000	0.972	0.095	0.787	0.932	0.980	0.880	0.703	0.960
	CLUTRR	1.111	1.000	1.065	1.532	0.854	0.991	0.949	1.057	1.166	1.130
DeepSeekCoder	Django	2.190	1.000	0.597	2.183	0.955	0.930	0.909	0.990	0.952	0.944

Our broad set of controlled comparisons highlights that the adaptation space of our column-restricted LoRA variants is sufficient for competitive downstream performance, while offering reduced computational cost and peak GPU memory as additional benefits.

4.2 Generalizability under Restricted Adaptation

The generalization error, $\mathcal{G}(\mathbf{W})$ (Definition 1), is difficult to evaluate directly in practice, as the true distribution of a feature space and label space, $\mathcal{X} \times \mathcal{Y}$, cannot be obtained. Therefore, we cannot directly use the theoretical bounds on $\mathcal{G}(\mathbf{W})$ in Table 1. Since test samples are i.i.d. from $(\mathcal{X} \times \mathcal{Y})$, as an alternative, the difference between the loss of a model on a collection of unseen test samples and the loss on its training set approximates how well the model generalizes to the true distribution of the instance space it was trained on. Therefore, we approximate $\mathcal{G}(\mathbf{W}) \approx \mathbb{E}(\mathcal{L}_{\text{test}}) - \mathcal{L}_{\text{train}}$. As the size of the test set increases, the difference approaches the actual $\mathcal{G}(\mathbf{W})$ of the model; see discussion in §E.5. We report the approximate generalizability of all fine-tuned models in Tables 3 via the epoch with the best task-specific metric averaged over five seeds; see additional results in §E.5, Table 15.

From Table 3, for TinyLlama on CLUTRR, cLA obtains the best generalization error, 38.2% lower than that of LoRA and 7.5% lower than Asymmetric LoRA. On ViT-Base fine-tuned to Office-Home, most column-restricted variants have smaller generalization errors than LoRA and their corresponding asymmetric variants, $8.7 - 21.8\%$ lower than their parent variant.

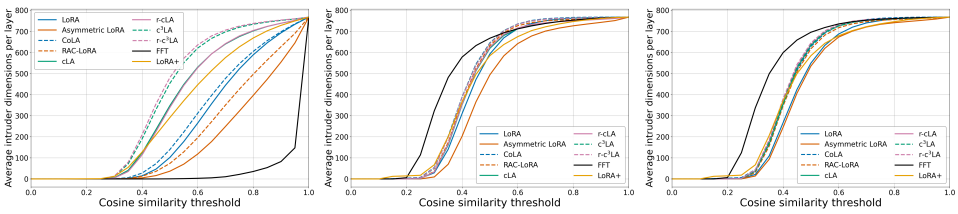
Since our theoretical bound on $\mathcal{G}(\mathbf{W})$ is an upper bound, it does not necessarily distinguish between the restricted and corresponding non-restricted methods under the current assumptions, while empirical behavior does. We noticed the column-restricted variants do not always preserve the empirical generalization error of their existing counterparts; rather, their empirical generalization error is more task- and model-sensitive than their existing counterparts. E.g., c^3LA and r-c^3LA on DeBERTa-v3 fine-tuned on TREC-50. This suggests that the variability may be related to column sensitivity. Thus, the training-speed benefit comes with a task-dependent cost in empirical generalization error.

4.2.1 Diagnostic Analysis

Table 3 shows that restricting the adaptation space can preserve empirical generalization, but the effect is model- and task-dependent. We examine two existing geometric proxies, loss landscapes (Li et al., 2018) and intruder dimensions (Shuttleworth et al., 2025), to determine whether they consistently explain these empirical trends.

(*i*) **Loss landscapes** sharpness does not consistently track the empirical generalization error. In Figure 1, ViT-Base fine-tuned on CIFAR-10 via FFT produces a sharper minimum than LoRA, yet has a smaller generalization error. In Figure 8, this discrepancy occurs for RoBERTa-Base on CoLA, where RAC-LoRA exhibits a sharper landscape than Asymmetric LoRA while attaining a smaller empirical generalization error; see §E.4.1. Thus, in our experiments, sharper minima do not provide a consistent explanation for differences in empirical generalization across fine-tuning strategies.

(*ii*) **Intruder dimension.** We next examine whether intruder dimensions provide a more informative diagnostic. Shuttleworth et al. (2025) relate intruder dimensions to performance on the pretraining task; we ask whether the same measure tracks empirical generalization after downstream adaptation.



(a) RoBERTa-Base (CoLA) (b) ViT-Base (Office-Home) (c) ViT-Base (CIFAR-10)

Figure 4: **The average intruder-dimension counts** per layer across cosine-similarity thresholds for RoBERTa-Base fine-tuned on CoLA, ViT-Base fine-tuned on Office-Home, and ViT-Base fine-tuned on CIFAR-10 using the studied methods. Methods with corresponding chain variants (Asymmetric LoRA with RAC-LoRA, cLA with c^3LA, r-cLA with r-c^3LA) share a color; dashed lines denote the chain variants.

The correspondence is again inconsistent. Figure 4 shows that FFT on ViT-Base/CIFAR-10 has many intruder dimensions while exhibiting a smaller empirical generalization error than LoRA. Likewise, chain methods can increase intruder counts without increasing the error; on RoBERTa-Base/CoLA, RAC-LoRA has more intruders but a smaller mean error than Asymmetric LoRA; see Table 15.

Key Takeaway. Neither loss-landscape sharpness nor intruder-dimension count consistently explains the empirical generalization behavior observed under restricted adaptation. These diagnostics remain useful for characterizing the geometry of fine-tuned models. Our results indicate that they should not be interpreted as standalone proxies for analyzing the effect of restricting the adaptation space.

5 Conclusion

This paper explicitly asks how much adaptation space LoRA needs for effective downstream adaptation. We study this question by controlling restrictions on the LoRA update. Cheap LoRA (cLA) and c^3LA provide controlled ways to impose increasingly structured restrictions.

Across 10 pretrained models and 14 datasets, substantial restrictions of the LoRA adaptation space often preserve competitive downstream performance while reducing training cost. These gains arise from restricting where adaptation is performed, rather than reducing the nominal number of trainable parameters. At the same time, the results are not uniform; in some model-task settings, restricting the adaptation space improves empirical generalization, whereas in others it increases the empirical generalization gap. Thus, the useful amount and location of adaptation space are not universal.

Our theoretical analysis provides a complementary perspective. Under standard assumptions, the generalization-error upper bounds for the restricted variants resemble those for their unrestricted counterparts. The bounds therefore do not distinguish when a particular column restriction will help or hurt empirically. However, our experiments show that the choice of accessible subspace can nevertheless affect downstream behavior. *Therefore, the theoretical and empirical results should be viewed as complementary rather than as a single criterion for predicting the performance of a restricted adaptation space.*

The diagnostic analyses further support this interpretation. Loss-landscape sharpness and intruder dimensions capture specific geometric or spectral properties of fine-tuned models, but neither consistently tracks the empirical generalization error across our settings. *We consequently view these quantities as secondary diagnostics rather than universal explanations of the observed adaptation-space effects. The central empirical evidence remains the direct comparison of downstream performance, empirical generalization errors, and computational cost.*

Taken together, these results suggest that effective parameter-efficient adaptation does not always require unrestricted access to the nominal LoRA adaptation space. A substantial portion of that space can often be removed while retaining much of the downstream performance. However, the amount and location of useful adaptation depend on the model, task, and selected subspace.

More broadly, our results motivate *adaptation space* as a complementary lens to parameter count and rank for studying PEFT. Rather than asking only how many parameters are trainable, one can ask which directions, coordinates, or subspaces are actually accessible during adaptation. The experiments suggest that this distinction can expose substantial redundancy in LoRA's nominal adaptation space, while also revealing that such redundancy is model- and task-dependent. Determining which subspaces are most useful, and whether they can be selected adaptively rather than prescribed in advance, remains an important future direction.

AI USE STATEMENT

We used generative AI tools, including ChatGPT and Codex, to assist with research, writing, and implementation. ChatGPT was used to improve the clarity and organization of the manuscript and to assist with literature searches. Codex was used for code refinement and debugging. The authors developed the core research ideas, mathematics, and initial implementation. AI-generated suggestions served as feedback rather than independent contributions. The authors reviewed all AI-assisted text and tested AI-assisted code in the corresponding experiments. The authors take full responsibility for the final content and results of this work.

REPRODUCIBILITY STATEMENT

Algorithm pseudocode is provided in Appendix C, and the assumptions and proofs for the theoretical results are presented in Appendices B and D. Section 4 describes the experimental setup. Tables 5 and 6 summarize the models, tasks, evaluation metrics, and main-experiment hyperparameters, including random seeds. Appendix E provides implementation details and the protocols for the ablation, efficiency, and diagnostic studies. Code and experiment configurations are available anonymously at `https://anonymous.4open.science/r/column-space-adaptation/`.

REFERENCES

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John Patrick Cunningham. LoRA learns less and forgets less. *Transactions on Machine Learning Research*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4133–4145, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Cooper Doyle, Andy Hu, Rebecca Chan, and Anna Leontjeva. Learning adapter rank via symmetry breaking. *arXiv preprint arXiv:2506.22809v4*, 2026. Version 4, May 12, 2026.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

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

Bingsheng He, Naga K Govindaraju, Qiong Luo, and Burton Smith. Efficient gather and scatter operations on graphics processors. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, pp. 1–12, 2007.

Haonan He, Peng Ye, Yuchen Ren, Yuan Yuan, Luyang Zhou, Shucun Ju, and Lei Chen. GoRA: Gradient-driven adaptive low rank adaptation. *arXiv preprint arXiv:2502.12171*, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

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

Ignacio Hounie, Charilaos Kanatsoulis, Arnuv Tandon, and Alejandro Ribeiro. LoRTA: Low Rank Tensor Adaptation of Large Language Models. *arXiv preprint arXiv:2410.04060*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

W. Ronny Huang, Zeyad Emam, Micah Goldblum, Liam Fowl, J. K. Terry, Furong Huang, and Tom Goldstein. Understanding generalization through visualizations. In *Proceedings on I Can't Believe It's Not Better! at NeurIPS Workshops*, volume 137 of *Proceedings of Machine Learning Research*, pp. 87–97. PMLR, 2020.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with LoRA. *arXiv preprint arXiv:2312.03732*, 2023.

Junsu Kim, Jaeyeon Kim, and Ernest K Ryu. Lora training provably converges to a low-rank global minimum or it fails loudly (but it probably won't fail). *arXiv preprint arXiv:2502.09376*, 2025.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Dongyue Li and Hongyang Zhang. Improved regularization and robustness for fine-tuning in neural networks. *Advances in Neural Information Processing Systems*, 34:27249–27262, 2021.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Zhenyu Ding, Haoying Wang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable Scale: Part I, Step Law — Optimal Hyperparameter Scaling Law in Large Language Model Pretraining, 2025.

Tao Li, Zhengbao He, Yujun Li, Yasheng Wang, Lifeng Shang, and Xiaolin Huang. Flat-LoRA: Low-rank adaptation over a flat loss landscape. *arXiv preprint arXiv:2409.14396*, 2024.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. Relora: High-rank training through low-rank updates. In *International Conference on Learning Representations*, volume 2024, pp. 49405–49421, 2024.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Grigory Malinovsky, Umberto Michieli, Hasan Abed Al Kader Hammoud, Taha Ceritli, Hayder Elesedy, Mete Ozay, and Peter Richtárik. Randomized asymmetric chain of lora: The first meaningful theoretical framework for low-rank adaptation. *arXiv preprint arXiv:2410.08305*, 2024.

Pravin Nair. Softmax is $1/2$-Lipschitz: A tight bound across all ℓ_p norms. *arXiv preprint arXiv:2510.23012*, 2025.

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

Lai-Man Po, Yuyang Liu, Haoxuan Wu, Tianqi Zhang, Wing-Yin Yu, Zhuohan Wang, Zeyu Jiang, and Kun Li. SBoRA: Low-rank adaptation with regional weight updates. In *Neural Information Processing: ICONIP 2024*, volume 2294 of *Communications in Computer and Information Science*, pp. 387–401. Springer, 2026. doi: 10.1007/978-981-96-7008-6_28.

Kaustubh Ponkshe, Raghav Singhal, Eduard Gorbunov, Alexey Tumanov, Samuel Horvath, and Praneeth Vepakomma. Initialization using update approximation is a silver bullet for extremely efficient low-rank fine-tuning. *arXiv preprint arXiv:2411.19557*, 2024.

Daniel Russo and James Zou. How much does your data exploration overfit? controlling bias via information usage. *IEEE Transactions on Information Theory*, 66(1):302–323, 2020.

Reece Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. LoRA vs full fine-tuning: An illusion of equivalence. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving lora in privacy-preserving federated learning. In *International conference on learning representations*, volume 2024, pp. 17978–17994, 2024.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37: 9565–9584, 2024.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5018–5027, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, 2018. doi: 10.18653/v1/W18-5446.

Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-rank Adaptation with Gradient Approximation. *Advances in Neural Information Processing Systems*, 37:54905–54931, 2024.

Sunghyeon Woo, Sol Namkung, Sunwoo Lee, Inho Jeong, Beomseok Kim, and Dongsuk Jeon. PaCA: Partial Connection Adaptation for Efficient Fine-Tuning. In *International Conference on Learning Representations*, 2025.

Wenhan Xia, Chengwei Qin, and Elad Hazan. Chain of lora: Efficient fine-tuning of language models via residual learning. *arXiv preprint arXiv:2401.04151*, 2024.

Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. *Advances in neural information processing systems*, 30, 2017.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. In *Advances in neural information processing systems*, volume 32, 2019.

Lingling Xu, Haoran Xie, S Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512v2*, 2023.

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. Autolora: Automatically tuning matrix ranks in low-rank adaptation based on meta learning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5048–5060, 2024.

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez De Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in low-rank adapters of foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62369–62385. PMLR, 2024.

CONTENTS

1 Introduction **1**

2 Fine-Tuning Through Structured Adaptation Spaces **2**

2.1 Full fine-tuning (FFT) and LoRA 3

2.2 Evolution of LoRA 3

2.3 From LoRA to Structured Adaptation 3

3 Theoretical Insights **4**

3.1 Setup and Generalization Error 5

3.2 On the Generalization of Different LoRA Variants 5

4 Empirical Insights **6**

4.1 Quality of the Fine-Tuned Models 6

4.2 Generalizability under Restricted Adaptation 8

4.2.1 Diagnostic Analysis 8

5 Conclusion **9**

A Evolution of LoRA — Continued **15**

B Relationship between PaCA and cLA **16**

B.1 Introducing New Artifacts to PaCA 16

B.2 Applying PaCA's Convergence Result to cLA 17

C Pseudo Code of Column-Restricted LoRA Variants **18**

D Theoretical Results **19**

D.1 Generalization 19

D.1.1 Inequalities Used 20

D.1.2 Proof of Theorem 1 21

D.1.3 Neural Network with No Activation Function—Special Case of Theorem 1 23

D.1.4 Consistency of the Bounds in Theorem 1 23

D.1.5 Adapting Theorem 1 to the Attention Mechanism 24

D.1.6 Adapting Theorem 1 Under Special Cases 26

E Addendum to Experimental Results **28**

E.1 Implementation Details 28

E.2 The Effects of Learning Rate, Scaling Factor, and Chain Reset Frequency on Performance Metrics Over Various Ranks 29

E.3 Computational Cost, Memory, and Efficiency 34

E.3.1 Leveraging the Computationally-Efficient Structure of Our Methods 34

E.3.2 Experiments 34
E.4 Loss-Landscape and Spectral Diagnostics 36
E.4.1 Loss Landscapes 36
E.4.2 Intruder Dimensions 38
E.5 Generalization Error—Continued 38
E.5.1 Extended Generalization Error Results 39
E.6 Complete Test Performance and Training-Time Results 39

F Limitations and Discussion **41**

G Table of Notations **42**

Organization of the Appendix. We organize the Appendix with the following structure: In §A, we discuss popular contemporary LoRA variants; this is a continuation of §2.2 of the main paper. In §B, we expand on the relationship between our methods and PaCA. In §C, we give the pseudocode of our proposed LoRA variants, cLA, c^3LA and their randomized variants. §D contains the proofs to the theorems in §3. Particularly, it contains the proofs for Theorem 1 and Theorem 6. Additionally, this section contains the statement and proof of Theorem 1 adapted to the attention mechanism; see Theorem 4 in §D.1.5. In §E, we discuss the implementation details and extend our empirical study by including various ablation studies and developing discussion topics. This section acts as an addendum to §4 of the main paper. For notations used in this paper, we refer to Table 17 in §G.

A Evolution of LoRA — Continued

Below, we discuss a few popular LoRA variants.

Chain of LoRA (CoLA) Xia et al. (2024) increases LoRA's performance without substantially increasing compute or memory costs. After fine-tuning $\mathbf{B}^1\mathbf{A}^1$ for the downstream task to obtain $\hat{\mathbf{B}}^1\hat{\mathbf{A}}^1$, CoLA merges $\hat{\mathbf{B}}^1\hat{\mathbf{A}}^1$ into the base weights and continues training with a new $\mathbf{B}^2\mathbf{A}^2$ on the same task, treating $\mathbf{W}_0 + \frac{\alpha}{r}\hat{\mathbf{B}}^1\hat{\mathbf{A}}^1$ as the base weights. Denote $\mathbf{W}^{(k,BA)} := \mathbf{W}_0 + \sum_{j=1}^{k}\frac{\alpha}{r}\hat{\mathbf{B}}^j\hat{\mathbf{A}}^j$ and $\mathbf{W}^{(0,BA)} = \mathbf{W}_0$ for convenience. CoLA of chain length k solves:

$$\text{For } j \in [k], \quad \hat{\mathbf{B}}^j\hat{\mathbf{A}}^j \approx \text{argmin}_{\mathbf{B}^j\mathbf{A}^j}\left[\mathcal{L}(\mathbf{W}_0^{(j-1,BA)} + \frac{\alpha}{r}\hat{\mathbf{B}}^j\hat{\mathbf{A}}^j)\right] \tag{6}$$

to obtain the fine-tuned model, $f_{\mathbf{W}^{(k,BA)}}$. CoLA simulates a higher-rank approximation of a single LoRA update Lialin et al. (2024) and claims to reduce LoRA's failure (Kim et al., 2025).

Asymmetric LoRA Zhu et al. (2024) modifies LoRA adaptation for each layer by freezing one of the low-rank matrices, conventionally, A to A_0, initializing the frozen matrix via a Normal distribution, and setting the trainable matrix to 0, and solves:

$$\hat{\mathbf{B}} \approx \text{argmin}_{\mathbf{B}}[\mathcal{L}(\mathbf{W}_0 + \frac{\alpha}{r}\mathbf{B}\mathbf{A}_0) = \frac{1}{|N|}\sum_{i=1}^{|N|}\ell(f_{\mathbf{W}_0+\frac{\alpha}{r}\mathbf{B}\mathbf{A}_0}(x_i), y_i)], \tag{7}$$

to obtain the fine-tuned model $f_{\mathbf{W}_0+\hat{\mathbf{B}}\mathbf{A}_0}$. Under trainable-parameter constraints, Asymmetric LoRA competes with LoRA Zhu et al. (2024) and retains the Lipschitz smoothness of the loss function, which LoRA does not Sun et al. (2024).

Randomized Asymmetric Chain of LoRA (RAC-LoRA) Malinovsky et al. (2024) combines Asymmetric LoRA and CoLA. RAC-LoRA fixes one of the low-rank matrices (conventionally A), initializing via some fixed distribution of matrices $\mathcal{D}$, and sets the trainable one to 0. Like CoLA, the trained $\hat{\mathbf{B}}^1\mathbf{A}_0^1$ is then merged into the base weights, and a new $\mathbf{B}\mathbf{A}_0$ is trained on the same task. Denote $\mathbf{W}^{(k,B)} := \mathbf{W}_0 + \sum_{j=1}^{k}\frac{\alpha}{r}\hat{\mathbf{B}}^j\mathbf{A}_0^j$ and $\mathbf{W}^{(0,B)} = \mathbf{W}_0$. RAC-LoRA of chain length k solves:

$$\text{For } j \in [k], \quad \hat{\mathbf{B}}^j \approx \text{argmin}_{\mathbf{B}^j}\left[\mathcal{L}(\mathbf{W}_0^{(j-1,B)} + \frac{\alpha}{r}\hat{\mathbf{B}}^j\mathbf{A}_0^j)\right] \tag{8}$$

to obtain the fine-tuned model $f_{\mathbf{W}^{(k,B)}}$.

LoRA+ Hayou et al. (2024) applies separate learning rates $\{\gamma_B^i, \gamma_A^i\}$ to the adapter matrices, $\{B^i, A^i\}$ of each layer, respectively, and maintains the identical structure to LoRA. LoRA+ prioritizes a substantially higher learning rate $(2 - 16\times)$ for B.

Other LoRA variants. There are other popular LoRA variants, such as HydraLoRA Tian et al. (2024), designed for fine-tuning on datasets with high heterogeneity. LoRA-SB Ponkshe et al. (2024) simulates the FFT process within low-rank subspaces by adding a trainable $r \times r$ matrix R, initializing BRA based on the SVD of the first step of FFT, and freezing B, A. QLoRA Dettmers et al. (2023) fine-tunes quantized LLMs. AdaLoRA Zhang et al. (2023) uses varying rank by layer and uses an SVD initialization. SoRA Ding et al. (2023) introduces sparsity in the low-rank updates. DoRA Liu et al. (2024) separates fine-tuning the direction and magnitude components of the model. AutoLoRA Zhang et al. (2024) trains each LoRA update as a sum of rank-one matrices and learns which to discard during training. DyLoRA Valipour et al. (2023) concentrates the more important features in the first columns and rows of B and A, respectively. LoRA-GA Wang et al. (2024) improves LoRA convergence by initializing low-rank adapters using a gradient approximation of full fine-tuned updates. LoRTA Hounie et al. (2024) replaces matrix-based adapters with a low-rank tensor factorization that unifies model updates across layers and attention heads. Flat-LoRA Li et al. (2024) seeks flat minima in the full parameter space to improve model generalization. BayesLoRA Doyle et al. (2026) learns the effective adapter rank through Bayesian low-rank variational dropout. rsLoRA Kalajdzievski (2023) stabilizes LoRA training across ranks by correcting the scaling factor from $1/r$ to $1/\sqrt{r}$, preventing gradient collapse. GoRA He et al. (2025) is a gradient-based LoRA variant that performs adaptive rank allocation and initializes adapter weights using compressed gradients. PaCA Woo et al. (2025) proposes fine-tuning randomly selected connections within pretrained weights to improve training speed and reduce overhead. Our independently developed column-restricted methods, particularly r-cLA, are similar to SBoRA-FA (Po et al., 2026); they serve as vehicles for a central contribution of this work: a systematic theoretical and empirical analysis of the benefits and task-dependent limitations of restricting LoRA's adaptation space.

B Relationship between PaCA and cLA

During fine-tuning, PEFT methods, such as LoRA adapters, must be processed with the backbone and cannot be merged, which limits their hardware utilization. PaCA Woo et al. (2025) is motivated by the training-time inefficiencies of adapter-based PEFT. PaCA fine-tunes a random subset of the pretrained model's columns explicitly and uses partial activations to form gradients, improving throughput and lowering activation-memory cost Woo et al. (2025). In contrast, cLA stays within the LoRA family to analyze column-restricted LoRA structures and for simple LoRA-based adapter deployment. cLA fixes its A matrix to $[I_r|\mathbf{0}]$, forcing a column update through B's projection of A; $\Delta W = B[I_r|\mathbf{0}] = [B|\mathbf{0}]$. Additionally, an alternate variant of cLA has the opportunity to update the A matrix and freeze the B matrix, forcing cLA to fine-tune a restricted subset of the row space of the pretrained model. As fine-tuning B is inherently more effective than fine-tuning A Zhu et al. (2024), this suggests that applying PaCA to a fixed subset of the rows of the pretrained matrix instead of the columns would degrade performance.

B.1 Introducing New Artifacts to PaCA

Since we established a connection between the LoRA PEFT family and PaCA via the column-restricted LoRA variants, we were interested in seeing how some performance-enhancing artifacts of LoRA PEFT methods migrate to PaCA, such as chain construction to PaCA, which we call C-PaCA, where we periodically resample the columns PaCA is updating. To have a fairer comparison to our column-restricted method, cLA, we introduce a deterministic PaCA variant, D-PaCA, which updates the same columns as cLA. PaCA and r-cLA both update a random subset of r columns of each adapted layer, and C-PaCA and r-c^{3}LA effectively implement chaining behavior to PaCA and r-cLA, respectively.

Experiments. To showcase the performance of PaCA and the column-restricted LoRA variants compared to baseline SOTA LoRA variants and full fine-tuning, we fine-tune ViT-Tiny and ViT-Base on the Office-Home and CIFAR-10 datasets, and RoBERTa-Base and RoBERTa-Large on the MRPC

and CoLA datasets using the aforementioned methods with rank $r = 16$ and 30 epochs, averaged over three random seeds. To align with the other experiments in our paper as well as LoRA's introduction paper Hu et al. (2021), we adapt only the query and value matrices of each attention head, fully fine-tune the classification heads of the models, and report those results in Table 4.

Discussion of results. In Table 4, the difference in accuracy of each column-restricted LoRA method and their PaCA counterpart (cLA and D-PaCA, r-cLA and PaCA, r-c^{3}LA and C-PaCA) is negligible, empirically validating how these methods connect LoRA and PaCA. In Table 4, C-PaCA's relative performance to PaCA was consistent with applying chain construction to the LoRA PEFT family. E.g., if CoLA outperformed LoRA for a particular row, then often C-PaCA outperformed PaCA. For efficiency details of PaCA and the sparse-induced LoRA variants compared to SOTA variants, see §E.3.

Table 4: Test accuracy (%) of ViT-Tiny and ViT-Base models fine-tuned on Office-Home and CIFAR-10, and RoBERTa-Base and RoBERTa-Large models fine-tuned on MRPC and CoLA, averaged over three seeds (0,1,2) for various LoRA and PaCA methods. The value in parentheses is normalized to LoRA's test accuracy (or MCC for the CoLA dataset), for $r = 16$, adapting the query and value matrices of the attention layers as well as the classification head. Light-gray columns identify our column-restricted variants. Red marks a strictly higher displayed performance or normalized ratio than Asym for cLA and r-cLA, or than RAC for c^{3}LA and r-c^{3}LA; each entry is compared separately. Ties are not highlighted.

Model	Dataset	Baselines		LoRA Variants	Parent Method	Column-Restricted Variants		PaCA Variants		Parent Method	Column-Restricted Variants		Chain of PaCA
		FFT	LoRA	CoLA	Asym	cLA	r-cLA	PaCA	D-PaCA	RAC	c^{3}LA	r-c^{3}LA	C-PaCA
ViT-Tiny	Office-Home	54.7 (0.803)	68.2 (1.000)	68.2 (1.000)	68.4 (1.003)	67.4 (0.989)	67.8 (0.995)	67.4 (0.989)	66.6 (0.976)	68.4 (1.003)	67.4 (0.989)	67.9 (0.995)	67.2 (0.985)
	CIFAR-10	93.5 (0.997)	93.7 (1.000)	93.8 (1.001)	94.2 (1.005)	94.0 (1.003)	93.9 (1.001)	94.0 (1.003)	94.1 (1.004)	94.2 (1.005)	93.9 (1.002)	93.8 (1.001)	94.0 (1.003)
ViT-Base	Office-Home	68.4 (0.850)	80.5 (1.000)	80.5 (1.000)	80.2 (0.995)	79.9 (0.992)	79.8 (0.991)	80.1 (0.994)	80.3 (0.997)	80.4 (0.998)	79.9 (0.992)	79.8 (0.991)	80.0 (0.994)
	CIFAR-10	95.0 (0.963)	98.6 (1.000)	98.6 (1.000)	99.0 (1.004)	98.6 (0.999)	98.7 (1.001)	98.6 (1.000)	98.8 (1.002)	99.0 (1.004)	98.6 (0.999)	98.8 (1.001)	98.5 (0.999)
RoBERTa-Base	MRPC	90.8 (1.010)	89.9 (1.000)	90.1 (1.003)	90.1 (1.002)	88.7 (0.986)	89.5 (0.996)	89.3 (0.993)	88.9 (0.988)	89.7 (0.997)	88.7 (0.987)	89.5 (0.996)	88.8 (0.987)
	CoLA	62.7 (0.953)	65.8 (1.000)	61.2 (0.931)	59.3 (0.901)	58.1 (0.883)	59.5 (0.904)	57.9 (0.880)	57.1 (0.868)	60.0 (0.911)	58.4 (0.887)	57.6 (0.876)	57.6 (0.875)
RoBERTa-Large	MRPC	91.3 (1.019)	89.6 (1.000)	90.8 (1.014)	91.0 (1.016)	90.1 (1.006)	89.5 (1.000)	89.6 (1.000)	90.4 (1.010)	91.2 (1.019)	90.5 (1.010)	90.0 (1.005)	90.3 (1.008)
	CoLA	69.7 (1.047)	66.6 (1.000)	68.4 (1.027)	69.3 (1.042)	65.0 (0.976)	67.1 (1.008)	67.3 (1.012)	67.1 (1.009)	68.1 (1.023)	66.4 (0.997)	66.4 (0.997)	65.4 (0.982)

B.2 Applying PaCA's Convergence Result to cLA

PaCA updates only a subset of columns in each layer. Writing the l-th layer weight as a list of columns $W_l = [w_1^l, w_2^l, \ldots, w_{d_{\text{out}}}^l]$ and letting $P_l = [i_1^l, i_2^l, \cdots, i_r^l]$ denote the selected column indices, PaCA's masked column update is

$$W_l^{t+1} = W_l^t - \eta \Delta W_l^t = W_l^t - \eta[0, \nabla_{i_1^l w_l^t}, \ldots, \nabla_{i_r^l w_l^t}, \ldots, 0].$$

Theorem 2. *(Loss Convergence of Partial Connections [Theorem 1 Woo et al. (2025)]) If the gradient of the loss $\mathcal{L}(W, X)$ is Lipschitz continuous with Lipschitz Constant $L_{\mathcal{L}}$, and only the partial connections are updated, then*

$$\mathcal{L}(W^{t+1}, X^{t+1}) \leq \mathcal{L}(W^t, X^t) - \eta(1 - \frac{\eta L_{\mathcal{L}}}{2})\|\nabla_{P^t}\|^2,$$

where η is the learning rate and $\|\nabla_{P^t}\|^2$ is the masked/restricted gradient only consisting of the gradient for the columns in P_l for each layer, $l \in [L]$.

In particular, the significance of Theorem 2 can be accredited to the fact that a learning rate of $0 < \eta < 2/L_{\mathcal{L}}$ ensures that the loss decreases after each iteration. We use PaCA's Theorem 2 to demonstrate similar convergence behavior as cLA and formalize the result in Theorem 3.

Theorem 3. *(Loss Convergence of cLA [Theorem 2 applied to cLA]) If the gradient of the loss $f(W, X)$ is Lipschitz continuous with Lipschitz Constant $L_{\mathcal{L}}$, and only the columns from cLA are*

updated, then

$$\mathcal{L}(W^{t+1}, X^{t+1}) \leq \mathcal{L}(W^t, X^t) - \eta(1 - \frac{\eta L_{\mathcal{L}}}{2})\|\nabla_W \mathcal{L}(W^t) A_0^T A_0\|^2.$$

Proof. In cLA, the frozen factor is $A_0 = [I_r|0]$, so the effective weight is

$$W^t = W_0 + \frac{\alpha}{r} B^t A_0.$$

Observe that only the first r columns of W can change ($\Delta W = B[I_r|0] = [B|0]$). Since W depends linearly on B, $W^t = W_0 + \frac{\alpha}{r} B^t A$, by the chain rule we have:

$$\nabla_B \mathcal{L}(W^t) = \nabla_B \mathcal{L}(W_0 + \frac{\alpha}{r} B^t A) = \frac{\alpha}{r} \nabla_W \mathcal{L}(W^t) A^\top.$$

Through gradient descent, we have:

$$B^{t+1} = B^t - \gamma \nabla_B \mathcal{L}(W^t) = B^t - \gamma \frac{\alpha}{r} \nabla_W \mathcal{L}(W^t) A^\top$$

Denote $\mathcal{K} = A_0^T A_0 = \text{Diag}(\overbrace{1, ..., 1}^{1 \text{ to } r}, 0, ..., 0)$.Observe the following:

$W^{t+1} - W^t = \frac{\alpha}{r}(B^{t+1} - B^t)A = -\gamma \frac{\alpha^2}{r^2} \nabla_W \mathcal{L}(W^t) A^\top A = -\gamma \frac{\alpha^2}{r^2} \nabla_W \mathcal{L}(W^t) \mathcal{K} = -\eta \nabla_W \mathcal{L}(W^t) \mathcal{K}$

In particular, for cLA we have $\mathcal{K}$ acting as a mask that selects only the first r columns in each layer. Thus, if we define $P_l^{\text{cLA}} = \{1, 2, \ldots, r\}$ for every layer l, then the masked gradient $\nabla_W \mathcal{L}(W^t)\mathcal{K}$ is exactly the restricted gradient ∇_{P^t} in Theorem 2 (with a deterministic choice of P_l rather than a random subset). Therefore, under the same Lipschitz-gradient assumption on $\mathcal{L}$ with constant $L_{\mathcal{L}}$, Theorem 2 applies directly to the effective weights W^t induced by cLA, resulting in

$$\mathcal{L}(W^{t+1}, X^{t+1}) \leq \mathcal{L}(W^t, X^t) - \eta\Big(1 - \frac{\eta L_{\mathcal{L}}}{2}\Big)\big\|\nabla_W \mathcal{L}(W^t)\mathcal{K}\big\|^2.$$

Hence the result. □

The same step-size condition $0 < \eta < 2/L_{\mathcal{L}}$ ensures a decrease in the loss for cLA updates. Equivalently, in terms of the B update step size, using $\gamma = \eta \frac{r^2}{\alpha^2}$ implies the condition

$$0 < \gamma < \frac{2r^2}{L_{\mathcal{L}} \alpha^2},$$

which guarantees that each iteration decreases the loss.

C Pseudo Code of Column-Restricted LoRA Variants

In this Section, we present the pseudocode of our proposed LoRA variants, cLA (Algorithm 1), random-cLA (Algorithm 2), c^3LA (Algorithm 3), and r-c^3LA (Algorithm 4).

Algorithm 1 Cheap LoRA (cLA)

Parameters: Loss function $\mathcal{L}$ and model $f_{\mathbf{W}}(\cdot)$. Pretrained weights $\mathbf{W}_0 = (W_0^1, ..., W_0^L)$, where $W_0^i \in \mathbb{R}^{n_i \times m_i}$. rank $r \ll \min\{m_i, n_i\}_{i \in [L]}$, learning rate $\gamma > 0$, scaling factor $\alpha > 0$, total training iterations T.
Initialize $A_0^j = [I_r \,|\, \mathbf{0}_{r \times (m_j - r)}]$; $B^{0,j} = \mathbf{0}$ for $j \in [L]$
for $t = 1, ..., T$ **do**
  forward pass with LoRA modules
  backward pass then update $\mathbf{B}^t$
  for $j = 1, ..., L$ **do**
    $B^{t,j} = B^{t-1,j} - \gamma \frac{\alpha}{r} \nabla_j \mathcal{L}(\mathbf{W}_0 + \frac{\alpha}{r} \mathbf{B}^{t-1} \mathbf{A}_0)\, \text{Diag}(\overbrace{1, ..., 1}^{1 \text{ to } r}, 0, ..., 0)$
  end for
end for
$\hat{j} = \text{argmin}_{j \in [T]}\, \mathcal{L}(\mathbf{W}_0 + \frac{\alpha}{r} \mathbf{B}^j \mathbf{A}_0)$ or task-based metric.
Return Fine-tuned weights $\mathbf{W}_0 + \frac{\alpha}{r} \mathbf{B}^{\hat{j}} \mathbf{A}_0$

Algorithm 2 random Cheap LoRA (r-cLA)

Parameters: Loss function $\mathcal{L}$ and model $f_{\mathbf{W}}(\cdot)$. Pretrained weights $\mathbf{W}_0 = (W_0^1, ..., W_0^L)$, where $W_0^i \in \mathbb{R}^{m_i \times n_i}$. rank $r \ll \min\{m_i, n_i\}_{i\in[L]}$, learning rate $\gamma > 0$, scaling factor $\alpha > 0$, total training iterations T.
Initialize
  $\xi_j = \text{randint}(0, \lfloor \frac{n_j}{r} \rfloor - 1)$ for $j \in [L]$
  $A_0^j = \left[\mathbf{0}_{r\times\xi_j} \mid I_r \mid \mathbf{0}_{r\times(n_j-\xi_j-r)}\right]$; $B^{0,j} = \mathbf{0}$ for $j \in [L]$
for $t = 1, ..., T$ **do**
  forward pass with LoRA modules
  backward pass then update $\mathbf{B}^t$
  for $j = 1, ..., L$ **do**
    $B^{t,j} = B^{t-1,j} - \gamma\frac{\alpha}{r}\nabla_j\mathcal{L}(\mathbf{W}_0 + \frac{\alpha}{r}\mathbf{B}^{t-1}\mathbf{A}_0)\,\text{Diag}(0, ..., 0, \overbrace{1, ..., 1}^{\xi_j+1 \text{ to } \xi_j+r}, 0, ..., 0)$
  end for
end for
$\hat{j} = \text{argmin}_{j\in[T]}\,\mathcal{L}(\mathbf{W}_0 + \frac{\alpha}{r}\mathbf{B}^j\mathbf{A}_0)$ or task-based metric.
Return Fine-tuned weights $\mathbf{W}_0 + \frac{\alpha}{r}\mathbf{B}^{\hat{j}}\mathbf{A}_0$

Algorithm 3 Circulant Chain of Cheap LoRA (c^3LA)

Parameters: Loss function $\mathcal{L}$ and model $f_{\mathbf{W}}(\cdot)$. Pretrained weights $\mathbf{W}_0^{(0)} = (W^1, ..., W^L)$, where $W^i \in \mathbb{R}^{m_i \times n_i}$. rank $r \ll \min\{m_i, n_i\}_{i\in[L]}$, learning rate $\gamma > 0$, scaling factor $\alpha > 0$, total training iterations T, chain-length $k <= T$.
Initialize $A_0^j = [I_r \mid \mathbf{0}_{r\times(n_j-r)}]$; $B^{0,j} = \mathbf{0}$ for $j \in [L]$, current chain $c = 0$.
for $t = 1, ..., T$ **do**
  if $t \equiv 0 \pmod{\lfloor \frac{T}{k} \rfloor}$ **then**
    $c = c + 1$
    Merge LoRA to backbone weights $\mathbf{W}_0^{(c)} = \mathbf{W}_0^{(c-1)} + \frac{\alpha}{r}\mathbf{B}^{t-1}\mathbf{A}_0$
    Re-initialize with $\mathbf{A}_0$ shifted by r:
      $A_0^j = [\mathbf{0}_{r\times cr} \mid I_r \mid \mathbf{0}_{r\times n_i-r-cr}]$; $B^{t-1,j} = \mathbf{0}$ for $j \in [L]$
  end if
  forward pass with LoRA modules
  backward pass then update $\mathbf{B}^t$
  for $j = 1, ..., L$ **do**
    $B^{t,j} = B^{t-1,j} - \gamma\frac{\alpha}{r}\nabla_j\mathcal{L}(\mathbf{W}_0^{(c)} + \frac{\alpha}{r}\mathbf{B}^{t-1}\mathbf{A}_0)\,\text{Diag}(0, ..., 0, \overbrace{1, ..., 1}^{cr \text{ to } (c+1)_r}, 0, ..., 0)$
  end for
end for
$\hat{c}, \hat{j} = \text{argmin}_{j\in[\lfloor \frac{T}{k} \rfloor], c\in[k]}\,\mathcal{L}(\mathbf{W}_0^{(c)} + \frac{\alpha}{r}\mathbf{B}^{cj}\mathbf{A}_0)$ or task-based metric.
Return Fine-tuned weights $\mathbf{W}_0^{\hat{c}} + \frac{\alpha}{r}\mathbf{B}^{\hat{c}\hat{j}}\mathbf{A}_0$

D THEORETICAL RESULTS

This section complements Section 3 in the main paper.

D.1 GENERALIZATION

Theorem 1 provides a pointwise deterministic bound that relates the generalization error of the pretrained model $\mathbf{W}_0$ to that of the fine-tuned model $\mathbf{W}_0 + \Delta\mathbf{W}$, rather than a uniform probabilistic PAC bound over a hypothesis class. This construction matches our intended use so that practitioners can be theoretically guided by the structural characteristics of a fine-tuning procedure.

In this section, we give a detailed proof of the generalization error bound. We start by listing the inequalities used in this section.

Algorithm 4 Random Circulant Chain of Cheap LoRA (r-c^3LA)

Parameters: Loss function $\mathcal{L}$ and model $f_{\mathbf{W}}(\cdot)$. Pretrained weights $\mathbf{W}_0^{(0)} = (W^1, ..., W^L)$, where $W^i \in \mathbb{R}^{m_i \times n_i}$. rank $r \ll \min\{m_i, n_i\}_{i\in[L]}$, learning rate $\gamma > 0$, scaling factor $\alpha > 0$, total training iterations T, chain-length $k <= T$.
Initialize
$\xi_j = \text{randint}(0, \lfloor \frac{n_j}{r} \rfloor - 1)$ for $j \in [L]$.
$A_0^j = \left[\mathbf{0}_{r\times\xi_j} \mid I_r \mid \mathbf{0}_{r\times(n_j-\xi_j-r)}\right]$; $B^{0,j} = \mathbf{0}$ for $j \in [L]$, current chain $c = 0$.
for $t = 1, ..., T$ **do**
  if $t \equiv 0 \pmod{\lfloor \frac{T}{k} \rfloor}$ **then**
    $c = c + 1$
    Merge LoRA to backbone weights $\mathbf{W}_0^{(c)} = \mathbf{W}_0^{(c-1)} + \frac{\alpha}{r}\mathbf{B}^{t-1}\mathbf{A}_0$
    Re-initialize with $\mathbf{A}_0$ shifted by a new random variable ξ_j':
      $A_0^j = \left[\mathbf{0}_{r\times\xi_j'} \mid I_r \mid \mathbf{0}_{r\times(n_j-\xi_j'-r)}\right]$; $B^{t-1,j} = \mathbf{0}$ for $j \in [L]$
  end if
  forward pass with LoRA modules
  backward pass then update $\mathbf{B}^t$
  for $j = 1, ..., L$ **do**
    $B^{t,j} = B^{t-1,j} - \gamma\frac{\alpha}{r}\nabla_j\mathcal{L}(\mathbf{W}_0 + \frac{\alpha}{r}\mathbf{B}^{t-1}\mathbf{A}_0)\,\text{Diag}(0, ..., 0, \overbrace{1, ..., 1}^{\xi_j+1 \text{ to } \xi_j+r}, 0, ..., 0)$
  end for
end for
$\hat{c}, \hat{j} = \text{argmin}_{j\in[\lfloor \frac{T}{k} \rfloor], c\in[k]}\, \mathcal{L}(\mathbf{W}_0^{(c)} + \frac{\alpha}{r}\mathbf{B}^{cj}\mathbf{A}_0)$ or task-based metric.
Return Fine-tuned weights $\mathbf{W}_0^{\hat{c}} + \frac{\alpha}{r}\mathbf{B}^{\hat{c}\hat{j}}\mathbf{A}_0$

D.1.1 Inequalities Used

1. If $A, B \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$, then the Triangle-Inequality gives:
$$\|(A + B)x\| \le \|Ax\| + \|Bx\|. \tag{9}$$
2. For $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$, we have:
$$\|Ax\| \le \|A\|_2\|x\|. \tag{10}$$
3. By Assumption 3, we have:
$$\|\sigma(Ax)\| \le \|\sigma(Ax) - \sigma(0)\| + \|\sigma(0)\| \le L_\sigma\|Ax\| + \|\sigma(0)\|. \tag{11}$$
4. For a finite collection of matrices, $\{A_1, \cdots, A_k\}$; $A_i \in \mathbb{R}^{m\times n}$, we have:
$$\text{rank}(\sum_{i=1}^{k} A_i) \le \sum_{i=1}^{k} \text{rank}(A_i). \tag{12}$$
5. Let $\mathbf{I}(X; Y)$ denote the mutual information between random variables X and Y. It measures how much the knowledge of one random variable reveals about the other, i.e.,
$$\mathbf{I}(X; Y) = D(P_{(X,Y)}\|P_X \otimes P_Y) = \sup_F\{\int F dP_{XY} - \log\int e^F d(P_X \otimes P_Y)\},$$
where F is a bounded, measurable function Xu & Raginsky (2017). Let T be a deterministic map for $A \in \mathbb{R}^{m\times n}$. Then the **Data Processing Inequality (DPI)** gives us $\mathbf{I}(T(A); N) \le \mathbf{I}(A; N)$, where N denotes the training dataset. If T is a bijective mapping then **DPI** gives us Zhu et al. (2024):
$$\mathbf{I}(A; N) = \mathbf{I}(T(A); N). \tag{13}$$

We make the following general assumptions to prove our generalization result.

Assumption 1. ***(Boundedness of input vectors)*** *The input vectors are bounded, i.e., there exists a constant $C \ge 0$ such that $\|x\| \le C$, for all $x \in \mathcal{X}$.*

Assumption 2. ***(Lipschitz continuity of the loss)*** *The loss function, $\ell(\cdot) : \mathbb{R}^d \to \mathbb{R}$ is $L_\mathcal{L}$-Lipschitz continuous, i.e., $|\ell(f_{\boldsymbol{W}}(x), y) - \ell(f_{\boldsymbol{W}'}(x), y)| \le L_\mathcal{L}\|f_{\boldsymbol{W}}(x) - f_{\boldsymbol{W}'}(x)\|$ for all $\boldsymbol{W}, \boldsymbol{W}' \in \mathbb{R}^d$ and $(x, y) \in \mathcal{X} \times \mathcal{Y}$.*

Assumption 3. ***(Lipschitz continuity of activation)*** *The vector-valued activation function, $\sigma_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$, for each layer, i, is L_{σ_i}-Lipschitz continuous, i.e., $\|\sigma_i(u) - \sigma_i(v)\| \leq L_{\sigma_i}\|u - v\|$, for all $u, v \in \mathbb{R}^{n_i}$.*

Now, we are set to prove Theorem 1. However, first, we furnish the full version of the theorem, including the details of the correction terms, $\Phi_{\Delta \mathbf{W}}$ and $\Phi_{\mathbf{W}_0}$, which we deliberately omitted in the main paper to not overwhelm the readers.

D.1.2 Proof of Theorem 1

Theorem 1. ***(Generalization bounds)*** *Let $f_{\mathbf{W}_0+\Delta\mathbf{W}}(x)$ be a L-layer fine-tuned DNN, where $\boldsymbol{W}_0+\Delta\boldsymbol{W}$ is the fine-tuned model. Let the loss function for fine-tuning and the activation functions follow Assumptions 1–3. Then $|\mathcal{G}(\boldsymbol{W}_0 + \Delta\boldsymbol{W}) - \mathcal{G}(\boldsymbol{W}_0)| \leq \Phi_{\Delta\boldsymbol{W}}$, and $|\mathcal{G}(\boldsymbol{W}_0 + \Delta\boldsymbol{W}) - \mathcal{G}(\Delta\boldsymbol{W})| \leq \Phi_{\boldsymbol{W}_0}$. Thus, we have:*

$$\mathcal{G}(\boldsymbol{W}_0 + \Delta\boldsymbol{W}) \leq \min\left(\mathcal{G}(\boldsymbol{W}_0) + \Phi_{\Delta\boldsymbol{W}}, \mathcal{G}(\Delta\boldsymbol{W}) + \Phi_{\boldsymbol{W}_0}\right),$$

where

$$\begin{aligned}\Phi_{\Delta\boldsymbol{W}} := {} & 2L_{\mathcal{L}}C\prod_{i=1}^{L} L_{\sigma_i}\left[\prod_{i=1}^{L}(\|W_0^i\|_2 + \|\Delta W^i\|_2) - \prod_{i=1}^{L}\|W_0^i\|_2\right] \\ & + 2L_{\mathcal{L}}\sum_{k=1}^{L-1}\|\sigma_k(0)\|\prod_{j=k+1}^{L} L_{\sigma_j}\left[\prod_{j=k+1}^{L}(\|W_0^j\|_2 + \|\Delta W^j\|_2) - \prod_{j=k+1}^{L}\|W_0^j\|_2\right],\end{aligned}$$

and

$$\begin{aligned}\Phi_{\boldsymbol{W}_0} := {} & 2L_{\mathcal{L}}C\prod_{i=1}^{L} L_{\sigma_i}\left[\prod_{i=1}^{L}(\|W_0^i\|_2 + \|\Delta W^i\|_2) - \prod_{i=1}^{L}\|\Delta W^i\|_2\right] \\ & + 2L_{\mathcal{L}}\sum_{k=1}^{L-1}\|\sigma_k(0)\|\prod_{j=k+1}^{L} L_{\sigma_j}\left[\prod_{j=k+1}^{L}(\|W_0^j\|_2 + \|\Delta W^j\|_2) - \prod_{j=k+1}^{L}\|\Delta W^j\|_2\right],\end{aligned}$$

are the correction terms.

Proof. Let $h_0 = h_0^0 = x$. For each layer $i \in [L]$, define

$$h_i := \sigma_i((W_0^i + \Delta W^i)h_{i-1})$$

and

$$h_i^0 := \sigma_i(W_0^i h_{i-1}^0).$$

Thus, $h_L = f_{\mathbf{W}_0+\Delta\mathbf{W}}(x)$ and $h_L^0 = f_{\mathbf{W}_0}(x)$. We additionally define $d_i := \|h_i - h_i^0\|$.

We first bound the magnitude of h_i. Assumption 3 and Inequalities (10) and (11) give

$$\begin{aligned}\|h_i\| &= \|\sigma_i((W_0^i + \Delta W^i)h_{i-1})\| \\ &\leq L_{\sigma_i}\|(W_0^i + \Delta W^i)h_{i-1}\| + \|\sigma_i(0)\| \\ &\leq L_{\sigma_i}(\|W_0^i\|_2 + \|\Delta W^i\|_2)\|h_{i-1}\| + \|\sigma_i(0)\|.\end{aligned}$$

Applying this relation recursively and using $\|x\| \leq C$ gives

$$\begin{aligned}\|h_i\| \leq {} & C\prod_{j=1}^{i}\left[L_{\sigma_j}(\|W_0^j\|_2 + \|\Delta W^j\|_2)\right] \\ & + \sum_{k=1}^{i}\|\sigma_k(0)\|\prod_{j=k+1}^{i}\left[L_{\sigma_j}(\|W_0^j\|_2 + \|\Delta W^j\|_2)\right].\end{aligned} \tag{14}$$

We bound the difference between fine-tuned and pretrained hidden states using Assumption 3:

$$\begin{aligned}d_i &= \|\sigma_i((W_0^i + \Delta W^i)h_{i-1}) - \sigma_i(W_0^i h_{i-1}^0)\| \\ &\leq L_{\sigma_i}\|(W_0^i + \Delta W^i)h_{i-1} - W_0^i h_{i-1}^0\| \\ &= L_{\sigma_i}\|\Delta W^i h_{i-1} + W_0^i(h_{i-1} - h_{i-1}^0)\| \\ &\leq L_{\sigma_i}\|\Delta W^i\|_2\|h_{i-1}\| + L_{\sigma_i}\|W_0^i\|_2 d_{i-1}.\end{aligned}$$

We show by induction that this recurrence gives

$$d_i \le C \prod_{j=1}^{i} L_{\sigma_j} \left[\prod_{j=1}^{i} (\|W_0^j\|_2 + \|\Delta W^j\|_2) - \prod_{j=1}^{i} \|W_0^j\|_2 \right] + \sum_{k=1}^{i-1} \|\sigma_k(0)\| \prod_{j=k+1}^{i} L_{\sigma_j} \left[\prod_{j=k+1}^{i} (\|W_0^j\|_2 + \|\Delta W^j\|_2) - \prod_{j=k+1}^{i} \|W_0^j\|_2 \right]. \tag{15}$$

For $i = 1$, we have

$$d_1 \le C L_{\sigma_1} \|\Delta W^1\|_2,$$

which agrees with (15). Assume (15) holds through layer $i-1$. Substituting (14) and the induction hypothesis into the recurrence for d_i collects the input-dependent products according to

$$\begin{aligned} &\|\Delta W^i\|_2 \prod_{j=1}^{i-1} (\|W_0^j\|_2 + \|\Delta W^j\|_2) \\ &\quad + \|W_0^i\|_2 \left[\prod_{j=1}^{i-1} (\|W_0^j\|_2 + \|\Delta W^j\|_2) - \prod_{j=1}^{i-1} \|W_0^j\|_2 \right] \\ &= \prod_{j=1}^{i} (\|W_0^j\|_2 + \|\Delta W^j\|_2) - \prod_{j=1}^{i} \|W_0^j\|_2. \end{aligned}$$

For every activation-offset term indexed by $k < i$, the same identity applies over the suffix $j = k+1, \dots, i$. Hence, (15) follows for layer i, completing the induction.

Setting $i = L$ gives

$$\begin{aligned} \|f_{\mathbf{w}_0+\Delta\mathbf{w}}(x) - f_{\mathbf{w}_0}(x)\| &\le C \prod_{i=1}^{L} L_{\sigma_i} \left[\prod_{i=1}^{L} (\|W_0^i\|_2 + \|\Delta W^i\|_2) - \prod_{i=1}^{L} \|W_0^i\|_2 \right] \\ &\quad + \sum_{k=1}^{L-1} \|\sigma_k(0)\| \prod_{j=k+1}^{L} L_{\sigma_j} \left[\prod_{j=k+1}^{L} (\|W_0^j\|_2 + \|\Delta W^j\|_2) - \prod_{j=k+1}^{L} \|W_0^j\|_2 \right] \\ &= \frac{\Phi_{\Delta\mathbf{w}}}{2L_{\mathcal{L}}}. \end{aligned}$$

Now that we have an upper bound for the difference of our hypotheses, we estimate the difference in terms of true risk. By Assumption 2,

$$\begin{aligned} &|\hat{\mathcal{L}}_{\text{global}}(\mathbf{W}_0 + \Delta\mathbf{W}) - \hat{\mathcal{L}}_{\text{global}}(\mathbf{W}_0)| \\ &= |\mathbb{E}_{\mathcal{X},\mathcal{Y}\sim\nu}[\ell(f_{\mathbf{w}_0+\Delta\mathbf{w}}(X), Y) - \ell(f_{\mathbf{w}_0}(X), Y)]| \\ &\le \mathbb{E}_{\mathcal{X},\mathcal{Y}\sim\nu}\left[|\ell(f_{\mathbf{w}_0+\Delta\mathbf{w}}(X), Y) - \ell(f_{\mathbf{w}_0}(X), Y)|\right] \\ &\le L_{\mathcal{L}} \mathbb{E}_{\mathcal{X},\mathcal{Y}\sim\nu}[\|f_{\mathbf{w}_0+\Delta\mathbf{w}}(X) - f_{\mathbf{w}_0}(X)\|] \\ &\le \frac{\Phi_{\Delta\mathbf{w}}}{2}. \end{aligned}$$

Similarly, for the empirical risk,

$$\begin{aligned} &|\mathcal{L}(\mathbf{W}_0 + \Delta\mathbf{W}) - \mathcal{L}(\mathbf{W}_0)| \\ &= \left| \frac{1}{|N|} \sum_{i=1}^{|N|} [\ell(f_{\mathbf{w}_0+\Delta\mathbf{w}}(x_i), y_i) - \ell(f_{\mathbf{w}_0}(x_i), y_i)] \right| \\ &\le \frac{1}{|N|} \sum_{i=1}^{|N|} |\ell(f_{\mathbf{w}_0+\Delta\mathbf{w}}(x_i), y_i) - \ell(f_{\mathbf{w}_0}(x_i), y_i)| \\ &\le \frac{\Phi_{\Delta\mathbf{w}}}{2}. \end{aligned}$$

Therefore, using Definition 1,

$$\begin{aligned}|\mathcal{G}(\mathbf{W}_0+\Delta\mathbf{W})-\mathcal{G}(\mathbf{W}_0)| &\le |\hat{\mathcal{L}}_{\text{global}}(\mathbf{W}_0+\Delta\mathbf{W})-\hat{\mathcal{L}}_{\text{global}}(\mathbf{W}_0)| \\ &\quad + |\mathcal{L}(\mathbf{W}_0+\Delta\mathbf{W})-\mathcal{L}(\mathbf{W}_0)| \\ &\le \Phi_{\Delta\mathbf{W}}.\end{aligned}$$

Consequently,

$$\mathcal{G}(\mathbf{W}_0+\Delta\mathbf{W}) \le \mathcal{G}(\mathbf{W}_0)+\Phi_{\Delta\mathbf{W}}.$$

Bound around $f_{\Delta\mathbf{W}}$. Let $f_{\Delta\mathbf{W}}$ denote the network that uses ΔW^i as the weight of each layer with the same activation functions. The same argument follows after exchanging the roles of W_0^i and ΔW^i at every layer. Thus,

$$|\mathcal{G}(\mathbf{W}_0+\Delta\mathbf{W})-\mathcal{G}(\Delta\mathbf{W})| \le \Phi_{\mathbf{W}_0},$$

and therefore

$$\mathcal{G}(\mathbf{W}_0+\Delta\mathbf{W}) \le \mathcal{G}(\Delta\mathbf{W})+\Phi_{\mathbf{W}_0}.$$

Combining both inequalities, we obtain

$$\mathcal{G}(\mathbf{W}_0+\Delta\mathbf{W}) \le \min\{\mathcal{G}(\mathbf{W}_0)+\Phi_{\Delta\mathbf{W}}, \mathcal{G}(\Delta\mathbf{W})+\Phi_{\mathbf{W}_0}\}.$$

Hence, the result. □

D.1.3 Neural Network with No Activation Function—Special Case of Theorem 1

We can simplify Theorem 1 for a neural network with no nonlinear activation functions, i.e., $\sigma_i = I_{n_i}$ for all $i \in [L]$. We additionally include the simplest case of a one-layer linear network.

Corollary 1. *Let Assumption 1 hold, and let the loss follow Assumption 2. Let $\sigma_i = I_{n_i}$ for all $i \in [L]$. Then*

$$\begin{aligned}\mathcal{G}(\boldsymbol{W}_0+\Delta\boldsymbol{W}) \le \min\Big\{&\mathcal{G}(\boldsymbol{W}_0)+2CL_{\mathcal{L}}\left[\prod_{i=1}^{L}(\|W_0^i\|_2+\|\Delta W^i\|_2)-\prod_{i=1}^{L}\|W_0^i\|_2\right], \\ &\mathcal{G}(\Delta\boldsymbol{W})+2CL_{\mathcal{L}}\left[\prod_{i=1}^{L}(\|W_0^i\|_2+\|\Delta W^i\|_2)-\prod_{i=1}^{L}\|\Delta W^i\|_2\right]\Big\}.\end{aligned}$$

Remark 1. Let Assumption 1 hold, and $\mathcal{L}$ follow Assumption 2. If $L = 1$, i.e., the model consists of only 1 layer, then we have:

$$\mathcal{G}(\mathbf{W}_0+\Delta\mathbf{W}) \le \min(\mathcal{G}(\mathbf{W}_0)+2CL_{\mathcal{L}}\|\Delta W\|_2, \mathcal{G}(\Delta\mathbf{W})+2CL_{\mathcal{L}}\|W_0\|_2).$$

D.1.4 Consistency of the Bounds in Theorem 1

We first consider the case where the fine-tuned update is zero, i.e., $\|\Delta W^i\|_2 = 0$, for all $i \in [L]$. In this case, $f_{\mathbf{W}_0+\Delta\mathbf{W}} = f_{\mathbf{W}_0}$. From the definition of $\Phi_{\Delta\mathbf{W}}$, we have:

$$\begin{aligned}\Phi_{\Delta\mathbf{W}} &= 2L_{\mathcal{L}}C\prod_{i=1}^{L}L_{\sigma_i}\left[\prod_{i=1}^{L}(\|W_0^i\|_2+0)-\prod_{i=1}^{L}\|W_0^i\|_2\right] \\ &\quad + 2L_{\mathcal{L}}\sum_{k=1}^{L-1}\|\sigma_k(0)\|\prod_{j=k+1}^{L}L_{\sigma_j}\left[\prod_{j=k+1}^{L}(\|W_0^j\|_2+0)-\prod_{j=k+1}^{L}\|W_0^j\|_2\right] = 0.\end{aligned}$$

Thus, Theorem 1 gives:

$$|\mathcal{G}(\mathbf{W}_0+\Delta\mathbf{W})-\mathcal{G}(\mathbf{W}_0)| \le 0,$$

confirming that

$$\mathcal{G}(\mathbf{W}_0+\Delta\mathbf{W}) = \mathcal{G}(\mathbf{W}_0)$$

when $\Delta\mathbf{W} = 0$.

Similarly, consider the update-only comparison when $\|W_0^i\|_2 = 0$, for all $i \in [L]$. In this case, $f_{\mathbf{W}_0+\Delta\mathbf{W}} = f_{\Delta\mathbf{W}}$. From the definition of $\Phi_{\mathbf{W}_0}$, we have:

$$\Phi_{\mathbf{W}_0} = 2L_{\mathcal{L}}C\prod_{i=1}^{L} L_{\sigma_i}\left[\prod_{i=1}^{L}(0+\|\Delta W^i\|_2) - \prod_{i=1}^{L}\|\Delta W^i\|_2\right] + 2L_{\mathcal{L}}\sum_{k=1}^{L-1}\|\sigma_k(0)\|\prod_{j=k+1}^{L} L_{\sigma_j}\left[\prod_{j=k+1}^{L}(0+\|\Delta W^j\|_2) - \prod_{j=k+1}^{L}\|\Delta W^j\|_2\right] = 0.$$

Hence,

$$|\mathcal{G}(\mathbf{W}_0+\Delta\mathbf{W}) - \mathcal{G}(\Delta\mathbf{W})| \le 0,$$

and therefore

$$\mathcal{G}(\mathbf{W}_0+\Delta\mathbf{W}) = \mathcal{G}(\Delta\mathbf{W})$$

when $\mathbf{W}_0 = 0$.

Thus, both correction terms vanish when the corresponding perturbation is zero. The upper bounds therefore recover the exact generalization error when the fine-tuned model and the reference model being compared are identical, without leaving an additional nonzero residual term.

D.1.5 Adapting Theorem 1 to the Attention Mechanism

Theorem 1 applies to any architecture that can be written as a composition of linear maps and Lipschitz maps, under bounded input. We therefore view transformer blocks as fitting the theorem. Let embedded inputs be bounded, and let X denote the current input sequence.

The MLP sub-blocks follow the same structure as standard DNNs, so it suffices if their activation functions are Lipschitz Goodfellow et al. (2016). Transformers also include residual connections and normalization. Residual connections take the form $X + F(X)$, which preserves the same decomposition up to a constant He et al. (2016). LayerNorm is applied elementwise across features; under bounded activations, LayerNorm is Lipschitz on the bounded set, and therefore it can be treated as another Lipschitz map in the composition Xu et al. (2019). For the multi-head attention (MHA) sub-blocks, the sequence X is projected into queries, keys, and values such that $Q = XW_Q$, $K = XW_K$, $V = XW_V$, and the attention output is projected by W_O. Each head forms attention weights from $QK^\top$ using softmax and masks and then directly applies them to V Vaswani et al. (2017).

Define the linear map $\mathcal{T}(X) := [XW_Q, |XW_K|XW_V]$ and define $\sigma(\mathcal{T}(X)) = \sigma_{(Q,K,V)}(\cdot) = \mathrm{Softmax}(\frac{QK^\top}{\sqrt{d_k}} + M)V$, where M is an attention mask. Then one attention head can be written as $\mathcal{V}(X) = (\sigma_{(Q,K,V)} \circ \mathcal{T}(X))\, W_O$. We treat $\sigma_{(Q,K,V)}$ as a Lipschitz operator and denote the Lipschitz constant by L_σ. In particular, Softmax is $\frac{1}{2}$-Lipschitz uniformly across all ℓ_p norms Nair (2025).

Consider $\|\sigma(\mathcal{T}_{\mathbf{W}_0+\Delta\mathbf{W}}(X))W_O^{\mathbf{W}_0+\Delta\mathbf{W}} - \sigma(\mathcal{T}_{\mathbf{W}_0}(X))W_O^{\mathbf{W}_0}\|$. We expand it in the same way as in the proof of Theorem 1. Denote $Z_\Delta = \sigma_{(Q,K,V)} \circ (\mathcal{T}_{\mathbf{W}_0+\Delta\mathbf{W}}(X))$ and $Z_0 = \sigma_{(QKV)} \circ (\mathcal{T}_{\mathbf{W}_0}(X))$. Let $W_O^{\mathbf{W}_0+\Delta\mathbf{W}} = W_{O,0} + \Delta W_O$. Then,

$$\begin{aligned}
\|\sigma(\mathcal{T}_{\mathbf{W}_0+\Delta\mathbf{W}}(X))W_O^{\mathbf{W}_0+\Delta\mathbf{W}} - \sigma(\mathcal{T}_{\mathbf{W}_0}(X))W_O^{\mathbf{W}_0}\| &\overset{\text{By construction}}{=} \|Z_\Delta W_O^{\mathbf{W}_0+\Delta\mathbf{W}} - Z_0 W_O^{\mathbf{W}_0}\| \\
&\overset{By\ W_O\ \text{Definition}}{=} \|Z_\Delta(W_{O,0}+\Delta W_O) - Z_0 W_{O,0}\| \\
&\overset{\text{By Triangle Inequality}}{\le} \|(Z_\Delta - Z_0)W_{O,0}\| + \|Z_\Delta \Delta W_O\| \\
&\overset{\text{Inequality (10)}}{\le} \|Z_\Delta - Z_0\|\,\|W_{O,0}\|_2 + \|Z_\Delta\|\,\|\Delta W_O\|_2. \quad (\star)
\end{aligned}$$

Now, we consider bounding the term $\|Z_\Delta - Z_0\|$ as follows:

$$\begin{aligned}
\|Z_\Delta - Z_0\| &\overset{\text{By construction}}{=} \|\sigma(\mathcal{T}_{\mathbf{W}_0+\Delta\mathbf{W}}(X)) - \sigma(\mathcal{T}_{\mathbf{W}_0}(X))\| \\
&\overset{\text{Assumption 3}}{\le} L_\sigma\|\mathcal{T}_{\mathbf{W}_0+\Delta\mathbf{W}}(X) - \mathcal{T}_{\mathbf{W}_0}(X)\|.
\end{aligned}$$

Expanding $\mathcal{T}_{\mathbf{W}_0+\Delta\mathbf{W}}(X) - \mathcal{T}_{\mathbf{W}_0}(X)$ we obtain:

$$\begin{aligned}\mathcal{T}_{\mathbf{W}_0+\Delta\mathbf{W}}(X) - \mathcal{T}_{\mathbf{W}_0}(X) &\overset{\text{By construction}}{=} X(W_{QKV,0} + \Delta W_{QKV}) - XW_{QKV,0} \\ &= X\Delta W_{QKV},\end{aligned}$$

which implies

$$\|\mathcal{T}_{\mathbf{W}_0+\Delta\mathbf{W}}(X) - \mathcal{T}_{\mathbf{W}_0}(X)\| \overset{\text{Inequality 10}}{\leq} \|X\|\,\|\Delta W_{QKV}\|_2. \qquad (\triangle)$$

Finally, we obtain the result:

$$\begin{aligned}&\|\sigma(\mathcal{T}_{\mathbf{W}_0+\Delta\mathbf{W}}(X))W_O^{\mathbf{W}_0+\Delta\mathbf{W}} - \sigma(\mathcal{T}_{\mathbf{W}_0}(X))W_O^{\mathbf{W}_0}\| \\ &\overset{(\star)\text{ and }(\triangle)}{\leq} L_\sigma\|X\|\,\|\Delta W_{QKV}\|_2\,\|W_{O,0}\|_2 + \|Z_\Delta\|\,\|\Delta W_O\|_2.\end{aligned}$$

Hence, the main difference for Theorem 1 between transformers and deep neural networks appears in the MHA block, where subtracting the fine-tuned and pretrained models produces an additional term $\|Z_\Delta\|\|\Delta W_O\|_2$ from the output projection update. This term is controlled under our setup since X is bounded and $V = X(W_V + \Delta W_V)$ is linear, we have $\|V\| \leq \|X\|\|W_V + \Delta W_V\|_2$. Thus $\|Z_\Delta\| = \|\mathrm{Softmax}(\cdot)\mathrm{V}\| \leq \mathrm{C_Z}$ for some constant C_Z depending on the input bound and $\|W_V + \Delta W_V\|_2$.

If W_O is frozen ($\Delta W_O = 0$), the additional term vanishes, and the bound simplifies to $L_\sigma\|X\|\,\|\Delta W_{QKV}\|_2\,\|W_{O,0}\|_2$. This has the same proof structure as in Theorem 1, with $\|W_{O,0}\|_2$ contributing as an additional spectral factor. The attention mask M is fixed, so it does not contribute to the parameter difference. Also, with $W_{QKV} = [W_Q\, W_K\, W_V]$ where we concatenate the projection matrices, we have $\|\Delta W_{QKV}\|_2 \leq \sqrt{\|\Delta W_Q\|_2^2 + \|\Delta W_K\|_2^2 + \|\Delta W_V\|_2^2}$ by properties of block matrices. This leads to the development of Theorem 4.

Theorem 4. ***(Upperbound on self-attention fine-tuned update)*** *Let an input sequence X be bounded, such that $\|X\| \leq C$. Consider one attention head as $\mathcal{V}_{\mathbf{W}}(X) = (\sigma_{(QKV)} \circ \mathcal{T}_{\mathbf{W}}(X))W_O, \mathcal{T}_{\mathbf{W}}(X) := [XW_Q \mid XW_K \mid XW_V]$, where $\sigma_{(QKV)}(\cdot) = \mathrm{Softmax}(\frac{QK^\top}{\sqrt{d_k}} + M)V$ and M is a fixed mask, let $X \in \mathbb{R}^{\hat{n}\times\hat{m}}$. Then the difference in attention-head outputs satisfies*

$$\|\mathcal{V}_{\mathbf{W}_0+\Delta\mathbf{W}}(X) - \mathcal{V}_{\mathbf{W}_0}(X)\| \leq C(L_\sigma\|\Delta W_{QKV}\|_2\|W_{O,0}\|_2 + \sqrt{\hat{n}}\|W_{V,0} + \Delta W_V\|_2\|\Delta W_O\|_2),$$

where ΔW_{QKV} denotes a concatenated update $\Delta W_{QKV} = [\Delta W_Q \mid \Delta W_K \mid \Delta W_V]$, L_σ represents the Lipschitz constant of $\mathrm{Softmax}(\frac{\mathrm{QK^\top + M}}{\sqrt{\mathrm{d_k}}})\mathrm{V}$.

Proof. We have already shown that

$$\|\mathcal{V}_{\mathbf{W}_0+\Delta\mathbf{W}}(X) - \mathcal{V}_{\mathbf{W}_0}(X)\| \leq L_\sigma\|X\|\,\|\Delta W_{QKV}\|_2\,\|W_{O,0}\|_2 + \|Z_\Delta\|\,\|\Delta W_O\|_2.$$

We assume $\|X\| \leq C$, hence

$$\|\mathcal{V}_{\mathbf{W}_0+\Delta\mathbf{W}}(X) - \mathcal{V}_{\mathbf{W}_0}(X)\| \leq CL_\sigma\,\|\Delta W_{QKV}\|_2\,\|W_{O,0}\|_2 + \|Z_\Delta\|\,\|\Delta W_O\|_2.$$

We aim to upper bound Z_Δ; $Z_\Delta = \sigma_{(QKV)} \circ (\mathcal{T}_{\mathbf{W}_0+\Delta\mathbf{W}}(X)) = \mathrm{Softmax}(\frac{QK^\top}{\sqrt{d_k}})V$ which gives

$$\|Z_\Delta\| = \|\mathrm{Softmax}(\frac{QK^\top}{\sqrt{d_k}} + M)V\| \leq \|\mathrm{Softmax}(\frac{QK^\top}{\sqrt{d_k}} + M)\|_2\|V\| \leq \sqrt{\hat{n}}\|V\|$$

$$\sqrt{\hat{n}}\|V\| = \sqrt{\hat{n}}\|X(W_{V,0} + \Delta W_V)\| \leq \sqrt{\hat{n}}\|X\|\|W_{V,0} + \Delta W_V\|_2 \leq C\sqrt{\hat{n}}\|W_{V,0} + \Delta W_V\|_2.$$

To justify the $\sqrt{\hat{n}}$ factor, define the attention weight matrix

$$S := \mathrm{Softmax}(\frac{QK^\top}{\sqrt{d_k}} + M) \in \mathbb{R}^{\hat{n}\times\hat{n}},$$

where Softmax is applied row-wise. Each row of S is a probability vector, hence $S_{ij} \geq 0$ and $\sum_{j=1}^{\hat{n}} S_{ij} = 1$ for all i. Therefore $\|S\|_\infty = 1$. Moreover, since each column sum satisfies $\sum_{i=1}^{\hat{n}} S_{ij} \leq \hat{n}$, we have $\|S\|_1 \leq \hat{n}$. Using the inequality $\|A\|_2 \leq \sqrt{\|A\|_1\|A\|_\infty}$ gives $\|S\|_2 \leq \sqrt{\|S\|_1\|S\|_\infty} \leq \sqrt{\hat{n}}$. Consequently,

$$\|Z_\Delta\| = \|SV\| \leq \|S\|_2\|V\| \leq \sqrt{\hat{n}}\|V\|.$$

Hence, we conclude with

$$\|\mathcal{V}_{\mathbf{W}_0+\Delta\mathbf{W}}(X) - \mathcal{V}_{\mathbf{W}_0}(X)\| \leq C(L_\sigma\|\Delta W_{QKV}\|_2\|W_{O,0}\|_2 + \sqrt{\hat{n}}\|W_{V,0} + \Delta W_V\|_2\|\Delta W_O\|_2).$$

□

D.1.6 ADAPTING THEOREM 1 UNDER SPECIAL CASES

To adapt Theorem 1 under special cases, we need the following general assumptions.

Assumption 4. *The loss function, $\ell(\cdot) : \mathbb{R}^d \to \mathbb{R}$, is 1-Lipschitz, i.e., $|\ell(f_{\boldsymbol{W}}(x), y) - \ell(f_{\boldsymbol{W}'}(x), y)| \leq \|f_{\boldsymbol{W}}(x) - f_{\boldsymbol{W}'}(x)\|$ for all $\boldsymbol{W}$, $\boldsymbol{W}' \in \mathbb{R}^d$ and $(x, y) \in \mathcal{X} \times \mathcal{Y}$.*

Assumption 5. *The loss function, $\ell(\cdot) : \mathbb{R}^d \to \mathbb{R}$, is bounded, i.e., there exists a constant $C_2 \geq 0$ such that $\|\ell(f_{\boldsymbol{W}}(x), y)\| \leq C_2$, for all $\boldsymbol{W} \in \mathbb{R}^d$ and $(x, y) \in \mathcal{X} \times \mathcal{Y}$.*

(*i*) **Perturbing around $\mathcal{G}(\mathbf{W}_0)$.** First, we adapt Theorem 4.1 in Li & Zhang (2021) into our notation and quote it below.

Theorem 5. ***(PAC-Bayes generalization bound for fine-tuning)****[Li & Zhang (2021), Theorem 4.1] Let Assumption 1 hold with the requirement that $C \geq 1$. Let the loss function, $\mathcal{L}$, follow Assumptions 4 and 5. Let $\|W_0^{(i)}\|_2 \leq \mathcal{A}_i$ with fixed $\mathcal{A}_i > 1$, $\|\Delta W^{(i)}\| \leq Q_i$, for all $i \in [L]$ and $V = \max_{i\in[L]}\{m_i, n_i\}$. Let ϵ and δ be arbitrary small values. Then with probability $1 - 2\delta$, the following inequality holds:*

$$\mathcal{G}(\mathbf{W}_0+\Delta\mathbf{W}) \leq \epsilon+C_2\sqrt{\frac{\frac{36}{\epsilon^2}C^2V\log(4LVC_2)(\sum_{i=1}^{L}\frac{\prod_{j=1}^{L}(\mathcal{A}_j+Q_j)}{\mathcal{A}_i+Q_i})^2(\sum_{i=1}^{L}Q_i^2)+3\ln\frac{|N|}{\delta}+8}{|N|}}.$$

We now use Theorem 5 to obtain a bound for $\mathcal{G}(\mathbf{W}_0)$. The following Theorem gives that.

Theorem 6. *Using the Assumptions made for Theorem 1 and Theorem 5, the following inequality holds with probability at least $1 - 2\delta$:*

$$\mathcal{G}(\boldsymbol{W}_0 + \Delta\boldsymbol{W}) \leq \epsilon + C_2\sqrt{\frac{3\ln\frac{|N|}{\delta}+8}{|N|}} + \Phi_{\Delta\boldsymbol{W}}.$$

Proof. We wish to find $\mathcal{G}(\mathbf{W}_0)$, and note that if we never train the model, we obtain the expression $\|W_0^i - W_0^i\|_2 = 0$. Thus, we can use $Q_i = 0$ for all $i \in [L]$ and obtain:

$$\begin{aligned}
\mathcal{G}(\mathbf{W}_0) &\overset{\text{Theorem 5}}{\leq} \epsilon + C_2\sqrt{\frac{\frac{36}{\epsilon^2}C^2V\log(4LVC_2)(\sum_{i=1}^{L}\frac{\prod_{j=1}^{L}(\mathcal{A}_j+Q_j)}{\mathcal{A}_i+Q_i})^2(\sum_{i=1}^{L}Q_i^2)+3\ln\frac{|N|}{\delta}+8}{|N|}} \\
&\overset{Q_i=0; i\in[L]}{=} \epsilon + C_2\sqrt{\frac{\frac{36}{\epsilon^2}C^2V\log(4LVC_2)(\sum_{i=1}^{L}\frac{\prod_{j=1}^{L}(\mathcal{A}_j+0)}{\mathcal{A}_i+0})^2(\sum_{i=1}^{L}0^2)+3\ln\frac{|N|}{\delta}+8}{|N|}} \\
&= \epsilon + C_2\sqrt{\frac{3\ln\frac{|N|}{\delta}+8}{|N|}}.
\end{aligned}$$

Now that we have an upper bound for $\mathcal{G}(\mathbf{W}_0)$, we can apply Theorem 1 and obtain the following:

$$\begin{aligned}
\mathcal{G}(\mathbf{W}_0 + \Delta\mathbf{W}) &\overset{\text{Theorem 1}}{\leq} \mathcal{G}(\mathbf{W}_0) + \Phi_{\Delta\mathbf{W}} \\
&\leq \epsilon + C_2\sqrt{\frac{3\ln\frac{|N|}{\delta}+8}{|N|}} + \Phi_{\Delta\mathbf{W}}.
\end{aligned}$$

This concludes the proof. □

(*i*) **Perturbing around $\mathcal{G}(\mathcal{A})$.** First, we make another assumption on the loss function and then adapt Theorem 1 in Xu & Raginsky (2017) to our notation.

Assumption 6. *The loss function, $\ell(\cdot) : \mathbb{R}^d \to \mathbb{R}$, is σ-sub-gaussian, i.e., $\mathbb{E}(e^{\lambda[\ell(f_{\mathbf{W}}(X),Y)-\mathbb{E}(\ell(f_{\mathbf{W}}(X),Y))]}) \leq e^{\frac{\lambda^2\sigma^2}{2}}$ for all $\lambda \in \mathbb{R}$, $\mathbf{W} \in \mathbb{R}^d$.*

Theorem 7. ***(Upper bound on generalization error using mutual information)****[Theorem 1 Xu & Raginsky (2017)] Let $\mathcal{A}$ denote a LoRA-based algorithm that outputs $\{\boldsymbol{\Delta}\mathbf{W}_i\}_{i\in[L]}$ on a fine-tuning dataset, N. By ν we denote the underlying distribution of the input space, $\mathcal{X}$, of which the elements*

of the fine-tuning dataset N are chosen following i.i.d. Let Assumption 6 hold. Then we have the following:

$$\mathcal{G}(\mathcal{A})_\nu \leq \sqrt{\frac{2\sigma^2 \mathbf{I}(\{\mathbf{\Delta W}_i\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W})}{|N|}}.$$

Let the loss function $\mathcal{L}$ follow Assumption 6. We present the generalization error upper bounds of the LoRA variants in Table 1. For this, we use the inequality $\mathcal{G}(\mathbf{W}_0 + \mathcal{A}) \leq \mathcal{G}(\mathcal{A}) + \Phi_{\mathbf{W}_0}$, where $\mathcal{G}(\mathcal{A})$ is upper bounded by the use of Lemma 1 quoted below.

Lemma 1. *(Upperbound on mutual-information)[Xu & Raginsky (2017)] Let $\{\mathbf{\Delta W}_i\}_{i\in[L]}$ be an update to a learning algorithm. Then the mutual information is upper bounded by the uniform distribution over an updated support set, i.e., $\mathbf{I}(\Delta\{\mathbf{W}_i\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W}) \leq \ln 2^{qp} = qp \ln 2$, where q represents the number of bits the learning algorithm is designed on, and p is the number of trainable parameters. Thus, with the use of Theorem 7, if Assumption 6 holds, then $\mathcal{G}(\mathcal{A}) \leq \sqrt{\frac{2\sigma^2 qp \ln 2}{|N|}}$.*

How do we arrive at the bounds of different LoRA variants?

(*a*) **LoRA+** has $\mathcal{G}(\mathcal{A})$ upper bounded by $\sqrt{\frac{2rq\sigma^2 \ln 2 \sum_{i=1}^{L}(m_i+n_i)}{|N|}}$. The learning rate does not alter the number of trainable parameters, which leads LoRA+ to possess the same upper bounds as LoRA. We note a unique observation regarding this claim, as $\gamma_A \to 0$, LoRA+ takes the lowered generalization error bound of Asymmetric LoRA since the adapter matrix, A, is no longer trainable.

(*b*) **cLA** has the fine-tuned update $B[I_r|\mathbf{0}_{m_i-r}]$, where $[I_r|\mathbf{0}_{m_i-r}]$ is a fixed constant orthogonal matrix. Thus, by using data processing inequality (13), the mutual information between the two is preserved, i.e.,

$$\mathbf{I}(\{B_i[I_r|\mathbf{0}_{m_i-r}]_i\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W}) = \mathbf{I}(\{B_i\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W}).$$

Similar to Zhu et al. (2024), we upper bound mutual information by the uniform distribution of a model's support; particularly $\mathbf{I}(\{B_i\}_{i\in[L]}; N|\mathbf{\Delta W}; \mathbf{W}) \leq qr \ln 2 \sum_{i=1}^{L} n_i$, by Lemma 1. Finally, by Theorem 7, we obtain the result $\mathcal{G}(\mathcal{A}) \leq \sqrt{\frac{2rq\sigma^2 \ln 2 \sum_{i=1}^{L} n_i}{|N|}}$.

(*c*) **c^{3}LA** has the fine-tuned update $B_1[I_r|\mathbf{0}_{m_i-r}]+B_2[\mathbf{0}_r|I_r|\mathbf{0}_{m_i-2r}]+\cdots+B_k[\mathbf{0}_{r(k-1)}|I_r|\mathbf{0}_{m_i-kr}]$. This expansion can be simplified by $\sum_{j=1}^{k} B_j[0_{r\times r(j-1)} \mid I_r \mid 0_{r\times(m_i-rj)}] = [B_1 \mid B_2 \mid \cdots \mid B_k|\mathbf{0}_{n_i(m_i-kr)}]$. Using (12), we can upper bound the rank of $\sum_{j=1}^{k} B^j[0_{r\times r(j-1)} \mid I_r \mid 0_{r\times(m_i-rj)}]$ by kr. Thus, the mapping $[B_1|\cdots|B_k] \to \mathbf{\Delta W}$ is injective and can be inverted by slicing the last $n_i - kr$ columns. Using DPI, this leads to the expression

$$\mathbf{I}(\{\sum_{j=1}^{k} B_i^j[0_{r\times r(j-1)} \mid I_r \mid 0_{r\times(m_i-rj)}]\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W}) = \mathbf{I}(\{[B_1|\cdots|B_k]_i\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W}).$$

We upper bound $\mathbf{I}(\{[B_1|\cdots|B_k]_i\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W})$ by $qrk \ln 2 \sum_{i=1}^{L} n_i$, using Lemma 1. Hence, by Theorem 7, we obtain: $\mathcal{G}(\mathcal{A}) \leq \sqrt{\frac{2rq\sigma^2 k \ln 2 \sum_{i=1}^{L} n_i}{|N|}}$.

(*d*) **CoLA** has the update structure $\mathbf{\Delta W} = \sum_{j=1}^{k} B^j A^j$. Using inequality (12), we upper bound the rank of each layer's update by kr. By Lemma 1, we upper bound $\mathbf{I}(\{\sum_{j=1}^{L} B_i^j A_i^j\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W})$ by $qrk \ln 2 \sum_{i=1}^{L}(m_i + n_i)$. Hence, we obtain $\mathcal{G}(\mathcal{A}) \leq \sqrt{\frac{2rq\sigma^2 k \ln 2 \sum_{i=1}^{L}(m_i+n_i)}{|N|}}$, by Theorem 7.

(*e*) **RAC-LoRA** has the fine-tuned update $\sum_{j=1}^{k} B^j Q^j$, where we consider each Q^j to be a frozen orthogonal matrix. This update can be represented by $\sum_{j=1}^{k} B^j Q^j = [B^1|B^2|\cdots|B^k][Q^1|Q^2|\cdots|Q^k]^T$, where we can invert $[B^1|B^2|\cdots|B^k][Q^1|Q^2|\cdots|Q^L]^T]$ to $[B^1|B^2|\cdots|B^L]$. Thus by using inequality (12), DPI, and Lemma 1 we have $\mathbf{I}(\{[B^1|B^2|\cdots|B^L]_i[Q^1|Q^2|\cdots|Q^L]_i^T]\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W}) = \mathbf{I}(\{B^1|B^2|\cdots|B^L]_i]\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W})$, which is

$$\mathbf{I}(\{B^1|B^2|\cdots|B^L]_i]\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W}) \leq qrk \ln 2 \sum_{i=1}^{L} n_i.$$

Hence, by Theorem 7, we have the result: $\mathcal{G}(\mathcal{A}) \leq \sqrt{\frac{2rq\sigma^2 k \ln 2 \sum_{i=1}^{L} n_i}{|N|}}$.

Generalization Upperbound on PaCA

(f) **PaCA** has an identical update to cLA if the fine-tuned columns are first r columns of the pretrained backbone. This suggests that any generalization guarantee driven by the update motivates the same generalization behavior for PaCA as cLA. Particularly, since PaCA updates $r \sum_{i=1}^{L} n_i$ trainable entries (the same degrees of freedom as cLA's B matrices), Lemma 1 yields $I(\{\Delta W_i\}_{i\in[L]}; N|\mathcal{A}; \mathbf{W}) \leq qr \ln 2 \sum_{i=1}^{L} n_i$. Plugging into Theorem 7, we get $\mathcal{G}(\mathcal{A}) \leq \sqrt{\frac{2rq\sigma^2 \ln 2 \sum_{i=1}^{L} n_i}{|N|}}$, matching the generalization upper bound of cLA from Table 1.

E Addendum to Experimental Results

In §E.1, we summarize the quality metrics and trainable parameters used for training the models in Table 16 and provide the specific hyperparameters for fine-tuning each model for each dataset in Table 6. In §E.2, we present ablation studies on the effects of learning rate (γ), scaling factor (α), and chain reset indices on the resulting test accuracy and test loss for varying ranks. In §E.3, we comment on the potential of our methods by leveraging the computational benefits of the column-restricted variants. In §E.4.2 and §E.4.1, we extend 4.2.1 with the implementation details of the loss landscapes and provide additional loss landscapes and intruder dimension results. In §E.5, we extend §4.2 with empirical results on generalization.

E.1 Implementation Details

Table 5 summarizes the benchmarks, quality metrics, and trainable parameters of the models involved in this paper.

Table 5: **Summary of the benchmarks, quality metrics, and trainable parameters for our main experiments.** For LoRA and Asymmetric LoRA methods, we report their percentage of trainable parameters relative to FFT. Note that for Asymmetric methods we double the rank compared to LoRA to more closely match trainable parameters, this will not be a perfect match for non-square query and value matrices as observed in TinyLlama and LLaMA-3-8B.

Task	Model	Pretrained On	Fine-Tuned On	Trainable (FFT)	LoRA (%)	Asym (%)	Quality Metric
Natural Language Processing	RoBERTa-Base	English language corpora	MRPC	124.647M	0.948	0.948	F1
			CoLA	124.647M	0.948	0.948	MCC
	RoBERTa-Large	English language corpora	MRPC	355.362M	0.739	0.739	F1
			CoLA	355.362M	0.739	0.739	MCC
	DeBERTa v2 XXL	English language corpora	MRPC	1.567B	0.301	0.301	Accuracy
			TREC-50	1.567B	0.306	0.306	Accuracy
			PAWS	1.567B	0.301	0.301	Accuracy
	DeBERTa v3 Base	English language corpora	MRPC	184.424M	0.321	0.321	Accuracy
			RTE	184.424M	0.321	0.321	Accuracy
			STS-B	184.423M	0.320	0.320	Pearson
			TREC-50	184.461M	0.341	0.341	Accuracy
			PAWS	184.424M	0.321	0.321	Accuracy
	GPT2-Small	WebText	E2E	124.440M	6.169	6.169	Perplexity
Image Classification	ViT-Tiny	ImageNet-1K	Office-Home	5.537M	2.890	2.890	Accuracy
			CIFAR-10	5.526M	2.703	2.703	Accuracy
	ViT-Base	ImageNet-21K then ImageNet-1K	Office-Home	85.849M	0.745	0.745	Accuracy
			CIFAR-10	85.806M	0.696	0.696	Accuracy
Coding Generation	DeepSeek-Coder-Base	Repo-Level Code Corpus	Django	1.346B	0.234	0.234	Exact Match
Logical Reasoning	TinyLlama	SlimPajama	OpenBookQA	1.035B	0.219	0.158	Accuracy
			FOLIO	1.035B	0.218	0.157	Accuracy
			LogiQA	1.035B	0.219	0.158	Accuracy
			CLUTRR	1.035B	0.222	0.161	Accuracy
	LlaMA-3-8B	Large-scale multilingual corpora	OpenBookQA	7.505B	0.091	0.070	Accuracy
			CLUTRR	7.505B	0.092	0.071	Accuracy

We implement the framework in Python using PyTorch Paszke et al. (2019). We train all models with the ADAM optimizer Kingma & Ba (2014). Training (of most models) was performed on one 80GB NVIDIA H100 GPU. The ablation studies on ViT-Tiny in Tables 8, 10, and 12 were trained using

one NVIDIA V100 GPU. We provide the hyperparameter settings, i.e., learning rates, learning rate scheduler, chain reset frequency, weight decay, batch size, training epochs, maximum token length or image resolution, and random seeds for the experiments in Table 6.

Table 6: **Summary of the main-table hyperparameters.** Learning rates are listed as (FFT, LoRA, Asym). The LoRA group comprises of LoRA, CoLA, and LoRA+ ($r = 16$); the Asym group comprises of Asymmetric LoRA, RAC-LoRA, cLA, c^3LA, r-cLA, and r-c^3LA ($r = 32$). Both adapter groups use the same base learning rate and $\alpha = 2r$, giving $\alpha/r = 2$; FFT has no adapter rank. All experiments use Adam with $(\beta_1, \beta_2, \epsilon) = (0.9, 0.999, 1e^{-8})$. Reported checkpoints maximize the validation task metric (accuracy, F1, MCC, Pearson correlation, or exact match), except E2E, which minimizes validation perplexity. Chain frequency is measured in epochs and applies only to CoLA, RAC-LoRA, c^3LA, and r-c^3LA; scheduled learning rates are preserved across resets. Weight decay is listed as (FFT, all adapter methods). Epochs specify the training budget. ViT, RoBERTa, and GPT2 use global L_2 gradient clipping at 1; the other trainers do not clip gradients. For LoRA+, $\eta_B = 16\eta_A$: the listed adapter rate is η_A, except DeBERTa-v3-Base/TREC-50, where it is η_B and $\eta_A = \eta_B/16$. FFT classifier-head groups in the ViT/RoBERTa trainer use ten times the listed FFT base rate.

Model	Dataset	Scheduler (Warmup LR, Ratio)	Learning Rates (FFT, LoRA, Asym)	Chain reset frequency	Weight decay (FFT, adapters)	Batch size	Epochs	Max length or Image size	Seeds
RoBERTa-Base	MRPC	Linear($1e^{-6}$, 0.1)	($1e^{-5}$, $3e^{-4}$, $3e^{-4}$)	3	(0.01, 0)	32	20	128	(0, 1, 2, 3, 4)
	CoLA	Linear($1e^{-6}$, 0.1)	($1e^{-5}$, $3e^{-4}$, $3e^{-4}$)	3	(0.01, 0)	32	20	128	(0, 1, 2, 3, 4)
RoBERTa-Large	MRPC	Linear($1e^{-6}$, 0.1)	($1e^{-5}$, $3e^{-4}$, $3e^{-4}$)	3	(0.01, 0)	32	20	128	(0, 1, 2, 3, 4)
	CoLA	Linear($1e^{-6}$, 0.1)	($1e^{-5}$, $3e^{-4}$, $3e^{-4}$)	3	(0.01, 0)	32	20	128	(0, 1, 2, 3, 4)
DeBERTa v2 XXL	MRPC	Constant	($1e^{-5.5}$, $1e^{-4.5}$, $1e^{-4.5}$)	5	0	8	25	512	(0, 1, 2, 3, 4)
	TREC-50	Constant	($1e^{-5.5}$, $1e^{-4.5}$, $1e^{-4.5}$)	5	0	8	25	512	(0, 1, 2, 3, 4)
	PAWS	Constant	($1e^{-6.5}$, $1e^{-4.5}$, $1e^{-4.5}$)	5	0	8	10	512	(0, 1, 2, 3, 4)
DeBERTa v3 Base	MRPC	Constant	($1e^{-5}$, $1e^{-3.5}$, $1e^{-3.5}$)	5	0	8	40	512	(0, 1, 2, 3, 4)
	RTE	Constant	($1e^{-4.75}$, $1e^{-3.5}$, $1e^{-3.5}$)	5	0	8	40	512	(0, 1, 2, 3, 4)
	STS-B	Constant	($1e^{-4.75}$, $1e^{-3.5}$, $1e^{-3.5}$)	5	0	8	40	512	(0, 1, 2, 3, 4)
	TREC-50	Constant	($1e^{-4.75}$, $1e^{-3.25}$, $1e^{-3.25}$)	5	0	8	40	512	(0, 1, 2, 3, 4)
	PAWS	Constant	($1e^{-5}$, $1e^{-3.5}$, $1e^{-3.5}$)	5	0	8	20	512	(0, 1, 2, 3, 4)
GPT2-Small	E2E	Linear($1e^{-6}$, 0.1)	($5e^{-5}$, $3e^{-4}$, $3e^{-4}$)	1	(0.01, 0)	16	30	64	(0, 1, 2, 3, 4)
ViT-Tiny	Office-Home	Cosine($1e^{-6}$, 0.05)	($3e^{-4}$, $1e^{-3}$, $1e^{-3}$)	5	(0.05, 0)	64	30	224	(0, 1, 2, 3, 4)
	CIFAR-10	Cosine($1e^{-6}$, 0.05)	($3e^{-4}$, $1e^{-3}$, $1e^{-3}$)	5	(0.05, 0)	64	30	224	(0, 1, 2, 3, 4)
ViT-Base	Office-Home	Cosine($1e^{-6}$, 0.05)	($3e^{-4}$, $1e^{-3}$, $1e^{-3}$)	5	(0.05, 0)	64	30	224	(0, 1, 2, 3, 4)
	CIFAR-10	Cosine($1e^{-6}$, 0.05)	($3e^{-4}$, $1e^{-3}$, $1e^{-3}$)	5	(0.05, 0)	64	30	224	(0, 1, 2, 3, 4)
DeepSeek-Coder Base	Django	Constant	($1e^{-5.5}$, $1e^{-4.5}$, $1e^{-4.5}$)	1	0	8	5	512	(0, 1, 2, 3, 4)
TinyLlama	OpenBookQA	Constant	($1e^{-6.25}$, $1e^{-3.75}$, $1e^{-3.75}$)	2	0	8	10	512	(0, 1, 2, 3, 4)
	FOLIO	Constant	($1e^{-5}$, $1e^{-3.75}$, $1e^{-3.75}$)	2	0	8	10	512	(0, 1, 2, 3, 4)
	LogiQA	Constant	($1e^{-5.75}$, $1e^{-4}$, $1e^{-4}$)	2	0	8	10	512	(0, 1, 2, 3, 4)
	CLUTRR	Constant	($1e^{-6.25}$, $1e^{-5.25}$, $1e^{-5.25}$)	2	0	8	10	512	(0, 1, 2, 3, 4)
Llama 3 8B	OpenBookQA	Constant	($1e^{-5.25}$, $1e^{-3.75}$, $1e^{-3.75}$)	2	0	4	5	384	(0, 1, 2, 3, 4)
	CLUTRR	Constant	($1e^{-5.25}$, $1e^{-4.25}$, $1e^{-4.25}$)	2	0	4	5	384	(0, 1, 2, 3, 4)

E.2 The Effects of Learning Rate, Scaling Factor, and Chain Reset Frequency on Performance Metrics Over Various Ranks

The ideal learning rate of an LLM tends to scale inversely with its size Li et al. (2025). Many papers suggest a default scaling factor of $2r$ Biderman et al. (2024); Shuttleworth et al. (2025). Malinovsky et al. (2024) suggests that for sufficiently low learning rates, performing a chain reset every epoch is optimal. We validate the first claim under LoRA fine-tuning methods via ablation studies over learning rates presented in Tables 7-8. Similarly, we assess the scaling factor baseline choice in Tables 9 and 10, and the optimal chain reset frequency in Table 11. For the ablation studies, we fine-tuned DeBERTaV3-Base on the MRPC, TREC-50, and PAWS for learning rate, MRPC and TREC-50 for scaling factor, and MRPC, CoLA, RTE, and TREC-50 for chain reset over various ranks. We then re-ran the same experiments on ViT-Tiny fine-tuned on the Office-Home and CIFAR-10 datasets. We ran for 30 epochs.

Table 7: Test accuracies obtained by fine-tuning DeBERTa v3 on MRPC, TREC-50, and PAWS varying learning rates (columns), ranks (rows), and LoRA PEFT methods. We center our search at $1e^{-4}$. The learning rate for all methods decreases with increasing rank; the relationship between learning rate and model size observed in LLMs Li et al. (2025) persists when fine-tuning via LoRA methods. Chain methods and their non-chain counterparts produce the best results in similar learning rate ranges; therefore, chain resets do not influence the optimal learning rate. We repeated the experiment with ViT-Tiny on Table 8. Light-gray subtables identify cLA and c^{3}LA. Red marks a strictly higher displayed accuracy than Asym for cLA, or than RAC for c^{3}LA, at the same model, dataset, rank, and column setting. Ties are not highlighted.

DeBERTa v3 LoRA MRPC

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	66.4	66.4	79.9	84.2	85.5	87.3	88.1	66.4	66.4
4	66.4	74.4	81.8	84.6	85.6	87.9	87.7	66.4	66.4
8	66.4	76.1	83.0	85.1	86.9	87.3	87.1	66.4	66.4
16	66.4	78.1	83.1	85.0	86.8	87.9	72.9	66.4	66.4
32	66.4	80.1	83.9	84.9	87.1	87.9	66.4	66.4	66.4
64	76.1	81.7	84.6	85.1	88.0	81.5	66.4	66.4	66.4
128	77.5	81.9	84.9	86.2	88.5	87.6	66.4	66.4	66.4

DeBERTa v3 Asym MRPC

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	66.4	66.4	66.4	68.1	81.4	85.3	86.1	86.9	86.0
4	66.4	66.4	66.4	76.0	82.9	84.2	85.7	86.8	85.6
8	66.4	66.4	66.5	80.6	84.2	85.2	86.5	86.3	72.3
16	66.4	66.4	76.7	82.6	84.3	86.2	86.8	87.3	66.4
32	66.4	66.6	79.6	83.1	84.6	86.1	87.3	75.0	66.4
64	66.4	77.7	81.7	82.5	84.7	86.1	87.4	66.4	66.4
128	69.0	79.6	82.2	84.3	86.1	80.1	66.4	66.4	66.4

DeBERTa v3 cLA MRPC

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	66.4	66.4	66.4	71.0	82.4	84.0	85.0	83.1	80.4
4	66.4	66.4	70.2	79.0	84.9	85.5	85.9	85.5	66.4
8	66.4	66.4	73.7	81.5	84.6	84.5	85.2	85.5	66.4
16	66.4	67.0	76.1	81.9	83.3	84.4	85.3	78.8	66.4
32	66.4	74.4	79.6	80.8	83.2	85.0	85.2	66.4	66.4
64	66.4	77.0	80.8	81.8	84.5	85.1	86.7	66.4	66.4
128	71.3	78.6	79.5	82.0	85.5	87.1	84.8	66.4	66.4

DeBERTa v3 CoLA MRPC

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	66.5	66.5	75.9	82.2	85.7	87.7	88.3	66.5	66.5
4	66.5	66.5	76.7	82.7	85.1	87.8	87.2	66.5	66.5
8	66.5	66.5	79.1	84.0	86.8	88.3	86.9	66.5	66.5
16	66.5	75.3	80.8	84.0	86.4	88.9	79.6	66.5	66.5
32	66.5	76.8	82.3	83.9	86.5	88.0	66.5	66.5	66.5
64	69.3	79.6	83.3	85.7	87.1	88.5	66.5	66.5	66.5
128	75.8	80.7	83.1	85.9	88.2	88.0	66.5	66.5	66.5

DeBERTa v3 RAC MRPC

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	66.5	66.5	66.5	66.5	66.5	76.5	82.0	87.0	86.2
4	66.5	66.5	66.5	66.5	74.0	79.4	84.6	85.4	85.5
8	66.5	66.5	66.5	66.5	76.9	82.8	86.3	87.5	76.7
16	66.5	66.5	66.5	74.1	78.8	83.0	86.9	87.2	66.5
32	66.5	66.5	66.5	76.7	83.9	86.3	87.7	72.4	66.5
64	66.5	66.5	74.2	80.4	84.1	87.9	87.3	66.5	66.5
128	66.5	66.5	76.1	82.3	86.4	87.6	72.0	66.5	66.5

DeBERTa v3 c^3LA MRPC

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	66.5	66.5	66.5	66.5	73.0	80.1	85.6	86.4	72.3
4	66.5	66.5	66.5	66.8	74.6	83.1	85.7	87.2	66.5
8	66.5	66.5	66.5	72.6	79.5	86.2	86.8	85.5	66.5
16	66.5	66.5	66.5	75.6	82.8	86.0	86.5	78.6	66.5
32	66.5	66.5	71.9	79.1	84.5	86.8	86.8	66.5	66.5
64	66.5	66.5	73.9	82.4	86.3	87.2	86.4	66.5	66.5
128	66.5	66.5	81.9	86.3	87.9	86.7	72.3	66.5	66.5

DeBERTa v3 LoRA TREC-50

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	3.2	10.9	10.9	39.1	59.5	76.6	86.9	10.9	10.9
4	10.1	10.9	10.9	42.3	70.6	82.3	87.7	10.9	10.9
8	10.9	10.9	10.9	50.0	70.6	84.7	90.1	10.9	10.9
16	1.4	10.9	10.9	50.0	73.0	89.3	88.3	10.9	10.9
32	1.4	10.9	42.9	59.5	76.4	89.1	10.9	10.9	10.9
64	10.9	10.9	48.2	66.1	82.9	87.1	10.9	10.9	10.9
128	10.9	10.9	58.1	71.6	86.1	10.9	10.9	10.9	10.9

DeBERTa v3 Asym TREC-50

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	10.9	10.9	10.9	32.5	46.2	80.6	86.5	82.5	26.6
4	10.9	10.9	10.9	33.3	58.9	85.1	88.1	85.7	10.9
8	10.9	10.9	10.9	40.1	73.6	87.3	86.7	84.5	10.9
16	10.9	10.9	10.9	42.9	78.8	89.1	86.9	82.7	10.9
32	10.9	10.9	10.9	57.5	83.1	90.9	91.7	56.5	10.9
64	10.9	10.9	41.9	73.0	88.5	90.7	87.5	10.9	10.9
128	10.9	10.9	52.2	78.8	89.9	90.9	10.9	10.9	10.9

DeBERTa v3 cLA TREC-50

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	10.9	10.9	10.9	10.9	10.9	40.9	71.2	81.3	34.5
4	9.5	10.9	10.9	10.9	35.3	61.9	79.6	82.7	10.9
8	0.4	10.9	10.9	10.9	53.0	71.8	83.1	86.3	10.9
16	10.9	10.9	10.9	40.7	60.7	84.3	85.5	87.5	10.9
32	10.1	10.1	10.9	47.0	70.0	86.1	88.9	10.9	10.9
64	10.9	10.9	42.7	62.3	76.2	86.9	89.1	52.4	10.9
128	3.6	10.9	50.2	67.1	85.1	86.7	66.7	10.9	10.9

DeBERTa v3 CoLA TREC-50

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	10.9	10.9	42.1	54.4	71.8	88.1	89.1	10.9	10.9
4	10.9	10.9	42.7	58.3	81.7	84.5	88.3	10.9	10.9
8	10.9	10.9	42.9	65.5	82.3	87.1	90.9	10.9	10.9
16	10.9	10.9	39.9	66.7	84.9	87.1	68.1	10.9	10.9
32	10.9	26.8	53.2	71.4	85.3	86.7	10.9	10.9	10.9
64	10.9	37.5	58.5	75.6	86.9	10.9	10.9	10.9	10.9
128	10.9	43.5	66.1	82.7	86.7	10.9	10.9	10.9	10.9

DeBERTa v3 RAC TREC-50

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	10.9	10.9	10.9	10.9	34.9	46.6	72.8	85.9	87.5
4	1.2	10.9	10.9	10.9	38.5	59.7	82.9	88.3	89.5
8	10.9	10.9	10.9	10.9	43.8	68.1	82.9	88.1	72.4
16	10.1	10.9	10.9	13.3	59.5	78.0	87.1	90.3	10.9
32	10.9	10.9	10.9	45.6	70.6	84.9	88.5	88.1	10.9
64	10.9	10.9	34.9	50.6	74.2	86.7	87.7	10.9	10.9
128	2.0	10.9	42.9	62.3	84.5	88.9	87.3	10.9	10.9

DeBERTa v3 c^3LA TREC-50

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	10.9	10.9	10.9	34.7	42.3	79.2	66.7	62.1	10.9
4	10.9	10.9	19.4	34.7	56.9	86.5	87.7	73.8	10.9
8	10.9	10.9	20.6	36.9	68.7	87.3	78.8	66.7	10.9
16	10.9	10.9	10.9	43.8	76.8	88.5	83.9	71.0	10.9
32	10.9	10.9	38.7	58.1	84.3	88.1	80.4	34.3	10.9
64	10.9	10.9	45.8	74.2	85.9	89.1	80.4	10.9	10.9
128	10.9	35.1	56.5	79.2	85.9	90.9	10.9	10.9	10.9

DeBERTa v3 LoRA PAWS

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	92.1	93.7	94.0	94.2	94.7	94.5	94.0	55.8	55.8
4	92.3	94.1	94.0	94.4	94.2	94.8	94.0	55.8	55.8
8	92.4	93.5	94.3	94.3	94.7	94.5	93.5	55.8	55.8
16	93.1	94.0	94.4	94.6	94.6	93.6	55.8	55.8	55.8
32	93.8	93.9	94.7	94.5	94.7	55.8	55.8	55.8	55.8
64	93.6	94.1	94.6	94.6	94.6	93.5	55.8	55.8	50.0
128	94.0	94.2	94.4	94.7	94.7	55.8	55.8	55.8	50.0

DeBERTa v3 Asym PAWS

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	55.8	55.8	86.9	92.3	93.7	93.9	93.9	94.4	93.4
4	55.8	55.8	90.3	92.5	93.2	94.0	94.4	94.2	92.7
8	55.8	55.8	92.3	93.1	94.1	94.7	94.7	94.2	55.8
16	55.8	55.8	92.4	94.1	94.0	94.5	94.6	94.0	55.8
32	55.8	92.9	93.6	94.4	94.4	94.6	94.1	92.9	55.8
64	55.8	92.5	94.1	94.1	94.8	94.8	93.4	55.8	55.8
128	91.7	92.9	94.1	94.4	94.6	94.7	91.8	55.0	44.2

DeBERTa v3 cLA PAWS

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	55.8	55.8	55.8	90.5	92.5	94.0	93.8	93.7	55.8
4	55.8	55.8	89.3	90.9	93.1	93.8	94.0	93.7	55.8
8	55.8	55.8	89.7	92.6	92.9	94.3	93.9	93.7	55.8
16	55.8	55.8	91.5	93.0	93.7	94.5	94.1	55.8	55.8
32	55.8	89.5	91.6	93.4	93.8	94.2	94.1	55.8	55.8
64	55.8	90.0	93.2	93.8	93.9	94.3	93.5	55.8	55.8
128	87.2	92.5	93.6	93.9	94.7	94.2	55.8	55.8	55.8

DeBERTa v3 CoLA PAWS

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	55.8	89.8	92.5	93.9	94.6	94.7	94.2	55.8	55.8
4	55.8	90.6	92.9	94.0	94.8	94.8	94.0	55.8	55.8
8	55.8	89.3	93.2	94.1	94.3	94.1	93.0	55.8	55.8
16	55.8	91.8	93.7	94.3	94.5	94.7	55.8	55.8	55.8
32	89.9	92.9	94.3	94.5	94.3	94.8	55.8	55.8	55.8
64	90.5	93.3	94.5	94.5	94.8	93.2	55.8	55.8	55.8
128	91.8	93.8	94.7	95.1	95.0	92.7	55.8	55.8	44.2

DeBERTa v3 RAC PAWS

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	55.8	55.8	55.8	89.0	93.3	94.1	93.8	94.4	93.4
4	55.8	55.8	91.0	93.0	93.5	93.9	94.4	93.8	90.6
8	55.8	55.8	89.4	93.4	93.8	94.5	94.2	94.3	88.9
16	55.8	55.8	92.6	92.8	94.2	95.1	94.7	93.5	55.8
32	55.8	91.0	92.6	93.8	94.2	94.0	94.7	55.8	55.8
64	55.8	92.7	93.5	94.3	94.8	94.5	93.8	55.8	55.8
128	91.8	93.1	94.2	94.3	94.6	94.6	55.8	55.8	55.8

DeBERTa v3 c^3LA PAWS

Rank/LR	1e-6	1e-5.5	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2
2	55.8	55.8	55.8	91.4	93.3	93.8	93.9	93.9	55.8
4	55.8	55.8	89.8	91.2	93.7	94.0	94.0	93.3	55.8
8	55.8	55.8	91.8	93.7	93.9	94.7	93.5	93.3	55.8
16	55.8	55.8	92.6	93.5	94.1	94.1	93.7	55.8	55.8
32	55.8	92.0	93.2	94.0	94.1	94.4	93.9	55.8	55.8
64	91.3	92.2	93.8	94.1	93.9	94.2	93.7	55.8	55.8
128	92.5	93.3	94.0	94.3	94.8	55.8	55.8	55.8	55.8

As shown in Tables 7 and 8, Asymmetric LoRA methods are more sensitive to varying learning rates than methods that train both matrices B, A. We notice that the cLA has a wide variety of acceptable learning rates. Across varying ranks, cLA and c^3LA sometimes outperform their parent variant on the same learning rate and rank combination (as indicated by the red text).

Table 8: Test accuracies obtained by fine-tuning ViT-Tiny on CIFAR-10 and Office-Home over varying learning rates (columns), ranks (rows), and LoRA PEFT methods. We center our search at $1e^{-3}$. Consistent with the results of 7, the learning rate for all methods decreases with increasing rank. Chain methods and their non-chain counterparts produce the best results in similar learning rate ranges. Light-gray subtables identify cLA and c^{3}LA. Red marks a strictly higher displayed accuracy than Asym for cLA, or than RAC for c^{3}LA, at the same model, dataset, rank, and column setting. Ties are not highlighted.

ViT-Tiny LoRA CIFAR-10

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	89.08	90.91	92.85	92.92	93.56	91.76	87.13	49.75	10.70
4	89.38	91.55	93.49	94.50	94.18	95.11	81.16	17.82	11.15
8	89.56	92.04	94.00	95.20	95.68	95.65	77.10	11.85	10.00
16	90.03	92.69	94.36	95.69	95.91	91.27	59.09	13.52	10.08
32	90.85	93.05	95.14	96.14	96.26	87.87	17.79	18.07	10.67
64	91.79	94.00	95.30	96.44	96.43	81.73	14.33	11.30	13.21
128	92.47	94.71	96.03	96.50	96.17	64.03	11.16	12.00	11.41

ViT-Tiny Asym CIFAR-10

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	85.34	89.63	90.86	91.64	92.03	91.88	90.85	90.18	80.81
4	86.82	90.29	91.66	92.58	93.32	92.73	91.63	87.88	78.91
8	88.34	90.87	92.35	93.34	93.71	93.81	93.74	86.88	64.43
16	89.23	91.61	93.18	94.23	94.65	95.08	90.39	82.52	50.44
32	90.12	92.17	93.81	95.20	95.15	95.36	93.94	73.55	34.13
64	91.20	93.11	94.66	95.77	96.08	92.99	92.18	53.07	24.52
128	92.27	94.03	95.36	96.09	94.56	94.97	69.81	27.77	16.58

ViT-Tiny cLA CIFAR-10

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	87.30	89.65	91.13	92.12	92.49	91.70	88.92	82.43	51.06
4	88.06	90.47	91.91	92.00	93.45	92.05	88.78	73.27	11.47
8	89.14	91.38	93.13	93.36	93.22	90.72	85.78	68.40	13.39
16	90.43	92.27	93.91	94.86	94.53	94.11	78.83	44.03	17.11
32	91.32	93.37	94.91	95.63	95.33	89.55	71.95	30.90	19.23
64	92.50	94.17	95.82	96.26	95.54	83.63	50.91	21.61	15.50
128	93.86	95.39	96.50	96.45	94.85	75.52	28.65	19.95	31.06

ViT-Tiny CoLA CIFAR-10

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	87.90	90.43	92.13	93.79	93.77	93.57	87.13	20.52	12.61
4	88.43	90.77	92.73	94.30	95.21	94.49	83.74	17.82	11.15
8	88.96	91.26	93.37	94.95	95.39	94.74	77.09	11.85	10.00
16	89.50	92.00	93.86	95.26	95.72	91.27	62.12	15.42	11.25
32	90.02	92.63	94.54	95.72	95.96	87.87	19.10	18.07	12.41
64	91.18	93.51	95.13	96.07	96.01	81.73	14.33	17.70	13.23
128	92.06	94.20	95.56	96.17	95.54	65.78	17.45	10.30	11.03

ViT-Tiny RAC CIFAR-10

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	85.61	89.68	90.98	91.87	92.72	91.94	91.47	89.51	80.23
4	86.64	90.35	91.89	92.96	93.71	93.46	93.49	87.88	77.56
8	88.26	90.87	92.40	93.65	94.09	94.33	90.69	86.88	64.43
16	89.26	91.75	93.26	94.40	95.31	94.24	89.56	82.52	50.44
32	90.27	92.28	94.05	95.43	95.54	95.68	93.14	73.55	24.92
64	91.12	93.20	94.86	95.91	94.78	95.35	82.86	53.07	24.52
128	92.19	94.03	95.56	96.11	96.10	94.21	69.81	27.77	16.58

ViT-Tiny c^3LA CIFAR-10

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	87.29	89.83	91.49	92.48	93.04	91.35	88.65	80.39	53.28
4	88.57	90.76	92.50	93.36	93.98	91.55	87.25	73.27	11.47
8	89.69	91.84	93.52	94.68	94.80	90.72	85.78	68.40	27.84
16	90.63	92.69	94.33	95.35	95.30	90.04	81.13	44.03	17.11
32	91.39	93.68	95.12	95.57	95.30	89.55	71.95	21.94	17.60
64	92.86	94.71	95.92	96.41	95.07	83.63	41.60	21.61	14.03
128	93.83	95.38	96.43	96.22	94.89	75.52	27.84	19.95	14.48

ViT-Tiny LoRA Office-Home

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	47.33	65.03	70.80	75.55	77.81	75.93	77.08	40.57	1.80
4	48.57	64.90	71.36	75.33	77.85	78.50	77.55	26.51	2.01
8	49.42	64.81	72.08	75.93	79.39	78.88	59.09	4.28	2.61
16	50.41	65.33	72.30	77.21	79.26	79.69	49.17	1.97	4.53
32	50.53	65.50	73.54	79.09	79.56	66.01	36.43	2.44	2.05
64	51.86	66.35	74.99	78.97	79.86	61.48	10.77	1.75	2.09
128	54.68	66.82	76.70	79.91	80.33	53.53	2.78	1.75	2.91

ViT-Tiny Asym Office-Home

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	43.91	63.06	70.20	73.45	74.65	74.48	74.22	75.29	74.09
4	44.72	63.83	70.59	73.71	75.76	73.92	76.14	75.37	51.69
8	45.79	64.47	71.31	74.65	75.93	75.50	75.84	75.72	40.49
16	47.20	65.80	72.21	75.12	77.13	75.67	76.87	54.77	21.42
32	49.68	66.27	72.47	76.66	78.50	77.94	77.04	45.28	13.85
64	51.52	67.04	74.13	77.81	79.31	78.37	57.46	23.56	8.85
128	53.53	68.06	75.25	79.35	80.72	77.85	46.81	10.65	2.05

ViT-Tiny cLA Office-Home

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	44.04	64.04	70.12	73.45	75.89	75.72	74.35	73.19	30.14
4	46.09	64.86	70.84	74.90	76.53	75.37	74.99	54.81	2.01
8	47.50	65.16	72.12	75.16	76.57	76.87	75.25	45.10	2.86
16	50.75	65.63	72.98	77.00	78.37	77.17	59.30	36.55	3.42
32	52.93	66.99	74.39	77.85	76.83	76.83	51.13	12.61	2.35
64	55.96	68.53	75.67	79.14	79.26	63.06	35.49	4.02	2.69
128	61.22	71.95	77.34	79.78	77.77	53.91	17.44	3.42	3.42

ViT-Tiny CoLA Office-Home

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	45.28	64.64	70.50	74.48	76.96	76.70	74.05	40.57	2.01
4	46.43	64.94	70.63	74.39	76.87	76.61	72.60	26.51	2.01
8	47.71	65.11	70.97	75.12	77.73	76.96	59.09	3.04	5.77
16	49.47	64.51	71.40	75.63	78.71	77.68	49.17	1.97	2.22
32	50.32	65.20	72.12	77.38	80.38	66.01	36.43	2.44	2.05
64	51.05	65.63	73.79	78.32	79.35	61.48	10.77	1.75	2.69
128	52.29	66.35	75.29	79.48	79.52	52.29	2.78	1.75	4.10

ViT-Tiny RAC Office-Home

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	44.38	63.75	70.54	73.54	75.29	76.14	75.16	70.97	54.85
4	44.76	64.04	70.71	74.01	76.36	74.22	72.85	66.44	51.69
8	45.83	65.07	71.65	74.78	76.31	77.04	75.97	64.73	40.49
16	47.29	65.71	72.55	75.29	77.94	77.47	75.46	67.38	21.42
32	49.64	66.10	72.60	77.47	79.26	78.79	75.07	45.28	13.85
64	51.60	67.12	74.39	78.32	79.91	75.33	57.46	23.56	9.49
128	53.53	67.76	75.72	79.26	80.63	76.74	46.81	10.65	2.05

ViT-Tiny c^3LA Office-Home

	1e-5	1e-4.5	1e-4	1e-3.5	1e-3	1e-2.5	1e-2	1e-1.5	1e-1
2	45.02	64.51	70.29	73.88	76.36	75.93	73.07	57.46	30.14
4	46.30	64.94	70.80	74.31	77.17	76.19	66.65	54.81	2.01
8	49.21	65.37	72.68	75.93	77.51	72.60	64.73	45.10	3.04
16	51.13	66.44	73.15	77.55	78.71	71.23	59.30	36.55	2.48
32	53.31	67.55	74.22	79.26	79.22	67.38	51.13	12.01	2.35
64	56.18	69.22	76.06	79.52	78.24	63.06	35.49	6.50	3.59
128	61.18	72.00	77.38	79.91	78.45	53.91	17.44	8.38	3.42

For our ablation study on scaling factor shown in Tables 9 and 10, the use of $\alpha = 2r$ works as a baseline given how often it was the best choice. With Asymmetric methods, the ideal scaling factor tends to be larger; this follows from the number of trainable parameters decreasing, requiring a larger effective learning rate, as the scaling factor can be interpreted as a scale on the learning rate.

Table 9: Test accuracies of fine-tuning DeBERTa v3 on MRPC and TREC-50 over varying scaling factors (columns), ranks (rows), and LoRA PEFT methods. The standard baseline $2r$ often was the best, and asymmetric methods preferred higher scaling factors. Light-gray subtables identify cLA and c^{3}LA. Red marks a strictly higher displayed accuracy than Asym for cLA, or than RAC for c^{3}LA, at the same model, dataset, rank, and column setting. Ties are not highlighted.

DeBERTa v3 LoRA MRPC

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	87.2	88.3	88.5	88.1	87.4
8	86.9	86.1	89.2	87.0	66.5
16	87.8	88.9	89.1	66.5	66.5

DeBERTa v3 Asym MRPC

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	75.5	79.9	80.4	85.0	84.2
8	76.7	82.1	83.6	86.1	86.9
16	79.2	81.4	84.8	84.8	86.1

DeBERTa v3 cLA MRPC

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	86.2	86.0	86.3	86.4	86.4
8	86.6	84.8	85.4	85.5	85.9
16	86.8	86.9	86.2	86.2	86.4

DeBERTa v3 CoLA MRPC

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	87.8	88.9	88.5	89.2	87.1
8	89.6	87.4	88.7	87.2	86.3
16	89.2	87.6	86.9	87.2	66.5

DeBERTa v3 RAC MRPC

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	75.6	79.4	82.2	85.0	85.7
8	77.8	81.6	84.6	85.7	87.2
16	80.4	84.5	85.0	85.6	87.0

DeBERTa v3 c^3LA MRPC

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	79.3	83.3	86.5	88.5	86.1
8	78.3	84.9	86.9	87.6	86.9
16	85.0	85.7	87.3	85.8	66.5

Table 9: DeBERTa v3 scaling-factor ablation (continued): TREC-50.

DeBERTa v3 LoRA TREC-50

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	88.9	89.7	90.7	83.1	90.3
8	88.7	90.7	91.3	85.3	75.4
16	91.1	91.5	90.7	88.5	10.9

DeBERTa v3 Asym TREC-50

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	79.8	84.7	87.7	90.5	89.9
8	82.9	87.7	84.5	89.3	90.7
16	89.3	86.7	90.7	91.3	89.7

DeBERTa v3 cLA TREC-50

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	57.9	74.8	80.0	82.9	86.3
8	73.6	76.6	83.7	82.5	87.3
16	79.6	80.2	87.9	88.1	86.1

DeBERTa v3 CoLA TREC-50

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	92.1	91.9	92.5	90.7	91.1
8	91.7	89.7	90.9	90.3	85.5
16	91.9	92.3	86.1	87.3	10.9

DeBERTa v3 RAC TREC-50

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	60.1	72.6	81.0	85.5	88.7
8	75.2	81.5	85.7	88.1	89.7
16	83.1	86.3	87.7	90.3	78.0

DeBERTa v3 c^3LA TREC-50

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	$2r$	$4r$
4	73.8	81.5	83.1	89.3	88.7
8	78.2	82.3	83.9	84.7	81.7
16	83.1	85.5	85.3	87.5	86.3

Table 10: Test accuracies obtained by fine-tuning ViT-Tiny on Office-Home and CIFAR-10 over varying scaling factors (columns), ranks (rows), and LoRA PEFT methods. The standard baseline $2r$ often was the best, and asymmetric methods preferred higher scaling factors. Light-gray subtables identify cLA and c^{3}LA. Red marks a strictly higher displayed accuracy than Asym for cLA, or than RAC for c^{3}LA, at the same model, dataset, rank, and column setting. Ties are not highlighted.

ViT-Tiny LoRA CIFAR-10

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	93.9	94.1	94.0	94.0
8	94.8	95.7	95.7	95.8
16	95.8	96.1	96.1	95.2

ViT-Tiny Asym CIFAR-10

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	91.7	92.2	92.8	92.4
8	92.6	93.1	93.7	94.
16	93.1	94.0	94.4	94.4

ViT-Tiny cLA CIFAR-10

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	92.0	92.6	93.1	93.5
8	93.4	93.4	93.5	93.4
16	94.3	94.5	94.5	94.5

ViT-Tiny CoLA CIFAR-10

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	94.3	94.5	94.5	95.3
8	94.7	94.9	95.3	95.1
16	95.0	95.5	95.7	96.2

ViT-Tiny RAC CIFAR-10

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	91.8	92.6	93.2	94.2
8	92.8	93.4	94.0	94.8
16	93.6	94.3	94.8	95.6

ViT-Tiny c^3LA CIFAR-10

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	92.7	93.4	93.4	94.4
8	93.8	94.4	94.6	94.8
16	94.8	95.3	95.2	95.3

ViT-Tiny LoRA Office-Home

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	76.8	77.1	77.8	77.9
8	76.9	77.9	78.5	79.4
16	77.9	78.4	79.2	79.4

ViT-Tiny Asym Office-Home

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	74.0	74.5	75.2	75.6
8	74.5	75.1	75.6	76.2
16	75.2	75.9	76.4	76.9

ViT-Tiny cLA Office-Home

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	75.3	75.9	76.5	76.5
8	76.3	76.5	76.6	76.5
16	76.4	76.9	77.3	78.4

ViT-Tiny CoLA Office-Home

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	75.5	75.9	76.6	77.8
8	76.0	76.4	77.4	79.6
16	76.3	77.2	78.2	79.4

ViT-Tiny RAC Office-Home

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	74.3	74.6	75.6	76.0
8	74.8	75.2	75.9	77.1
16	75.2	75.5	76.4	77.7

ViT-Tiny c^3LA Office-Home

Rank/α	$\frac{r}{4}$	$\frac{r}{2}$	r	2r
4	75.6	75.9	76.3	77.3
8	76.4	77.0	76.9	77.5
16	76.6	77.7	78.4	78.5

Tables 11 and 12 examine the effect of chain-reset frequency on test accuracy for DeBERTa-v3-Base and ViT-Tiny, respectively, across the tested datasets and ranks.

Table 11: Test accuracies obtained by fine-tuning DeBERTa v3 on MRPC, CoLA, TREC-50, and RTE using chain LoRA methods CoLA, RAC, and c^3LA over varying ranks and chain reset frequencies. We do not observe a clear correlation between the optimal chain reset frequency and rank. Light-gray rows identify c^3LA. Red marks a strictly higher displayed accuracy than RAC at the same model, dataset, rank, and reset frequency. Ties are not highlighted.

DeBERTa v3 MRPC

Variant	Rank	Chain Reset Frequency 1	2	5	10	15	20
CoLA	4	88.0	86.8	89.2	88.1	86.7	86.7
	8	87.8	88.0	87.2	87.2	86.7	87.2
	16	66.5	87.2	87.2	87.2	87.2	87.2
RAC	4	68.3	77.4	85.0	85.7	85.7	86.6
	8	68.1	82.0	85.7	86.4	85.7	85.6
	16	69.1	84.2	85.6	86.1	86.5	86.3
c^3LA	4	84.8	86.7	87.2	85.2	85.8	85.2
	8	85.2	87.7	86.6	86.7	85.3	86.9
	16	87.6	87.0	86.7	86.6	86.6	87.6

DeBERTa v3 TREC-50

Variant	Rank	Chain Reset Frequency 1	2	5	10	15	20
CoLA	4	91.3	91.1	89.9	88.5	90.5	91.3
	8	92.7	91.1	85.3	10.9	92.7	90.5
	16	10.9	10.9	93.1	91.7	92.1	65.1
RAC	4	84.3	84.1	85.5	84.1	86.3	86.5
	8	88.3	88.5	88.1	88.7	87.7	88.9
	16	87.7	91.5	90.3	89.9	89.9	88.9
c^3LA	4	86.3	88.1	89.3	85.9	88.9	88.9
	8	86.1	89.3	84.7	83.7	86.1	90.7
	16	89.7	90.5	87.5	91.1	87.3	88.1

DeBERTa v3 CoLA

Variant	Rank	Chain Reset Frequency 1	2	5	10	15	20
CoLA	4	86.9	86.5	86.2	86.6	87.1	86.7
	8	85.5	85.1	85.1	85.1	85.1	85.1
	16	84.2	69.1	69.1	69.1	69.1	69.1
RAC	4	87.0	86.7	87.7	87.4	88.0	87.7
	8	87.5	87.8	87.8	87.5	86.6	86.6
	16	86.7	86.9	87.3	87.0	87.0	87.6
c^3LA	4	86.4	86.6	86.1	85.8	86.0	86.3
	8	86.0	86.1	86.1	86.2	86.3	86.3
	16	86.2	85.7	86.3	85.6	85.4	86.7

DeBERTa v3 RTE

Variant	Rank	Chain Reset Frequency 1	2	5	10	15	20
CoLA	4	82.9	84.4	85.1	83.7	85.4	86.2
	8	88.2	84.6	84.8	87.1	87.1	86.7
	16	85.1	52.6	81.4	84.8	84.3	73.5
RAC	4	82.3	83.0	81.6	82.1	82.4	82.4
	8	85.5	86.8	86.4	87.5	87.5	87.5
	16	84.2	84.4	84.1	83.5	83.7	84.3
c^3LA	4	79.0	77.9	72.6	71.9	74.0	72.4
	8	80.0	80.3	76.6	73.9	76.2	75.4
	16	85.0	82.5	83.6	83.0	82.9	82.4

Table 12: Test accuracies obtained by fine-tuning ViT-Tiny on Office-Home and CIFAR-10 using chain LoRA methods CoLA, RAC, and c^3LA over varying ranks and chain reset frequencies. Light-gray rows identify c^3LA. Red marks a strictly higher displayed accuracy than RAC at the same model, dataset, rank, and reset frequency. Ties are not highlighted.

ViT-Tiny Office-Home								ViT-Tiny CIFAR-10							
		Chain Reset Frequency								Chain Reset Frequency					
Variant	Rank	1	2	5	10	15	20	Variant	Rank	1	2	5	10	15	20
CoLA	4	76.4	76.5	77.2	77.2	77.6	77.8	CoLA	4	94.5	94.8	94.7	95.3	94.0	94.0
	8	77.6	77.1	78.3	77.3	78.7	79.6		8	95.1	95.1	95.3	94.9	94.7	94.5
	16	77.9	77.9	78.8	78.6	79.4	79.1		16	95.5	95.5	95.7	96.0	96.0	96.2
RAC	4	75.5	75.8	76.0	75.7	75.7	75.7	RAC	4	94.2	94.0	94.0	93.4	92.4	92.5
	8	77.1	76.1	76.3	76.6	76.1	76.4		8	94.5	94.8	94.5	94.2	94.1	94.0
	16	77.4	77.7	77.7	77.0	77.3	77.0		16	95.6	95.2	95.3	95.1	95.0	94.3
c^3LA	4	76.7	77.3	77.3	76.3	76.5	76.6	c^3LA	4	94.4	94.2	94.0	93.9	92.7	92.7
	8	77.5	76.9	77.2	76.7	76.8	76.8		8	93.6	94.2	94.8	94.5	93.4	93.4
	16	77.5	78.1	78.4	78.5	78.1	78.4		16	94.0	93.6	95.3	95.1	94.8	95.0

E.3 Computational Cost, Memory, and Efficiency

In this Section, we discuss the computational cost, peak memory, and throughput of our column-restricted LoRA variants. We also contrast these metrics with PaCA and its variants. We start by describing our compute-efficient implementation.

E.3.1 Leveraging the Computationally-Efficient Structure of Our Methods

We leverage the structure of our cheap variants to avoid computing Ax as a dense general matrix-matrix multiplication (GEMM) throughout training. Consider a single layer $W^i \in \mathbb{R}^{n_i \times m_i}$ adapted via r-cLA to $W^i + BA$, where $B \in \mathbb{R}^{n_i \times r}$ and A is constructed as follows: sample r random numbers without replacement from $[m_i]$, $\{c_1, c_2, \cdots, c_r\} \subset [m_i]$. For row j in A, the c_j^{th} element is one, and all other elements are zero, thus $Ax = [x_{c_1}, \cdots, x_{c_r}]^\top$. For cLA choose $\{c_1, \cdots, c_r\} = \{1, \cdots, r\}$.

In each forward pass, we compute $(W^i)(x) + B(A(x))$ where $x \in \mathbb{R}^{m_i}$. For d-cLA, we directly calculate $A(x)$ as a GEMM resulting in $r(2m_i - 1)$ floating point operations (FLOPs). For cLA, we store the columns $\{c_1, \cdots, c_r\}$ in memory when A is constructed, then directly construct $[x_{c_1}, \cdots, x_{c_r}]^\top$. Figure 5 visually compares the two methods.

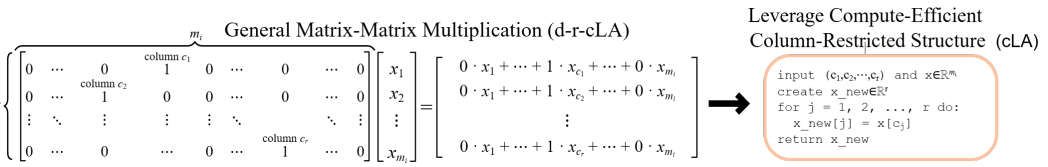


Figure 5: **A comparative visualization of the general matrix-matrix multiplication in d-r-cLA (left) and the `gather` (indexing) operation used in r-cLA (right).** The `gather` operation He et al. (2007) saves $r(2m_i - 1)$ FLOPs for each input x in each adapted layer of the forward pass, at the cost of creating minor memory overhead.

E.3.2 Experiments

To showcase the benefit of our methods and directly compare with PaCA, we fine-tune ViT-Tiny and ViT-Base on the Office-Home and CIFAR-10 datasets, and RoBERTa-Base and RoBERTa-Large on the MRPC and CoLA datasets using full fine-tuning, some base LoRA variants, the column-restricted

Table 13: Throughput, runtime, and trainable parameter count, and peak allocated GPU memory for fine-tuning various text and vision models on various datasets using various PEFT methods, including LoRA, PaCA, and their connecting variants, averaged over three seeds (0,1,2). We adapt the query and value matrices of the attention layers as well as the classification head. When adapting a small enough number of layers such that the PEFT methods are faster than FFT, PaCA reduces peak GPU memory by around 5–15% compared to LoRA under these rank (not parameter) matched baselines; a similar reduction is observed with the column-restricted LoRA variants. The prefix d- denotes a dense-multiplication implementation. Gray identifies our column-restricted methods. Red marks higher throughput or lower runtime/memory than Asym for cLA and d-r-cLA, comparing displayed values and normalized ratios separately.

Metric	Model	Dataset	Baselines		LoRA Variants	Parent Method	Column-Restricted Variants		PaCA Variants	Column-Restricted Variants		Chain of PaCA
			FFT	LoRA	CoLA	Asym	cLA	d-r-cLA	PaCA	r-c^{3}LA	d-r-c^{3}LA	C-PaCA
Throughput (Samples/s)	ViT-Tiny	Office-Home	360.4 (1.004)	358.8 (1.000)	361.6 (1.008)	361.5 (1.007)	358.3 (0.998)	359.9 (1.003)	344.3 (0.959)	362.1 (1.009)	359.4 (1.002)	364.1 (1.015)
		CIFAR-10	1068.6 (1.006)	1062.4 (1.000)	1004.0 (0.945)	1094.2 (1.030)	1111.8 (1.046)	1083.4 (1.020)	1159.5 (1.091)	1109.5 (1.044)	1081.9 (1.018)	1139.6 (1.073)
	ViT-Base	Office-Home	362.6 (1.004)	361.3 (1.000)	357.5 (0.990)	359.3 (0.994)	360.8 (0.999)	361.2 (1.000)	351.4 (0.973)	362.5 (1.003)	360.4 (0.997)	342.2 (0.947)
		CIFAR-10	1060.4 (1.013)	1047.2 (1.000)	1031.7 (0.985)	1077.7 (1.029)	1038.9 (0.992)	1070.3 (1.022)	1132.6 (1.082)	1104.3 (1.054)	1061.7 (1.014)	1139.5 (1.088)
	RoBERTa-Base	MRPC	806.5 (0.974)	828.3 (1.000)	725.5 (0.876)	875.6 (1.057)	920.9 (1.112)	829.3 (1.001)	942.0 (1.137)	910.6 (1.099)	880.3 (1.063)	930.7 (1.124)
		CoLA	816.1 (1.125)	725.5 (1.000)	834.8 (1.151)	894.2 (1.233)	929.0 (1.281)	829.1 (1.143)	765.3 (1.055)	929.8 (1.282)	834.3 (1.150)	875.3 (1.206)
	RoBERTa-Large	MRPC	387.7 (0.888)	436.6 (1.000)	437.8 (1.003)	466.9 (1.069)	483.6 (1.108)	468.9 (1.074)	499.7 (1.144)	453.9 (1.040)	439.3 (1.006)	480.8 (1.101)
		CoLA	394.7 (0.893)	441.9 (1.000)	417.9 (0.946)	468.1 (1.059)	463.1 (1.048)	468.1 (1.059)	480.3 (1.087)	454.3 (1.028)	471.2 (1.066)	506.0 (1.145)
Runtime (Minutes)	ViT-Tiny	Office-Home	50.6 (0.989)	51.2 (1.000)	50.5 (0.986)	50.4 (0.984)	50.8 (0.992)	51.0 (0.996)	53.8 (1.050)	50.3 (0.983)	50.6 (0.989)	50.3 (0.983)
		CIFAR-10	23.0 (0.992)	23.2 (1.000)	24.7 (1.067)	22.4 (0.969)	22.2 (0.956)	22.7 (0.979)	21.3 (0.919)	22.2 (0.959)	22.7 (0.980)	21.5 (0.928)
	ViT-Base	Office-Home	50.3 (0.999)	50.4 (1.000)	51.0 (1.012)	50.8 (1.008)	50.4 (1.000)	50.6 (1.004)	52.7 (1.047)	50.4 (1.000)	50.6 (1.005)	53.3 (1.058)
		CIFAR-10	23.2 (0.987)	23.5 (1.000)	23.5 (0.999)	22.8 (0.969)	23.8 (1.011)	22.9 (0.976)	21.7 (0.922)	22.2 (0.946)	22.9 (0.974)	21.6 (0.918)
	RoBERTa-Base	MRPC	2.5 (1.016)	2.5 (1.000)	2.9 (1.155)	2.3 (0.941)	2.3 (0.926)	2.5 (1.016)	2.2 (0.890)	2.3 (0.919)	2.3 (0.950)	2.2 (0.885)
		CoLA	5.5 (0.878)	6.2 (1.000)	5.4 (0.864)	5.1 (0.813)	4.9 (0.791)	5.5 (0.885)	5.9 (0.951)	4.9 (0.786)	5.5 (0.878)	5.2 (0.833)
	RoBERTa-Large	MRPC	5.1 (1.100)	4.7 (1.000)	4.7 (1.004)	4.5 (0.961)	4.2 (0.909)	4.4 (0.952)	4.2 (0.902)	4.7 (1.001)	4.8 (1.037)	4.4 (0.939)
		CoLA	11.3 (1.110)	10.1 (1.000)	10.8 (1.068)	9.7 (0.951)	9.9 (0.972)	9.7 (0.953)	9.5 (0.938)	9.9 (0.977)	9.6 (0.949)	9.0 (0.888)
Trainable Parameters	ViT-Tiny	Office-Home	$5.7e^6$ (16.793)	$3.4e^5$ (1.000)	$3.4e^5$ (1.000)	$2.7e^5$ (0.783)	$2.7e^5$ (0.783)	$2.7e^5$ (0.783)	$2.7e^5$ (0.783)	$2.7e^5$ (0.783)	$2.7e^5$ (0.783)	$2.7e^5$ (0.783)
		CIFAR-10	$5.5e^6$ (36.994)	$1.5e^5$ (1.000)	$1.5e^5$ (1.000)	$7.6e^4$ (0.506)	$7.6e^4$ (0.506)	$7.6e^4$ (0.506)	$7.6e^4$ (0.506)	$7.6e^4$ (0.506)	$7.6e^4$ (0.506)	$7.6e^4$ (0.506)
	ViT-Base	Office-Home	$8.7e^7$ (63.708)	$1.4e^6$ (1.000)	$1.4e^6$ (1.000)	$1.1e^6$ (0.783)	$1.1e^6$ (0.783)	$1.1e^6$ (0.783)	$1.1e^6$ (0.783)	$1.1e^6$ (0.783)	$1.1e^6$ (0.783)	$1.1e^6$ (0.783)
		CIFAR-10	$8.6e^7$ (143.606)	$6e^5$ (1.000)	$6e^5$ (1.000)	$3e^5$ (0.506)	$3e^5$ (0.506)	$3e^5$ (0.506)	$3e^5$ (0.506)	$3e^5$ (0.506)	$3e^5$ (0.506)	$3e^5$ (0.506)
	RoBERTa-Base	MRPC	$1.2e^8$ (105.459)	$1.2e^6$ (1.000)	$1.2e^6$ (1.000)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)
		CoLA	$1.2e^8$ (105.459)	$1.2e^6$ (1.000)	$1.2e^6$ (1.000)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)	$8.9e^5$ (0.750)
	RoBERTa-Large	MRPC	$3.6e^8$ (135.401)	$2.6e^6$ (1.000)	$2.6e^6$ (1.000)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)
		CoLA	$3.6e^8$ (135.401)	$2.6e^6$ (1.000)	$2.6e^6$ (1.000)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)	$1.8e^6$ (0.700)
Peak GPU Memory (GB)	ViT-Tiny	Office-Home	2.22 (1.280)	1.74 (1.000)	1.74 (1.000)	1.58 (0.911)	1.58 (0.912)	1.58 (0.911)	1.61 (0.927)	1.58 (0.912)	1.58 (0.911)	1.61 (0.927)
		CIFAR-10	2.29 (1.340)	1.71 (1.000)	1.71 (1.000)	1.57 (0.918)	1.57 (0.918)	1.57 (0.918)	1.57 (0.918)	1.57 (0.918)	1.57 (0.918)	1.57 (0.918)
	ViT-Base	Office-Home	7.29 (1.221)	5.97 (1.000)	5.97 (1.000)	5.12 (0.857)	5.12 (0.857)	5.12 (0.857)	5.11 (0.856)	5.12 (0.857)	5.12 (0.857)	5.11 (0.856)
		CIFAR-10	7.05 (1.437)	4.91 (1.000)	4.91 (1.000)	4.44 (0.904)	4.44 (0.904)	4.44 (0.904)	4.40 (0.896)	4.44 (0.905)	4.44 (0.904)	4.40 (0.896)
	RoBERTa-Base	MRPC	3.82 (1.658)	2.30 (1.000)	2.30 (1.000)	2.18 (0.948)	2.28 (0.990)	2.18 (0.948)	2.16 (0.940)	2.28 (0.990)	2.18 (0.948)	2.16 (0.940)
		CoLA	3.34 (1.812)	1.84 (1.000)	1.84 (1.000)	1.76 (0.953)	1.76 (0.953)	1.76 (0.953)	1.74 (0.943)	1.76 (0.953)	1.76 (0.954)	1.74 (0.943)
	RoBERTa-Large	MRPC	9.34 (1.783)	5.24 (1.000)	5.17 (0.988)	4.81 (0.919)	4.81 (0.919)	4.81 (0.919)	4.69 (0.895)	4.81 (0.919)	4.94 (0.944)	4.77 (0.910)
		CoLA	7.93 (2.026)	3.91 (1.000)	3.91 (1.000)	3.71 (0.949)	3.71 (0.948)	3.71 (0.949)	3.59 (0.917)	3.71 (0.948)	3.71 (0.949)	3.66 (0.934)

variants (cLA, r-cLA) and our column-restricted variants without the optimization of Subsection E.3.1, using dense GEMMs instead, and PaCA with rank $r = 16$ for 30 epochs, averaged over three seeds on a single NVIDIA H100 PCIe GPU. To align with the other experiments in our paper, as well as LoRA Hu et al. (2021), we adapt only the query and value matrices of each attention head, fully fine-tune the classification heads of the models, and report those results in Table 13. Note that these supplementary experiments for comparing to PaCA use $r = 16$ for all adapter methods and three seeds, while our main paper's results are parameter-matched comparisons which use $r = 16$ for the two-factor methods and $r = 32$ for the one-factor methods.

Discussion of experiments. In Table 13, PaCA is around 10-11% faster than LoRA while often reducing peak GPU memory by 5-15%, with our LoRA variants generally 8-10% faster than LoRA while reducing peak GPU memory similarly to PaCA, ranging from 5-15% as well. Our methods retain LoRA's structure, which simplifies task switching and adapter storage.

Key Takeaway. Column-restricted adaptation remain competitive with their counterparts (both PaCA and LoRA) while realizing the benefit of both methods. relative to LoRA, they save training-step time and peak GPU memory; relative to PaCA, they retain LoRA's storage-efficient structure and ease of swapping tasks. For the specific results of a rank-matched study, see Table 13.

E.4 Loss-Landscape and Spectral Diagnostics

We provide the detailed diagnostic comparisons supporting §4.2.1. These analyses describe how fine-tuning changes loss geometry and spectral structure. We use them to complement, rather than replace, the direct measurements of downstream performance and empirical $\mathcal{G}(\mathbf{W})$. The hyperparameters for these Figures is reported in Table 14.

Table 14: **Hyperparameters for the diagnostic figures.**

Model	Dataset	Learning rates (FFT, LoRA, Asym)	Chain reset frequency	Batch	Epochs	Max length / image size
ViT-Base	Office-Home	$(3e^{-4}, 1e^{-3}, 1e^{-3})$	5	64	30	224
ViT-Base	CIFAR-10	$(3e^{-4}, 1e^{-3}, 1e^{-3})$	5	64	30	224
RoBERTa-Base	CoLA	$(1e^{-5}, 3e^{-4}, 3e^{-4})$	3	32	20	128

Shared settings. Adam Kingma & Ba (2014) with $(\beta_1, \beta_2, \epsilon) = (0.9, 0.999, 1e^{-8})$; adapter rank $r = 16$ and scaling $\alpha = 2r$; global L_2 gradient clipping at 1; seeds $(12, 22, 32)$. LoRA and Asym denote the two-factor and one-factor adapter groups, respectively. For LoRA+, $\gamma_B = 16\gamma_A$.

Model-specific settings. ViT-Base uses a cosine scheduler with (warmup learning rate, warmup ratio) $= (1e^{-6}, 0.05)$ and weight decay (FFT, LoRA) $= (0.05, 0)$. RoBERTa-Base uses a linear scheduler with $(1e^{-6}, 0.1)$ and weight decay (FFT, LoRA) $= (0.01, 0)$.

E.4.1 Loss Landscapes

Comparisons across models and tasks. Figure 6 compares ViT-Base fine-tuned on Office-Home using PCA directions with RoBERTa-Base fine-tuned on CoLA using random directions. Figure 8 contrasts non-chain methods with their chain counterparts. The construction details below describe how these 3D visualizations are obtained. We treat landscape geometry as a diagnostic of the resulting models, rather than as a substitute for their measured test performance or empirical $\mathcal{G}(\mathbf{W})$.

3D landscapes. We obtained the top two principal directions of the model's update path via PCA of the update matrix $[\mathbf{W}^0 - \mathbf{W}^T; ...; \mathbf{W}^{T-1} - \mathbf{W}^T]$, where $\{\mathbf{W}^t\}_{t=0}^T$. denotes the sequence of model weights during optimization. Let δ, η be those two directions. For random directions, we generate them via a Gaussian distribution. For LoRA methods, we merged the adapters into the base weights before calculating. We normalize the directions similar to the methods of Li et al. (2018). We plot the function $f(\alpha, \beta) := \mathcal{L}(\mathbf{W} + \alpha\delta + \beta\eta)$ over a 51^2 grid of α, β values uniformly distributed over $[-2, 2] \times [-2, 2]$, we use mini-batches of size 12 when finding the values for $\mathcal{L}$.

Comparison between using random or PCA directions. To understand the differences between the loss landscapes of the models in the PCA directions compared to random directions, we plotted the loss landscape of ViT-Base fine-tuned on CIFAR-10 in both PCA directions (top) and random directions (bottom) in Figure 7. For random directions, the FFT landscape is substantially smoother;

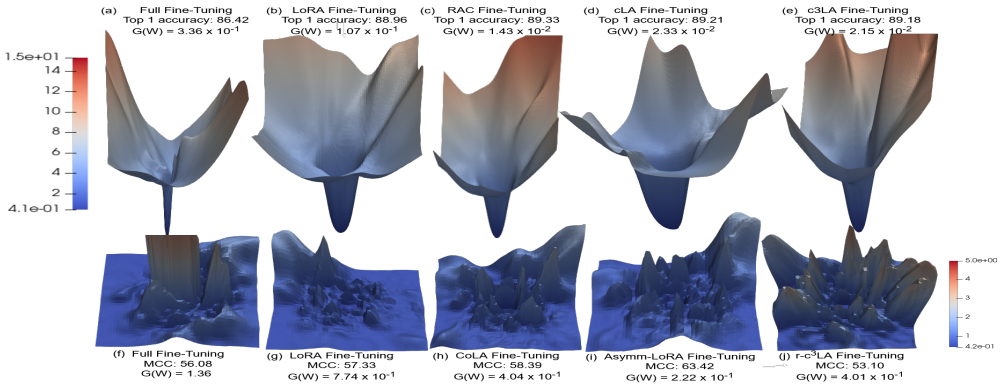


Figure 6: Loss landscapes of ViT-Base fine-tuned on Office-Home (top row) using PCA directions, and RoBERTa-Base fine-tuned on CoLA (bottom row) using random directions.

this is consistent with Li et al. (2018), but this is inconsistent with the loss landscapes of RoBERTa-Base with random direction in Figure 8, where chain methods produce spikier landscapes with no substantial change in generalizability.

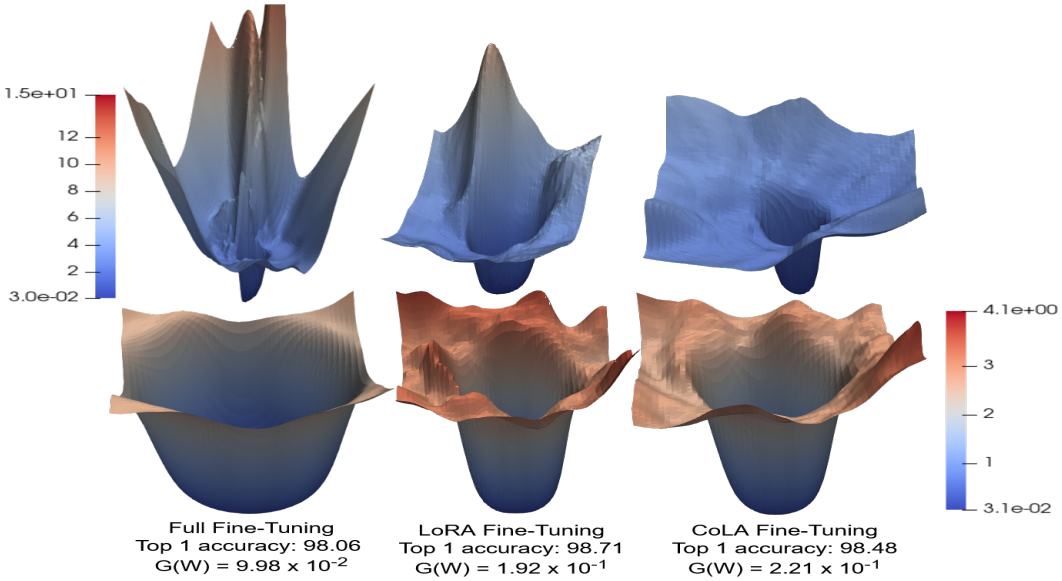


Figure 7: 3D loss landscapes of ViT-Base Dosovitskiy et al. (2020) pretrained on ImageNet-1K Deng et al. (2009) and fine-tuned on CIFAR-10 Krizhevsky (2009) using the PCA directions of the model weight updates (top) and random directions (bottom).

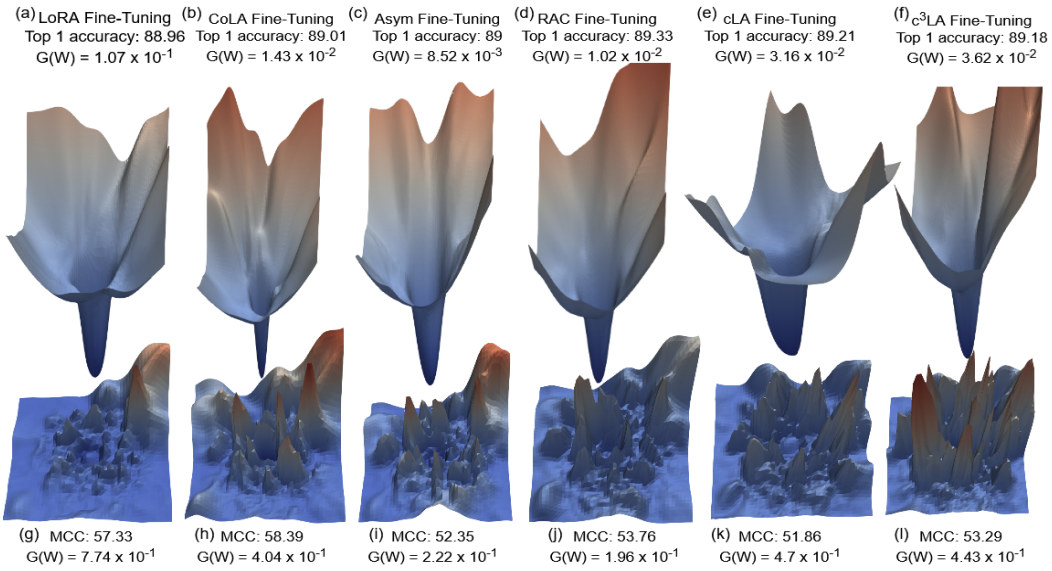


Figure 8: 3D loss landscapes of ViT-Base Dosovitskiy et al. (2020) pretrained on ImageNet-1K Deng et al. (2009) and fine-tuned on Office-Home Venkateswara et al. (2017) (top) and RoBERTa-Base Liu et al. (2019) pretrained on a corpus of English text fine-tuned on CoLA Wang et al. (2018) (bottom) using the non-chain then chain variants of each LoRA method. The chain variants consistently produce sharper landscapes than the non-chain variants. The expected increased generalization error is not observed in less space-restrictive adaptation methods; see Table 15.

E.4.2 Intruder Dimensions

Comparison with empirical generalization. The intruder-dimension plots and their empirical interpretation appear in §4.2.1, Figure 4. Here we provide the definition and threshold computation used in that comparison. This downstream loss-gap analysis is distinct from evaluating retention of pretraining-task performance.

Given the pretrained and fine-tuned models, $\mathbf{W}_0$ and $\mathbf{W}_0 + \Delta\mathbf{W}$ we find intruder dimensions as follows: first, we decompose each layer of $\mathbf{W}_0$ and $\mathbf{W}_0 + \Delta\mathbf{W}$ into their corresponding SVDs, $U^i\Sigma^i{V^i}^T_{(\mathbf{W}_0)^i}$ and $U^i\Sigma^i{V^i}^T_{(\mathbf{W}_0+\Delta\mathbf{W})^i}, i \in [L]$, respectively. Then, given a threshold $\varepsilon \in (0, 1)$, a singular vector $u^{j,i}_{(\mathbf{W}_0+\Delta\mathbf{W})}$ in $U^i_{(\mathbf{W}_0+\Delta\mathbf{W})}$ is an intruder dimension if for all $u^{k,i}_{(\mathbf{W}_0)}$ in $U^i_{(\mathbf{W}_0)}$, the expression, $\frac{|\langle u^{j,i}_{(\mathbf{W}_0+\Delta\mathbf{W})}, u^{k,i}_{(\mathbf{W}_0)}\rangle|}{\|u^{j,i}_{(\mathbf{W}_0+\Delta\mathbf{W})}\|\|u^{k,i}_{(\mathbf{W}_0)}\|} < \varepsilon$. For ε small enough, this indicates the vector $u^{j,i}_{(\mathbf{W}_0+\Delta\mathbf{W})}$ is almost orthogonal to all vectors in $U^i_{(\mathbf{W}_0)}$. We denote these vectors as *intruder dimensions.*

E.5 Generalization Error—Continued

Let $\mathcal{X} \times \mathcal{Y}$ be our input space and label space with ν distribution of pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, our dataset $N = \{(x_1, y_1), ..., (x_n, y_n)\}$ where each (x_i, y_i) is i.i.d. from ν distribution of $\mathcal{X} \times \mathcal{Y}$, thus the distribution over our dataset does not represent the true distribution of input-output pairs from our instance space. Let $\mathcal{H}$ be our hypothesis space, where $w \in \mathcal{H}; w(x_i) = \hat{y}_i$ thus, we are concerned with how accurately w can adapt to the true distribution ν of $\mathcal{X} \times \mathcal{Y}$. This can be addressed by the generalization error of our hypothesis $w \in \mathcal{H}$ given our loss function ℓ. The true risk of w over $\mathcal{X} \times \mathcal{Y}$ given ℓ is $\mathcal{L}_{\text{global}}(w) := \mathbb{E}_{\mathcal{X},\mathcal{Y}}[\ell(w(x), y)] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(w(x), y)d\nu$, while empirical risk is $\mathcal{L} := \frac{1}{n}\sum^n \ell(w(x_i), y_i); (x_i, y_i) \in N$. Let M denote the full dataset, where $M = N \cup T$, N being the train dataset, and T being the test dataset. In practice, the empirical risk can be computed based on N, and the test dataset, T, can be used to show how well the model has generalized. N and T are independent samples from ν; their distributions approximate ν but differ due to random and finite sampling. Although $\mathcal{L}_{\text{test}} - \mathcal{L}_{\text{train}}$ is not a true testament for calculating the generalization error of a model, it can be used as a heuristic for determining generalization. Understanding how stable these

models are to small weight perturbations provides insight into their reliability and reputability for practical use.

E.5.1 Extended Generalization Error Results

We extend the generalization gaps Table in 3 to report the empirical generalization errors for each method and the ratio $\frac{\mathcal{G}(\cdot)}{\mathcal{G}(\text{LoRA})}$ in Table 15. Under this normalization, our cheap LoRA variants vary from substantially outperforming LoRA to being worse. This same trend is observed among their counterparts, Asymmetric LoRA and RAC-LoRA, that have less restrictive adaptation spaces.

Table 15: Generalization error approximations, $\mathcal{G}(\mathbf{W})$, for the fine-tuning methods over various models and datasets. Upper entries report means and sample standard deviations ($\pm$) over five seeds. Parentheses report the mean errors normalized to LoRA's mean error within each model and dataset combination pair. LoRA, CoLA, and LoRA+ use $r = 16$; asymmetric methods use $r = 32$. Light-gray columns identify our column-restricted variants. Red marks a strictly smaller displayed mean or normalized ratio than Asym for cLA and r-cLA, or than RAC for c^{3}LA and r-c^{3}LA; each entry is compared separately.

Model	Dataset	Baselines		LoRA Variants		Parent Method	Column-Restricted Variants		Parent Method	Column-Restricted Variants	
		FFT	LoRA	CoLA	LoRA+	Asym	cLA	r-cLA	RAC	c^{3}LA	r-c^{3}LA
ViT-Tiny	Office-Home	0.090 ± 0.023 (0.099)	0.913 ± 0.045 (1.000)	0.793 ± 0.134 (0.869)	0.960 ± 0.014 (1.052)	0.860 ± 0.071 (0.942)	1.008 ± 0.032 (1.104)	0.980 ± 0.068 (1.073)	0.961 ± 0.062 (1.052)	1.073 ± 0.040 (1.176)	1.098 ± 0.067 (1.203)
	CIFAR-10	0.007 ± 0.004 (0.027)	0.260 ± 0.011 (1.000)	0.243 ± 0.036 (0.934)	0.252 ± 0.015 (0.968)	0.274 ± 0.013 (1.057)	0.308 ± 0.019 (1.185)	0.317 ± 0.015 (1.222)	0.279 ± 0.011 (1.076)	0.330 ± 0.015 (1.271)	0.315 ± 0.014 (1.211)
ViT-Base	Office-Home	0.081 ± 0.014 (0.219)	0.372 ± 0.039 (1.000)	0.447 ± 0.073 (1.201)	0.443 ± 0.270 (1.191)	0.402 ± 0.040 (1.079)	0.339 ± 0.066 (0.911)	0.367 ± 0.010 (0.985)	0.451 ± 0.099 (1.213)	0.407 ± 0.124 (1.094)	0.353 ± 0.084 (0.948)
	CIFAR-10	$0.005 \pm 7.13e^{-4}$ (0.048)	0.100 ± 0.012 (1.000)	0.099 ± 0.031 (0.994)	0.013 ± 0.004 (0.134)	0.060 ± 0.032 (0.597)	0.113 ± 0.009 (1.136)	0.091 ± 0.045 (0.916)	0.078 ± 0.037 (0.784)	0.077 ± 0.037 (0.768)	0.089 ± 0.045 (0.895)
DeBERTa v2 XXL	MRPC	0.213 ± 0.158 (1.511)	0.141 ± 0.056 (1.000)	0.133 ± 0.069 (0.944)	0.149 ± 0.127 (1.053)	0.120 ± 0.032 (0.849)	0.243 ± 0.120 (1.722)	0.240 ± 0.100 (1.700)	0.139 ± 0.057 (0.988)	0.426 ± 0.105 (3.023)	0.501 ± 0.166 (3.552)
	TREC-50	0.402 ± 0.083 (1.585)	0.254 ± 0.045 (1.000)	0.243 ± 0.031 (0.959)	0.319 ± 0.104 (1.258)	0.213 ± 0.029 (0.840)	0.268 ± 0.022 (1.056)	0.260 ± 0.052 (1.027)	0.225 ± 0.029 (0.887)	0.412 ± 0.042 (1.623)	0.392 ± 0.113 (1.546)
	PAWS	0.132 ± 0.049 (1.612)	0.082 ± 0.044 (1.000)	0.065 ± 0.027 (0.796)	0.003 ± 0.007 (0.041)	0.074 ± 0.020 (0.906)	0.098 ± 0.022 (1.199)	0.109 ± 0.025 (1.329)	0.068 ± 0.024 (0.837)	0.129 ± 0.025 (1.580)	0.135 ± 0.037 (1.650)
DeBERTa v3 Base	MRPC	0.599 ± 0.134 (0.965)	0.620 ± 0.170 (1.000)	0.819 ± 0.157 (1.319)	0.026 ± 0.045 (0.041)	0.646 ± 0.153 (1.041)	0.721 ± 0.120 (1.162)	0.808 ± 0.150 (1.303)	0.654 ± 0.273 (1.054)	0.867 ± 0.271 (1.398)	0.985 ± 0.281 (1.587)
	RTE	0.942 ± 0.450 (1.427)	0.660 ± 0.163 (1.000)	0.705 ± 0.371 (1.068)	0.005 ± 0.004 (0.007)	0.984 ± 0.211 (1.491)	1.070 ± 0.094 (1.620)	1.163 ± 0.397 (1.760)	0.950 ± 0.358 (1.438)	1.544 ± 0.383 (2.339)	1.643 ± 0.474 (2.488)
	STS-B	0.312 ± 0.032 (1.006)	0.310 ± 0.057 (1.000)	0.280 ± 0.061 (0.904)	0.464 ± 0.548 (1.497)	0.266 ± 0.046 (0.858)	0.359 ± 0.017 (1.160)	0.316 ± 0.095 (1.019)	0.314 ± 0.033 (1.014)	0.333 ± 0.028 (1.073)	0.334 ± 0.018 (1.079)
	TREC-50	0.534 ± 0.060 (0.982)	0.544 ± 0.101 (1.000)	0.687 ± 0.264 (1.263)	0.726 ± 0.124 (1.335)	0.755 ± 0.061 (1.388)	0.794 ± 0.061 (1.458)	0.643 ± 0.133 (1.182)	0.774 ± 0.183 (1.422)	1.114 ± 0.203 (2.048)	0.901 ± 0.312 (1.656)
	PAWS	0.243 ± 0.089 (2.129)	0.114 ± 0.026 (1.000)	0.166 ± 0.042 (1.448)	$9.25e^{-5} \pm 8.37e^{-5}$ ($8.09e^{-4}$)	0.191 ± 0.081 (1.671)	0.221 ± 0.088 (1.932)	0.184 ± 0.077 (1.605)	0.208 ± 0.060 (1.820)	0.324 ± 0.159 (2.836)	0.195 ± 0.068 (1.708)
RoBERTa-Base	MRPC	0.131 ± 0.006 (0.254)	0.515 ± 0.270 (1.000)	0.411 ± 0.123 (0.797)	0.599 ± 0.203 (1.162)	0.245 ± 0.100 (0.476)	0.569 ± 0.363 (1.105)	0.711 ± 0.227 (1.380)	0.241 ± 0.103 (0.468)	0.716 ± 0.287 (1.389)	0.563 ± 0.200 (1.092)
	CoLA	0.076 ± 0.033 (0.182)	0.416 ± 0.154 (1.000)	0.372 ± 0.113 (0.893)	0.560 ± 0.139 (1.346)	0.242 ± 0.106 (0.580)	0.684 ± 0.039 (1.642)	0.483 ± 0.206 (1.161)	0.197 ± 0.059 (0.474)	0.612 ± 0.074 (1.470)	0.475 ± 0.182 (1.140)
RoBERTa-Large	MRPC	0.232 ± 0.069 (0.400)	0.581 ± 0.339 (1.000)	0.513 ± 0.152 (0.884)	0.008 ± 0.007 (0.013)	0.352 ± 0.208 (0.607)	0.686 ± 0.284 (1.182)	0.357 ± 0.239 (0.614)	0.216 ± 0.138 (0.373)	0.624 ± 0.255 (1.075)	0.641 ± 0.195 (1.103)
	CoLA	0.100 ± 0.041 (0.326)	0.307 ± 0.181 (1.000)	0.249 ± 0.138 (0.813)	0.050 ± 0.042 (0.161)	0.246 ± 0.150 (0.801)	0.265 ± 0.163 (0.864)	0.394 ± 0.248 (1.286)	0.271 ± 0.070 (0.884)	0.441 ± 0.143 (1.437)	0.409 ± 0.072 (1.335)
TinyLlama	OpenBookQA	2.685 ± 0.510 (2.779)	0.966 ± 0.296 (1.000)	1.398 ± 0.421 (1.447)	0.231 ± 0.177 (0.239)	1.242 ± 0.544 (1.285)	1.838 ± 0.739 (1.902)	1.756 ± 0.657 (1.817)	1.270 ± 0.580 (1.314)	1.872 ± 0.880 (1.937)	1.918 ± 1.032 (1.984)
	FOLIO	0.617 ± 0.343 (0.814)	0.757 ± 0.452 (1.000)	0.777 ± 0.330 (1.025)	0.355 ± 0.245 (0.469)	0.879 ± 0.348 (1.160)	0.997 ± 0.323 (1.316)	0.887 ± 0.269 (1.171)	1.025 ± 0.278 (1.354)	1.082 ± 0.561 (1.428)	1.224 ± 0.373 (1.615)
	LogiQA	2.531 ± 0.686 (1.184)	2.137 ± 0.238 (1.000)	1.789 ± 0.176 (0.837)	0.183 ± 0.136 (0.086)	1.574 ± 0.055 (0.737)	1.961 ± 0.095 (0.918)	2.106 ± 0.110 (0.986)	1.908 ± 0.091 (0.893)	2.532 ± 0.242 (1.185)	2.391 ± 0.381 (1.119)
	CLUTRR	4.570 ± 0.392 (2.118)	2.158 ± 0.202 (1.000)	1.433 ± 0.110 (0.664)	4.012 ± 0.198 (1.860)	1.441 ± 0.298 (0.668)	1.333 ± 0.080 (0.618)	1.578 ± 0.263 (0.731)	1.412 ± 0.224 (0.655)	1.387 ± 0.107 (0.643)	1.398 ± 0.132 (0.648)
Llama3-8B	OpenBookQA	0.360 ± 0.073 (0.780)	0.461 ± 0.080 (1.000)	0.449 ± 0.138 (0.972)	0.044 ± 0.050 (0.095)	0.363 ± 0.108 (0.787)	0.430 ± 0.077 (0.932)	0.452 ± 0.150 (0.980)	0.406 ± 0.113 (0.880)	0.324 ± 0.024 (0.703)	0.443 ± 0.171 (0.960)
	CLUTRR	2.369 ± 0.516 (1.111)	2.133 ± 0.289 (1.000)	2.272 ± 0.624 (1.065)	3.268 ± 0.484 (1.532)	1.822 ± 0.296 (0.854)	2.113 ± 0.283 (0.991)	2.024 ± 0.496 (0.949)	2.254 ± 0.512 (1.057)	2.487 ± 0.337 (1.166)	2.411 ± 0.207 (1.130)
DeepseekCoder	Django	0.184 ± 0.005 (2.190)	0.084 ± 0.004 (1.000)	0.050 ± 0.001 (0.597)	0.184 ± 0.004 (2.183)	0.080 ± 0.002 (0.955)	0.078 ± 0.003 (0.930)	0.077 ± 0.001 (0.909)	0.083 ± 0.002 (0.990)	0.080 ± 0.002 (0.952)	0.080 ± 0.002 (0.944)
GPT2-Small	E2E	0.015 ± 0.005 (0.082)	0.180 ± 0.022 (1.000)	0.177 ± 0.009 (0.982)	0.191 ± 0.019 (1.062)	0.185 ± 0.012 (1.027)	0.200 ± 0.006 (1.107)	0.187 ± 0.003 (1.037)	0.184 ± 0.011 (1.019)	0.185 ± 0.014 (1.025)	0.192 ± 0.009 (1.067)

E.6 Complete Test Performance and Training-Time Results

Table 16 reports the complete test-performance and training-time results. Figure 9 gives the full-range counterparts of the main-paper summary plots.

Table 16: Performance of fine-tuned models with adapter rank $r = 16$ for LoRA, CoLA, and LoRA+, and $r = 32$ for the asymmetric methods; FFT has no adapter rank. Each cell reports test performance at the best-validation checkpoint, with total training-step time over in parentheses (minutes; forward, backward, and optimizer phases, excluding input loading, evaluation, and separately timed initialization). Values are means ± sample standard deviations over five seeds. Light-gray columns identify our column-restricted variants. Red marks a strictly better displayed mean than Asym for cLA and r-cLA, or than RAC for c^3LA and r-c^3LA, separately for performance and training-step time. Lower runtime and GPT2-Small/E2E perplexity are better; for other performance metrics, higher is better. Benchmarking details are given in Appendix §E.

Model	Dataset	Baselines		LoRA Variants		Parent Method	Column-Restricted Variants		Parent Method	Column-Restricted Variants	
		FFT	LoRA	CoLA	LoRA+	Asym	cLA	r-cLA	RAC	c^3LA	r-c^3LA
ViT-Tiny	Office-Home	61.29 ± 1.36 (4.11 ± 0.22)	79.97 ± 1.04 (3.79 ± 0.09)	79.53 ± 0.74 (3.84 ± 0.16)	79.09 ± 0.37 (3.98 ± 0.21)	79.37 ± 0.80 (3.46 ± 0.03)	78.76 ± 0.45 (3.38 ± 0.30)	78.72 ± 0.67 (3.41 ± 0.26)	79.77 ± 0.77 (3.45 ± 0.03)	78.72 ± 0.70 (3.39 ± 0.27)	79.25 ± 0.35 (3.38 ± 0.27)
	CIFAR-10	88.30 ± 0.54 (13.51 ± 0.38)	95.90 ± 0.11 (12.93 ± 0.36)	95.63 ± 0.24 (13.09 ± 0.44)	95.12 ± 0.16 (13.32 ± 0.53)	95.62 ± 0.08 (12.20 ± 0.09)	94.99 ± 0.27 (12.06 ± 1.01)	95.11 ± 0.20 (12.94 ± 1.64)	95.73 ± 0.07 (12.08 ± 0.13)	95.06 ± 0.23 (13.15 ± 1.88)	95.21 ± 0.24 (13.13 ± 1.79)
ViT-Base	Office-Home	84.10 ± 0.90 (4.30 ± 0.08)	90.51 ± 0.31 (3.91 ± 0.11)	90.57 ± 0.50 (3.98 ± 0.09)	86.27 ± 1.04 (4.01 ± 0.21)	90.28 ± 0.54 (3.60 ± 0.02)	89.98 ± 0.45 (3.53 ± 0.16)	90.20 ± 0.32 (3.56 ± 0.19)	90.29 ± 0.23 (3.59 ± 0.04)	89.66 ± 0.24 (3.62 ± 0.15)	90.35 ± 0.51 (3.63 ± 0.21)
	CIFAR-10	97.93 ± 0.11 (16.67 ± 0.22)	98.32 ± 0.15 (14.40 ± 0.42)	98.26 ± 0.05 (14.49 ± 0.37)	97.84 ± 0.07 (14.61 ± 0.55)	98.50 ± 0.08 (13.98 ± 0.18)	98.42 ± 0.10 (13.90 ± 0.72)	98.36 ± 0.13 (13.84 ± 0.70)	98.60 ± 0.13 (13.86 ± 0.04)	98.33 ± 0.17 (13.92 ± 0.76)	98.40 ± 0.04 (14.81 ± 1.77)
DeBERTa v2 XXL	MRPC	86.02 ± 4.11 (79.16 ± 0.32)	88.63 ± 0.77 (56.33 ± 0.08)	87.73 ± 0.87 (57.29 ± 1.91)	81.77 ± 9.42 (56.67 ± 0.28)	87.25 ± 0.93 (55.89 ± 0.21)	85.18 ± 0.63 (55.66 ± 0.49)	85.18 ± 0.98 (56.07 ± 0.77)	86.76 ± 1.01 (55.91 ± 0.16)	83.64 ± 1.02 (55.47 ± 0.29)	84.12 ± 0.85 (55.93 ± 0.86)
	TREC-50	92.56 ± 0.68 (59.28 ± 0.33)	91.96 ± 0.26 (45.91 ± 1.21)	92.40 ± 0.68 (48.99 ± 2.32)	84.28 ± 11.34 (46.91 ± 2.52)	91.96 ± 0.46 (43.06 ± 0.61)	91.60 ± 0.28 (41.41 ± 0.48)	91.72 ± 0.78 (41.79 ± 0.86)	91.80 ± 0.28 (44.64 ± 0.94)	91.08 ± 0.48 (41.31 ± 0.60)	91.12 ± 0.52 (41.52 ± 0.76)
	PAWS	94.89 ± 0.33 (457.53 ± 2.90)	94.98 ± 0.22 (332.14 ± 1.01)	95.05 ± 0.36 (334.52 ± 3.74)	63.55 ± 17.34 (332.62 ± 1.34)	94.94 ± 0.41 (329.32 ± 1.09)	95.00 ± 0.14 (329.09 ± 3.57)	95.14 ± 0.14 (328.16 ± 1.33)	95.01 ± 0.31 (329.99 ± 1.83)	94.89 ± 0.18 (328.27 ± 3.81)	94.95 ± 0.17 (329.20 ± 2.78)
DeBERTa v3 Base	MRPC	87.88 ± 0.44 (17.11 ± 0.13)	87.93 ± 0.14 (14.17 ± 0.19)	88.24 ± 0.70 (14.75 ± 0.57)	70.11 ± 7.90 (14.53 ± 0.31)	86.92 ± 0.70 (14.14 ± 0.36)	85.83 ± 0.72 (13.73 ± 0.23)	85.55 ± 0.88 (13.74 ± 0.14)	86.41 ± 0.57 (13.94 ± 0.19)	87.07 ± 0.54 (13.81 ± 0.33)	86.48 ± 0.84 (13.65 ± 0.23)
	RTE	81.44 ± 5.30 (32.86 ± 0.20)	85.04 ± 3.19 (29.11 ± 0.15)	83.45 ± 2.64 (29.03 ± 0.06)	53.81 ± 1.79 (29.12 ± 0.11)	81.44 ± 4.15 (28.96 ± 0.10)	79.28 ± 2.00 (28.84 ± 0.07)	78.13 ± 3.04 (28.75 ± 0.08)	81.87 ± 3.79 (29.03 ± 0.14)	80.29 ± 3.93 (28.84 ± 0.12)	80.58 ± 4.23 (28.93 ± 0.14)
	STS-B	92.08 ± 0.17 (29.22 ± 0.11)	90.45 ± 2.73 (25.31 ± 0.10)	90.59 ± 2.09 (25.32 ± 0.14)	35.44 ± 49.28 (25.62 ± 0.44)	91.61 ± 0.35 (25.17 ± 0.25)	90.72 ± 0.68 (24.41 ± 0.18)	90.85 ± 0.45 (24.43 ± 0.11)	91.50 ± 0.47 (24.99 ± 0.18)	91.26 ± 0.49 (24.55 ± 0.08)	91.23 ± 0.38 (24.45 ± 0.07)
	TREC-50	92.20 ± 0.75 (20.44 ± 0.59)	89.56 ± 3.30 (18.82 ± 0.24)	89.60 ± 0.69 (19.60 ± 0.57)	90.92 ± 0.58 (19.37 ± 0.57)	90.84 ± 0.99 (17.87 ± 0.51)	90.20 ± 0.40 (17.09 ± 0.51)	90.56 ± 0.52 (16.91 ± 0.18)	90.84 ± 0.82 (17.52 ± 0.16)	91.56 ± 0.48 (16.97 ± 0.39)	90.88 ± 1.29 (17.39 ± 0.54)
	PAWS	94.72 ± 0.07 (130.96 ± 2.58)	94.60 ± 0.17 (110.89 ± 1.46)	94.63 ± 0.26 (111.70 ± 1.92)	55.80 ± 0.01 (110.44 ± 1.69)	94.64 ± 0.29 (108.03 ± 0.69)	94.63 ± 0.13 (106.36 ± 1.51)	94.39 ± 0.25 (108.58 ± 2.07)	94.65 ± 0.24 (109.67 ± 3.66)	94.52 ± 0.26 (107.98 ± 2.09)	94.55 ± 0.19 (109.48 ± 5.43)
RoBERTa-Base	MRPC	89.09 ± 0.43 (1.19 ± 0.04)	90.14 ± 0.42 (1.15 ± 0.07)	90.35 ± 0.30 (1.05 ± 0.02)	89.19 ± 0.51 (1.06 ± 0.04)	89.63 ± 0.82 (1.00 ± 0.03)	88.64 ± 0.27 (0.99 ± 0.07)	89.35 ± 0.36 (0.98 ± 0.09)	89.42 ± 0.38 (0.99 ± 0.03)	89.00 ± 0.12 (0.99 ± 0.09)	89.37 ± 0.31 (0.99 ± 0.08)
	CoLA	53.74 ± 1.45 (2.72 ± 0.08)	63.11 ± 2.70 (2.43 ± 0.02)	60.97 ± 4.29 (2.53 ± 0.19)	61.71 ± 3.27 (2.56 ± 0.11)	60.64 ± 2.62 (2.28 ± 0.06)	57.24 ± 2.82 (2.16 ± 0.05)	59.13 ± 3.89 (2.15 ± 0.03)	61.37 ± 3.25 (2.30 ± 0.08)	57.10 ± 3.77 (2.18 ± 0.04)	58.20 ± 2.83 (2.16 ± 0.05)
RoBERTa-Large	MRPC	89.91 ± 0.28 (2.39 ± 0.05)	91.35 ± 0.38 (2.00 ± 0.04)	91.45 ± 0.36 (2.02 ± 0.10)	80.99 ± 2.50 (2.02 ± 0.06)	89.61 ± 3.46 (1.87 ± 0.06)	90.59 ± 0.79 (1.74 ± 0.05)	89.78 ± 0.77 (1.77 ± 0.11)	88.88 ± 5.00 (1.90 ± 0.05)	90.48 ± 0.77 (1.75 ± 0.04)	90.67 ± 0.35 (1.76 ± 0.08)
	CoLA	59.78 ± 3.02 (5.54 ± 0.13)	68.32 ± 1.77 (4.61 ± 0.04)	68.96 ± 2.98 (4.63 ± 0.14)	35.18 ± 32.15 (4.73 ± 0.13)	67.35 ± 2.99 (4.31 ± 0.10)	65.43 ± 1.17 (3.99 ± 0.04)	65.92 ± 4.03 (3.99 ± 0.03)	67.54 ± 1.61 (4.27 ± 0.03)	65.59 ± 1.71 (4.00 ± 0.02)	66.14 ± 3.58 (4.04 ± 0.10)
TinyLlama	OpenBookQA	51.12 ± 5.51 (10.74 ± 0.21)	56.72 ± 3.48 (8.07 ± 0.14)	56.28 ± 3.26 (8.31 ± 0.20)	35.16 ± 6.59 (8.37 ± 0.25)	51.88 ± 1.72 (7.86 ± 0.13)	50.04 ± 4.06 (7.35 ± 0.11)	50.12 ± 6.59 (7.38 ± 0.03)	51.52 ± 2.92 (7.88 ± 0.22)	50.36 ± 4.10 (7.36 ± 0.16)	50.80 ± 4.72 (7.44 ± 0.04)
	FOLIO	59.22 ± 5.74 (1.84 ± 0.02)	64.12 ± 5.48 (1.73 ± 0.06)	58.82 ± 5.14 (1.73 ± 0.02)	52.94 ± 5.88 (1.73 ± 0.02)	63.73 ± 2.30 (1.64 ± 0.02)	59.61 ± 6.29 (1.59 ± 0.07)	62.35 ± 1.32 (1.61 ± 0.13)	61.57 ± 4.30 (1.64 ± 0.02)	60.20 ± 4.73 (1.55 ± 0.04)	62.35 ± 3.77 (1.56 ± 0.04)
	LogiQA	47.51 ± 5.46 (34.31 ± 0.13)	45.15 ± 3.19 (21.18 ± 0.06)	43.51 ± 4.14 (21.78 ± 0.75)	29.27 ± 3.22 (21.26 ± 0.11)	41.20 ± 3.85 (20.81 ± 0.18)	39.99 ± 2.57 (20.02 ± 0.12)	41.31 ± 2.65 (20.25 ± 0.28)	42.42 ± 3.42 (20.76 ± 0.10)	41.34 ± 2.64 (20.05 ± 0.16)	43.09 ± 3.10 (20.09 ± 0.13)
	CLUTRR	41.34 ± 1.56 (26.36 ± 0.70)	36.77 ± 3.02 (20.07 ± 0.16)	38.72 ± 1.02 (20.81 ± 0.78)	39.14 ± 1.88 (20.27 ± 0.32)	36.15 ± 3.10 (19.18 ± 0.16)	36.77 ± 1.61 (18.08 ± 0.15)	36.07 ± 1.69 (18.37 ± 0.19)	36.62 ± 2.74 (19.17 ± 0.13)	36.47 ± 0.96 (18.24 ± 0.14)	37.16 ± 2.32 (18.31 ± 0.08)
Llama3-8B	OpenBookQA	88.60 ± 0.86 (21.95 ± 0.11)	87.64 ± 1.23 (11.71 ± 0.23)	87.12 ± 1.29 (11.92 ± 0.30)	24.84 ± 1.80 (11.67 ± 0.09)	87.84 ± 1.37 (11.56 ± 0.20)	88.32 ± 0.72 (10.73 ± 0.27)	86.88 ± 1.36 (10.86 ± 0.31)	86.44 ± 0.88 (11.22 ± 0.13)	87.92 ± 1.15 (10.72 ± 0.18)	87.32 ± 0.77 (10.84 ± 0.16)
	CLUTRR	52.29 ± 6.97 (53.38 ± 0.49)	55.69 ± 3.04 (28.87 ± 0.62)	57.10 ± 2.87 (28.59 ± 0.12)	42.35 ± 2.92 (28.86 ± 0.56)	53.87 ± 3.18 (27.87 ± 0.44)	54.01 ± 3.89 (26.08 ± 0.49)	55.06 ± 3.55 (26.13 ± 0.31)	55.27 ± 3.01 (27.49 ± 0.29)	54.98 ± 3.71 (26.00 ± 0.40)	54.24 ± 3.00 (27.15 ± 2.83)
DeepseekCoder	Django	85.15 ± 0.47 (19.90 ± 0.40)	82.74 ± 0.41 (15.25 ± 0.11)	82.77 ± 0.14 (16.11 ± 1.16)	84.42 ± 0.43 (15.15 ± 0.11)	82.47 ± 0.25 (14.64 ± 0.10)	82.79 ± 0.14 (13.81 ± 0.40)	82.73 ± 0.36 (14.87 ± 0.29)	82.76 ± 0.16 (14.74 ± 0.08)	82.93 ± 0.29 (14.25 ± 0.17)	82.86 ± 0.22 (14.31 ± 0.11)
GPT2-Small	E2E	5.01 ± 0.03 (32.49 ± 0.32)	3.33 ± 0.03 (39.88 ± 0.59)	3.37 ± 0.01 (41.08 ± 2.39)	3.30 ± 0.04 (40.64 ± 1.91)	3.42 ± 0.03 (37.29 ± 0.40)	3.38 ± 0.02 (35.56 ± 0.55)	3.36 ± 0.01 (36.14 ± 0.52)	3.39 ± 0.02 (37.74 ± 0.57)	3.36 ± 0.03 (36.33 ± 0.78)	3.36 ± 0.02 (36.49 ± 0.77)

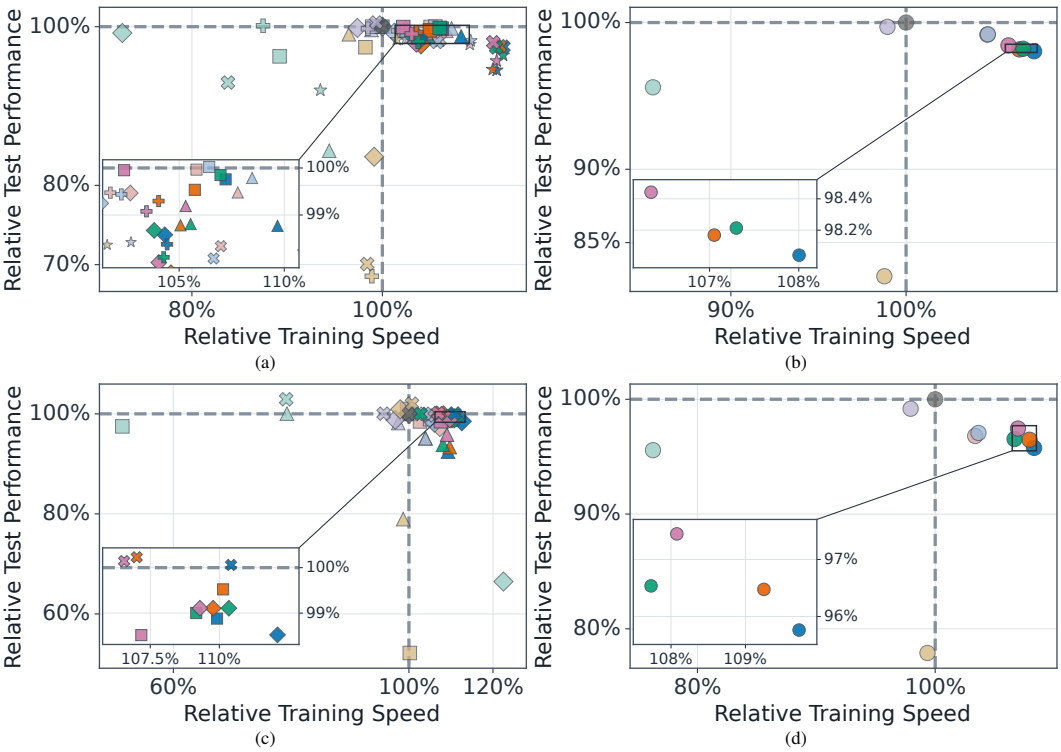


Figure 9: **Test performance and training-step speedup relative to LoRA for all fine-tuned models, datasets, and methods over five seeds.** The *top row* presents only the encoder models (e.g., ViT, RoBERTa, DeBERTa); the *bottom row* presents the decoder models (e.g., TinyLlaMA, LLaMA-3-8B, GPT2-Small, DeepSeekCoder). In the *first column*, we show the performance of each fine-tuning method relative to LoRA averaged over each of the model's fine-tuning tasks (e.g., performance of cLA on ViT-Tiny averaged over Office-Home and CIFAR-10 is given by ▲); in the *second column*, we show the performance of each fine-tuning method relative to LoRA averaged over all models and tasks (e.g., performance of cLA averaged over ViT-Tiny fine-tuned on Office-Home and CIFAR-10, ViT-Base on Office-Home and CIFAR-10, DeBERTa-v2-XXL on MRPC, TREC-50, PAWS, and so on is given by ●). We define the colors and shapes in Table 2. These are the full range counterparts to Figure 3.

F Limitations and Discussion

cLA and c^{3}LA particularly train only a small subsection of our pretrained model at a time; this can lead to greater variance in generalizability in comparison to alternate LoRA variants. Often, we observed that cLA and c^{3}LA performed as well as their corresponding asymmetric baselines, Asymmetric LoRA and RAC, while being less expensive. The nature of the methods they were inspired by already had a frozen matrix component; we leave it up to researchers to study more potential identity-based LoRA variants to save computational resources.

G TABLE OF NOTATIONS

Table 17: Table of notations.

Notation	Definition
$\|x\|$	The ℓ_2 norm of a vector, x
$\|A\|$	The Frobenius norm of a matrix, A
$\|A\|_2$	The spectral norm of a matrix, A
L	Number of layers in a deep neural network
W^i	i^{th} layer of network
$\mathbf{W}$	$(W^1, ..., W^L)$
x	Input to the network
$f_{\mathbf{W}}(x)$	$\sigma_L(W^L \cdots \sigma_3(W^3\sigma_2(W^2\sigma_1(W^1(x))...)))$
$\sigma_i(\cdot)$	i^{th} layer non-linear activation function
N_{pre}	pre-training dataset $(x_i, y_i)_{i=1}^{\|N_{\text{pre}}\|}$
$\ell_{\text{pre}}(\cdot)$	pre-training loss function
$\mathbf{W}_0$	pre-training weights
$\Delta\mathbf{W}$	FFT weight-update
$\Delta\hat{\mathbf{W}}$	FFT argmin update
$\ell(\cdot)$	fine-tuning loss function
$\mathbf{BA}$	LoRA weight-update
$\hat{\mathbf{B}}\hat{\mathbf{A}}$	LoRA argmin weight update
k	Chain-length of chain methods (CoLA, RAC, c^3LA)
$\mathbf{B}^j\mathbf{A}^j$	CoLA j^{th} chain weight update
$\hat{\mathbf{B}}^j\hat{\mathbf{A}}^j$	CoLA j^{th} chain argmin weight update
$\mathbf{W}_0^{(k,BA)}$	k chains of CoLA updates, where $\mathbf{W}_0^{(k,BA)} := \mathbf{W}_0 + \sum_{j=1}^k \hat{\mathbf{B}}^j\hat{\mathbf{A}}^j$
$\mathbf{A}_0$	Frozen A layers
$\mathbf{BA}_0$	Assymetric LoRA weight update
$\hat{\mathbf{B}}\mathbf{A}_0$	Assymetric LoRA argmin weight update
$\mathbf{B}^j\mathbf{A}_0^j$	RAC-LoRA j^{th} chain weight update
$\hat{\mathbf{B}}^j\mathbf{A}_0^j$	RAC-LoRA j^{th} chain argmin weight update
$\mathbf{W}_0^{(k,B)}$	k chains of RAC-LoRA updates, where $\mathbf{W}_0^{(k,B)} := \mathbf{W}_0 + \sum_{j=1}^k \hat{\mathbf{B}}^j\mathbf{A}_0^j$
$\mathbf{B}^c$	Cheap LoRA (cLA) weight update
$\hat{\mathbf{B}}^c$	cLA argmin weight update
$\mathbf{B}^{c^3,j}$	Circulant chain of cheap LoRA's (c^3LA) j^{th} chain weight update
$\hat{\mathbf{B}}^{c^3}$	c^3LA j^{th} chain argmin weight update
$\mathbf{W}_0^{(k,B^{c^3})}$	k chains of c^3LA updates, where $\mathbf{W}_0^{(k,B^{c^3})} := \mathbf{W}_0 + \sum_{j=1}^k \hat{\mathbf{B}}^{c^3,j}$
L_G	Lipschitz constant for the gradient of the loss function.
$\mathcal{X}$	feature space of the network
$\mathcal{Y}$	label space of the network
$\hat{\mathcal{L}}_{\text{global}}(\cdot)$	true risk of an input network