LoRA Fine-Tuning Without GPUs: A CPU-Efficient Meta-Generation Framework for LLMs

Reza Arabpour [1 2] Haitz Sáez de Ocáriz Borde [3] Anastasis Kratsios [1 2]

Abstract

Low-Rank Adapters (LoRAs) have transformed the fine-tuning of Large Language Models (LLMs) by enabling parameter-efficient updates. However, their widespread adoption remains limited by the reliance on GPU-based training. In this work, we propose a theoretically grounded approach to LoRA fine-tuning designed specifically for users with limited computational resources, particularly those restricted to standard laptop CPUs. Our method learns a meta-operator that maps any input dataset, represented as a probability distribution, to a set of LoRA weights by leveraging a large bank of pre-trained adapters for the Mistral-7B-Instruct-v0.2 model. Instead of performing new gradient-based updates, our pipeline constructs adapters via lightweight combinations of existing LoRAs directly on CPU. While the resulting adapters do not match the performance of GPU-trained counterparts, they consistently outperform the base Mistral model on downstream tasks, offering a practical and accessible alternative to traditional GPU-based fine-tuning.

1. Introduction

As models and datasets scale up, full fine-tuning becomes increasingly unrealistic for most practitioners. The largest foundation models—often built by tech giants with almost unlimited compute (Touvron et al., 2023; OpenAI, 2023; Bai et al., 2023; Qwen et al., 2025; DeepSeek-AI et al., 2025)—can have hundreds of billions of parameters, making traditional fine-tuning for individuals prohibitively expensive. Parameter-efficient fine-tuning (PEFT) methods (He et al., 2022; Pfeiffer et al., 2020; Ding et al., 2022; Yu et al., 2022; Han et al., 2024) offer a workaround: instead of updating all weights, they tweak a small subset, slashing compute and storage costs while maintaining reasonable performance. Among these, the Low-Rank Adapter (LoRA) (Hu et al., 2021) approach has become standard due to combined simplicity and surprisingly powerful effectiveness. Nevertheless, for modern massive LLMs, LoRA fine-tuning can still be long and heavy. Thus, the following question becomes necessary:

Can one generate new low-rank adapters to fine-tune large language models on new tasks without the need for GPUs?

We address this concern by introducing a *zero-shot LoRA meta-generation procedure aimed at CPU-only users*. Our approach takes novel datasets, each potentially containing a variable number of instances, as input. It then outputs LoRA weights for a pre-trained LLM, where the prediction relies on a combination of instances from an existing bank of LoRAs (Gabrielsson et al., 2024). Importantly, the way in which these combinations are performed is lightweight enough to be computable on a standard contemporary CPU in a few minutes (see Table 2 in Appendix C), with no need for GPU clusters.

Main Contribution Our principled LoRA meta-generation pipeline provides lightweight, "cheap" LoRAs that approach the performance of GPU-fine-tuned models (which are often inaccessible to many) and outperform the base "non-finetuned" model. These contributions are theoretically grounded in Proposition 1 and Theorem 1. Together, these demonstrate that, with high probability, a ReLU Multi-Layer Perceptron (MLP) architecture, designed to run efficiently on a CPU, can identify the optimal coefficients for combining existing LoRAs. These optimal LoRA mixture coefficients, as defined in (3) (representing a weighted sum of pre-trained LoRA parameters), are determined based on the given dataset alignment features. This process effectively minimizes the downstream task loss, which quantifies the model's error on new, specific tasks. Additionally, our work also provides nearly-optimal closed-form solutions through lightweight, neural network-free alternatives (e.g., the Attentional or Normalized approaches). Interestingly, our experiments

[1]Department of Mathematics, McMaster University, Hamilton, Canada [2]Vector Institute, Toronto, Canada [3]University of Oxford, Oxford, United Kingdom. Correspondence to: Reza Arabpour <arabpour@mcmaster.ca>, Anastasis Kratsios <kratsioa@mcmaster.ca>, Haitz Sáez de Ocáriz Borde <chri6704@ox.ac.uk>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

reveal that the neural network-free variants of our pipeline perform comparably to the theoretically near-optimal neural network solution (the MLP-based approach).

Section 2 provides a discussion of related work concerning LoRA. We introduce the preliminaries for formalizing datasets as probability distributions in Section 3. Section 4 presents our LoRA generation pipelines. Their respective theoretical guarantees are later detailed in Section 5, and experimentally validated in Section 6.

2. Related Work

Since its introduction, the utility of LoRA (Hu et al., 2021) has expanded significantly beyond classical LLM post-training and language. It is now employed in diverse fields such as vision language models (Li et al., 2023) and vision Transformers (Dong et al., 2023). LoRA has also proven valuable in image generative modeling for rapid Stable Diffusion fine-tuning and personalization (Rombach et al., 2022; Gal et al., 2022; Ruiz et al., 2022; Roich et al., 2022), and for score distillation (Wang et al., 2023), although more principled LoRA-free methods have recently emerged (Lukoianov et al., 2024). Its application even extends to fine-tuning base models into reasoning models using reinforcement learning (Wang et al., 2025), and in the development of new adapters for Graph Neural Networks and Graph Transformers (Papageorgiou et al., 2025).

Alongside this expanding applicability, numerous LoRA variants have emerged, often aiming to further reduce computational overhead. For instance, quantization offers a way to lower memory consumption both during training (Gholami et al., 2021; Dettmers et al., 2023; Guo et al., 2024) and after (Yadav et al., 2023). The number of trainable parameters can also be reduced through adaptive rank allocation (Zhang et al., 2023). Further inspired by ideas around weight or projection reuse (Frankle and Carbin, 2018; Ramanujan et al., 2020), strategies to decrease trainable LoRA parameters include learning diagonal rescaling of frozen random B and A matrices (VeRA) (Kopiczko et al., 2024), deriving B and A from the SVD of the pre-trained W_0 and optimizing a smaller matrix in the resulting space (SVDiff) (Han et al., 2023), learning a linear combination of fixed random matrices (NOLA) (Koohpayegani et al., 2023), and fine-tuning with orthogonal matrices (BOFT) (Liu et al., 2024). LoRAs have also been explored from a more theoretical viewpoint (Zeng and Lee, 2024; Zhu et al., 2024; Kratsios et al., 2025).

Note that our focus here is on LoRA generation on CPU, which none of the aforementioned works explore. We would like to reiterate that all our pipelines, *including those using artificial neural networks* can be trained solely using CPUs.

3. Preliminaries

Datasets as Probability Distributions To describe our pipeline, we first need a unified framework for datasets with a varying number of instances. As such, we fix dimensions $d, D \in \mathbb{N}_+$. Given our training datasets $D_1, \dots, D_N \subset \mathcal{X}$ for some (non-empty) compact input domain $\mathcal{X} \subseteq \mathbb{R}^{d+D}$ corresponding to one of N possible down-stream tasks $\mathcal{T}_1, \dots, \mathcal{T}_N$ which our Transformer model (Mistral-7B-Instruct-v0.2) $f_\theta : \mathbb{R}^d \to \mathbb{R}^D$, whose parameters $\theta \in \mathbb{R}^p$ lie in a $p \gg 0$ dimensional Euclidean parameter space. Since the entries of each dataset are permutation-invariant, then, following the synthetic data generation literature, e.g. (Zamanlooy et al., 2024), it is natural to represent each dataset D_n as an empirical distribution (probability measure) via

$$P_{D_n} = \frac{1}{N_m} \sum_{(x,y) \in D_m} \delta_{(x,y)} \tag{1}$$

on the domain $\mathcal{X}$ where $N_m \overset{\text{def.}}{=} \# D_n$; i.e. $P_{D_n} = \sum_{m=1}^{N_m} w_m \delta_{(x_m, y_m)}$ with $w_m = 1/N_m$ for each $m = 1, \dots, N_m$. The support of the measure P_{D_n}, namely, $\{(x_1, y_1), \dots, (x_m, y_m)\}$ represent instances in D_n and the weights $w_m \in [0,1]$ sum to 1, i.e. w belongs to the N_m simplex $\Delta_{N_m} \overset{\text{def.}}{=} \{u \in [0,1]^{N_m} : \sum_{i=1}^{N_m} u_i = 1\}$, and represent the relative frequency of instance of data-point in D_m. We denote the set of probability measures on $\mathcal{X}$ by $\mathcal{P}(\mathcal{X})$.

Pipeline Inputs and Distributional Alignment Scores We then choose a *data-similarity score* where $\rho : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to [0, \infty]$. For this, we choose a (dis)similarity metric between probability distributions (measures) on $\mathcal{X}$, e.g. an information-theoretic divergence such as Kullback Leibler (KL) divergence or a metric such as the 1-Wasserstein distance $\mathcal{W}_1$. This dissimilarity score then allows us to extract *alignment scores* between any new dataset D (encoded as a probability measure P_D on the data-domain $\mathcal{X}$) and every dataset $(D_n)_{n=1}^N$ in our database via $\text{align} : \mathcal{P}(\mathcal{X}) \to \Delta_N$

$$\text{align}(P_D) \overset{\text{def.}}{=} \text{Softmax}\left((\rho(P_D, P_{D_n})_{n=1}^N \right). \tag{2}$$

Once the (softmax-normalized) alignment scores are computed, they are passed to a network, here in our "proof of concept" we use a simple MLP (trained on CPU), which yields a set of *mixture weights* $W_D \in \Delta_N$. These mixture weights are then used to combine the pre-trained LoRA weights $\theta_1, \dots, \theta_N$, from out database. Note that each LoRA weight θ_n was specialized for task $\mathcal{T}_n$ and pre-trained on dataset D_n. The output of our model is thus simply the mixture of LoRAs

$$D \mapsto P_D \mapsto \sum_{n=1}^N W_D \, \theta_n \tag{3}$$

and lies in the *convex hull* of the pre-trained LoRA weights $\theta_1, \dots, \theta_N$ in the parameter space $\mathbb{R}^p$. Therefore, we only need to learn (or compute, as we will see in Section 4) the mapping in (3). Based on this we are able to obtain LoRA weights with no fine-tuning, directly *out-of-the-box*.

4. LoRA Generation Pipelines for CPU

We now mathematically formalize our end-to-end cheap LoRA pipelines. Further details on how these were practically implemented can be found in Appendix C.1. Our main theoretical guarantee (Theorem 1) is general enough to apply not only to LoRAs fed into transformers but also to nearly any mixture-of-expert-based parameter prediction pipeline.

4.1. Setup

Let $d, D \in \mathbb{N}_+$. Let $\ell : \mathbb{R}^D \times \mathbb{R}^D \to [0, \infty)$ be Lipschitz. Let $f : \mathbb{R}^p \times \mathbb{R}^d \to \mathbb{R}^D$ be a locally-Lipschitz model, mapping the parameters $\theta \in \mathbb{R}^p$ and an input $x \in \mathbb{R}^d$ to an output $f_\theta(x) \in \mathbb{R}^D$. Also, We are given a pre-trained model $\theta_0 \in \mathbb{R}^p$. Purely for simplicity, we consider the standardized data-domain $\mathcal{X} = [0, 1]^{d+D}$. Following (Rothfuss et al., 2023). We henceforth fix a *task distribution* $\mathbb{P} \in \mathcal{P}(\mathcal{S})$ quantifying the probability of selecting any one dataset in $\mathcal{S}$ at random. We consider a metric space of datasets $\mathcal{D} \subseteq \mathcal{P}([0, 1]^{d+D})$ metrized by ρ, where the topology generated by ρ is no coarser than the topology of convergence in distribution. We fix a $K \in \mathbb{N}_+$ datasets paired with "fine-tuned" model parameters $(D_1, \Delta\theta_1), \dots (D_K, \Delta\theta_K)$ in $\mathcal{D} \times \mathbb{R}^p$. Let

$$\operatorname{co}(\Delta\theta) \overset{\text{def.}}{=} \{\vartheta \in \mathbb{R}^p : (\exists w \in \Delta_K)\, \vartheta = \sum_{k=1}^{K} w_k \Delta\theta_K\}$$

where $\Delta_K \overset{\text{def.}}{=} \{w \in [0, 1]^K : \sum_{k=1}^{K} w_k = 1\}$.

4.2. Very-Cheap LoRAs: Attentional Approach

Consider the following approach which maps any new incoming dataset D to the following mixture of LoRAs

$$\mathcal{C}_{\text{Att}}(D) \overset{\text{def.}}{=} \underbrace{[\operatorname{softmin} \circ \operatorname{align}(D)]^\top}_{\text{LoRA Alignment Scores}} \underbrace{(\Delta\theta_1, \dots, \Delta\theta_K)}_{\text{Pre-Trained LoRAs}} \quad (4)$$

We refer to the pipeline in (4) as our *attentional approach* since the dataset $D_1, \dots, D_K$ play a similar role to the *keys* in attention mechanisms (Vaswani et al., 2017). The LoRA alignment scores in (4) are analogous to contextual *alignment scores*, and the pre-trained LoRA parameters play a similar role to the *value matrices* in (Vaswani et al., 2017). The softmin is used instead of a softmax since maximal distance alignment happens when two datasets have a distance of 0 from one another, not some arbitrarily large number. We examine a *normalized* version of distance vector (4) in our experiments, see Appendix C.1.4 for details.

4.3. Cheap Nearly-Optimal LoRAs: Neural Approach

Our neural approach injects non-linear flexibility into how the distances are mapped to the LoRA alignment scores in (4) using a deep learning model $\mathcal{C} : \mathcal{D} \to \operatorname{co}(\Delta\theta)$; in this paper, this will always be an MLP. This allows our cheap LoRA approach to learn how to detect and align complicated non-linear alignments between the new dataset and those defining each pre-trained task. This *neural* approach thus sends any dataset D to the following mixture of LoRAs

$$\mathcal{C}(D) \overset{\text{def.}}{=} \underbrace{[\operatorname{softmin} \circ \hat{f} \circ \operatorname{align}(D)]^\top}_{\text{Neural-LoRA Alignment Scores}} \underbrace{(\Delta\theta_1, \dots, \Delta\theta_K)}_{\text{Pre-Trained LoRAs}} \quad (5)$$

where $\hat{f} : \mathbb{R}^K \to \mathbb{R}^K$ is an MLP with activation function ς, and we write $\operatorname{align}(D)$ in place of $\operatorname{align}(P_D)$ understanding the correspondence $D \to P_D$ as implicit.

5. Theoretical Guarantees

We now provide guarantees on the optimality of both our main approaches. We also demonstrate the existence of an oracle optimizer, yielding the best possible LoRA if the user had access to complete information on the task distribution.

5.1. Attentional Approach

Our cheapest out-of-the-box LoRA pipeline (4) are optimal in a PAC-Bayesian sense of (Alquier et al., 2016).

Proposition 1 (Existence: Optimal Oracles for Fine-Tuning). *For every $K \in \mathbb{N}_+$ and $\{(D_k, \Delta\theta_k)\}_{k=1}^K \subset \mathcal{D} \times \mathbb{R}^p$ with each D_k finite and non-empty. For every $\alpha > 0$ and each dataset $D \in \mathcal{D}$, the LoRA Alignment Scores in (4) satisfy*

$$\underbrace{\operatorname{softmin} \circ \operatorname{align}(D)}_{\textit{LoRA Alignment Scores}} \in \operatorname{argmin}_{w \in \Delta_k} \underbrace{\frac{1}{K} \sum_{k=1}^{K} w_k\, \rho(D, D_k)}_{\textit{Dataset Alignment}} + \underbrace{\frac{1}{\alpha} \sum_{k=1}^{K} w_k \log(w_k)}_{\textit{Entropic Penalty}}$$

Proof. See Appendix B.1. □

5.2. Neural Approach

The attentional pipeline, in (4), only checks for the *alignment* of a dataset with the datasets previously used for training the adapters in the bank. In contrast our neural approach, in (5), optimizes for the *downstream performance* of the predicted mixture of LoRA experts. Surprisingly, at least theoretically, one only needs a small MLP between the distance vector and softmin normalization layers to perform this out-of-the-box downstream (near) optimal LoRA generation. Our first guarantee for the neural approach demonstrates the existence of a map, i.e., an oracle predictor, which returns the best possible downstream optimization.

Proposition 2 (Existence: Optimal Oracles for Fine-Tuning). *For every dataset $D \in \mathcal{D}$ there exists an oracle parameter $\vartheta^\star \in \operatorname{co}(\Delta\theta)$ satisfying*

$$\underbrace{\mathbb{E}_{(X,Y)\sim\mathcal{D}}\Big[\ell(f_{\theta+\vartheta^*}(X),Y)\Big]}_{\textit{Oracle Error}} = \underbrace{\inf_{\Delta\theta\in\operatorname{co}(\Delta\theta)} \mathbb{E}_{(X,Y)\sim\mathcal{D}}\Big[\ell(f_{\theta+\Delta\theta}(X),Y)\Big]}_{\textit{Optimal Error}}.$$

Proof. See Appendix B.2. □

Our next and main results show that our pipeline can implement the optimal downstream mixture of LoRA predictors to achieve precision. Our result only relies on one structural regularity condition on our data, guaranteeing that: the *inverse problem* of recording a dataset/measure from its distance measurements to the available datasets/measures is possible. Effectively, this means that the metric dimension, in the graph-theoretic sense (see (Tillquist et al., 2023) for details), of the space $\mathcal{D}$ is exactly K.

Assumption 1 (Well-Posed Inverse Problem). *Let $(\mathcal{D},\rho)$ be compact and suppose that ρ metrizes the weak topology (convergence in distribution) on $\mathcal{D}$. We require that: the map* align : $\mathcal{D}\to[0,\infty)^K$ *injectively maps any $D\in\mathcal{D}$ to*

$$\operatorname{align}(D) \overset{\text{def.}}{=} \big(\rho(D,D_k)\big)_{k=1}^K.$$

Theorem 1 (ε-Optimal Cheap Fine-tuning). *Let $\varsigma:\mathbb{R}\to\mathbb{R}$ be a Lipschitz activation function which is differentiable with non-zero derivative at least on point. For every $0<\varepsilon\leq 1$, there is a MLP $\mathcal{C}:\mathbb{R}^K\to\mathbb{R}^K$ with activation function ς such that the ϵ-optimal selection property:*

$$\underbrace{\mathbb{E}_{(X,Y)\sim\mathcal{D}}\Big[\ell(f_{\theta+\mathcal{C}(D)}(X),Y)\Big]}_{\textit{Cheap Fine-Tuning}} \leq \underbrace{\inf_{\Delta\theta\in\operatorname{co}(\Delta\theta)} \mathbb{E}_{(X,Y)\sim\mathcal{D}}\Big[\ell(f_{\theta+\Delta\theta}(X),Y)\Big]}_{\textit{Fine-Turning Oracle}} + \varepsilon$$

holds with $\mathbb{P}$-probability at-least $1-\varepsilon$.

Proof. See Appendix B.2. □

6. Experimental Results

A comprehensive evaluation was conducted to assess the performance of three distinct approaches (Attentional, Normalized, and Neural) in conjunction with four established distance metrics (or divergences): Wasserstein Distance (WD), Kullback–Leibler (KL) divergence, Jensen-Shannon (JS) divergence, and Maximum Mean Discrepancy (MMD). This evaluation aimed to systematically compare the outputs generated by each combination of approach and metric. The primary evaluation criterion for the quality of the generated adapters was Rouge-L, a metric ranging from 0 to 1 that quantifies similarity based on the overlap of the longest common subsequences between generated and reference outputs (Lin, 2004). We also include Exact Match (EM) results in Appendix D.1.

Our experimental setup utilized the Mistral-7B-Instruct-v0.2 model (Jiang et al., 2023) and a dataset comprising 502 English dataset-adapter pairs sourced from the Lots-of-LoRAs HuggingFace repository (Gabrielsson et al., 2024). Further technical details regarding the implementation are provided in Appendix C.

Our experimental setup highlighted a key distinction in resource usage: the actual computation and adaptation of the LoRA adapters were performed exclusively on the CPU. GPUs, however, were essential only for the evaluation phase. This is because each adapted LLM, after being modified by our pipeline, needed to be loaded onto a GPU to generate outputs on its respective test set. To thoroughly assess the performance of each approach-distance (or divergence) metric pairing, we executed the entire pipeline twelve times for each of the 502 datasets. This exhaustive process covered every unique combination of approaches and distance metrics. Following the generation of outputs, the Rouge-L score was calculated for the test set of each dataset, and the reported values reflect the average of these scores across all runs.

6.1. Performance Comparison and Analysis

Our work is benchmarked against two key performance indicators. First, the performance of the base foundation model without any fine-tuning, representing a scenario where an end-user with limited computational resources applies a foundation model to a new dataset: this yielded an average and standard deviation Rouge-L score of 0.192 ± 0.181. Second, we compare against the performance of a GPU-fine-tuned model, achieved without hardware limitations, which obtained an Rouge-L score of 0.746 ± 0.265. Table 1 presents the average and standard deviation of Rouge-L performance for all approaches across the four distance (or divergence) metrics on the downstream task.

The JS divergence-based Normalized approach achieved the highest score, with an average Rouge-L of 0.520. This represents an improvement of 0.328 over the base model's score of 0.192. It is worth mentioning that even our Attentional approach, despite its simplicity, significantly outperforms the base foundation model across all distance metrics. Interestingly, the neural approach does not seem to justify the additional computational cost, as its performance improvement over the Attentional and Normalized approaches is generally minimal or even worse.

Table 1. Performance of our cheap LoRA pipelines.

Approach	WD	KL	JS	MMD
Attentional (std. dev.)	0.426 (±0.290)	0.501 (±0.272)	0.486 (±0.270)	0.486 (±0.270)
Normalized (std. dev.)	0.495 (±0.267)	0.488 (±0.269)	**0.520** (±0.277)	0.497 (±0.269)
Neural (std. dev.)	0.494 (±0.265)	0.482 (±0.268)	0.484 (±0.272)	0.493 (±0.270)

7. Conclusion

In conclusion, our work presents a practical, simple, and theoretically supported pipeline for generating LoRAs suitable for fine-tuning LLMs using only a CPU. This pipeline significantly reduces the typically required computational demands, making fine-tuning accessible even to users with limited hardware resources or on edge devices with privacy constraints.

We proved the existence of a lightweight ReLU MLP backbone, runnable on a CPU, that can reliably approximate optimal LoRA adapter weights and biases, thereby effectively minimizing downstream task loss in Theorem 1. Surprisingly, the simplest versions of our pipeline (Attentional and Normalized) achieved performance matching that of the MLP backbone version, further demonstrating the efficiency and power of our approach.

Our experiments, using the Mistral-7B-Instruct-v0.2 model on 502 diverse datasets, demonstrate substantial improvements over the baseline model, with the best configuration achieving a 0.328 increase in performance (Rouge-L score) over the base model, bridging more than half of the performance gap between the base model and the GPU fine-tuned reference. While our CPU-generated adapters do not yet match the performance of GPU-trained adapters, they provide a compelling alternative in resource-limited settings.

Future work could explore the applicability of these approaches to other language models as more LoRA adapter banks become open-source, as well as to tasks beyond NLP. Likewise, it would be of interest to better understand how many LoRA adapters would be required to generate new, high-quality adapters—that is, what size of bank is necessary. We expect this to depend on the task, data modalities, and possibly even the model architecture. Finally, our method could also potentially be used for LoRA initialization (pre-heating) before fine-tuning on GPU.

References

Hugo Touvron et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

OpenAI. Gpt-4 technical report, 2023. Available at `https://openai.com/research/gpt-4`.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023. URL `https://arxiv.org/abs/2309.16609`.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying

He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0RDcd5Axok.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. AdapterHub: A framework for adapting transformers. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.7. URL https://aclanthology.org/2020.emnlp-demos.7.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models, 2022. URL https://arxiv.org/abs/2203.06904.

Bruce X.B. Yu, Jianlong Chang, Lingbo Liu, Qi Tian, and Chang Wen Chen. Towards a unified view on visual parameter-efficient transfer learning. *arXiv preprint arXiv:2210.00788*, 2022.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey, 2024. URL https://arxiv.org/abs/2403.14608.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. 2021. URL https://arxiv.org/abs/2106.09685.

Rickard Brüel Gabrielsson, Jiacheng Zhu, Onkar Bhardwaj, Leshem Choshen, Kristjan Greenewald, Mikhail Yurochkin, and Justin Solomon. Compress then serve: Serving thousands of loRA adapters with little overhead, 2024. URL https://openreview.net/forum?id=hHNVn4hFPk.

Xin Li, Dongze Lian, Zhihe Lu, Jiawang Bai, Zhibo Chen, and Xinchao Wang. Graphadapter: Tuning vision-language models with dual knowledge graph. In *Advances in Neural Information Processing Systems*, 2023.

Wei Dong, Dawei Yan, Zhijun Lin, and Peng Wang. Efficient adaptation of large vision transformer via adapter re-composing. In *Advances in Neural Information Processing Systems*, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. *arXiv preprint arXiv:2208.12242*, 2022.

Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Transactions on Graphics (TOG)*, 42(1): 1–13, 2022.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ppJuFSOAnM.

Artem Lukoianov, Haitz Sáez de Ocáriz Borde, Kristjan Greenewald, Vitor Campagnolo Guizilini, Timur Bagautdinov, Vincent Sitzmann, and Justin Solomon. Score distillation via reparametrized DDIM. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=4DcpFagQ9e.

Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Burak Bilgin, Ollie Liu, and Willie Neiswanger. Tina: Tiny reasoning models via lora, 2025. URL https://arxiv.org/abs/2504.15777.

Pantelis Papageorgiou, Haitz Sáez de Ocáriz Borde, Anastasis Kratsios, and Michael M. Bronstein. Graph low-rank adapters of high regularity for graph neural networks and graph transformers. In *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, 2025. URL https://openreview.net/forum?id=gxhZj6uvFC.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference, 2021.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *ArXiv*, abs/2305.14314, 2023. URL https://api.semanticscholar.org/CorpusID:258841328.

Han Guo, Philip Greengard, Eric P. Xing, and Yoon Kim. Lq-lora: Low-rank plus quantized matrix decomposition for efficient language model finetuning, 2024.

Prateek Yadav, Leshem Choshen, Colin Raffel, and Mohit Bansal. Compeft: Compression for communicating parameter efficient updates via sparsification and quantization. *arXiv preprint arXiv:2311.13171*, 2023.

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

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

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

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

Ligong Han, Yinxiao Li, Han Zhang, Peyman Milanfar, Dimitris Metaxas, and Feng Yang. Svdiff: Compact parameter space for diffusion fine-tuning. *arXiv preprint arXiv:2303.11305*, 2023.

Soroush Abbasi Koohpayegani, KL Navaneet, Parsa Nooralinejad, Soheil Kolouri, and Hamed Pirsiavash. Nola: Networks as linear combination of low rank random basis, 2023.

Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, and Bernhard Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization. In *ICLR*, 2024.

Yuchen Zeng and Kangwook Lee. The expressive power of low-rank adaptation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=likXVjmh3E.

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

Anastasis Kratsios, Tin Sum Cheng, Aurelien Lucchi, and Haitz Sáez de Ocáriz Borde. Sharp generalization bounds for foundation models with asymmetric randomized low-rank adapters, 2025. URL https://arxiv.org/abs/2506.14530.

Behnoosh Zamanlooy, Mario Diaz, and Shahab Asoodeh. Locally private sampling with public data. *arXiv preprint arXiv:2411.08791*, 2024.

Jonas Rothfuss, Martin Josifoski, Vincent Fortuin, and Andreas Krause. Scalable pac-bayesian meta-learning via the pac-optimal hyper-posterior: from theory to practice. *The Journal of Machine Learning Research*, 24(1):18474–18535, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Pierre Alquier, James Ridgway, and Nicolas Chopin. On the properties of variational approximations of gibbs posteriors. *Journal of Machine Learning Research*, 17(236):1–41, 2016.

Richard C Tillquist, Rafael M Frongillo, and Manuel E Lladser. Getting the lay of the land in discrete space: A survey of metric dimension and its applications. *SIAM Review*, 65(4):919–962, 2023.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-1013.

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. Mistral 7b. 2023. URL https://arxiv.org/abs/2310.06825.

Renjie Wang, Cody Hyndman, and Anastasis Kratsios. The entropic measure transform. *Canad. J. Statist.*, 48 (1):97–129, 2020. ISSN 0319-5724,1708-945X. doi: 10.1002/cjs.11537. URL https://doi.org/10.1002/cjs.11537.

Charalambos D. Aliprantis and Kim C. Border. *Infinite dimensional analysis*. Springer, Berlin, third edition, 2006. ISBN 978-3-540-32696-0; 3-540-32696-0. A hitchhiker's guide.

James R. Munkres. *Topology*. Prentice Hall, Inc., Upper Saddle River, NJ, second edition, 2000. ISBN 0-13-181629-2.

Olav Kallenberg. *Foundations of modern probability*, volume 99 of *Probability Theory and Stochastic Modelling*. Springer, Cham, third edition, 2021. ISBN 978-3-030-61871-1; 978-3-030-61870-4. doi: 10.1007/978-3-030-61871-1. URL https://doi.org/10.1007/978-3-030-61871-1.

Alexander S. Kechris. *Classical descriptive set theory*, volume 156 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995. ISBN 0-387-94374-9. doi: 10.1007/978-1-4612-4190-4. URL https://doi.org/10.1007/978-1-4612-4190-4.

Achim Klenke. *Probability theory—a comprehensive course*. Universitext. Springer, Cham, third edition, 2020. ISBN 978-3-030-56402-5; 978-3-030-56401-8. doi: 10.1007/978-3-030-56402-5. URL https://doi.org/10.1007/978-3-030-56402-5.

J. Dugundji. An extension of Tietze's theorem. *Pacific J. Math.*, 1:353–367, 1951. ISSN 0030-8730,1945-5844. URL http://projecteuclid.org/euclid.pjm/1103052106.

Anastasis Kratsios and Léonie Papon. Universal approximation theorems for differentiable geometric deep learning. *J. Mach. Learn. Res.*, 23:Paper No. [196], 73, 2022. ISSN 1532-4435,1533-7928.

Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks. In *Conference on learning theory*, pages 2306–2327. PMLR, 2020.

Rickard Brüel Gabrielsson. Lots of loras. https://huggingface.co/Lots-of-LoRAs, 2025.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, and Daniel Khashabi. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. 2022. URL https://arxiv.org/abs/2204.07705.

Funding

A. Kratsios and R. Arabpour acknowledge financial support from an NSERC Discovery Grant No. RGPIN-2023-04482 and No. DGECR-2023-00230. They also acknowledge that resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute[1].

A. Additional Background

This appendix presents any additional background required for the formulation of our main results, proofs of our guarantees, and additional experimental details.

A.1. Foundation Model Fine-tuning and Attention Layers

In modern LLMs, fine-tuning all parameters can be computationally expensive and memory-intensive. LoRA (Hu et al., 2021) provides an efficient alternative by introducing low-rank updates to pre-trained weight matrices, particularly focusing on attention layers in transformer-based models. Given query Q, key K, and value V matrices, the standard attention mechanism computes the attention scores

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \tag{6}$$

where d_k is the dimension of the keys and queries.

A.2. Low-Rank Adapter (LoRA) Fine-tuning

In large transformers, these weight matrices dominate the parameter count, making them an ideal target for LoRA's efficient fine-tuning. By applying low-rank updates to these matrices, LoRA achieves significant savings in memory and computation without retraining the entire model. Consider a pre-trained weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, typically representing the projection matrices in the attention mechanism. Instead of updating the entire matrix, LoRA modifies the weights by adding a low-rank perturbation:

$$W = W_0 + \Delta W, \tag{7}$$

where ΔW is constrained to have $\text{rank}(\Delta W) = r \leq \min(d_{\text{out}}, d_{\text{in}})$. To efficiently parameterize ΔW, LoRA decomposes it as:

$$\Delta W = BA, \tag{8}$$

where $B \in \mathbb{R}^{d_{\text{out}} \times r}$ and $A \in \mathbb{R}^{r \times d_{\text{in}}}$. During fine-tuning, the original weights W_0 remain frozen, and only the parameters in A and B are optimized. In traditional full fine-tuning, the entire weight matrix is updated, requiring $d_{\text{out}} \cdot d_{\text{in}}$ trainable parameters. In contrast, the LoRA decomposition introduces only $r \cdot (d_{\text{in}} + d_{\text{out}})$ trainable parameters, which is more efficient when $r \ll \min(d_{\text{out}}, d_{\text{in}})$.

A.3. Distance Measures between Probability Distributions

We remind the reader of the necessary definitions required in formulating the distance between datasets, when interpreted as finitely supported probability measures (distributions). Given two probability distributions P and Q defined on a separable and complete (Polish) metric space $\mathcal{X}$ equipped with its Borel σ-algebra and metrized a metric $\rho : \mathcal{X}^2 \to [0, \infty)$, these measures of discrepancy (or divergences) are defined as follows:

Wasserstein Distance (WD). For distributions P and Q with cumulative distribution functions F_P and F_Q respectively, the 1-Wasserstein distance is defined as:

$$W_1(P, Q) \overset{\text{def.}}{=} \inf_{\pi} \int_{(x,y)\in\mathcal{X}^2} \rho(x, y)\, \pi(d(x, y)) \tag{9}$$

where the infimum is taken over all joint probability distributions π on $\mathcal{X} \times \mathcal{X}$ (with the product σ-algebra) whose marginals are P and Q.

[1]https://vectorinstitute.ai/partnerships/current-partners/

Kullback–Leibler Divergence (KL). The KL divergence measures the relative entropy between two distributions:

$$D_{KL}(P \parallel Q) \overset{\text{def.}}{=} \begin{cases} \int \log_{x\in\mathcal{X}} \frac{dP}{dQ}(x)P(dx) & : \text{if } P \ll Q \\ \infty & : \text{if } P \not\ll Q \end{cases} \tag{10}$$

where $P \ll Q$ denotes the absolute continuity of P with respect to Q, and $\frac{dP}{dQ}$ denotes the Radon-Nikodym derivative, or probability density, of P with respect to Q.

Jensen–Shannon Divergence (JS). The JS divergence is a symmetrized version of KL divergence:

$$D_{JS}(P \parallel Q) = \frac{1}{2}D_{KL}(P \parallel M) + \frac{1}{2}D_{KL}(Q \parallel M) \tag{11}$$

where $M = \frac{1}{2}(P + Q)$.

Maximum Mean Discrepancy (MMD). If $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ of functions over $\mathcal{X}$ with reproducing kernel function k; then we may also define the MMD between P and Q by

$$\text{MMD}^2(P, Q) = \mathbb{E}_{x,x'\sim P}[k(x, x')] - 2\mathbb{E}_{x\sim P,y\sim Q}[k(x, y)] + \mathbb{E}_{y,y'\sim Q}[k(y, y')]. \tag{12}$$

If $\mathcal{X}$ is $\mathbb{R}^d$ and $\mathcal{H} = L^2_\gamma(\mathbb{R}^d)$ for the standard Gaussian measure $\gamma \sim N(0, I_d)$, then k is often chosen to be a Gaussian kernel, i.e., $k(x, y) = \exp(-\frac{\|x-y\|^2}{2\sigma^2})$.

B. Proofs

We now prove the main result of our paper. We begin with the proof of our simplest result, Proposition 1.

B.1. Proof of Proposition 1

For any dataset D, note that $\operatorname{argmin}_{w\in\Delta_k} \frac{1}{K}\sum_{k=1}^K w_k\,\rho(D, D_k) + \frac{1}{\alpha}\sum_{k=1}^K w_k \log(w_k)$. Now, by (Wang et al., 2020, Proposition 1) its unique minimizer, which we denote by $w^\star_D$, is given by

$$w^\star_D = \frac{e^{-\rho(D,D_k)}}{\sum_{i=1}^K e^{-\rho(D,D_i)}} = \text{softmin} \circ \text{align}(D). \qquad \square$$

B.2. Simultaneous Proof of Theorem 1 and Proposition 2

We now derive Proposition 2 and Theorem 1 within the same proof, as their derivation is most naturally undertaken together.

Step 1 - Existence of a Measurable Selector
We will first set up the Measurable Maximum Theorem, see e.g. (Aliprantis and Border, 2006, Theorem 18.19). Consider the constant correspondence

$$\begin{aligned} \varphi : \mathcal{D} &\twoheadrightarrow 2^{\mathbb{R}^p} \\ D &\mapsto \text{co}(\Delta\theta). \end{aligned}$$

Since $\text{co}(\Delta\theta)$ is a closed, non-empty, and bounded set then the Heine-Borel theorem implies that $\text{co}(\Delta\theta)$ is compact. Whence φ is a correspondence with non-empty, compact, and convex values. Let $B \subseteq \mathcal{D}$ be a Borel set, then

$$\varphi(B) \overset{\text{def.}}{=} \bigcup_{D\in B} \varphi(D) = \bigcup_{D\in B} \text{co}(\Delta\theta) = \text{co}(\Delta\theta). \tag{13}$$

Since $\text{co}(\Delta\theta)$ is closed it is Borel; whence, φ is not only a weakly measurable correspondence (Aliprantis and Border, 2006, 18.1 Definition (1)) but it is also a Borel measurable correspondence (Aliprantis and Border, 2006, 18.1 Definition (3)). Thus, the correspondence φ satisfies the requirements of (Aliprantis and Border, 2006, Theorem 18.19).

Next, consider the objective function

$$\begin{aligned} L : \mathcal{D} \times \mathbb{R}^p &\to [0, \infty) \\ (D, B) &\mapsto \mathbb{E}_{(X,Y)\sim D}\big[\ell(f_{\theta+\vartheta}(X), Y)\big]. \end{aligned} \tag{14}$$

We will show that L is Carathéodory function by showing it is continuous. Since $\mathcal{D} \times \mathbb{R}^p$ is a product (topological) space, then (Munkres, 2000, Theorem 19.6) guarantees that f is continuous if and only if each of its component functions is continuous; we show the latter.

Fix $D \in \mathcal{D}$. Since $\mathcal{C}$ and the softmax function are locally Lipschitz, and since ℓ is Lipschitz, then their composition is locally Lipschitz. Whence, for each $(x, y) \in [0, 1]^{d+D}$ the map

$$\Lambda_{x,y} : \operatorname{co}(\Delta\theta)) \ni \vartheta \mapsto \ell(\mathcal{C}_{\theta+\vartheta}(x), y))$$

is λ-Lipschitz, for some $\lambda \geq 0$. By Jensen's inequality we have: for each $\vartheta_1, \vartheta_2 \in \operatorname{co}(\Delta\theta)$

$$\begin{aligned} &\Big|\mathbb{E}_{(X,Y)\sim\mathcal{D}}\big[\Lambda_{X,Y}(\vartheta_1)\big] - \mathbb{E}_{(X,Y)\sim\mathcal{D}}\big[\Lambda_{X,Y}(\vartheta_2)\big]\Big| \\ &= \Big|\mathbb{E}_{(X,Y)\sim\mathcal{D}}\big[\Lambda_{X,Y}(\vartheta_1) - \Lambda_{X,Y}(\vartheta_2)\Big| \\ &\leq \mathbb{E}_{(X,Y)\sim\mathcal{D}}\Big[\big|\Lambda_{X,Y}(\vartheta_1) - \Lambda_{X,Y}(\vartheta_2)\big|\Big] \\ &\leq \lambda\mathbb{E}_{(X,Y)\sim\mathcal{D}}\Big[\big\|\vartheta_1 - \vartheta_2\big\|\Big]. \end{aligned}$$

Thus, L is locally Lipschitz in its second argument; in particular, it is continuous in its second argument.

Now, we show continuity in its first argument. Fix $\vartheta \in \operatorname{co}(\Delta\theta)$. Let $(D_n)_{n=1}^{\infty}$ be a sequence in $\mathcal{D}$ converging to some measure $D \in \mathcal{D}$. Since d metrizes the (relative) weak topology in $\mathcal{P}([0, 1]^{d+D})$ relative to $\mathcal{D}$, then by Alexandrov's Portmanteau Theorem, see e.g. (Kallenberg, 2021, Theorem 5.25), for every continuous and bounded function $g \in C_b([0, 1]^{d+D})$ we have

$$\lim_{n\uparrow\infty} \big|\mathbb{E}_{(X,Y)\sim D_n}[g(X, Y)] - \mathbb{E}_{(X,Y)\sim D}[g(X, Y)]\big| = 0. \tag{15}$$

Since $\lambda_{x,y}(\vartheta)$ is locally-Lipschitz for each $\vartheta \in \operatorname{co}(\Delta\theta)$ and $[0, 1]^{d+D}$ is compact then $(x, y) \mapsto \lambda_{x,y}(\vartheta)$ is bounded (and of course continuous). Thus, we pay pick g in (15) to be $(x, y) \mapsto \lambda_{x,y}(\vartheta)$; whence,

$$\lim_{n\uparrow\infty} \big|\mathbb{E}_{(X,Y)\sim D_n}[\lambda_{X,Y}(\vartheta)] - \mathbb{E}_{(X,Y)\sim D}[\lambda_{X,Y}(\vartheta)]\big| = 0. \tag{16}$$

Thus, L is continuous in its first argument as well. Therefore, L is continuous, which implies that it is Carathéodory. This completes the verification of all the conditions of the Measurable Maximum Theorem, again see (Aliprantis and Border, 2006, Theorem 18.19 (2)), have been verified. Whence: 1) for each $D \in \mathcal{D}$ the argmin set

$$M(D) \overset{\text{def.}}{=} \Big\{\vartheta \in \operatorname{co}(\Delta\theta) : \mathbb{E}_{(X,Y)\sim\mathcal{D}}\Big[\ell(f_{\theta+\Delta\theta}(X), Y)\Big] = \ell^\star(D)\Big\}$$

is non-empty; where the corresponding oracle loss is given by

$$\ell^\star(D) \overset{\text{def.}}{=} \inf_{\Delta\theta\in\operatorname{co}(\Delta\theta)} \mathbb{E}_{(X,Y)\sim\mathcal{D}}\Big[\ell(f_{\theta+\Delta\theta}(X), Y)\Big].$$

This establishes Proposition 2. Moreover, (Aliprantis and Border, 2006, Theorem 18.19 (1) and (3)) further imply that there exists a measurable selector $S : \mathcal{D} \to \operatorname{co}(\Delta\theta)$; i.e. S is Borel measurable for each $D \in \mathcal{D}$ the following optimal selection property holds:

$$S(D) \in M(D). \tag{17}$$

Step 2 - Change of Domain
Next, we create a "copy" of S in "distance domain" $[0,\infty)^K$. By the well-posedness assumption made in Assumption 1, the map $\operatorname{align} : D \to [0,\infty)^K$ is injective. Thus, align is bijective onto its image. Since each component of align is given by the 1-Lipchitz, and therefore continuous, function $\mathcal{D} : D \mapsto \rho(D, D_k) \in [0,\infty)$; then (Munkres, 2000, Theorem 19.6) implies that align is continuous. Consequentially, align is a measurable bijection onto its image $\operatorname{align}(\mathcal{D})$. Thus, (Kechris, 1995, corollary 15.2) implies that align has a measurable inverse $\psi : \operatorname{align}(\mathcal{D}) \to \mathcal{D}$ on its image $\operatorname{align}(\mathcal{D})$; i.e.

$$\operatorname{align} \circ \psi = 1_{\operatorname{align}(\mathcal{D})} \text{ and } \psi \circ \operatorname{align} = 1_{\mathcal{D}}. \tag{18}$$

Define the map $S' : \operatorname{align}(\mathcal{D}) \to \operatorname{co}(\Delta\theta)$ by composition with ψ via

$$S' \overset{\text{def.}}{=} S \circ \psi.$$

Let $\tilde{S}$ be any measurable extension of S' to all of $\mathbb{R}^K$; e.g.

$$\tilde{S} \overset{\text{def.}}{=} S' I_{x \in \operatorname{align}(\mathcal{D})} + \Delta\theta_1 \, I_{x \notin \operatorname{align}(\mathcal{D})}.$$

By construction: for each $D \in \mathcal{D}$

$$\tilde{S} \circ \operatorname{align}(D) = S(D). \tag{19}$$

Step 3 - High-Probability of Continuity
Consider the pushforward (probability) measure $\mathbb{Q} \overset{\text{def.}}{=} \operatorname{align}_\sharp \mathbb{P}$ on $[0,\infty)^K$, supported on $\operatorname{align}(\mathcal{D})$. Now, by Lusin's Theorem, as formulated in (Klenke, 2020, Exercise 13.1.3), for every $\varepsilon \in (0,1]$ there exists a compact subset $K_\varepsilon \subset \operatorname{supp}(\mathbb{Q}) \subseteq \operatorname{align}(\mathcal{D})$ such that

$$\mathbb{Q}(K_\varepsilon) \geq 1 - \varepsilon \text{ and } \tilde{S}|_{K_\varepsilon} \in C(K_\varepsilon, \operatorname{co}(\Delta\theta)) \tag{20}$$

where $C(K_\varepsilon, \operatorname{co}(\Delta\theta))$ denotes the set of continuous functions from K_ε to $\operatorname{co}(\Delta\theta)$.

Since $\tilde{S}|_{K_\varepsilon}$ is continuous *and its image lies in a closed convex set* then the Dugundji-Tietze theorem, see (Dugundji, 1951, Theorem 4.1), implies that there exists a continuous extension $S_\varepsilon : \mathbb{R}^K \to \operatorname{co}(\Delta\theta)$; i.e.

$$S_\varepsilon(x) = \tilde{S}(x) \tag{21}$$

for all $x \in K_\varepsilon$.

Step 4 - Approximation by Models of the form (5)
Let $W : \mathbb{R}^{K-1} \to \mathbb{R}^K$ be the affine map of (Kratsios and Papon, 2022, Example 13). Then, by nearly identical computation to (Kratsios and Papon, 2022, Example 13), we find that the map

$$\begin{aligned} \mathbb{R}^K &\to \operatorname{co}(\Delta\theta) \\ w &\mapsto \operatorname{softmax}(W(w))^\top (L_1, \dots, L_K) \end{aligned} \tag{22}$$

also satisfies (Kratsios and Papon, 2022, Assumption 8). Since $\operatorname{softmin} = \operatorname{softmax}(-\cdot)$; set $\tilde{W} \overset{\text{def.}}{=} -W$, and note that, the result of (22) can be re-expressed as

$$\begin{aligned} \mathbb{R}^K &\to \operatorname{co}(\Delta\theta) \\ w &\mapsto \operatorname{softmin}(\tilde{W}(w))^\top (L_1, \dots, L_K) \end{aligned} \tag{23}$$

Since $\tilde{S}$ is continuous, $K_\varepsilon \subset \mathbb{R}^K$ is a non-empty compact set, and ς is a continuous activation function satisfying the Kidger-Lyons condition, of (Kidger and Lyons, 2020); namely it differentiable with non-zero derivative at at least one point in $\mathbb{R}$, then (an inconsequential mild variant) of (Kratsios and Papon, 2022, Theorem 37 (ii)) implies that: for every $\delta > 0$ (to be fixed retroactively) there exists an MLP $\hat{f} : \mathbb{R}^K \to \mathbb{R}^K$ with activation function ς such that the map

$$\hat{f} \overset{\text{def.}}{=} [\operatorname{softmin} \circ \mathcal{C}(\cdot)]^\top (L_1, \dots, L_K) : \mathbb{R}^K \to \operatorname{co}(\Delta\theta)$$

satisfies the uniform approximation guarantee

$$\max_{x\in K_\varepsilon} \big\|F_\delta(x) - S_\varepsilon(x)\big\| < \delta. \tag{24}$$

Now, by (19), the continuous extension property in (21), and the approximation guarantee in (24) we find that

$$\max_{D\in\psi(K_\varepsilon)} \big\|\hat{f}\circ \operatorname{align}(D) - S(D)\big\| = \max_{x\in K_\varepsilon} \big\|\hat{f} - S_\varepsilon(x)\big\| < \delta. \tag{25}$$

The continuity of L, defined in (14), implies that $\delta > 0$ may be taken to be small enough so that: for each $D \in \psi(K_\varepsilon)$

$$\big|L(D, \theta + \mathcal{C}_\delta) - L(D, \theta + S(D))\big| < \varepsilon. \tag{26}$$

Step 5 - ϵ-Optimality with high probability
Combining the ε-uniform approximation guarantee in (26) for $\mathcal{C}_\delta \circ \operatorname{align}$ with the optimality guarantee for S in (17) implies that: for each $D \in \psi(K_\varepsilon)$

$$L(D, \theta + \mathcal{C}_\delta) - \varepsilon \le L(D, \theta + S(D)) = \ell^\star(D). \tag{27}$$

Snow, since $D \mapsto L(D, \theta + \mathcal{C}_\delta)$ is the composition of continuous functions, it is continuous and therefore measurable; whence the set

$$M_\varepsilon^\star \overset{\text{def.}}{=} \big\{L(D, \theta + \mathcal{C}_\delta) - \varepsilon \le \ell^\star(D)\big\}$$

is Borel measurable and contains $\psi(K_\varepsilon)$. In particular, $\mathbb{P}(M_\varepsilon^\star)$ is well-defined. Finally, the lower-bound in (20) yields

$$\mathbb{P}(M_\varepsilon^\star) \ge \mathbb{P}(\psi(K_\varepsilon) \ge \mathbb{Q}(K_\varepsilon) \ge 1 - \varepsilon$$

which concludes our proof.

C. Implementation Details

In our implementation, we used the Lots-of-LoRAs HuggingFace repository (Gabrielsson, 2025), which contains 502 dataset-adapter pairs for Mistral-7B-Instruct-v0.2. From these 502 English datasets, 10 are manually selected to ensure diversity of tasks spanning classification, commonsense reasoning, and question generation domains for evaluation. Additionally, 492 datasets are randomly selected from the 1616 diverse natural language processing (NLP) tasks provided by (Wang et al., 2022). Each adapter comprises $p = 9,437,184$ parameters, stored as 32-bit floating-point numbers (approximately 36 MB).

To further reduce the computational load of our training procedure, we implemented several critical optimizations in Step 2:

1. **Symmetry exploitation**: For symmetric difference metrics (WD, JS, and MMD), we calculate only half of the possible $N \times N$ distances, reusing values obtained from calculations done for pair (i, j), where $i < j$, as the (j, i) pair as well.
2. **Pre-computation of probability distributions**: For metrics requiring probability density functions (KL and JS), we pre-calculate and cache these distributions for all datasets to avoid repeating these costly computations.
3. **Parallelization**: We also utilize multi-threading capabilities by assigning each distance calculation to a separate CPU thread, allowing these independent computations to be processed concurrently.

Table 2 reports the time elapsed at each stage of our LoRA generation pipeline, measured on a Dell XPS 15 (Intel i7-13700H, 14 cores, 64 GB RAM). All steps, except for the final inference and adapter loading, were executed using the CPU only across 502 datasets. Importantly, we ran this benchmark by predicting the adapter for each of the 502 datasets, assuming the remaining 501 were given, to evaluate the overall performance of our pipeline.

Excluding GPU inference and adapter loading, generating predicted adapters for 502 datasets across 12 methods took roughly 9 hours. In typical use—predicting one adapter using a single variate of our LoRA generation pipeline and metric—runtime is much lower: generating one adapter from 100 reference pairs takes 10–20 minutes, depending on compute, memory, network, and dataset size. Runtime scales roughly linearly with the number and size of reference datasets, as most steps run independently per dataset. However, full experimental runs involving pairwise comparisons (e.g., distance computations) scale quadratically with the number of datasets.

Table 2. Time elapsed for each step of the pipeline for all 502 datasets at once (CPU only).

Pipeline Step	**Time**
1. Dataset-Adapter pairs gathering:	
Downloading raw data	15 min
2. Datasets Pre-processing:	
Tokenization	10 min
3. Distribution similarity calculations:	
Wasserstein (WD)	3 hours
Kullback–Leibler (KL)	5 min
Jensen–Shannon (JS)	5 min
Maximum Mean Discrepancy (MMD)	1.5 hours
4. Distances Processing (Coefficients):	
Base attentional	3 min
Normalized	3 min
MLP-based	45 min
5. Adapter prediction:	
Calculating adapters and saving	5 min

C.1. Pipeline Steps

We evaluate three pipelines for predicting LoRA adapter parameters. The *Attentional* method is lightweight, using only matrix multiplications with no learned components. The *Normalized* method standardizes distance values to a normal distribution to stabilize the *SoftMin* stage. The *Neural* method trains a small CPU-based MLP to minimize MSE between predicted and actual adapter weights and biases.

C.1.1. Dataset-Adapter pairs gathering

Our approach relies on pre-existing fine-tuned adapters and their corresponding datasets. We begin by gathering a set of N datasets, denoted as $\{D_i\}_{i=1}^{N}$, where for each dataset, we also have the optimal adapters, $\{\theta_n\}_{n=1}^{N}$. These adapters are generated by fine-tuning the same base model, using the same adapter structure, on their respective datasets.

C.1.2. Datasets Pre-processing

Next, we tokenize each dataset using the base model's tokenizer, converting inputs and outputs into integer sequences. Formally, we apply a tokenizer $T : \mathcal{S} \to \mathbb{Z}^{l\times V}$ (where l being the length of the tokenized sequence and V being the tokenizer's vocabulary size) to map each string to its sequence of token IDs. The resulting sequences denoted $\{T(D_i)\}_{i=1}^{N}$, contain all the tokenized inputs followed by the outputs for each dataset. This preprocessing step is computationally efficient and highly parallelizable. We also extract the LoRA adapter parameters (weights and biases) from each fine-tuned model, reshape them into one-dimensional vectors, and stack them into a matrix $\theta_{\text{all}} \in \mathbb{R}^{N\times p}$ (N being the number of dataset-adapter pairs, and p the number of parameters per adapter). Thus, each row represents the parameters of a single adapter.

C.1.3. Distribution Distance Computation

A key step in our pipeline is computing the dissimilarity between datasets, which are treated as probability distributions over tokenized sequences. Given tokenized datasets $\{T(D_i)\}_{i=1}^{N}$, we compute pairwise distances using four established measures: the Wasserstein distance, Kullback–Leibler divergence, Jensen–Shannon divergence, and Maximum Mean Discrepancy, as defined in Appendix A.3. For each tokenized dataset $T(D_i)$, we calculate a distance vector $\delta_i = [\rho(T(D_i), T(D_1)), \rho(T(D_i), T(D_2)), \ldots, \rho(T(D_i), T(D_N))]$, where ρ is the chosen divergence metric. In practice, we mask the self-distance $\rho(T(D_i), T(D_i))$ by assigning it a large value prior to normalization. Note that here we are emphasizing the tokenization step using the $T(D_i)$ notation, whereas in the main text we often omit this.

C.1.4. Distances Processing (With different methodologies)

The goal here is to find how close each dataset is to the current dataset and to assign coefficients to them in such a way that these coefficients increase as the similarity increases.

Attentional Approach In this baseline approach, we directly apply the softmin function to the distance vectors, after masking the self-corresponding entry. For each dataset D_i, we calculate:

$$w_i(j) = \text{softmin}(\delta_i(j) \mid j \in 1, 2, ..., N, j \neq i) \tag{28}$$

where $\delta_i(j) = \rho(T(D_i), T(D_j))$ represents the distance between the tokenized datasets.

Attentional Approach - With Normalization In this variant, we normalize each distance vector to have zero mean ($\mu = 0$) and unit variance ($\sigma = 1$), effectively applying z-score standardization. This transformation is equivalent to applying a softmin with an adaptive temperature $\tau_i = \sigma_i$ (its own standard deviation). When σ_i is small, the temperature is low, leading to sharper, more peaked (i.e., sparse) coefficient distributions. Conversely, larger σ_i results in flatter distributions. Empirically, we observe that most σ_i values are small after masking the self-distance, which leads to sparser weights—and, interestingly, improved performance.

Neural Approach The third pipeline, justified by Theorem 1, uses a small MLP to map distance values to adapter weights. It minimizes the MSE between predicted and actual adapter parameters (weights and biases). The MLP used here has three fully connected layers, with the first two followed by layer normalization and ReLU activations.

$$h = \text{ReLU}(\text{Layer Normalization}(W_1 x + b_1)), \quad h \in \mathbb{R}^{4000} \tag{29}$$

$$\hat{h} = \text{ReLU}(\text{Layer Normalization}(W_2 h + b_2)), \quad \hat{h} \in \mathbb{R}^{4000} \tag{30}$$

$$\hat{y} = W_3 \hat{h} + b_3, \quad \hat{y} \in \mathbb{R}^{1} \tag{31}$$

where $x \in \mathbb{R}$ is a single distance value (scalar), $W_1 \in \mathbb{R}^{4000\times 1}$, $W_2 \in \mathbb{R}^{4000\times 4000}$, and $W_3 \in \mathbb{R}^{1\times 4000}$ are weight matrices, and $b_1 \in \mathbb{R}^{4000}$, $b_2 \in \mathbb{R}^{4000}$, and $b_3 \in \mathbb{R}^1$ are bias terms. We apply the MLP to transform all distance values:

$$w_i(j) = \text{softmin}(\text{MLP}(\delta_i(j)) \mid j \in 1, 2, ..., N, j \neq i) \tag{32}$$

C.1.5. Adapter Prediction

We make our prediction with a straightforward linear combination of existing adapters, weighted by the processed distances:

$$\hat{\theta}_i = \sum_{j=1, j\neq i}^{N} w_i(j)\theta_j. \tag{33}$$

This formulation effectively answers the key question: "Based on the distances between a new dataset and each of the datasets with known adapters, what proportion of information should the new adapters inherit from each of the fine-tuned (reference) adapters?" The processed distances serve as coefficients determining the knowledge transfer from each source adapter.

In our study, to make the predictions for all datasets more efficient, we construct a weight (coefficient) matrix $W \in \mathbb{R}^{N\times N}$ where row i contains the processed distances w_i, allowing us to compute all predictions simultaneously by leveraging hardware acceleration and vectorization.

Deployment and Inference Predicted adapters match the size of flattened fine-tuned adapters and can be reshaped to their original structure, ensuring full compatibility with existing LoRA inference pipelines. Once generated, they can be directly loaded for downstream use.

D. Further Experimental Evaluation

This appendix presents a detailed account of our experimental observations.

D.1. Exact Match Evaluation

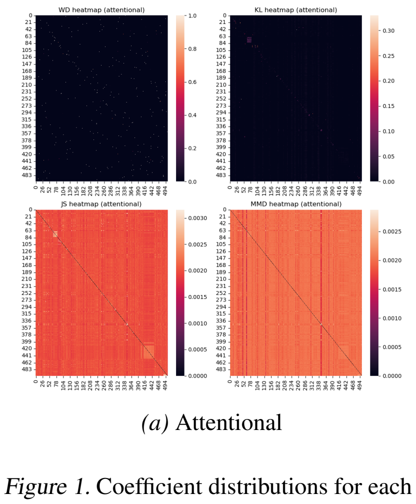
In addition to Rouge-L, we evaluate our LoRA generation pipelines using the Exact Match (EM) metric, which measures the fraction of test samples for which the model's output exactly matches the expected string. This is a particularly meaningful complement for classification-style tasks common in our dataset corpus, where outputs are short, well-defined, and often categorical. Without any fine-tuning, the Mistral model achieved a score of 0.016 ± 0.069. Ideally, if the user had access to GPUs, the GPU fine-tuned models would achieve an average exact match score of 0.654 ± 0.351). As shown in Table 3, each of our pipelines performs substantially better than the base foundation model, but as expected, it does not achieve the same predictive power as LLMs with fine-tuned LoRAs. Additionally, note that we observe a strong correlation between Rouge-L and EM scores across all methods and distance metrics. Both evaluation scores consistently rank the Normalized approach with JS as the top-performing configuration. While Rouge-L captures partial overlap between generated and reference sequences, EM provides a stricter binary signal of correctness. Despite this difference in granularity, the relative performance of the Attentional, Normalized, and Neural approaches remains consistent suggesting that improvements in soft sequence similarity are accompanied by gains in exact prediction accuracy.

Table 3. Exact match performance of our lightweight LoRA prediction pipelines.

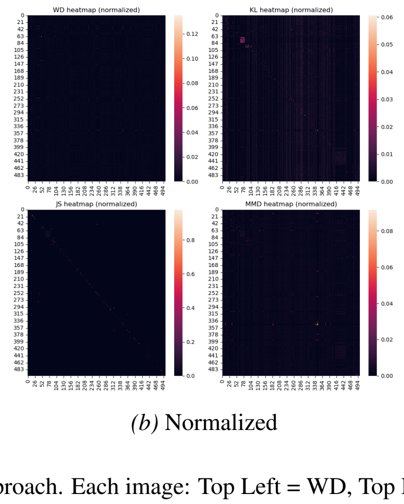
Approach	WD	KL	JS	MMD
Attentional	0.288 ± 0.297	0.344 ± 0.302	0.328 ± 0.296	0.327 ± 0.295
Normalized	0.338 ± 0.296	0.330 ± 0.297	0.373 ± 0.314	0.340 ± 0.298
Neural	0.338 ± 0.294	0.323 ± 0.295	0.325 ± 0.296	0.337 ± 0.298

D.2. Coefficient Distribution Analysis

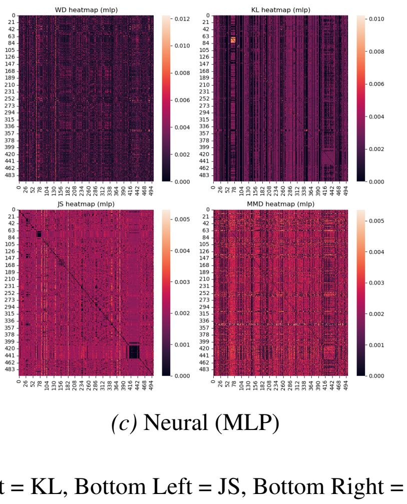
Figures 1a, 1b, and 1c below show the LoRA matrices produced by each approach across each of our datasets. In each visualization, both the horizontal and vertical axes list the dataset, and each of the $(i, j)^{th}$ pixel darkness indicators how much of the pre-trained LoRA from dataset i is used to predict the LoRA for dataset j. Darker pixels indicate lower coefficients, while brighter ones indicate higher weights assigned to a source adapter for each target dataset. Both axes correspond to dataset indices. Interestingly, the Normalized approach exhibits extreme sparsity: most weights are near zero, and each prediction is dominated by one or two reference adapters, as evidenced by the presence of isolated bright pixels in a largely dark matrix. In contrast, the Neural (MLP) and Attentional methods display greater dispersion in the coefficients, with many rows exhibiting moderate weights across a broader range of source adapters. This reflects a more distributed and nuanced reuse of prior adapters. Given that the Normalized approach exhibits slightly better performance in practice, this visualization may suggest that sparsity is important, but further investigation in follow-up work is encouraged.



(a) Attentional



(b) Normalized



(c) Neural (MLP)

Figure 1. Coefficient distributions for each approach. Each image: Top Left = WD, Top Right = KL, Bottom Left = JS, Bottom Right = MMD.

D.3. Sample Outputs Generated by Predicted Models

Below, we have included four randomly selected inputs, expected outputs, and the generated texts by a model with predicted adapters using our best configurations: the JS-divergence-based Normalized approach.

Input Prompt [Task ID 671]:
"Definition: In this task, you're given an ambiguous question (which can be answered in more than one way). Your task is to provide one question which clarifies the input question and it has one unique answer, and also provide an answer to the generated question. Generated question and answer should be separated with a new line.

Positive Example 1 -
Input: Lucifer how many episodes are in season 3?
Output: How many episodes of season 3 of Lucifer were there, including bonus episodes?
26

Positive Example 2 -
Input: What is the tallest ride at six flags over texas?
Output: What is the tallest roller coaster at six flags over texas 2001?
The Titan

Negative Example 1 -
Input: Lucifer how many episodes are in season 3?
Output: How many episodes of season 3 of Lucifer were there, including bonus episodes? 26

Negative Example 2 -
Input: What is the tallest ride at six flags over texas?
Output: What is the tallest roller coaster at six flags over Texas 2001?

Now complete the following example -
Input: When was season 7 of game of thrones released?
Output:

"

Expected Answer:
"When was season 7, episode 1 of game of thrones released through simulcast?
July 16, 2017"

Model Output:
"When was season 7 of game of thrones released in the US?
July 16, 2017

Input Prompt [Task ID 431]:
"Definition: In this task you are given a sentence. You must judge whether the object of the main clause is singular(like: apple) or plural(like: apartments). Label the instances as "Singular" or "Plural" based on your judgment.

Positive Example 1 -
Input: They both let out relieved breaths at the doctorś words.
Output: Plural

Positive Example 2 -
Input: He signaled the bartender and ordered one more drink.
Output: Singular

Negative Example 1 -
Input: I collect my painting supplies.
Output: Singular

Negative Example 2 -
Input: I saw the ramp leading back toward the surface.
Output: Plural

Now complete the following example -
Input: For some unknown reason, that second kiss wreaked more emotional havoc than the first had.
Output:

"

Expected Answer:
"Singular"

Model Output:
"Singular"

Input Prompt [Task ID 322]:
"Definition: In this task, you are given a public comment from online platforms. You are expected to classify the comment into two classes: threat and non-threat. Threat is a statement that someone will be hurt or harmed, especially if the person does not do something in particular.

Positive Example 1 -
Input: Comment: Enough already. Cut the head off this snake.
Output: Threat

Positive Example 2 -
Input: Comment: Disgusting comment, I pity anyone who has to live with you.
Output: Non-threat

Negative Example 1 -
Input: Comment: Bravo! Bravo!

Encore!! Encore!! Encore!!
Output: Threat

Negative Example 2 -
Input: Comment: ... after shooting her with her own gun.
Output: Non-threat

Now complete the following example -
Input: Comment: It's a good thing that right wing illiterates in farms don't read the newspaper.
Output:

"

Expected Answer:
"Non-threat"

Model Output:
"Non-threat"

Input Prompt [Task ID 1398]:
"Definition: Given a fact, create a question that can be answered using the fact. Construct the question such that it is unambiguous, has a unique answer and the answer can be given using the fact.

Positive Example 1 -
Input: Fact: deep sea animals live deep in the ocean
Output: Frilled sharks and angler fish live far beneath the surface of the ocean, which is why they are known as?

Positive Example 2 -
Input: Fact: as an object moves , the kinetic energy of that object will increase
Output: An example of lots of kinetic energy would be?

Negative Example 1 -
Input: Fact: water is often brackish in an estuary
Output: What is the sun made of?

Negative Example 2 -
Input: Fact: if a liquid disappears then that liquid probably evaporated
Output: What happens is water is mopped up?

Now complete the following example -
Input: Fact: as the use of a crop increases , the amount of crops planted will increase
Output:

"

Expected Answer:
"When the demand for corn rises?"

Model Output:
"Which crop is most likely to be planted in large quantities due to its high demand?"