GeLoRA: Geometric Adaptive Ranks For Efficient LoRA Fine-tuning

Abdessalam Ed-dib
New York University
ae2842@nyu.edu

Zhanibek Datbayev
Nace.ai
zhanibek@nace.ai

Amine Mohamed Aboussalah
New York University
ama10288@nyu.edu

Abstract

Fine-tuning large language models (LLMs) is computationally expensive because it requires updating all model parameters. Low-Rank Adaptation (LoRA) reduces this cost by modifying a subset of weights, but selecting the appropriate rank introduces a trade-off: lower ranks improve efficiency at the expense of expressivity, while higher ranks enhance performance but increase computational burden. Existing adaptive LoRA methods lack a theoretical foundation to guide this trade-off optimally. We propose Geometric Low-Rank Adaptation (GeLoRA), a principled approach that estimates the intrinsic dimensionality of hidden data representations to adaptively select LoRA ranks. We show theoretically and empirically that the intrinsic dimension serves as a lower bound for the optimal rank of LoRA matrices, enabling a balance between efficiency and expressivity. Extensive experiments on GLUE, SQuAD (with DeBERTa), and MT-Bench (with LLaMA) demonstrate that GeLoRA consistently outperforms recent adaptive LoRA methods by up to +1.0%, while simultaneously reducing computational time by 13.5% to 64.2%, depending on the baseline, under the same parameter budget.

1 Introduction

LLMs excel in natural language processing (NLP) tasks but require fine-tuning for effective personalization. However, fine-tuning the entire model is computationally expensive in terms of time and memory. Parameter-Efficient Fine-Tuning (PEFT) offers a solution by adjusting only a subset of the model parameters (Han et al., 2024). Among PEFT methods, LoRA decomposes weight updates into low-rank components, significantly reducing computational costs while achieving performance comparable to full fine-tuning (Hu et al., 2021).

LoRA and its variants assume that pre-trained models exhibit low "intrinsic dimensionality" (Aghajanyan et al., 2020; Li et al., 2018), implying that weight updates should similarly be low-rank. However, a key challenge lies in determining the optimal rank values, which involves balancing expressivity and computational efficiency. Expressivity refers to the model's ability to capture complex patterns in the data, while computational efficiency pertains to the speed and resource requirements for fine-tuning. Lower ranks reduce expressivity but enhance memory efficiency and speed, while higher ranks increase expressivity at the cost of more resources and computation time. The trade-off is evident: lower ranks reduce expressivity but enhance memory efficiency and computational speed, whereas higher ranks increase expressivity at the cost of greater memory usage, longer computation times, and most likely more data to learn weights reliably. Typically, ranks are set uniformly across all layers, with practitioners relying on trial-and-error to achieve a balance between expressivity and efficiency. This is time-consuming and may not always yield optimal results.

Recent studies (Valeriani et al., 2023) show that hidden representations of transformer models exhibit low intrinsic dimensionality, suggesting potential connections between the manifold of data representations and the manifold of model parameters. This raises the question: ***Is there a connection between the manifold of data representations and the manifold of model parameters?***

We theoretically examine the relationship between the dimensionality of data representations and the ranks of weight updates, deriving a lower bound for the optimal rank as a function of the intrinsic dimensionalities of each transformer block's input and output representations. These intrinsic dimensionalities are specific to each transformer block and depend on the underlying model and dataset. Although determining the exact intrinsic dimension of each hidden representation is impractical, reliable estimates can be obtained using ex-

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 25174–25196
November 4-9, 2025

isting methods. In this work, we adopt the Two Nearest Neighbors (TwoNN) method (Facco et al., 2017), a robust and widely used estimator for estimating the intrinsic dimension of data representations in neural networks.(Ansuini et al., 2019; Doimo et al., 2020; Valeriani et al., 2023; Cheng et al., 2023; Kvinge et al., 2023; Basile et al., 2024).

Building on this, we develop a principled approach to improve LoRA methods. Specifically, we introduce ***Geometric Low-Rank Adaptation (GeLoRA)***, a method that dynamically adjusts the rank of weight updates for each transformer block as a function of the intrinsic dimensionalities of its input and output data manifold representations, achieving a more efficient and expressive trade-off between model performance and computational cost. Our work makes three key contributions:

1. We establish a theoretical framework to explain when LoRA works well. Specifically, we derive a lower bound on the rank needed for LoRA to be effective, showing that this depends on the intrinsic dimensionalities of data representation manifolds at the inputs and outputs of transformer blocks.

2. Based on our framework, we introduce GeLoRA, which adjusts the LoRA ranks across model layers to align with the intrinsic dimensionalities of data representations.

3. We evaluate GeLoRA on several benchmarks, including GLUE, SQuAD, and MT-Bench, using encoder-based models (DeBERTa) and decoder-based large language models (LLaMA). GeLoRA consistently outperforms baselines such as standard LoRA and AdaLoRA, while keeping the total number of tunable parameters fixed.

2 Related work

In this section, we review PEFT methods for LLMs, examine their limitations and recent developments in LoRA techniques. We then explore the concept of intrinsic dimensionality, a key theoretical concept that we leverage in our proposed methodology.

2.1 Parameter-efficient fine-tuning

LLMs, such as GPT (Brown et al., 2020) and BERT (Devlin et al., 2019), have achieved state-of-the-art performance in various NLP tasks like sentiment analysis, machine translation, and question answering. However, while these models excel in general tasks, developing personalized models requires additional fine-tuning, which needs to be handled efficiently given the substantial computational costs involved. PEFT (Han et al., 2024) addresses this challenge by adjusting only a small subset of model parameters, significantly reducing resource consumption compared to full model fine-tuning.

Several approaches have emerged to optimize fine-tuning efficiency. One such approach, BitFit (Zaken et al., 2022), adjusts only the bias terms and task-specific head, leaving the rest of the model unchanged. Another approach uses adapter layers (Houlsby et al., 2019), which add small trainable layers, "adapters", to adapt the model to new tasks without modifying its weights. Context-based fine-tuning methods (Petrov et al., 2024) modify the input representation, such as in prefix tuning (Li and Liang, 2021), where task-specific parameters are appended to the input embedding to guide model responses. Finally, LoRA (Hu et al., 2021; Dettmers et al., 2023; Hayou et al., 2024) reduces the number of trainable parameters by decomposing update weight matrices into low-rank components. However, LoRA faces the challenge of determining an appropriate rank for these matrices, typically set uniformly across layers via a trial-and-error process, which can be suboptimal.

More recently, several LoRA variants have been developed to address the challenge of setting uniform rank values by dynamically adjusting the rank for each layer. These variants optimize rank allocation by computing importance scores and pruning unnecessary ranks, while ensuring that the total number of parameters does not exceed a fixed budget. Notable examples include AdaLoRA (Zhang et al., 2023), SaLoRA (Hu et al., 2023), SoRA (Ding et al., 2023), and ALoRA (Liu et al., 2024).

AdaLoRA adaptively distributes low-rank updates across model weights by parameterizing the update matrices via singular value decomposition (SVD), and evaluates the importance of each direction using sensitivity scores. Important directions receive higher ranks, while less significant ones are pruned. To reduce noise from mini-batch variability, it smooths the scores using moving averages and applies orthogonality regularization to preserve the SVD structure. However, the method increases computational overhead and relies on heuristic sensitivity scores without strong theoretical justification. SaLoRA instead learns the rank of each matrix using a binary gating mechanism, where a diagonal gate matrix selectively deactivates less impor-

tant components via a differentiable approximation of the ℓ_0-norm. While this improves efficiency, it may introduce training instability; orthogonality regularization (Brock et al., 2017) is used to mitigate this, though it also adds computational cost. Similarly, SoRA dynamically adjusts ranks during training using a sparse gating unit, optimized by minimizing the ℓ_0-norm via the proximal gradient method (Rockafellar, 1970). Although effective for adaptation, this approach lacks strong theoretical justification and may not generalize well. Lastly. ALoRA introduces AB-LoRA, a method that estimates the importance of each rank by measuring the performance impact of removing or isolating specific ranks. This guides a gradual pruning strategy that reallocates rank budgets from less important to more critical transformer modules. However, ablation-based evaluation is computationally expensive and may limit the method's scalability. In contrast, GeLoRA proposes a geometry-aware, theoretically grounded criterion for adaptive rank allocation. This enables efficient and effective fine-tuning while addressing both the theoretical limitations and computational inefficiencies of prior adaptive LoRA methods.

2.2 Intrinsic dimensionality

Intrinsic dimensionality (ID) refers to the minimum number of parameters or variables needed to describe the data without significant loss of information. While the data may be embedded in a high-dimensional space, it often lies near a much lower-dimensional structure. Traditional methods such as PCA (Fan et al., 2010) assume data lies near a linear subspace, approximating it by a flat, low-dimensional space within a high-dimensional feature space. In contrast, modern techniques capture nonlinear structures by exploiting geometric or statistical properties of local neighborhoods (Ceruti et al., 2012; Campadelli et al., 2015; Johnsson et al., 2015; Amsaleg et al., 2018; Albergante et al., 2019). Among these, TwoNN (Facco et al., 2017) stands out for its simplicity, efficiency, and minimal assumptions. TwoNN estimates the intrinsic dimension (ID) using only local information from the two nearest neighbors, avoiding the linearity assumptions inherent in PCA. It scales well to high-dimensional data and exhibits robustness to moderate noise, though performance degrades under severe noise conditions.

Its favorable trade-offs have led to broad adoption in machine learning. TwoNN is applied to analyze neural network latent spaces for model compression and feature disentanglement (Valeriani et al., 2023; Ansuini et al., 2019), detect nonlinear correlations in joint embeddings (e.g., text-image) (Basile et al., 2025), and identify anomalies by spotting deviations from expected ID in high-dimensional data (Anderberg et al., 2024). However, TwoNN's accuracy diminishes in regions with highly nonuniform data density or near manifold boundaries, where local uniformity and smoothness assumptions break down. Despite these limitations, TwoNN remains widely used due to its strong theoretical grounding and computational efficiency. More details are provided in Appendix A.1.

3 Geometric LoRA

3.1 Intuition

Consider a linear map $f : x \mapsto Wx$, where the matrix W has a low rank r. The low rank of W implies that f compresses the semantic information of x into a lower-dimensional space such that $\dim \Im mf = r$ [1]. Although the functions approximated by transformer blocks are far more complex than a linear map, we will later show that intrinsic dimension profiles can provide insights for selecting appropriate ranks for each layer of a language model. Specifically, we show that they provide a lower bound on the number of parameters required to effectively encode information. We develop a formal theoretical framework to rigorously characterize how the rank of hidden representations relates to the number of parameters required for effective fine-tuning within a transformer block.

3.2 Theoretical formulation

For consistency, we maintain the notation used in the original LoRA paper (Hu et al., 2021). Without loss of generality, we will focus on the language modeling problem, where the goal is to maximize conditional probabilities given a task-specific prompt. Each task can be represented by a dataset containing context-target pairs $\mathcal{Z} = \{(x_i, y_i)\}$, where both x_i and y_i are sequences of tokens. The primary objective is to accurately predict y_i given x_i. For example, in a summarization task, x_i represents the original content and y_i its summary. Mathematically, this can be modeled as follows:

$$\max_{\phi \in \Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log(\mathbb{P}_\phi(y_t \mid x, y_{<t}))$$

[1] $\Im mf$ denotes the range of the function f.

Here, Φ denotes the parameter set of the model, and $\mathbb{P}_\Phi$ represents the conditional probability describing the relationship between context and target pairs. This probability distribution can be viewed as a point on a neuromanifold $\mathcal{M} = \{\text{NN}_\phi \mid \phi \in \Phi\}$. The geometry of this manifold is characterized by the Fisher Information Matrix (FIM) (Fisher, 1922) with respect to ϕ, which is given by:

$$\mathcal{I}(\phi) = \mathbb{E}_{x\sim\mathbb{P}_{\text{data}}, y\sim\mathbb{P}(\cdot|x;\phi)}\left[\left(\frac{\partial}{\partial\phi}\log\mathbb{P}(y \mid x;\phi)\right)\left(\frac{\partial}{\partial\phi}\log\mathbb{P}(y \mid x;\phi)\right)^T\right]$$

The FIM defines a Riemannian metric on the learning parameter space (Amari, 2021; Čencov, 1982). However, learning models often exhibit singularities (Watanabe, 2009), meaning that the rank of the FIM is less than its full dimensionality, typically due to parameter redundancy or degeneracies in the model structure. This is particularly relevant in transformer models, which contain millions or even billions of parameters due to their deep and wide architectures. Such high-dimensional parameter spaces are prone to redundancy and strong correlations between parameters (Dalvi et al., 2020), leading to linear dependencies among the gradients of the log-likelihood with respect to different parameters. Moreover, optimization algorithms such as Stochastic Gradient Descent (SGD) (Ruder, 2017) tend to favor flatter minima during training (Jastrzębski et al., 2018), often resulting in plateau regions where gradient magnitudes are small. This further contribute to the singular or near-singular behavior of the FIM, as indicated by the presence of eigenvalues approaching zero.

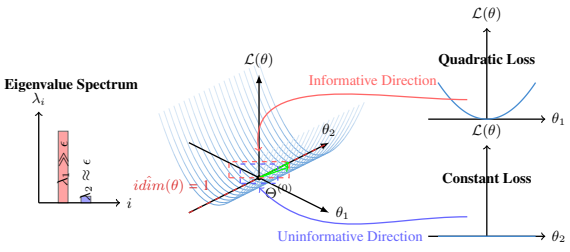

Figure 1: Locally near $\Theta^{(0)}$, the loss $\mathcal{L}(\theta_1, \theta_2) = \frac{1}{2}\theta_1^2$ depends solely on θ_1, with θ_2 lying in the kernel of the Hessian and thus not affecting the loss. Thus, the local low-loss region has intrinsic dimension one.

In this context, the rank of $\mathcal{I}(\phi)$, defined by the number of non-zero eigenvalues of the FIM, reflects the number of degrees of freedom (directions) at a point ϕ that can modify the probabilistic model $\mathbb{P}_\Phi(\cdot \mid \cdot)$. This is referred to as the local dimensionality (Sun and Nielsen, 2024). Figure 1 illustrates this concept, where the local dimensionality is 1, while the dimension of the space is 2.

Definition 3.1 (**Local Dimensionality**). The local dimensionality, denoted as $d(\phi)$, is defined as the rank of the information matrix $\mathcal{I}(\phi)$. It represents the number of parameters that need to be optimized in the model, indicating the effective dimensionality of the parameter space around the point ϕ.

Ideally, we aim to compute the local dimensionality of the parameter space at each gradient step. However, two primary challenges limit this approach. Firstly, the computational feasibility poses a significant obstacle, as computing the FIM at each step requires extensive computational resources. Secondly, the FIM behaves as a random matrix, typically maintaining full rank with probability 1, and it often has very small eigenvalues on the order of $\epsilon \in \mathbb{R}^+$ (Feng and Zhang, 2007). According to the Cramér-Rao bound, the variance of the parameter estimates is greater than or equal to $1/\epsilon$. Therefore, parameters associated with such small eigenvalues provide negligible information about the model and can be effectively considered uninformative. Disregarding parameters with very small eigenvalues leads us to the concept of intrinsic dimension, which can also be seen as the minimum number of parameters required to capture the local variance of the data points effectively. Thus, it represents a lower bound on the local dimensionality.

Theorem 3.2 (**Intrinsic Dimension as a Lower Bound**). *The intrinsic dimension* $\hat{idim}(\phi)$ *is a lower bound to the local dimensionality* $d(\phi)$.

$$d(\phi) \geq \hat{idim}(\phi).$$

There are still several major challenges that make it difficult to apply geometry-based fine-tuning methods in practice. First, calculating the FIM and determining its rank is computationally intensive, which becomes impractical for large models. Second, estimating the intrinsic dimensionality of the model's neuromanifold, defined by the space of probability distributions induced by neural network parameters, is intractable. In addition, the number of parameters that need to be fine-tuned (often guided by the rank of the FIM) is usually considered at the level of the entire model. This makes it difficult to adjust ranks locally for each layer or matrix, limiting flexibility and precision.

To address these limitations, we shift focus from the parameter space to the geometry of data representations as they propagate through the network.

Specifically, we leverage the fact that we have direct access to both the input data and its hidden representations across the transformer blocks of a LLM. Each block performs a nonlinear transformation that may alter the intrinsic geometry of the input manifold. By analyzing how the intrinsic dimensionality evolves across layers, we gain insight into the complexity and information content processed at each block. This perspective allows us to guide the local selection of LoRA ranks based on the intrinsic dimensions of the input and output data representations at each block. To formalize this idea, we introduce a theoretical rank-based bound that connects the intrinsic geometry of the data manifolds to the local structure of the model parameter space, enabling principled, geometric data-driven fine-tuning strategies.

Theorem 3.3 (**Rank Bound of Transformer Blocks**). *Let $\mathcal{M}$ denote a language model consisting of N transformer blocks. For each $i \in \{1, 2, \ldots, N\}$, the i-th transformer block is represented by $\mathcal{T}_i : \mathbb{R}^{n_{i-1}} \times \mathbb{R}^{p_{i-1}} \to \mathbb{R}^{n_i}$, which maps the hidden representation $\mathcal{H}_{i-1} \subset \mathbb{R}^{n_{i-1}}$ and parameters $\theta_{i-1} \in \mathbb{R}^{p_{i-1}}$ to the next hidden state $\mathcal{H}_i \subset \mathbb{R}^{n_i}$. Consider that the $\mathcal{H}_{i-1}$ lies on a **data manifold** $\mathcal{N}_{i-1}$ with **intrinsic dimension** d_{i-1} embedded in $\mathbb{R}^{n_{i-1}}$, and $\mathcal{H}_i$ lies on $\mathcal{N}_i$ with **intrinsic dimension** d_i embedded in $\mathbb{R}^{n_i}$. Let the **parameter manifold** $\Theta_{i-1} \subset \mathbb{R}^{p_{i-1}}$ be a submanifold locally defined around θ_{i-1}, with **intrinsic dimension** $d_{\theta_{i-1}}$, representing the number of independent parameters in the neighborhood. The rank of the transformer block $\mathcal{T}_i$, defined as*

$$\mathit{rank}(\mathcal{T}_i) = \max_{(x,\theta) \in \mathcal{H}_{i-1} \times \Theta_{i-1}} \mathit{rank}(J(\mathcal{T}_i, x, \theta)),$$

where $J(\mathcal{T}_i, x, \theta)$ is the Jacobian matrix of $\mathcal{T}_i$ evaluated at (x, θ), satisfies the following bound:

$$d_i \leq \mathit{rank}(\mathcal{T}_i) \leq d_{i-1} + d_\theta.$$

Based on Theorem 3.3, we derive Corollary 3.4, which provides a bound on the contribution of the parameter manifold to the model's expressiveness.

Corollary 3.4 (**Lower Bound on Parameter Dimensionality**). *Let $d_{\theta_{i-1}}$ denote the effective dimensionality of the parameter manifold, and let d_i and d_{i-1} represent the intrinsic dimensions of the data manifold at layers i and $i-1$, respectively. The dimensionality of the parameter manifold satisfies: $d_{\theta_{i-1}} \geq \max(d_i - d_{i-1}, 0)$.*

Corollary 3.4 establishes a geometric constraint on model expressivity: the parameter manifold must have at least $d_i - d_{i-1}$ effective degrees of freedom to support an increase in the intrinsic dimensionality of the data manifold across layers. When $d_i > d_{i-1}$, parameters must enable expansion into higher-dimensional features; when $d_i \leq d_{i-1}$, transformations are compressive or dimension-preserving and perform operations like filtering without increasing dimensionality.

Two important questions arise: (1) Is the bound potentially loose, that is, does it become ineffective when there is no increase in intrinsic dimensionality? (2) If the effective number of tunable parameters changes during optimization, must it be recomputed after each gradient step, and if so, does this introduce significant computational overhead?

When the inequality is loose and the computed lower bound is zero, certain LoRA weights are assigned a rank of zero. This typically occurs when the corresponding layers already encode task-relevant information, making fine-tuning unnecessary (Hartford et al., 2024). Freezing such layers can reduce computational overhead without degrading performance. Although recomputing the optimal number of parameters after each gradient step is computationally costly, models actually tend to compress data representations over time. Thus, ss training progresses, fewer parameters are needed to capture local variance in the hidden representations. This suggests that the number of parameters required for effective adaptation may naturally decrease over the course of training (Theorem 3.5).

Theorem 3.5 (**Transformer Rank Bound Dynamics**). *Let $\mathcal{T}_i$ denote the i-th transformer block in a language model, and let d_i^0 be the intrinsic dimension of the data manifold at its input at initialization. Define the initial parameter rank as $d_{\theta_i}^0 := \max(0, d_i^0 - d_{i-1}^0)$, representing the number of degrees of freedom needed to capture additional information at layer i. Let $d_{\theta_i}^t$ denote the effective rank of the transformer block after t training steps. Then, we have: $d_{\theta_i}^0 \geq d_{\theta_i}^t \quad$ for all $t \geq 0$.*

4 Methodology

Figure 2 provides a schematic overview of the GeLoRA methodology, which begins by computing the hidden representations of the data using a language model and determining the intrinsic dimensions of these hidden representations.

For each transformer block i, let d_{i-1} represent

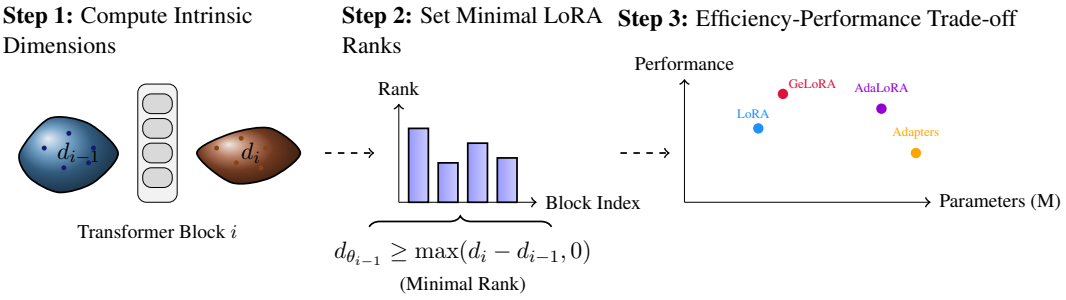

Figure 2: Overview of the GeLoRA methodology. The approach consists of three main steps: *(1)* analyzing the intrinsic dimensionality of data representations, *(2)* assigning minimal LoRA ranks based on these dimensions, and *(3)* performing efficient fine-tuning to balance computational cost and model expressivity.

the intrinsic dimension of the data manifold at the input and d_i represent the intrinsic dimension at the output. The minimum rank $d_{\theta_{i-1}}$ required for each block is determined based on the condition we derived in Corollary 3.4: $d_{\theta_{i-1}} \geq \max(d_i - d_{i-1}, 0)$, where the difference $\max(d_i - d_{i-1}, 0)$ indicates the necessary capacity to capture any dimensional expansion of the data manifold between consecutive layers. An adaptive scaling factor α_i is then applied across blocks to maintain a consistent ratio $\alpha_i / r_i = \text{const}$, preserving the proportion of adaptation strength relative to rank. This enables an efficient fine-tuning process that balances expressivity with computational efficiency.

To estimate the intrinsic dimension d of the hidden representations, we employ the TwoNN method (Facco et al., 2017). More details are provided in Appendix A.1. In high-dimensional settings, the TwoNN method tends to provide a conservative estimate, often serving as a lower bound for the true intrinsic dimension. To illustrate this, we conducted experiments on established benchmark datasets comparing the estimated values to the ground truth. The results of these experiments are summarized in Appendix C.2. To mitigate the risk of underestimating the intrinsic dimension, sometimes resulting in an inaccurate value of zero rank, we add a small offset of 1 to each rank lower bound. Moreover, we compute a lower bound on the rank of each transformer block, rather than for the individual key, query, value and output projection matrices within the block. Since the distribution of effective tunable parameters across these matrices is unknown, we adopt a uniform allocation strategy: the rank of each matrix in the block is set equal to the computed lower bound (i.e., the rank of the transformer block). Although this approach implicitly assumes equal contribution from each component, it provides a tractable approximation in the absence of a fine-grained estimation method.

$$d_{\theta_{K_{i-1}}} = d_{\theta_{Q_{i-1}}} = d_{\theta_{V_{i-1}}} = d_{\theta_{O_{i-1}}} = \max(d_i - d_{i-1}, 0) + 1,$$

where $d_{\theta_{K_{i-1}}}, d_{\theta_{Q_{i-1}}}, d_{\theta_{V_{i-1}}}$, and $d_{\theta_{O_{i-1}}}$ are, respectively, the ranks of the key, query, value and output matrices of the transformer block i. The pseudocode of GeLoRA is in Appendix B.3.

5 Experiments

5.1 Fine-tuning techniques and datasets

We evaluate GeLoRA on natural language understanding, question answering and instruction following tasks. For understanding, we use the GLUE benchmark (Wang et al., 2019). For question answering, we use SQuAD (Rajpurkar et al., 2016). Lastly, instruction following is trained on Airoboros (Durbin, 2024) and evaluated on MT-Bench (Zheng et al., 2023a). Dataset statistics are in Appendix D. We conduct experiments on both encoder-only and decoder-only models. For the encoder-only family, we use DeBERTaV3 (He et al., 2021), which is widely adopted in prior PEFT benchmarks for natural language understanding and question answering (Hu et al., 2021; Ding et al., 2023; Qiang et al., 2024; Chang et al., 2025; Zhang et al., 2023). For the decoder-only family, we use LLaMA 3 (8B) (Grattafiori et al., 2024), a state-of-the-art model designed for generative tasks.

We compare GeLoRA against various baselines, including weight update tuning (Zaken et al., 2022), adapter-based methods (Houlsby et al., 2019; Pfeiffer et al., 2021), and LoRA variants (Hu et al., 2021; Ding et al., 2023; Zhang et al., 2023). For LoRA-based methods, the maximum rank is set to 4, that is, four times the rank budget of GeLoRA, while for adapters, the reduction factor is set to the maximum to ensure comparable parameter budgets.

Table 1: Results with DeBERTaV3-base on SQuADv1.1 and SQuADv2.0. Here # Params is the number of trainable parameters. The best results are shown in **red bold**, and second best results are shown in **red bold**.

	# PARAMS	SQUADV1.1 EM	SQUADV1.1 F1	SQUADV2.0 EM	SQUADV2.0 F1
FULL FT	183.83M	86.12 ± 0.28	92.68 ± 0.13	83.03 ± 0.49	86.21 ± 0.51
HADAPTER	647.45×10^{-3}M	84.58 ± 0.20	91.57 ± 0.13	80.79 ± 1.10	84.28 ± 1.13
PADAPTER	309.89×10^{-3}M	83.00 ± 0.06	90.57 ± 0.10	78.17 ± 0.95	81.94 ± 0.94
LORA$_{r=2}$	150.54×10^{-3} M	84.45 ± 0.35	91.35 ± 0.25	**83.15 ± 0.77**	86.16 ± 0.74
LORA$_{r=1}$	75.27×10^{-3}M	86.23 ± 0.16	92.51 ± 0.16	81.09 ± 0.66	84.22 ± 0.63
ADALORA$_{r=1}$	147.55×10^{-3}M	**86.31 ± 0.30**	92.58 ± 0.21	81.12 ± 0.35	84.23 ± 0.31
ADALORA$_{r=2}$	221.33×10^{-3}M	86.27 ± 0.31	**92.61 ± 0.27**	**81.68 ± 0.51**	**84.80 ± 0.50**
GELORA	80.34×10^{-3}M	**86.72 ± 0.27**	**92.84 ± 0.20**	**83.15 ± 0.22**	**86.25 ± 0.24**

5.2 Experimental setting

We implemented all algorithms using PYTORCH, based on the publicly available HUGGINGFACE TRANSFORMERS (Wolf et al., 2020) code-base. For optimization, we used the ADAMW optimizer (Loshchilov and Hutter, 2019), which features parameters set to $\epsilon = 10^{-6}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$, and we fixed the batch size to 32. The numerical results were averaged over five runs with random seeds and we report standard deviations to ensure statistical robustness. The alpha rank ratio for low-rank adaptation techniques was fixed at 32, consistent with prior work (Hu et al., 2021; Zhang et al., 2023), and was not fine-tuned further. For estimating intrinsic dimension, we used the SCIKIT-DIMENSION package (Bac et al., 2021). All experiments were conducted on 8 NVIDIA A100-SXM4 GPUs. Additional training details can be found in Appendix F.

5.3 Numerical results

5.3.1 Question answering: SQuAD

Our experimental results demonstrate the efficiency of GeLoRA against baseline approaches on the SQuADv1.1 and SQuADv2.0 benchmarks. GeLoRA achieves better performance with EM/F1 scores of $\mathbf{86.72/92.84}$ and $\mathbf{83.15/86.25}$, respectively, surpassing other fine-tuning techniques while using only a fraction of trainable parameters. Table 1 reveals consistent performance improvements over existing parameter-efficient methods. GeLoRA outperforms LoRA variants by margins of $\mathbf{0.45 - 2.27}$ points in EM score on SQuADv1.1, with similar gains observed on SQuADv2.0. The performance delta is more pronounced when compared to adapter-based methods, showing improvements of $\mathbf{2.14}$ and $\mathbf{3.72}$ points over HAdapter and PAdapter, respectively, on the SQuADv1.1 dataset.

5.3.2 Instruction following: MT-Bench

We fine-tuned LLaMA 3 on the Airoboros dataset (Durbin, 2024) and evaluated its performance using the MT-Bench dataset (Zheng et al., 2023a). For evaluation, we used the LLM-as-a-judge approach, where responses generated by the fine-tuned LLaMA model were assessed by another LLM (Zheng et al., 2023b). In our case, we used the instruct version of Mistral (Jiang et al., 2023), which assigned scores from 1 to 10, with 1 representing the lowest quality and 10 the highest quality. To ensure consistency, we set the temperature to 0.1 and performed eight evaluation runs. The results remained unchanged across these runs, and are summarized in Table 3.

Table 3: Comparison of fine-tuning techniques on instruction tuning task (MT-Bench) using LLaMA 3. The average performance score across 8 evaluation runs is reported for each configuration. The best results are shown in **red bold**, and second best in **blue bold**..

METHOD	PARAMS	AVG SCORE
BASE MODEL	-	8.04
LORA$_{r=1}$	851.97×10^{-3}M	9.04
LORA$_{r=2}$	1703.94×10^{-3}M	8.94
LORA$_{r=4}$	3407.87×10^{-3}M	8.97
ADALORA$_{r=1}$	1704.19×10^{-3}M	**9.06**
ADALORA$_{r=2}$	2556.29×10^{-3}M	8.69
ADALORA$_{r=4}$	7668.86×10^{-3}M	**9.06**
GELORA	985.09×10^{-3}M	**9.09**

GeLora achieves the highest average score of 9.09, making it the best-performing fine-tuning method for this task. LoRA-based methods perform similarly, with rank 1 scoring 9.04, while higher ranks (2, 4, and 8) score slightly lower (8.94–8.97). AdaLoRA shows more variation, with rank 1 and rank 4 both scoring 9.06, while rank 2 scores 8.69. BitFit records the lowest score among fine-tuned models at 8.70, though it still surpasses the base model, which scores 8.04.

Table 2: Results with DeBERTaV3-base on GLUE test set. The best results for each dataset are highlighted in **red bold**, while the second-best results are **blue bold**. We report the average correlation for STS-B. *Full FT* represent full fine-tuning, *HA Adapter* represents Houlsby Adapters, and *PF Adapter* represents Pfeiffer Adapters.

Method	# Params	CoLA	STS-B	MRPC	QNLI	SST-2	RTE	QQP	MNLI	Average
FULL FT	184.42M	68.28 ± 1.39	91.32 ± 0.45	73.53 ± 3.25	93.81 ± 0.21	94.68 ± 0.30	73.67 ± 1.33	88.54 ± 0.23	89.65 ± 0.19	84.19
BITFIT	105.22×10^{-3}M	68.66 ± 1.87	89.40 ± 0.57	85.20 ± 1.56	92.10 ± 0.13	94.54 ± 0.30	75.11 ± 2.52	86.25 ± 0.27	86.04 ± 0.58	84.66
HA ADAPTER	647.45×10^{-3}M	68.46 ± 1.08	91.26 ± 0.13	86.76 ± 0.44	93.52 ± 0.40	95.32 ± 0.35	80.43 ± 2.78	89.08 ± 0.06	89.07 ± 0.19	86.74
PF ADAPTER	619.79×10^{-3}M	68.59 ± 1.43	89.85 ± 0.13	88.24 ± 1.07	93.33 ± 0.30	**95.55 ± 0.41**	79.14 ± 2.95	88.60 ± 0.14	88.82 ± 0.07	86.52
LORA$_{r=1}$	75.27×10^{-3}M	**69.68 ± 0.92**	88.29 ± 3.28	**88.43 ± 1.37**	93.83 ± 0.13	95.04 ± 0.43	80.29 ± 1.33	**90.41 ± 0.05**	89.64 ± 0.15	**86.95**
LORA$_{r=2}$	148.99×10^{-3}M	69.04 ± 1.51	88.60 ± 3.09	87.75 ± 0.69	93.79 ± 0.17	95.04 ± 0.22	80.43 ± 1.60	90.78 ± 0.11	89.77 ± 0.39	86.90
SORA$_{r=1}$	75.27×10^{-3}M	61.78 ± 2.37	78.88 ± 6.55	87.45 ± 3.06	88.66 ± 0.68	91.94 ± 0.52	**82.32 ± 2.49**	84.08 ± 0.66	82.09 ± 0.82	82.15
SORA$_{r=2}$	148.99×10^{-3}M	67.85 ± 1.33	84.33 ± 3.90	88.04 ± 2.00	89.76 ± 0.41	91.40 ± 0.32	78.84 ± 3.74	83.80 ± 0.46	83.50 ± 0.76	83.44
ADALORA$_{r=1}$	147.55×10^{-3}M	69.28 ± 0.33	**92.08 ± 0.15**	84.61 ± 0.91	93.84 ± 0.15	95.07 ± 0.42	74.96 ± 3.82	89.92 ± 0.10	**90.12 ± 0.20**	86.23
ADALORA$_{r=2}$	221.33×10^{-3}M	64.76 ± 1.49	91.56 ± 0.12	87.25 ± 0.93	**94.07 ± 0.12**	**95.44 ± 0.34**	**81.87 ± 0.95**	90.12 ± 0.08	**90.13 ± 0.26**	86.90
GELORA	TABLE 4	**70.96 ± 0.96**	**91.66 ± 0.48**	**89.9 ± 0.79**	**93.87 ± 0.23**	95.05 ± 0.24	81.29 ± 1.64	**90.81 ± 0.12**	89.84 ± 0.22	**87.92**

5.3.3 Natural language understanding: GLUE benchmark

Our experiments on the GLUE benchmark (Table 2) show that GeLoRA achieves the highest overall performance, with an average score of 87.92, surpassing strong parameter-efficient baselines such as HA Adapter (86.74), LoRA (86.95), and AdaLoRA (86.90). GeLoRA outperforms other techniques on CoLA (70.96), STS-B (91.66), and MRPC (89.90), while also remaining highly competitive on other tasks such as SST-2 (95.05) and RTE (81.29). Compared to full fine-tuning (84.19) and other adapter-based methods, GeLoRA consistently yields superior or comparable results across diverse datasets.

A key advantage of GeLoRA lies in its adaptive parameter allocation strategy (Table 4), which assigns only 0.09M–0.13M parameters depending on task complexity. This is orders of magnitude smaller than full fine-tuning (184.42M) and remains within the efficiency range of other PEFT approaches (0.08M–0.22M). By allocating parameters selectively rather than uniformly, GeLoRA avoids unnecessary overhead on tasks for which pretrained representations already capture relevant features. This design not only improves efficiency but also better aligns with the intuition that tasks vary in difficulty and benefit unevenly from additional adaptation. A natural question that arises is: what happens if we simply increase the ranks in LoRA, or introduce additional complexity in adaptive variants? To explore this, we evaluate these methods in a high-rank setting by applying an offset to the GeLoRA lower bound. Concretely, the ranks are defined as

$$r_{K_i} = r_{Q_i} = r_{V_i} = r_{O_i} = \max(d_{i+1} - d_i, 0) + o,$$

where o is the applied offset.

The results in Table 5 show that, on average, GeLoRA still outperforms LoRA and its adaptive variants even in high-rank settings. Moreover, methods such as SoRA and AdaLoRA benefit more from the higher ranks than in the low-rank regime since they are able to prune unnecessary capacity during training. In contrast, GeLoRA already incorporates this inductive bias by design, estimating task-appropriate ranks before fine-tuning without relying on post-hoc adjustments. This observation highlights that simply increasing rank does not guarantee better performance. Instead, principled strategies for parameter allocation, as implemented in GeLoRA, are more effective in balancing efficiency and accuracy.

Table 4: GeLoRA parameters and mean ranks across GLUE tasks. Values in parentheses show rounded ranks.

TASK	# PARAMS ($\times 1e^3$)	MEAN RANK
CoLA	99.84M	1.33 (1)
STS-B	111.36M	1.50 (2)
MRPC	130.56M	1.75 (2)
QNLI	93.70M	1.25 (1)
SST-2	87.55M	1.17 (1)
RTE	130.56M	1.75 (2)
MNLI	100.61M	1.33 (1)
QQP	118.27M	1.58 (2)

5.3.4 Computational Efficiency

Finally, we evaluate the efficiency of different techniques under a fixed budget constraint. The reported times cover only the fine-tuning stage. For GeLoRA, there is also a one-time preprocessing step to estimate the intrinsic dimension. Training time was measured across eight datasets, with experiments run for 20 epochs on all datasets except RTE (50 epochs) and QQP and MNLI (10 epochs each). All experiments were conducted on identical hardware, eight NVIDIA A100-SXM4 GPUs, with a fixed batch size of 32, and the ranks of LoRA and its variants were adjusted to match the rounded mean rank of GeLoRA.

Table 5: Results with DeBERTaV3-base on GLUE test set using higher ranks. The best results are shown in **bold red**, while the second-best are **bold blue**.

Method	# Params	CoLA	STS-B	MRPC	QNLI	SST-2	RTE	QQP	MNLI	Average
LoRA$_{r=4}$	297.22 $\times 10^{-3}$M	67.52 ± 0.38	89.84 ± 1.36	89.12 ± 2.09	93.77 ± 0.10	95.39 ± 0.40	81.73 ± 1.55	91.06 ± 0.08	89.71 ± 0.37	87.26
LoRA$_{r=8}$	$\times 10^{-3}$M	68.92 ± 0.84	89.02 ± 1.41	68.81 ± 34.87	79.42 ± 2.83	94.84 ± 0.55	93.53 ± 0.16	88.67 ± 0.90	77.61 ± 20.96	82.60
LoRA$_{r=16}$	$\times 10^{-3}$M	66.28 ± 1.03	89.80 ± 1.22	63.06 ± 33.59	73.81 ± 7.46	93.97 ± 1.05	91.79 ± 0.22	68.09 ± 9.82	55.79 ± 24.52	75.32
LoRA$_{r=64}$	$\times 10^{-3}$M	59.73 ± 6.21	88.53 ± 1.18	62.15 ± 35.88	69.06 ± 7.24	94.01 ± 0.41	91.86 ± 0.44	68.18 ± 5.00	64.25 ± 28.47	74.72
SoRA$_{r=4}$	297.22 $\times 10^{-3}$M	63.47 ± 1.99	81.68 ± 7.93	87.06 ± 1.15	90.04 ± 0.67	92.46 ± 0.59	86.09 ± 2.69	84.16 ± 0.46	85.14 ± 0.34	83.76
SoRA$_{r=8}$	$\times 10^{-3}$M	66.92 ± 3.67	89.22 ± 0.62	89.12 ± 1.00	84.93 ± 2.69	94.22 ± 0.47	91.36 ± 0.20	84.04 ± 0.33	85.07 ± 0.37	85.61
SoRA$_{r=16}$	$\times 10^{-3}$M	65.09 ± 2.35	90.59 ± 2.37	81.69 ± 9.15	84.93 ± 1.97	93.99 ± 0.45	91.88 ± 0.19	84.60 ± 0.73	85.91 ± 0.51	84.84
SoRA$_{r=64}$	$\times 10^{-3}$M	64.24 ± 1.03	90.98 ± 0.73	85.66 ± 4.60	86.67 ± 1.92	94.40 ± 0.42	92.22 ± 0.17	81.85 ± 1.02	81.70 ± 1.02	84.72
AdaLoRA$_{r=4}$	442.66 $\times 10^{-3}$M	68.62 ± 1.22	90.54 ± 0.23	84.31 ± 1.45	**94.11 ± 0.12**	95.39 ± 0.44	79.71 ± 1.24	91.57 ± 0.22	90.27 ± 0.18	86.81
AdaLoRA$_{r=8}$	$\times 10^{-3}$M	69.95 ± 0.92	87.94 ± 1.14	**91.93 ± 0.10**	82.64 ± 1.74	95.03 ± 0.19	**93.90 ± 0.33**	**91.70 ± 0.17**	**90.31 ± 0.13**	**87.92**
AdaLoRA$_{r=16}$	$\times 10^{-3}$M	**70.73 ± 1.58**	86.96 ± 1.57	**91.50 ± 0.14**	82.73 ± 1.44	**95.57 ± 0.37**	**93.83 ± 0.13**	**91.78 ± 0.38**	**90.28 ± 0.15**	**87.92**
AdaLoRA$_{r=64}$	$\times 10^{-3}$M	69.07 ± 1.63	87.94 ± 1.01	**91.93 ± 0.34**	82.73 ± 1.70	95.44 ± 0.20	**93.83 ± 0.14**	81.70 ± 1.02	90.07 ± 0.50	86.59
GeLoRA	Table 4	**70.96 ± 0.96**	**91.66 ± 0.48**	89.9 ± 0.79	93.87 ± 0.23	95.05 ± 0.24	81.29 ± 1.64	90.81 ± 0.12	89.84 ± 0.22	**87.92**

Table 6: Training computational cost (runtime) in seconds for DeBERTaV3-base fine-tuning on GLUE tasks. The runtime for each fine-tuning is indicated in seconds. The best results for each dataset are highlighted in **bold**.

Dataset	GeLoRA	LoRA	AdaLoRA	BitFit	HAdapter	PAdapter
CoLA	**85.68 ± 2.27**	100.95 ± 10.53	165.43 ± 0.28	157.27 ± 1.07	117.98 ± 0.07	113.52 ± 0.11
STS-B	**59.13 ± 3.26**	78.26 ± 6.92	157.50 ± 8.36	122.68 ± 0.40	84.51 ± 0.06	81.27 ± 0.04
MRPC	**40.42 ± 0.30**	58.75 ± 1.73	112.61 ± 1.36	94.93 ± 0.34	57.41 ± 0.10	55.09 ± 0.03
QNLI	**736.57 ± 3.34**	865.76 ± 4.11	2328.60 ± 24.81	1341.47 ± 21.03	1254.14 ± 1.21	1205.86 ± 1.83
SST-2	**475.58 ± 5.10**	482.38 ± 5.11	1140.65 ± 2.25	871.10 ± 5.05	807.91 ± 0.57	775.33 ± 0.56
RTE	**75.62 ± 0.29**	116.28 ± 7.30	207.89 ± 4.42	80.5 ± 0.24	104.38 ± 0.06	100.40 ± 0.11
Average	**245.5**	283.73	685.45	444.66	404.39	388.58

Table 9 shows that GeLoRA requires less fine-tuning time than competing methods. By contrast, AdaLoRA incurs extra overhead from maintaining moving averages of importance scores and computing gradients for orthogonality regularization. BitFit also requires training a task-specific head, which adds complexity and increases training time.

On the other hand, the pre-processing step for GeLoRA (estimating the intrinsic dimension) takes between 7 minutes and 1 hour on a single A100 GPU with 8 CPU cores, depending on dataset size. For very large datasets, this cost can be reduced using principled subsampling strategies, which are valid under the local uniformity assumption of the ID estimator (Appendix A.1.2). We view this one-time cost as reasonable, since it enables a more principled approach to parameter-efficient fine-tuning and gaining a deeper understanding of the underlying mechanisms.

6 Conclusion

In this work, we introduced GeLoRA, a geometry-aware method for adaptive low-rank fine-tuning. By leveraging intrinsic dimensionality estimates of hidden representations, GeLoRA assigns principled LoRA ranks per layer, balancing expressivity and efficiency. Our theoretical framework connects data representation geometry to parameter optimization needs, and our empirical results show consistent gains over existing PEFT methods on GLUE, SQuAD, and MT-Bench, with significantly fewer trainable parameters.

Limitations

GeLoRA presents a theoretically motivated approach to adaptive low-rank fine-tuning by leveraging intrinsic dimensionality estimates of hidden representations. However, the method's effectiveness depends heavily on the accuracy of the Two Nearest Neighbors (TwoNN) estimator, which is known to underestimate intrinsic dimensions in high-dimensional or non-uniform settings (Appendix A.1). Although we mitigate this with a fixed offset, this correction is an ad hoc heuristic and lacks theoretical justification. Additionally, the derived theoretical bounds connecting intrinsic dimensionality to LoRA ranks are only lower bounds and may be loose in practice. The tightness of these bounds and whether they meaningfully constrain the true optimal ranks remains an open question.

Another limitation lies in the scalability of GeLoRA. Our proposed methodology has been evaluated on benchmark datasets using DeBERTaV3 and LLaMA 3, leaving its behavior on larger models, multilingual LLMs, and multi-modal architectures untested. Moreover, although our approach reduces training cost, the preprocessing step to estimate intrinsic dimensions might introduce a nontrivial overhead, especially in larger-scale settings. Online, parallel computing, or approximate versions of this step could improve efficiency without sacrificing accuracy.

Future work will explore more robust and scalable intrinsic dimension estimators, extend the framework to multimodal and multilingual tasks, and investigate dynamic or sample-specific rank assignment.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *Preprint*, arXiv:2012.13255.

Luca Albergante, Jonathan Bac, and Andrei Zinovyev. 2019. Estimating the effective dimension of large biological datasets using fisher separability analysis. *Preprint*, arXiv:1901.06328.

Shun-Ichi Amari. 2021. Information geometry. *Int. Stat. Rev.*, 89(2):250–273.

Laurent Amsaleg, Oussama Chelly, Teddy Furon, Stéphane Girard, Michael E Houle, Ken-Ichi Kawarabayashi, and Michael Nett. 2018. Extreme-value-theoretic estimation of local intrinsic dimensionality. *Data Mining and Knowledge Discovery*, 32(6):1768–1805.

Alastair Anderberg, James Bailey, Ricardo J. G. B. Campello, Michael E. Houle, Henrique O. Marques, Miloš Radovanović, and Arthur Zimek. 2024. Dimensionality-aware outlier detection: Theoretical and experimental analysis. *Preprint*, arXiv:2401.05453.

Alessio Ansuini, Alessandro Laio, Jakob H. Macke, and Davide Zoccolan. 2019. Intrinsic dimension of data representations in deep neural networks. *Preprint*, arXiv:1905.12784.

Jonathan Bac, Evgeny M. Mirkes, Alexander N. Gorban, Ivan Tyukin, and Andrei Zinovyev. 2021. Scikit-dimension: A python package for intrinsic dimension estimation. *Entropy*, 23(10):1368.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, and Danilo Giampiccolo. 2006. The second pascal recognising textual entailment challenge. *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Lorenzo Basile, Santiago Acevedo, Luca Bortolussi, Fabio Anselmi, and Alex Rodriguez. 2025. Intrinsic dimension correlation: uncovering nonlinear connections in multimodal representations. *Preprint*, arXiv:2406.15812.

Lorenzo Basile, Nikos Karantzas, Alberto D'Onofrio, Luca Bortolussi, Alex Rodriguez, and Fabio Anselmi. 2024. Investigating adversarial vulnerability and implicit bias through frequency analysis. *Preprint*, arXiv:2305.15203.

Andrew Brock, Theodore Lim, J. M. Ritchie, and Nick Weston. 2017. Neural photo editing with introspective adversarial networks. *Preprint*, arXiv:1609.07093.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss,

Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Paola Campadelli, Elena Casiraghi, Claudio Ceruti, and Alessandro Rozza. 2015. Intrinsic dimension estimation: Relevant techniques and a benchmark framework. *Mathematical Problems in Engineering*, 2015:1–21.

N. N. Čencov. 1982. *Statistical decision rules and optimal inference*, volume 53 of *Translations of Mathematical Monographs*. American Mathematical Society, Providence, R.I. Translation from the Russian edited by Lev J. Leifman.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Claudio Ceruti, Simone Bassis, Alessandro Rozza, Gabriele Lombardi, Elena Casiraghi, and Paola Campadelli. 2012. Danco: Dimensionality from angle and norm concentration. *Preprint*, arXiv:1206.3881.

Yupeng Chang, Yi Chang, and Yuan Wu. 2025. Balora: Bias-alleviating low-rank adaptation to mitigate catastrophic inheritance in large language models. *Preprint*, arXiv:2408.04556.

Emily Cheng, Corentin Kervadec, and Marco Baroni. 2023. Bridging information-theoretic and geometric compression in language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12397–12420, Singapore. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. *Preprint*, arXiv:2004.04010.

Francesco Denti, Diego Doimo, Alessandro Laio, and Antonietta Mira. 2022. The generalized ratios intrinsic dimension estimator. *Sci Rep*, 12(1):20005.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Preprint*, arXiv:2305.14314.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. 2023. Sparse low-rank adaptation of pre-trained language models. *Preprint*, arXiv:2311.11696.

Diego Doimo, Aldo Glielmo, Alessio Ansuini, and Alessandro Laio. 2020. Hierarchical nucleation in deep neural networks. *Preprint*, arXiv:2007.03506.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Jon Durbin. 2024. airoboros-gpt4-1.4.1-mpt. `https://huggingface.co/datasets/jondurbin/airoboros-gpt4-1.4.1-mpt`. Accessed: 2024-11-28.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. 2017. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports*, 7(1):12140.

Mingyu Fan, Nannan Gu, Hong Qiao, and Bo Zhang. 2010. Intrinsic dimension estimation of data by principal component analysis. *Preprint*, arXiv:1002.2050.

Xinlong Feng and Zhinan Zhang. 2007. The rank of a random matrix. *Applied Mathematics and Computation*, 185(1):689–694.

R A Fisher. 1922. On the mathematical foundations of theoretical statistics. *Philos. Trans. R. Soc. Lond.*, 222(594-604):309–368.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Aldo Glielmo, Iuri Macocco, Diego Doimo, Matteo Carli, Claudio Zeni, Romina Wild, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. 2022. Dadapy: Distance-based analysis of data-manifolds in python. *Patterns*, page 100589.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

V. Guillemin and A. Pollack. 2010. *Differential Topology*. AMS Chelsea Publishing. AMS Chelsea Pub.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Preprint*, arXiv:2403.14608.

Eric Hartford, Lucas Atkins, Fernando Fernandes Neto, and David Golchinfar. 2024. Spectrum: Targeted training on signal to noise ratio. *Preprint*, arXiv:2406.06623.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *Preprint*, arXiv:2402.12354.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. *Preprint*, arXiv:2006.03654.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *Preprint*, arXiv:1902.00751.

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

Yahao Hu, Yifei Xie, Tianfeng Wang, Man Chen, and Zhisong Pan. 2023. Structure-aware low-rank adaptation for parameter-efficient fine-tuning. *Mathematics*, 11:NA.

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. 2018. Three factors influencing minima in sgd. *Preprint*, arXiv:1711.04623.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Kerstin Johnsson, Charlotte Soneson, and Magnus Fontes. 2015. Low bias local intrinsic dimension estimation from expected simplex skewness. *IEEE Trans Pattern Anal Mach Intell*, 37(1):196–202.

Henry Kvinge, Davis Brown, and Charles Godfrey. 2023. Exploring the representation manifolds of stable diffusion through the lens of intrinsic dimension. *Preprint*, arXiv:2302.09301.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. *Preprint*, arXiv:1804.08838.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *Preprint*, arXiv:2101.00190.

Zequan Liu, Jiawen Lyn, Wei Zhu, Xing Tian, and Yvette Graham. 2024. Alora: Allocating low-rank adaptation for fine-tuning large language models. *Preprint*, arXiv:2403.16187.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Rotem Mulayoff and Tomer Michaeli. 2020. Unique properties of flat minima in deep networks. *Preprint*, arXiv:2002.04710.

Aleksandar Petrov, Philip H. S. Torr, and Adel Bibi. 2024. When do prompting and prefix-tuning work? a theory of capabilities and limitations. *Preprint*, arXiv:2310.19698.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. *Preprint*, arXiv:2005.00247.

Rushi Qiang, Ruiyi Zhang, and Pengtao Xie. 2024. Bilora: A bi-level optimization framework for overfitting-resilient low-rank adaptation of large pre-trained models. *Preprint*, arXiv:2403.13037.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *Preprint*, arXiv:1606.05250.

R. Tyrrell Rockafellar. 1970. *Convex analysis*. Princeton Mathematical Series. Princeton University Press, Princeton, N. J.

Alex Rodriguez, Maria d'Errico, Elena Facco, and Alessandro Laio. 2018. Computing the free energy without collective variables. *J Chem Theory Comput*, 14(3):1206–1215.

Sebastian Ruder. 2017. An overview of gradient descent optimization algorithms. *Preprint*, arXiv:1609.04747.

Ke Sun and Frank Nielsen. 2024. A geometric modeling of occam's razor in deep learning. *Preprint*, arXiv:1905.11027.

Lucrezia Valeriani, Diego Doimo, Francesca Cuturello, Alessandro Laio, Alessio Ansuini, and Alberto Cazzaniga. 2023. The geometry of hidden representations of large transformer models. *Preprint*, arXiv:2302.00294.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. *Preprint*, arXiv:1804.07461.

Sumio Watanabe. 2009. *Algebraic Geometry and Statistical Learning Theory*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Lei Wu, Mingze Wang, and Weijie Su. 2022. The alignment property of sgd noise and how it helps select flat minima: A stability analysis. *Preprint*, arXiv:2207.02628.

Zeke Xie, Issei Sato, and Masashi Sugiyama. 2021. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. *Preprint*, arXiv:2002.03495.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *Preprint*, arXiv:2106.10199.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023a. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

A Mathematical Formalism

In this section, we provide the mathematical definitions, theorems, and algorithms that serve as the foundation for the methods and theorems presented in this paper.

A.1 Intrinsic Dimensionality

A.1.1 Definition and Example

Definition A.1 (Intrinsic Dimensionality). Let $\mathcal{M} \subseteq \mathbb{R}^D$ be a manifold embedded in a D-dimensional ambient space. The *intrinsic dimensionality (idim)* of $\mathcal{M}$ is defined as the smallest number of coordinates d such that all data points on $\mathcal{M}$ can be locally approximated by a d-dimensional Euclidean space. Formally, for every point $\mathbf{x} \in \mathcal{M}$, there exists a neighborhood $\mathcal{N}(\mathbf{x})$ and a smooth map $\phi : \mathbb{R}^d \to \mathbb{R}^D$ such that $\phi(\mathbb{R}^d) \cap \mathcal{N}(\mathbf{x}) = \mathcal{M} \cap \mathcal{N}(\mathbf{x})$.

In practical terms, the intrinsic dimensionality d represents the number of degrees of freedom required to describe the structure of $\mathcal{M}$, regardless of the ambient space's dimensionality D.

Example. Consider a helical curve $\mathcal{H}$ embedded in three-dimensional space ($\mathbb{R}^3$) (Figure 3) with the parametric representation:

$$\mathbf{x}(t) = \begin{bmatrix} r\cos(t) \\ r\sin(t) \\ ct \end{bmatrix}, \quad t \in \mathbb{R},$$

where $r > 0$ is the radius and $c > 0$ is the vertical scaling factor.

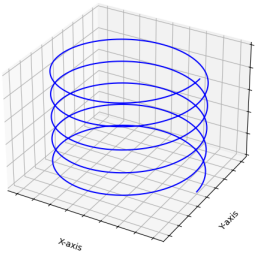

Figure 3: A helical curve in 3D space with an intrinsic dimension of 1, fully described by a single parameter despite its 3D embedding.

Although the helix is embedded in $\mathbb{R}^3$, the parameter t uniquely determines any point on the curve. Hence, the helix is a one-dimensional ($d = 1$) manifold, since it can be locally approximated by a one-dimensional Euclidean space ($\mathbb{R}^1$).

A.1.2 Methodology: Two Nearest Neighbors Estimator

Motivation. The *Two Nearest Neighbor (TwoNN)* method (Facco et al., 2017), is a technique designed to estimate the intrinsic dimension (idim) of a dataset using local geometric properties. It is particularly effective for NLP datasets due to several advantages. First, it is highly scalable, since it requires only the distances to the first two nearest neighbors of each data point, making it computationally efficient even for large datasets. Second, the method is robust, consistently producing reliable results in datasets of varying sizes and scales (Denti et al., 2022). Finally, the assumption of locally constant density is valid for NLP datasets. This has been empirically verified using the *Point Adaptive kNN (PAk)* method (Rodriguez et al., 2018; Valeriani et al., 2023).

Methodology. To estimate the intrinsic dimension, the TwoNN algorithm follows four key steps. The first step involves calculating the distances to the two nearest neighbors for each data point x_i: r_{i_1}, the distance to the first nearest neighbor, and r_{i_2}, the distance to the second nearest neighbor. Once these distances are obtained, the method computes the ratio $\mu_i = \frac{r_{i_2}}{r_{i_1}}$ for each data point.

Under the assumption of constant local density, the ratios μ_i follow a Pareto distribution, expressed as follows

$$p(\mu_i \mid d) = d\mu_i^{-d-1},$$

where d represents the intrinsic dimension. The cumulative distribution function (CDF) of the Pareto distribution is given by

$$F(\mu) = 1 - \mu^{-d}.$$

To approximate this CDF empirically, the computed ratios μ_i are sorted in ascending order, yielding $\mu_{\sigma(i)}$, where $\sigma(i)$ represents the index of the i-th smallest value. The empirical CDF is then defined as

$$F_{\text{emp}}(\mu_{\sigma(i)}) = \frac{i}{N},$$

where N is the total number of data points.

Finally, the intrinsic dimension d is estimated using a linear regression. A logarithmic transformation of the Pareto CDF is applied, yielding the relationship

$$-\log(1 - F_{\text{emp}}(\mu_{\sigma(i)})) \approx d \log(\mu_{\sigma(i)}).$$

By plotting $\log(\mu_{\sigma(i)})$ against $-\log(1 - F_{\text{emp}}(\mu_{\sigma(i)}))$, the slope of the resulting line provides an estimate of d.

In summary, the TwoNN method offers a simple yet effective approach to estimate the intrinsic dimension of datasets. To ensure robustness, the TwoNN estimator is applied to random subsets of the dataset of decreasing sizes (e.g., $N, N/2, N/4, \dots$), and the ID is chosen where the estimates stabilize.

Computational Complexity. The TwoNN estimator requires finding the two nearest neighbors for each data point in the dataset. This operation has a computational complexity of $O(n^2)$ for a naive approach or $O(n \log(n))$ when using optimized nearest neighbor search methods (e.g., KD-trees or ball trees). For a dataset with n points and a model with L transformer blocks (e.g., where distances need to be computed across L hidden representations), the overall complexity becomes:

$$O(L \cdot n \log(n)) \quad \text{or} \quad O(L \cdot n^2),$$

depending on the algorithm used for nearest neighbor computation.

Assumption Empirical Validation To empirically validate the local density assumption, we applied the *Point Adaptive kNN (PAk)* method, as implemented in DADApy (Glielmo et al., 2022). We determined that the probability density can be considered approximately constant within the first two nearest neighbors of each data point. This supports the validity of the local density assumption at the scale used for intrinsic dimensionality estimation, which is measured using the distances to the first two nearest neighbors. Table 7 summarizes the extent of constant density in various datasets, where this extent is defined as the minimum number of neighbors observed across all hidden representations of each dataset.

Table 7: Extent to which the local density is considered constant for each dataset.

Dataset	Component	Extent of Constant Density (Nearest Neighbors)
GLUE	CoLA	3
	SST-2	4
	MRPC	3
	STS-B	3
	QNLI	3
	RTE	2

A.2 Transformer Architecture

Definition A.2 (Single-head Self-attention Layer). Let $k, d \in \mathbb{N}$. Consider matrices

$Q, K, V \in \mathbb{R}^{k \times d}$. For any integer $n \in \mathbb{N}$ and vectors $x_1, \ldots, x_n \in \mathbb{R}^d$, self-attention with parameters (Q, K, V) maps the sequence $(x_1, \ldots, x_n) \in \mathbb{R}^{d \times n}$ to

$$f(x_1, \ldots, x_n) = \left(V \sum_{j=1}^{n} \text{softmax}\left(\frac{x_i^\top Q^\top K x_j}{\sqrt{k}} \right) x_j \right)_{1 \le i \le n} \in (\mathbb{R}^k)^n,$$

Definition A.3 (Multi-head Self Attention Layer). Let $d \in \mathbb{N}$ and H be a divisor of d. For $1 \le h \le H$, let $Q(h), K(h), V(h) \in \mathbb{R}^{k \times d}$ with $k := d/H$, and $W(h) \in \mathbb{R}^{d \times k}$. Multi-head self-attention with parameters $(Q(h), K(h), V(h), W(h))_{1 \le h \le H}$ maps any sequence $(x_1, \ldots, x_n) \in (\mathbb{R}^d)^n$ to

$$f_{\text{MH}}(x_1, \ldots, x_n) = \sum_{h=1}^{H} W(h) f^{(h)}(x_1, \ldots, x_n) \in (\mathbb{R}^d)^n,$$

where $f^{(h)}$ denotes single-head self-attention with parameters $(Q(h), K(h), V(h))$.

A.3 Differential Topology

Definition A.4 (Manifold). A *manifold* is a topological space M that is locally homeomorphic to Euclidean space $\mathbb{R}^n$, where n is the dimension of the manifold. More formally, for each point $p \in M$, there exists an open neighborhood $U_p \subseteq M$ and a homeomorphism $\varphi_p : U_p \to \mathbb{R}^n$ that maps the local neighborhood around p to an open subset of $\mathbb{R}^n$. The pair (U_p, φ_p) is called a *chart* or *coordinate chart* of the manifold at p. The function φ_p is called a *coordinate map* or *local coordinate system*, and its inverse φ_p^{-1} is called a *local parameterization*.

Theorem A.5 (Sard's Theorem). *Let $f : M \to N$ be a smooth map between smooth manifolds M and N, where $\dim(M) = m$ and $\dim(N) = n$. The set of critical values, $f(C_f) \subseteq N$, where $C_f = \{p \in M : \text{rank}(Df_p) < n\}$, has Lebesgue measure zero in N. In other words, the image of the critical set under a smooth map has measure zero in the target space N.*

Theorem A.6 (Rank Theorem). *Let M, N be smooth manifolds such that $\dim M = m, \dim N = n$, and let $f : M \to N$ be a smooth map with constant rank r. For each $p \in U$, there exists a chart (U, φ) centered at p, and a chart (V, ψ) centered at $f(p)$, with $f(U) \subset V$ such that*

$$\hat{f}(x^1, \ldots, x^r, x^{r+1}, \ldots, x^m) = \psi \circ f \circ \varphi^{-1}(x^1, \ldots, x^r, x^{r+1}, \ldots, x^m) = (x^1, \ldots, x^r, 0, \ldots, 0).$$

B GeLoRA: Framework and Theoretical Proofs

In this section, we provide the pseudocode for the GeLoRA framework along with detailed proofs of the theorems presented in this paper.

B.1 Mathematical Proofs

B.1.1 Proof of Theorem 3.2 – Intrinsic Dimension as a Lower Bound

Theorem B.1 (Intrinsic Dimension as a Lower Bound). *The intrinsic dimension $\hat{idim}(\phi)$ is a lower bound to the local dimensionality $d(\phi)$.*

$$d(\phi) \ge \hat{idim}(\phi).$$

Proof. The local dimensionality $d(\phi)$ of a neuromanifold is defined as the rank of the Fisher Information Matrix (FIM), which corresponds to the number of non-zero eigenvalues of the FIM. However, in practice, while the FIM is almost surely of full rank, many of its eigenvalues can be exceedingly small, on the order of $\epsilon \in \mathbb{R}^+$, where ϵ is a small positive threshold.

According to the Cramér-Rao bound, the variance of parameter estimates is inversely proportional to the eigenvalues of the FIM. Specifically, for an eigenvalue on the order of ϵ, the variance of the corresponding parameter is at least $1/\epsilon$. Parameters associated with such small eigenvalues contribute negligible information about the model and can therefore be considered effectively uninformative.

By disregarding parameters associated with small eigenvalues, we obtain the definition of the intrinsic dimension $\text{idim}(\phi)$, which represents the minimal number of parameters necessary to describe the structure of the manifold. The specific value of the intrinsic dimension depends on the threshold ϵ used to exclude eigenvalues below a certain magnitude. This threshold determines the uninformative directions that are discarded, yielding an estimate of the ground truth intrinsic dimension. Consequently, the estimated intrinsic dimension $\hat{\text{idim}}(\phi)$ is always less than or equal to the local dimensionality $d(\phi)$:

$$\hat{\text{idim}}(\phi) \le d(\phi).$$

Thus, the estimated intrinsic dimension $\hat{\text{idim}}(\phi)$ provides a lower bound for the local dimensionality $d(\phi)$, completing the proof. □

B.1.2 Proof of Theorem 3.3 – Rank Bound of Transformer Blocks

Theorem B.2 (Rank Bound of Transformer Blocks). *Let $\mathcal{M}$ denote a language model consisting of N transformer blocks. For each $i \in$*

$\{1, 2, \ldots, N\}$, *the i-th transformer block is represented by*

$$\mathcal{T}_i : \mathbb{R}^{n_{i-1}} \times \mathbb{R}^{p_{i-1}} \to \mathbb{R}^{n_i},$$

which maps the hidden state $\mathcal{H}_{i-1} \subset \mathbb{R}^{n_{i-1}}$ and parameters $\theta_{i-1} \in \mathbb{R}^{p_{i-1}}$ to the next hidden state $\mathcal{H}_i \subset \mathbb{R}^{n_i}$.

- *Assume that the $\mathcal{H}_{i-1}$ lies on a **data manifold** $\mathcal{N}_{i-1}$ with **intrinsic dimension** d_{i-1} embedded in $\mathbb{R}^{n_{i-1}}$, and $\mathcal{H}_i$ lies on $\mathcal{N}_i$ with **intrinsic dimension** d_i embedded in $\mathbb{R}^{n_i}$.*
- *Let the **parameter manifold** $\Theta_{i-1} \subset \mathbb{R}^{p_{i-1}}$ be a submanifold locally defined around θ_{i-1}, with **intrinsic dimension** $d_{\theta_{i-1}}$, representing the number of independent parameters in the neighborhood.*

The rank of the transformer block $\mathcal{T}_i$, defined as rank$(\mathcal{T}_i) = \max_{(x,\theta)\in\mathcal{H}_{i-1}\times\Theta_{i-1}}$ rank$(J(\mathcal{T}_i, x, \theta))$, where $J(\mathcal{T}_i, x, \theta)$ is the Jacobian matrix of $\mathcal{T}_i$ evaluated at (x, θ), satisfies the following bound:

$$d_i \leq \textit{rank}(\mathcal{T}_i) \leq d_{i-1} + d_{\theta_{i-1}}.$$

Proof. We aim to establish the rank bound for a single transformer block $\mathcal{T}_i$, which transforms the hidden state $\mathcal{H}_{i-1}$ into $\mathcal{H}_i$ using parameters θ_{i-1}. The transformation is modeled by a function:

$$\mathcal{T}_i : \mathbb{R}^{n_{i-1}} \times \mathbb{R}^{p_{i-1}} \to \mathbb{R}^{n_i}.$$

The rank of $\mathcal{T}_i$ is given by the rank of its Jacobian matrix:

$$J(\mathcal{T}_i, x, \theta) \in \mathbb{R}^{n_i \times (n_{i-1}+p_{i-1})}.$$

We now prove both the upper and lower bounds.

Lower Bound. Let $i \in \{1, 2, \ldots, N\}$, and consider the map $\mathcal{T}_i : \mathbb{R}^{n_{i-1}} \times \mathbb{R}^{p_{i-1}} \to \mathbb{R}^{n_i}$ to be the i-th transformer block, which maps the hidden state $\mathcal{H}_{i-1} \subset \mathbb{R}^{n_{i-1}}$ and parameters $\theta_{i-1} \in \mathbb{R}^{p_{i-1}}$ to the next hidden state $\mathcal{H}_i \subset \mathbb{R}^{n_i}$. Assume that $\text{idim}(\mathcal{H}_{i-1}) = d_{i-1}$ and $\text{idim}(\mathcal{H}_i) = d_i$.

Given that $\text{idim}(\mathcal{H}_{i-1}) = d_{i-1} \leq n_{i-1}$, we can define a smooth bijective parameterization $\phi : \mathcal{U} \to \mathbb{R}^{n_{i-1}}$ from an open set $\mathcal{U} \subset \mathbb{R}^{d_{i-1}}$ to an open subset $\mathcal{O} \subset \mathcal{H}_{i-1}$.

Similarly, since Θ_{i-1} is a submanifold of $\mathbb{R}^{p_{i-1}}$ with intrinsic dimension $d_{\theta_{i-1}}$, we can define a smooth bijective parameterization $\varphi : \mathcal{V} \to \mathbb{R}^{p_{i-1}}$ from an open set $\mathcal{V} \subset \mathbb{R}^{d_{\theta_{i-1}}}$ to an open subset $\mathcal{O}_\theta \subset \Theta_{i-1}$.

We now extend these parameterizations by considering the map $\psi : \mathcal{U} \times \mathcal{V} \to \mathbb{R}^{n_{i-1}} \times \mathbb{R}^{p_{i-1}}$ that maps each point $(x, \eta) \in \mathcal{U} \times \mathcal{V}$ to $(\phi(x), \varphi(\eta))$.

Since $\mathcal{T}_i$ is smooth almost everywhere, we can apply the constant rank [2] theorem for manifolds (Theorem A.6) to the composed map $\mathcal{T}_i \circ \psi$, obtaining:

$$\begin{aligned}\text{idim}(\mathcal{T}_i(\mathcal{H}_{i-1} \times \Theta_{i-1})) &= \text{rank}(\mathcal{T}_i \circ \psi) \\ &= \text{rank}(J_{\mathcal{T}_i \circ \psi}),\end{aligned}$$

where $J_{\mathcal{T}_i \circ \psi}$ is the Jacobian matrix of the composition $\mathcal{T}_i \circ \psi$.

Using the chain rule, the rank of the composition is bounded by the minimum rank of the individual Jacobians:

$$\text{rank}(J_{\mathcal{T}_i \circ \psi}) = \text{rank}(J_{\mathcal{T}_i} \cdot J_\psi) \leq \text{rank}(J_{\mathcal{T}_i}).$$

Thus, the dimension of $\mathcal{T}_i(\mathcal{H}_{i-1} \times \Theta_{i-1})$, which corresponds to the intrinsic dimension d_i of the hidden state $\mathcal{H}_i$, satisfies:

$$d_i = \text{idim}(\mathcal{H}_i) \leq \text{rank}(\mathcal{T}_i).$$

This completes the lower bound.

Upper Bound. By definition, $J(\mathcal{T}_i, x, \theta)$ maps perturbations in (x, θ) to changes in $\mathcal{H}_i$. The rank of this Jacobian is at most the sum of the intrinsic dimensions of the data manifold and the parameter manifold:

$$\text{rank}(J(\mathcal{T}_i, x, \theta)) \leq d_{i-1} + d_{\theta_{i-1}},$$

because the effective degrees of freedom in the input are constrained by d_{i-1} (intrinsic dimension of $\mathcal{N}_{i-1}$) and $d_{\theta_{i-1}}$ (intrinsic dimension of Θ_{i-1}). This establishes the upper bound:

$$\text{rank}(\mathcal{T}_i) \leq d_{i-1} + d_{\theta_{i-1}}.$$

Conclusion. Combining the upper and lower bounds, we obtain the desired inequality:

$$d_i \leq \text{rank}(\mathcal{T}_i) \leq d_{i-1} + d_{\theta_{i-1}}.$$

□

[2] By Sard's Theorem (Guillemin and Pollack, 2010) (Theorem A.5), critical points, where the Jacobian rank is lower, map to a set of measure zero. These regions of lower ranks contribute negligibly to the representation manifolds. Therefore, we can disregard them and focus only on regions where the rank is constant and maximal.

B.1.3 Proof of Corollary 3.4 – Lower Bound on Parameter Dimensionality

Corollary B.3 (Lower Bound on Parameter Dimensionality). *Let $d_{\theta_{i-1}}$ denote the effective dimensionality of the parameter manifold, and let d_i and d_{i-1} represent the intrinsic dimensions of the data manifold at layers i and $i-1$, respectively. The dimensionality of the parameter manifold satisfies:*

$$d_{\theta_{i-1}} \geq \max(d_i - d_{i-1}, 0).$$

Proof. From the established rank inequality:

$$d_i \leq \text{rank}(\mathcal{T}_i) \leq d_{i-1} + d_{\theta_{i-1}}.$$

we isolate:

$$d_i \leq d_{i-1} + d_{\theta_{i-1}} \Rightarrow d_i - d_{i-1} \leq d_{\theta_{i-1}}.$$

Since the dimensionality cannot be negative, we take the maximum with zero:

$$d_{\theta_{i-1}} \geq \max(d_i - d_{i-1}, 0).$$

This completes the proof. □

B.2 Proof of Theorem 3.5 – Transformer Rank Bound Dynamics

Theorem B.4 (Transformer Rank Bound Dynamics). *Let $i \in \{1, 2, \ldots, N\}$, and consider the process of fine-tuning a language model. The initial intrinsic dimensions d_i^0 of the data manifold are computed at the beginning of the training. We define $d_{\theta_{i-1}}^0 = \max(0, d_i^0 - d_{i-1}^0)$ as the initial rank of the parameters required for training, and we fix it during training.*

Let d_θ^t represent the evolving rank of the parameters of the transformer block $\mathcal{T}_i$ after the t-th gradient step. Then, for all t, the following inequality holds:

$$d_\theta^0 \geq d_\theta^t.$$

Proof. Let $\mathcal{T}_i$ denote the i-th transformer block. We define the initial effective rank of the parameter submanifold by

$$d_{\theta_{i-1}}^0 = \max(0, d_i^0 - d_{i-1}^0).$$

During fine-tuning, the parameters are updated via SGD:

$$\theta_{i-1}^{t+1} = \theta_{i-1}^t - \eta \nabla_{\theta_{i-1}} L(\theta_{i-1}^t),$$

where $L(\theta)$ is the loss function. Empirical and theoretical analyses have shown that SGD prefers flat minima (Xie et al., 2021; Wu et al., 2022; Mulayoff and Michaeli, 2020). In a flat minimum, the local Hessian has many small eigenvalues, indicating that perturbations along the corresponding directions have little effect on the loss. This effectively means that the learned parameters are constrained to a subspace with fewer independent directions than might be available in the full parameter space.

Thus, after each gradient update, the parameters θ_{i-1}^t evolve to lie in regions of the parameter space where the effective degrees of freedom, quantified by the effective rank $d_{\theta_{i-1}}^t$, are reduced. Hence, the effective rank cannot increase beyond the initial effective rank $d_{\theta_{i-1}}^0$. In other words, the dynamics of SGD ensure that

$$d_{\theta_{i-1}}^t \leq d_{\theta_{i-1}}^0 \quad \text{for all } t.$$

Therefore, we conclude that

$$d_\theta^0 \geq d_\theta^t \quad \text{for all } t,$$

which proves the theorem. □

B.3 GeLoRA: Pseudocode

B.3.1 Estimating Intrinsic Dimensions

Algorithm 1 EstimateIntrinsicDimension

Require: Hidden states matrix $\mathbf{X} = [x_1, \ldots, x_n]$
Ensure: Estimated intrinsic dimension d

```
for each point x_j in X do
    r_1(j) ← distance to nearest neighbor of x_j
    r_2(j) ← distance to second nearest neighbor of x_j
    μ_j ← r_2(j)/r_1(j)
end for
Fit empirical {μ_j} to F(μ|d) = 1 − μ^{−d} using regression or MLE
return d
```

B.3.2 Compute Intrinsic Dimensions of Data Representations

Algorithm 2 ComputeIntrinsicDimensions

Require: Model $\mathcal{M}$ with L layers, dataset $\mathcal{D}$
Ensure: Array of intrinsic dimensions $\mathbf{d}$ of size $L+1$

```
for i = 0 to L do
    X_i ← GetHiddenStates(M, D, layer = i)
    d_i ← EstimateIntrinsicDimension(X_i)
end for
return d
```

B.3.3 Setting LoRA Ranks and Scaling Factors

Algorithm 3 ComputeLoRAHyperparameters

Require: Intrinsic dimensions $\mathbf{d}$, constant c
Ensure: LoRA ranks $\mathbf{r}$ and scaling factors $\boldsymbol{\alpha}$

for $i = 0$ to $L - 1$ **do**
  dim_difference $\leftarrow \max(d_{i+1} - d_i, 0)$
  base_rank $\leftarrow$ dim_difference $+ 1$
  $r_{K_i}, r_{Q_i}, r_{V_i}, r_{O_i} \leftarrow$ base_rank
  $r_i \leftarrow$ base_rank ▷ Assume shared rank here
end for
for $i = 0$ to $L - 1$ **do**
  $\alpha_i \leftarrow c \cdot r_i$
end for
return $\mathbf{r}, \boldsymbol{\alpha}$

B.3.4 GeLoRA

Algorithm 4 GeLoRA (Geometry-aware Low-Rank Adaptation)

Require: Model $\mathcal{M}$ with L layers, dataset $\mathcal{D}$, constant c
Ensure: LoRA ranks and scaling factors applied to model

$\mathbf{d} \leftarrow$ COMPUTEINTRINSICDIMENSIONS($\mathcal{M}, \mathcal{D}$)
$\mathbf{r}, \boldsymbol{\alpha} \leftarrow$ COMPUTELORAHYPERPARAMETERS($\mathbf{d}, c$)
APPLYLORAPARAMETERS($\mathcal{M}, \mathbf{r}, \boldsymbol{\alpha}$)

C Supplementary Experiments

In this section, we present supplementary experiments that support our design decisions, offer further validation of our approach under higher budget constraints, and provide a more principled explanation of the intermediate task tuning technique.

C.1 Efficiency-Performance Diagram – GLUE Benchmark

To better understand how our method, GeLoRA, compares to other approaches on the GLUE benchmark and balances computational efficiency with expressivity, we present an efficiency-performance diagram. In this diagram, the x-axis represents performance metric, while the y-axis represents the number of parameters, measured in millions of parameters multiplied by $1e^{-3}$. We have omitted full fine-tuning from the plot for clarity, as the difference in scale is substantial. Additionally, since the parameter size of GeLoRA is dataset-dependent, we use the average parameter size to represent it in the diagram.

Figure 4 offers a comparative analysis of parameter-efficient fine-tuning methods evaluated on the GLUE benchmark, illustrating the trade-off between computational efficiency and model performance. GeLoRA emerges as the leading approach, achieving a performance of approximately 88.0% while maintaining high parameter efficiency, positioning it in the upper-left quadrant of the diagram. This placement indicates that GeLoRA effectively balances performance and parameter usage. In contrast, traditional LoRA variants (represented by blue triangles) demonstrate consistent but lower performance across different rank settings (r=1, r=2). Adapter-based methods (yellow markers) tend to cluster in the right portion of the graph, achieving competitive performance but requiring more parameters. AdaLoRA variants (purple markers) show mixed results depending on rank settings, while BitFit (green) has the smallest parameter footprint but the lowest performance. Overall, the diagram highlights how GeLoRA optimizes both computational efficiency and model performance, outperforming other methods in this balance.

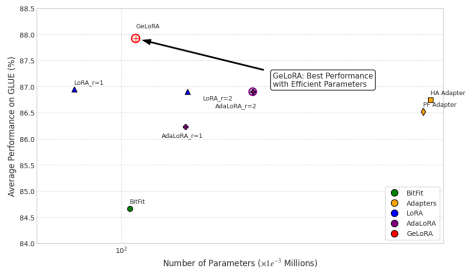

Figure 4: Comparison of parameter-efficient fine-tuning methods on GLUE benchmark. GeLoRA achieves the best performance (88.0%) with relatively few parameters compared to adapter-based alternatives, illustrating the efficiency-performance trade-off across different techniques.

C.2 Intrinsic Dimension Estimation with TwoNN in High Dimensions

In high-dimensional settings, the TwoNN method often underestimates the ground truth intrinsic dimension. To demonstrate this, we conducted experiments on benchmark datasets and compared the estimated values with the ground truth. Using the SCIKIT-DIMENSION package, we generated datasets representing different manifolds, each con-

taining 2500 data points. We set the embedding dimension to 4000 and the ground truth intrinsic dimension to 1000. The intrinsic dimension was then estimated using the TwoNN estimator with a decimation factor of 10, which removes 10% of the points identified as outliers. The results, summarized in Table 8, show that the estimated intrinsic dimension of around 200 is lower than the true value of 1000. To mitigate this issue, we introduced an offset when applying the GeLoRA methodology, ensuring that no layer is overlooked due to inaccurate intrinsic dimension estimates:

$$d_{\theta_{K_{i-1}}} = d_{\theta_{Q_{i-1}}} = d_{\theta_{V_{i-1}}} = d_{\theta_{O_{i-1}}} = \max(d_i - d_{i-1}, 0) + 1,$$

where $d_{\theta_{K_{i-1}}}, d_{\theta_{Q_{i-1}}}, d_{\theta_{V_{i-1}}}$, and $d_{\theta_{O_{i-1}}}$ are, respectively, the LoRA ranks of the Key, Query, Value and Output matrices of the transformer block i

C.3 Computational Complexity of GeLoRA Training

We evaluate the efficiency of different techniques under the same budget constraint. We measured the clock time for training across eight datasets, conducting experiments for 20 epochs on all datasets except for RTE, which was run for 50 epochs, and QQP and MNLI, which were run for 10 epochs each. To ensure a fair comparison between different techniques, we adjusted the ranks of LoRA and its variants to match the rounded mean rank of GeLoRA.

Table 9 shows GeLoRA delivers superior performance with lower computational overhead. In contrast, AdaLoRA and BitFit incur extra costs due to additional operations like importance score computation and task-specific head training.

C.4 Mean Ranks for Shifted GeLoRA

Table 10: GeLoRA parameters and mean ranks across GLUE tasks. Values in parentheses show rounded ranks.

TASK	# PARAMS ($\times 1e^3$)	MEAN RANK
CoLA	321.03M	4.33 (4)
STS-B	258.82M	3.50 (4)
MRPC	278.02M	3.75 (4)
QNLI	314.88M	4.25 (4)
SST-2	308.74M	4.17 (4)
RTE	278.02M	3.75 (4)

C.5 Intermediate Task Tuning: A Plausible Explanation

In our approach, the ranks used for fine-tuning are dataset-specific, as demonstrated in Table 4. This aligns with the widely adopted practice of intermediate task tuning, which is used to improve performance during fine-tuning. Intermediate task tuning involves fine-tuning a model on a different task from the target task as a preliminary warm-up step. Although this methodology is primarily intuitively motivated, rooted in the idea of learning common features and fostering common sense reasoning, its theoretical justification remains less clear. In this regard, our objective is to provide a plausible explanation for the effectiveness of this approach.

We focus on three tasks: MRPC (Dolan and Brockett, 2005), STS-B (Cer et al., 2017), and RTE (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007). Although each dataset has a specific focus, they all assess semantic relationships between pairs of texts, presenting a strong case for a sequential fine-tuning strategy. MRPC targets the identification of paraphrases, where two sentences convey the same idea using different wording. STS-B evaluates the degree of semantic similarity between sentences on a continuous scale ranging from 0 to 5. RTE determines whether one sentence entails another, reflecting a distinct aspect of semantic relationships. These tasks require the model to comprehend nuanced semantic properties, including synonyms, paraphrases, and entailment.

As a result, the underlying language representations in these datasets exhibit significant similarities. Consequently, we hypothesize that fine-tuning on MRPC can facilitate subsequent fine-tuning processes for STS-B and RTE.

We posit that the main reason for this improved performance is data compression, as the model learns features relevant to the target tasks during intermediate training. To evaluate this hypothesis, we theorize that the lower bound of the intrinsic dimensions will become looser after compression. Our experimental results support this hypothesis. For example, we observe a decrease in the mean intrinsic dimension for RTE (from 13.47 to 12.97), whereas the mean intrinsic dimension for STS-B remains consistent (from 13.19 to 13.01), albeit with a change in their profiles, as shown in Figure 5.

Moreover, we note similarities in the behavior

Table 8: Behavior of the TwoNN Estimator in High-Dimensional Data

Dataset	Intrinsic Dimension	Estimated Intrinsic Dimension	Description
M1_Sphere	1000	208.31	sphere linearly embedded
M4_Nonlinear	1000	260.36	Nonlinear manifold
M6_Nonlinear	1000	255.34	Nonlinear manifold
M8_Nonlinear	1000	269.03	Nonlinear (highly curved) manifold
M9_Affine	1000	197.46	Affine space
M10a_Cubic	1000	192.97	hypercube
M10b_Cubic	1000	190.91	hypercube
M10c_Cubic	1000	196.75	hypercube
M10d_Cubic	1000	195.61	hypercube
M12_Norm	1000	184.12	Isotropic multivariate Gaussian
Mbeta	1000	133.12	Manifold generated with a smooth nonuniform pdf
Mn1_Nonlinear	1000	193.82	Nonlinearly embedded manifold of high ID
Mn2_Nonlinear	1000	187.39	Nonlinearly embedded manifold of high ID

Table 9: Training computational cost (runtime) in seconds for DeBERTaV3-base fine-tuning on GLUE tasks. The runtime for each fine-tuning is indicated in seconds. The best results for each dataset are highlighted in **bold**.

Dataset	GeLoRA	LoRA	AdaLoRA	BitFit	HAdapter	PAdapter
CoLA	**85.68 ± 2.27**	100.95 ± 10.53	165.43 ± 0.28	157.27 ± 1.07	117.98 ± 0.07	113.52 ± 0.11
STS-B	**59.13 ± 3.26**	78.26 ± 6.92	157.50 ± 8.36	122.68 ± 0.40	84.51 ± 0.06	81.27 ± 0.04
MRPC	**40.42 ± 0.30**	58.75 ± 1.73	112.61 ± 1.36	94.93 ± 0.34	57.41 ± 0.10	55.09 ± 0.03
QNLI	**736.57 ± 3.34**	865.76 ± 4.11	2328.60 ± 24.81	1341.47 ± 21.03	1254.14 ± 1.21	1205.86 ± 1.83
SST-2	**475.58 ± 5.10**	482.38 ± 5.11	1140.65 ± 2.25	871.10 ± 5.05	807.91 ± 0.57	775.33 ± 0.56
RTE	**75.62 ± 0.29**	116.28 ± 7.30	207.89 ± 4.42	80.5 ± 0.24	104.38 ± 0.06	100.40 ± 0.11
Average	**245.5**	283.73	685.45	444.66	404.39	388.58

of different layers. The lower layers, responsible for basic features (such as syntax and grammar), remain largely unchanged. However, the higher layers, which capture more complex features, exhibit significant compression. The intermediate layers, as indicated by recent studies on the geometry of hidden representations (Valeriani et al., 2023), show a slight increase in their capacity due to the model's specialization in the semantics of the intermediate task.

Thus, the decrease in the mean intrinsic dimensions corresponds to a reduction in the lower bounds presented in Corollary 3.4. This loosening of the bounds indicates that the number of parameters required for optimal performance has decreased, leading to more efficient training.

D Datasets Statistics

D.1 GLUE Benchmark

We present the statistics for the GLUE (Wang et al., 2019) datasets used in our experiments in Table 11.

Table 11: Summary of the GLUE benchmark datasets.

Corpus	Task	#Train	#Dev	#Test	#Label	Metrics
CoLA	Acceptability	8.5k	1k	1k	2	Matthews Corr.
SST-2	Sentiment	67k	872	1.8k	2	Accuracy
RTE	NLI	2.5k	276	3k	2	Accuracy
MRPC	Paraphrase	3.7k	408	1.7k	2	Accuracy
QNLI	QA/NLI	108k	5.7k	5.7k	2	Accuracy
STS-B	Similarity	7k	1.5k	1.4k	–	Pearson/Spearman Corr.

D.2 SQUAD Datasets

We present the statistics for the SQUAD (Rajpurkar et al., 2016) datasets used in our experiments in Table 12.

Table 12: Statistics of the SQuAD dataset.

	# Train	# Validation
SQuAD v1.1	87,599	10,570
SQuAD v2.0	130,319	11,873

D.3 Airoboros Dataset

We present the statistics for the Airoboros (Durbin, 2024) dataset used in our experiments in Table 13.

Table 13: Statistics of the Airoboros dataset.

	# Train
Airobors	29,400

D.4 MT-BENCH Benchmark

We present the statistics for the MT-BENCH (Zheng et al., 2023b) dataset used in our experiments in Table 14.

Table 14: Statistics of the MT-BENCH dataset.

	# Samples
MT-BENCH	80

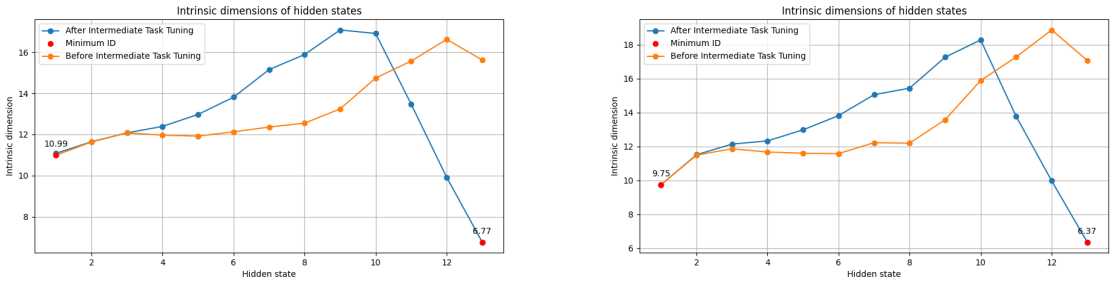

Figure 5: Intrinsic dimensions of hidden states before and after intermediate task tuning. Left: Results for RTE dataset showing a decrease in mean intrinsic dimension from 13.47 to 12.97 after intermediate training. Right: Results for STS-B dataset showing relatively consistent mean intrinsic dimensions (13.19 to 13.01) but with altered dimensional profiles. Both panels demonstrate how intermediate task tuning affects the model's representational space across different hidden state sizes.

E Examples of rank patterns

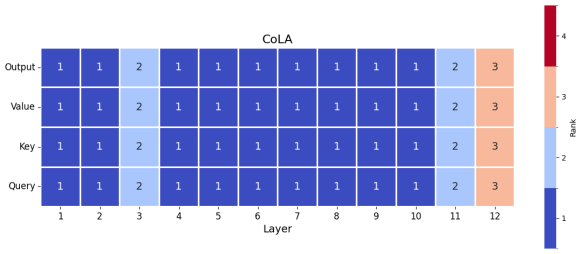

Figure 6: GeLoRA rank pattern for CoLA

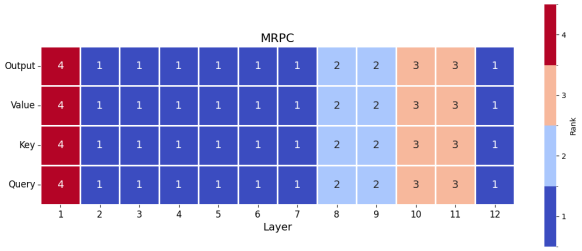

Figure 7: GeLoRA rank pattern for MRPC

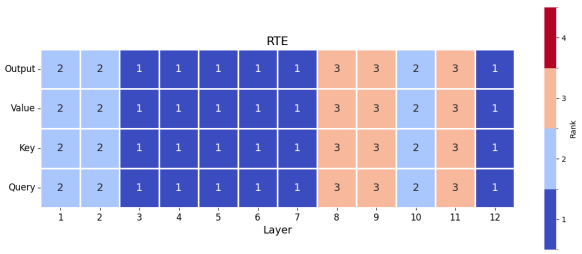

Figure 8: GeLoRA rank pattern for RTE

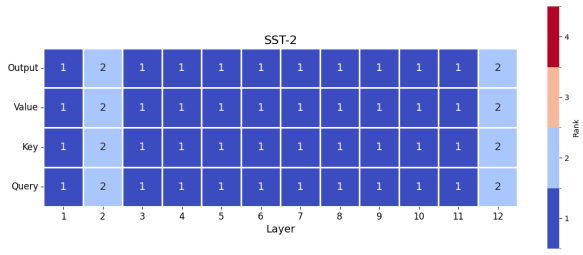

Figure 9: GeLoRA rank pattern for SST-2

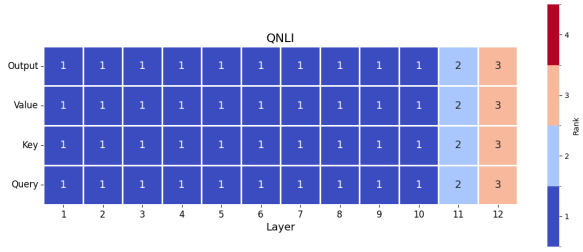

Figure 10: GeLoRA rank pattern for QNLI

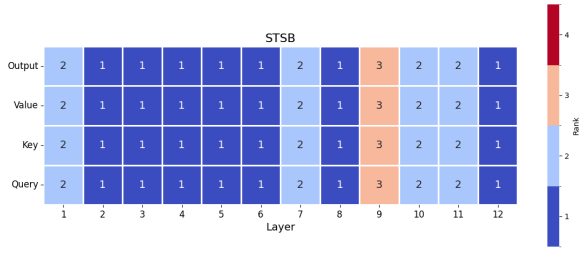

Figure 11: GeLoRA rank pattern for STSB

F Training Details

We employ OPTUNA to fine-tune the learning rates of the techniques employed. In the following, we summarize the optimal parameters identified in 10 trials, which were used in the fine-tuning process.

F.1 Instruction Following

Table 15: Learning Rates for Different Fine-tuning Methods on Instruction Tuning Dataset

Method	Airoboros
GeLoRA	1×10^{-6}
LoRA	1×10^{-6}
AdaLoRA	1×10^{-5}

F.2 Question Answering

Table 16: Learning Rates for Different Fine-tuning Methods on SQuAD Datasets

Method	SQuAD v1.1	SQuAD v2.0
GeLoRA	$8e^{-4}$	$8e^{-4}$
LoRA	$1e^{-4}$	$8e^{-4}$
Full Finetuning	$2e^{-5}$	$3e^{-5}$
BitFit	$8e^{-4}$	$8e^{-4}$
Houlsby Adapter	$1e^{-3}$	$1e^{-3}$
Pfeiffer Adapter	$1e^{-3}$	$1e^{-3}$
AdaLoRA	$1e^{-4}$	$8e^{-4}$

F.3 Natural Language Understanding

Table 17: Learning Rates for Different Fine-tuning Methods across GLUE Tasks

Method	CoLA	STS-B	MRPC	QNLI	SST-2	RTE	QQP	MNLI
GeLoRA	8×10^{-5}	2×10^{-4}	8×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}
LoRA	4×10^{-4}	1×10^{-4}	4×10^{-4}	2×10^{-4}	1×10^{-4}	3×10^{-4}	1×10^{-4}	4×10^{-4}
Full Finetuning	1×10^{-4}	1×10^{-4}	7×10^{-4}	1×10^{-4}	1×10^{-4}	9×10^{-5}	1×10^{-4}	1×10^{-4}
BitFit	8×10^{-4}	6×10^{-4}	9×10^{-4}	8×10^{-4}	3×10^{-4}	1×10^{-3}	1×10^{-4}	1×10^{-4}
Houlsby Adapter	3×10^{-4}	3×10^{-4}	3×10^{-3}	3×10^{-3}	1.2×10^{-3}	3×10^{-4}	3×10^{-4}	3×10^{-4}
Pfeiffer Adapter	3×10^{-4}	3×10^{-4}	3×10^{-3}	3×10^{-3}	1.2×10^{-3}	3×10^{-4}	3×10^{-4}	3×10^{-4}
AdaLoRA	5×10^{-4}	2×10^{-3}	1×10^{-3}	1×10^{-3}	8×10^{-4}	1.2×10^{-3}	5×10^{-4}	5×10^{-4}
SoRA	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}	8×10^{-4}	1.2×10^{-3}	8×10^{-4}	8×10^{-4}