HydraLoRA: An Asymmetric LoRA Architecture for Efficient Fine-Tuning

Chunlin Tian[†]
University of Macau
yc27402@um.edu.mo

Zhan Shi[†]
University of Texas at Austin
zshi17@cs.utexas.edu

Zhijiang Guo
University of Cambridge
zg283@cam.ac.uk

Li Li[*]
University of Macau
llili@um.edu.mo

Chengzhong Xu
University of Macau
czxu@um.edu.mo

Abstract

Adapting Large Language Models (LLMs) to new tasks through fine-tuning has been made more efficient by the introduction of Parameter-Efficient Fine-Tuning (PEFT) techniques, such as LoRA. However, these methods often underperform compared to full fine-tuning, particularly in scenarios involving complex datasets. This issue becomes even more pronounced in complex domains, highlighting the need for improved PEFT approaches that can achieve better performance. Through a series of experiments, we have uncovered two critical insights that shed light on the training and parameter inefficiency of LoRA. Building on these insights, we have developed *HydraLoRA*, a LoRA framework with an asymmetric structure that eliminates the need for domain expertise. Our experiments demonstrate that *HydraLoRA* outperforms other PEFT approaches, even those that rely on domain knowledge during the training and inference phases. Code is available.

1 Introduction

Large Language Models (LLMs; [10, 3, 36, 47, 48, 32, 33]) are notably powerful, yet their training involves substantial expense. Adapting a single LLM for multiple downstream applications via fine-tuning has emerged as a prevalent method to cater to specific domain needs, balancing performance with practicality. This approach, however, faces a significant challenge due to the extensive memory and computational resources required for full fine-tuning (FFT), i.e., fine-tuning all billions of parameters. A solution to this has been the development of more selective adaptation techniques, involving modifying only a portion of the parameters or integrating external modules designed for new tasks. Key methodologies in this sphere include LoRA [18], Adaptors [37, 17, 31], and many other variants [25, 24, 9, 14, 53], all part of what can be generally termed as Parameter-Efficient Fine-tuning (PEFT). PEFT strategies are characterized by freezing the backbone model parameters while only a minimal number of task-specific parameters are introduced and fine-tuned. This method substantially boosts efficiency in the phases of fine-tuning and subsequent deployment, marking a significant advancement in the practical use of LLMs.

While fine-tuning a small subset of parameters offers a streamlined approach for domain adaptation, it's well-recognized that model performance is closely tied to the number of parameters involved [22]. This intrinsic characteristic of methods like LoRA often results in them falling short of the FFT baseline, which updates all parameters, thereby creating a trade-off between efficiency and model quality. This issue of compromised quality in a low-parameter setting becomes even more pronounced in target domains characterized by complex sub-domains and diverse tasks. This situation presents a compelling research question:

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-0304

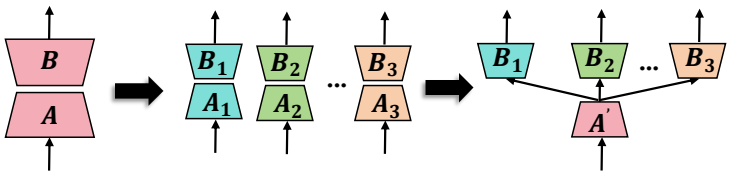


Figure 1: Illustration of LoRA architecture changes in *HydraLoRA*. Only the tunable parameters are shown in this Figure. (a) LoRA architecture with matrix A to achieve low rank and matrix B to recover. (b) under the same parameter count, a monolithic LoRA is split into multiple smaller A and B matrices to avoid training interference. (c) based on (b), *HydraLoRA* has an asymmetric structure that has a shared A matrix and multiple B matrices.

What is the optimal architecture that can deliver superior model performance while still capitalizing on the efficiency benefits of a reduced parameter footprint?

In our research, we carry out a series of exploratory experiments, applying LoRA to the LLaMA2 [48] model to adapt it to a new domain encompassing multiple downstream tasks. As shown in Figure 1(a), LoRA adds trainable pairs of rank decomposition matrices A and B in addition to existing weight matrices. Our in-depth analysis of LoRA's mechanics yields several insightful observations and leads to the formulation of key hypotheses. First, rather than employing a single LoRA for the entire domain, it proves more effective to deploy multiple, smaller LoRA heads, each dedicated to a specific downstream task (see Figure 1(b)). This suggests that domain or task interference might harmfully impact the training process. We further hypothesize that this interference originates from *"intrinsic components"*—sub-domains or distinct tasks—potentially unknown even to domain experts. Additionally, upon visualizing the parameters of LoRA, we discern a pattern: some parameters predominantly learn the commonalities across all data, while others focus on the unique aspects of each intrinsic component. From these observations, we posit that an optimal LoRA architecture should embody an explicit, asymmetric structure.

Building upon the observations, we propose an improved end-to-end LoRA framework, which we refer to as *HydraLoRA*. From the architecture perspective, unlike LoRA's symmetric structure, *HydraLoRA* has an asymmetric structure that has a shared A matrix and multiple B matrices (see Figure 1(c)). The shared A matrix is used by all samples for parameter efficiency. During the fine-tuning phase, *HydraLoRA* is designed to auto-identify "intrinsic components" and segregate training samples into distinct B matrices. During the inference phase, *HydraLoRA* leverages multiple B matrices using Mixture-of-Experts (MoE; [20, 40]) manner. Unlike prior work, *HydraLoRA* completely eliminates the need for human expertise and assumptions, showing better performance than using domain knowledge to guide the fine-tuning process.

2 Background and Motivation

2.1 LoRA Basics

LoRA [18] achieves comparable performances to fine-tuning on many benchmarks by freezing the pre-trained model weights W_0 and inserting trainable rank decomposition matrices into each layer of the pre-trained model. In particular, for each layer, LoRA uses two sequential low-rank matrices A and B to fit the residual weights for adaptation. The forward computation is written as follows:

$$y\prime = y + \Delta y = W_0 x + BAx \tag{1}$$

where $y \in R^{\,d}$ is the output and the $x \in R^{\,k}$ denotes the input. $B \in R^{\,d\times r}, A \in R^{\,r\times k}$ with $r \ll min(d,k)$. Normally matrix B is initialized with zeroes and matrix A is initialized with Kaiming Uniform [15] to force $\Delta\, y = 0$ at the beginning.

https://doi.org/10.52202/079017-0304

2.2 LoRA's Practical Dilemma

Parameter count has a clear impact on the performance of neural models [22, 33]. Yet, Parameter-Efficient Fine-tuning (PEFT) methods, such as Adapter [17] and prefix-tuning [25], focus on fine-tuning a limited set of parameters. These approaches present a practical dilemma: while restricting the number of tuned parameters is essential for training efficiency, it hinders the model's ability to learn from diverse datasets. This trade-off becomes particularly evident when considering corpus heterogeneity [2]. Figure 2 reveals a notable performance disparity between PEFT techniques and full fine-tuning (FFT), with the gap widening in scenarios involving a more diverse or heterogeneous training corpus.

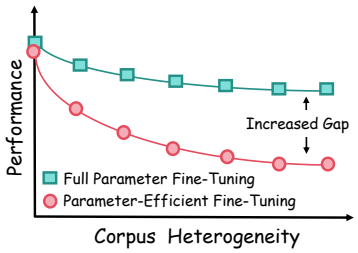


Figure 2: Performance impact of corpus heterogeneity on full fine-tuning vs. parameter-efficient fine-tuning. Heterogeneity signifies the diversity within the dataset, often leading to interference due to its varied content and style [2]. Parameter-efficient approaches are particularly sensitive, suffering greater performance losses in heterogeneous cases.

Table 1: Performance on instruction tuning with Dolly-15K [8] and evaluated with MMLU [16] with different ranks. For LoRA (Split) decomposes high-rank LoRA modules into smaller, equivalent low-rank components ($r \times n$). n is the number of LoRAs, r denotes the rank of each LoRA.

Schemes	$r \times n$	MMLU ↑	% Parameter
LoRA	8×1	43.22	0.062
LoRA	16×1	45.45	0.124
LoRA	32×1	46.59	**0.248**
LoRA (Split)	16×2	46.82	0.248
LoRA (Split)	8×4	**46.94**	0.248
LoRA (Split)	4×8	46.83	0.248

2.3 Observations

In this work, we aim for a PEFT approach that strikes a better balance between maximizing the learning capability for heterogeneous data and minimizing the number of parameters involved. A key goal is to ensure that our enhanced technique exhibits robust generalization across unseen tasks, independent of any prior task-specific knowledge. To achieve our objectives, we focus on LoRA and conduct a series of experiments as Table 1 to gain a deeper understanding of its mechanisms. Our methodology involves leveraging data from diverse tasks within a domain, and training distinct LoRA heads for each domain, leading to our first observation:

Observation I*: With the same parameter count, rather than employing a single LoRA for the entire domain dataset, it proves more effective to deploy multiple, smaller LoRA heads, each dedicated to a specific downstream task.*

This suggests that interference among tasks might harmfully impact the training process. Furthermore, we posit that this interference is NOT exclusive to this explicit multi-task training. This interference could happen in any training setting since all datasets inherently consist of multiple implicit *intrinsic components*, such as sub-domains or tasks within a domain that is even unknown to domain experts. To better understand how multiple LoRA heads mitigate the interference among intrinsic components, in Figure 3, we employ the t-SNE technique [49] to visualize the parameters of matrix A and B across all heads. This analysis yields another critical observation:

Observation II*: When multiple LoRA heads are trained individually on different data, the parameters of matrix A from different heads tend to converge, while those of matrix B are distinguishable.*

In detail, the parameters of matrix A across all heads exhibit a high degree of similarity, leading to their overlaps in the figure. Conversely, the parameters of matrix B from different heads are distinct and easily distinguishable. We posit that this divergence is an artifact of the initialization schemes, with matrix A inclined toward capturing commonalities across domains, while matrix B adapts to domain-specific diversities. The distinction between matrix A and B offers valuable insights for enhancing both parameter efficiency and effectiveness. From an efficiency standpoint, our hypothesis suggests that the parameters of matrix A could potentially be shared across multiple heads, thereby reducing redundancy. Regarding effectiveness, since the parameters of matrix B of different heads

https://doi.org/10.52202/079017-0304

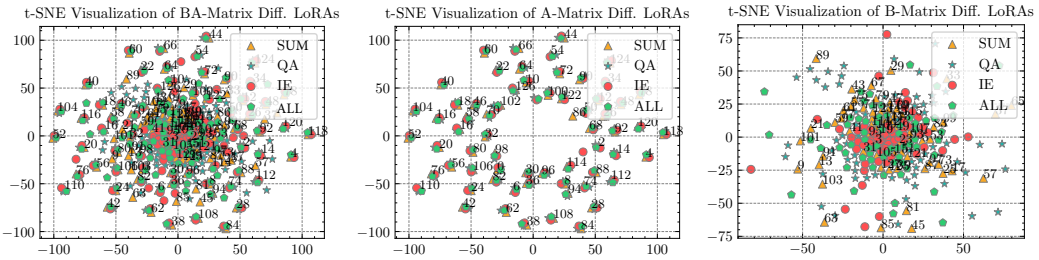


Figure 3: Breakdown analysis of LoRA modules. Compare fine-tuned LoRA modules of Dolly-15K [8] with three subtasks of Dolly-15K including "*summarization (Sum)*", "*closed QA (QA)*" and "*information extraction (IE)*" using t-SNE. Consider LLaMA2-7B (random seed=42), which contains 32 decoder layers, corresponding to 32 adaptive modules. Each module consists of {**0**: q_proj of A, **1**: q_proj of B, **2**: v_proj of A, **3**: v_proj of B} submodules. This makes a total of 32×4 submodules. Left displays all submodules. Center shows all even submodules, i.e. the A matrix. Right represents all odd submodules, i.e. the B matrix. It can be seen that the differences in the fine-tuned LoRA modules for different tasks arise mainly from the B matrix.

are dispersed, suggesting that using a single head to adapt to multiple domains might be less effective than using individual heads for each domain, which minimizes the interference between domains.

Building upon our observations, we propose an optimized LoRA architecture designed to enhance cost-effectiveness. In this architecture, we share the parameters of A matrix across various sub-domains or tasks to improve parameter efficiency, while deploying multiple B matrices, each tailored to handle different intrinsic components. This design allows for a more effective adaptation to the specific characteristics of each component. While these intrinsic components can be manually identified using prior knowledge of the training data, we also introduce end-to-end methods using Mixture-of-Experts (MoEs) [21], which will be detailed in the methodology section. This automatic approach facilitates flexibility and applicability, particularly in scenarios where prior knowledge is limited or unavailable.

3 *HydraLoRA*

In this section, we introduce the proposed *HydraLoRA*, an asymmetric LoRA architecture for efficient fine-tuning, as illustrated in Figure 1. After that, we show the workflow of *HydraLoRA* as Figure 4.

3.1 Asymmetric LoRA architecture

The LoRA method updates two low-rank matrices A and B, and uses AB as the change of a pretrained and frozen weight W_0 of a linear layer as shown in Eq. 1. The integral parameters are fine-tuned for the whole corpus in the original LoRA, which causes difficulty in learning the various knowledge aspects. Drawing from a detailed breakdown analysis of LoRA, a potential solution is to segment the entire LoRA into "Hydra" structured LoRA variants, that is, characterized by a central shared matrix A and several distinct matrices B, fostering a blend of shared knowledge and specialized functionalities. As Figure 1, *HydraLoRA* is to fine-tune LoRAs to achieve robust performance without redundancy, thereby benefiting the entire heterogeneous corpus. The asymmetric LoRA architecture can be formulated as:

$$\begin{aligned} W &= W_0 + \Delta W \\ &= W_0 + \sum_{i=1}^{N} \omega_i \cdot B_i A \end{aligned} \tag{2}$$

The matrics $B_i \in \mathbb{R}^{d \times r}$ and shared $A \in \mathbb{R}^{r \times k}$. The hyper-parameter N denotes the number of B matrices. The term ω_i modulates these contribution weights for head B_i.

3.2 Workflow of *HydraLoRA*

Figure 4 illustrates the workflow of *HydraLoRA*. Initially, *HydraLoRA* delves into the adaptive identification and initialization of LoRA modules within a heterogeneous corpus, aligning them with

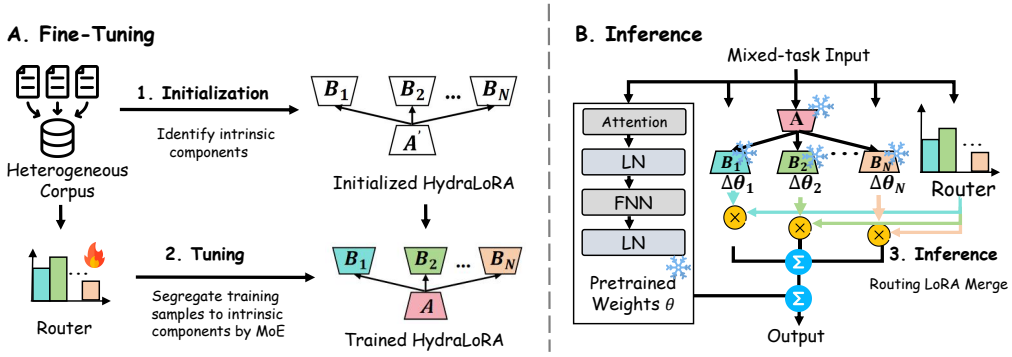


Figure 4: Architecture and workflow of *HydraLoRA*. During the fine-tuning stage, *HydraLoRA* first adaptively identifies and initializes N of intrinsic components without specific domain knowledge. It then employs a trainable MoE router that treats each intrinsic component as an expert to automatically segregate training samples into intrinsic components for fine-tuning. During the inference stage, *HydraLoRA* merges multiple B matrices flexibly and dynamically through a trained router.

task relevance through the application of k-means or developer-specified size. Subsequently, we propose a Mixture-of-Experts (MoE) framework that handles B matrices as expert adapters to ensure computational efficiency throughout the fine-tuning (Section 3.2.1) and inference (Section 3.2.2) stages by freezing the rest of the LLM parameters. During inference, it flexibly and dynamically merges multiple B matrices through the MoE router.

3.2.1 Fine-tuning

Motivated by Mixture-of-Experts (MoEs; [20, 40]), where experts are selectively activated by a gating mechanism (Router) in response to different inputs. In *HydraLoRA*, we substitute each expert with a lightweight LoRA adapter. During fine-tuning, while weights of LLMs remain frozen, the experts and router layers are trained from scratch. In order to achieve a unified approach to the distinct forward processes of multiple B matrices, we define a set of experts, denoted as $(E_1, \ldots, E_N)$, to learn the updated matrix ΔW. As *HydraLoRA* fine-tunes the experts using the heterogeneous corpus, the shared matrix A inherently captures collaborative knowledge to augment intra-gains, and different matrices B foster knowledge modularity to mitigate fine-tuning inter-offsets. Based on this structure, the forward process of *HydraLoRA* is expressed as:

$$y = W_0 x + \sum_{i=1}^{N} \omega_i E_i A x \quad (MoE) \tag{3}$$

where N denotes the number of experts, i.e., B matrices.To regulate these contributions, we introduce a gate function (router network) commonly consisting of a dense layer with trainable weights (transformation matrix) $W_g \in \mathbb{R}^{r \times N}$ followed by a softmax function which takes an intermediate token representation x as input and combines the output of each expert based on the gating scores $(\omega_1, \ldots, \omega_N)$:

$$\omega_i = \text{softmax}(W_g^T x) \quad (Router) \tag{4}$$

3.2.2 Inference

During inference, *HydraLoRA* merges adapters by enabling routing computation based on the input. Specifically, since matrices B operate as linear functions, we initially compute a weighted average of the experts. Following this, we apply a PEFT transformation using the combined expertise. The *HydraLoRA* significantly enhances training efficiency through an extremely parameter-efficient MoE formulation. Additionally, the intrinsic structural modularity of *HydraLoRA* facilitates rapid recovery and merging of the trained parameters during inference, leading to substantial memory savings.

https://doi.org/10.52202/079017-0304

Table 2: Comparative performance of different tuning schemes across multiple benchmarks on a single domain. 8-shot for GSM8K, zero-shot for others. #$\bar{B}$ refers to the average B matrix number.

Schemes	MMLU	Medical	Law	HumanEval		GSM8K	%Param	#A	#$\bar{B}$
				P@1	P@10				
LLaMA2-7B [48]	38.88	35.98	33.51	13.10	20.34	10.38	-	-	-
Full Fine-Tuning	49.91	46.78	46.08	20.24	32.93	25.70	100	-	-
Prompt Tuning [24]	39.91	37.59	35.02	13.66	21.55	13.18	0.001	-	-
P-Tuning$_{(256)}$ [29]	41.11	39.81	36.72	13.60	21.13	15.56	0.193	-	-
Prefix Tuning [25]	41.78	40.28	36.54	13.23	22.56	16.89	0.077	-	-
$(IA)^3$ [27]	40.45	37.12	35.25	13.54	23.17	13.98	0.009	-	-
AdaLoRA$_{(r=8)}$ [55]	44.32	42.83	39.36	14.81	23.78	19.51	0.093	1	1
LoRA$_{(r=8)}$ [18]	43.22	41.59	37.85	15.67	22.95	18.24	0.062	1	1
LoRA$_{(r=16)}$	45.45	43.10	39.64	16.71	25.60	20.32	0.124	1	1
LoRA$_{(r=32)}$	46.59	44.32	40.81	17.12	25.89	20.67	0.248	1	1
LoRA-Split$_{(4\times8)}$	46.94	45.28	41.35	18.20	26.85	21.92	0.248	4	4
HydraLoRA$_{(r=8)}$	**47.22**	**45.71**	**42.18**	**18.31**	**27.43**	**22.27**	**0.124**	1	3

4 Experiments

In this section, we detail the principal experiments. We begin with an overview of the experimental setup and implementation intricacies. Following this, we share our findings and offer a succinct interpretation.

4.1 Experiment Setting

Dataset and Benchmarks To explore the properties and commonalities of the LoRA asymmetric structure, we conduct experiments on both single and multiple domains to evaluate the effectiveness of *HydraLoRA* for profiling intrinsic components. • Single domain. 1) *General*: we fine-tune with the general instruction tuning databricks-dolly-15k [8] for generic language capability and evaluate with MMLU [16]. 2) *Medical*: we fine-tune with GenMedGPT and clinic-10k from ChatDoctor [26] for medicine applications and evaluate medical tasks in MMLU. 3) *Law*: we fine-tune with two legal instruction tuning datasets Lawyer-Instruct [1] and US-Terms [4] then evaluate with law tasks in MMLU. 4) *Math*: we fine-tune with the training split of GSM8K [7] for mathematical reasoning and evaluate with test set of GSM8K. 5) *Code*: we fine-tune with CodeAlpaca [5] for code generation and evaluate with HumanEval [6]. • Multi-task domain. We select a portion of the Flanv2 [51] datasets covering Natural Language Understanding (NLU) and Natural Language Generation (NLG), which can be grouped into 10 distinct task clusters. Then we evaluate it with the Big-Bench Hard (BBH) [45] benchmark. A detailed description of the benchmarks can be found in Appendix A.1.

Baselines • First, we compare *HydraLoRA* with different PEFT methods on single datasets: 1) *Full fine-tuning*; 2) *Prompt Tuning* [24]; 3) *P-Tuning* [29]; 4) *Prefix Tuning* [25]; 5) *IA*^{3} [27]; 6) *AdaLoRA* [55]. • Second, we extend the experiments exploring *HydraLoRA* on multiple datasets compared with more weighted average methods: 1) Lorahub [19] employs black-box optimization to learn weights of 20 randomly selected LoRAs for new tasks, using weighted averaging without needing gradient calculations. 2) LoRA MoE [52] combines lightweight experts (LoRA) with MoE architecture for high efficiency, generalizing to new tasks without prior knowledge. A detailed description of the baseline models can be found in Appendix A.2.

4.2 Overall Performance

The experimental results of *HydraLoRA* and the competing baselines are presented in Table 2 with a single domain and Table 3 with the mixed task domain. The evaluation of diverse tasks demonstrates that *HydraLoRA* consistently outperforms all other schemes. The performances rooted in LoRA outperform those of conventional PEFT methodologies. Compared to the default single LoRA configuration (rank=8), the Hydra architecture, enriched by the integration of several B matrices, effectively addresses the inherent conflicts among intrinsic components of the corpus. Furthermore, with equivalent parameters (rank=16), the model shows superior performance, confirming the ef-

Table 3: Comparative performance of different tuning schemes, including base model (Base), LoRA tuning (LoRA), LoraHub learning, multi-LoRA tuning with MoE inference (LoRA MoE) and our proposed *HydraLoRA* learning across mix-task domain on the BBH benchmark with LLaMA2-7B, LLaMA2-13B as the base LLM (3-shot).

Metrics		**Base** [48]	**LoRA** [18]	**Lorahub** [19]	**LoRA MoE** [52]	***HydraLoRA***
Performance	7B	31.6	36.8	39.7	40.3	**41.5**
	13B	38.4	40.1	41.9	43.7	**44.1**
# of A/B for training		0/0	1/1	48/48	48/48	1/10
# of A/B for inference		0/0	1/1	20/20	48/48	1/10
% Params		-	0.062	1.240	2.976	0.341

fectiveness of the adopted parameters. Based on Table 2 and Table 3, we propose three research questions that confirm the aforementioned observations.

RQ1: Is it more effective to use multiple smaller LoRA heads for specific tasks rather than one single LoRA for the entire domain dataset, given the same parameter count? Comparing the high-dimensional LoRA configuration with $r = 32$ against a segmented version using LoRA-Split, a variant introduced by HydraLoRA, which divides the model into four distinct components each with $r = 8$. That is, multiple vanilla LoRAs are directly utilized to capture the differences between data. We observe a noteworthy trend in the performance across a variety of tasks as detailed in Table 2. It illustrates the superior performance of LoRA-Split in comparison to the traditional LoRA approach, across all the evaluated scenarios. This enhancement in performance is a strong indication of the detrimental impact that task interference can have on the training process. By segregating the tasks into discrete components, LoRA-Split effectively minimizes the conflict and interference between tasks, thereby promoting a more efficient and focused environment.

The concept of LoRA-Split hinges on the construction of different intrinsic component compositions, employing LoRA as a foundational technique to strategically mitigate the interference conflict. This architectural innovation has proven to be a pivotal factor in enhancing model performance. However, it's important to note that while LoRA-Split marks a significant advancement in model efficiency and task handling, it also introduces a certain level of parameter redundancy. The segmented approach of LoRA-Split inevitably leads to an increase in the overall model parameters, which can be manifold in comparison to the traditional, singular LoRA model. This increase in parameters, while contributing to the model's robustness and capability to handle multiple tasks simultaneously, also poses new challenges in terms of computational resources and model optimization.

RQ2: Will multiple LoRA heads, individually trained on different data, improve efficiency by distinguishing matrix B parameters? We evaluated the Hydra structure LoRA — *HydraLoRA* that is characterized by a shared LoRA A matrix, while maintaining distinct B matrices that are trained separately. This configuration was meticulously compared with both the standard LoRA and the LoRA-Split approaches, emphasizing efficiency parameters.

According to the results presented in Table 2, unlike split which straightforwardly adopts multiple vanilla LoRAs, *HydraLoRA* adopts an asymmetric LoRA structure that not only improves parameter efficiency by separating the uses of A matrix for commonalities and B matrices for diversities with a notably smaller adapter parameter set, but also employs a trainable router to improve the composition of multiple B matrices that outperforms the LoRA-Split approach. This finding is significant as it suggests that *HydraLoRA* not only enhances performance efficiency but also boosts overall system effectiveness. This may be driven by 1) different B matrices capturing different features of the data-intrinsic knowledge, mitigating mutual interferences, and avoiding performance offsets. 2) Module A maintains the collaborative knowledge by taking the strengths of each and integrating them to improve the model performance.

RQ3: How does *HydraLoRA* fare against other merge methods in complex, multi-task domains, considering scalability and robustness? While we hypothesize that the asymmetry is mainly rooted in the different initialization methods of A and B matrices, it is possible that this behavior varies on different model architectures and datasets. Recent work confirms similar empirical observations [54, 13]. To the best of our ability, we extended the experiments exploring *HydraLoRA* on multiple

https://doi.org/10.52202/079017-0304

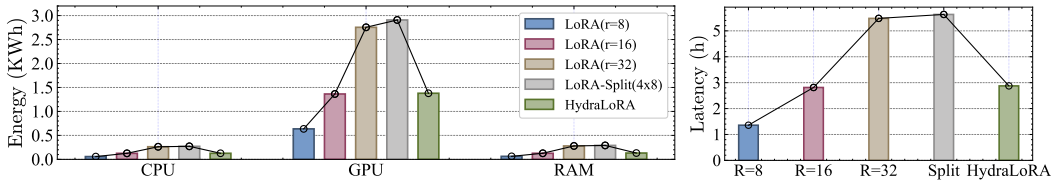


Figure 5: Energy consumption and latency during fine-tuning with different LoRA approaches (fine-tuning LLaMA2-7B with GSM-8K).

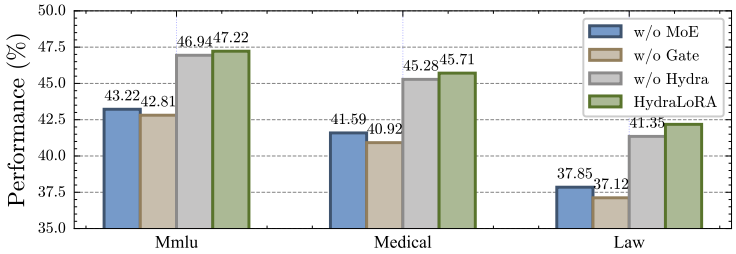


Figure 6: Comparative performance of ablation study for *HydraLoRA* across multiple benchmarks.

datasets. LoRA MoE and their variants typically aim at tackling multi-tasks by employing multiple independent LoRAs. This makes them suitable for handling various domains. However, for a single dataset like ours, a "default" MoE method might not be optimal. HydraLoRA addresses this by constructing asymmetric structures and utilizing multiple B matrices to capture the specific nuances within the single dataset. The effectiveness of this approach is demonstrated by the experimental results in Table 3.

4.3 Energy and Throughput Analysis

RQ4: How does the "Hydra" structure in *HydraLoRA* enhance system efficiency, particularly in reducing training energy consumption and latency? We evaluate the system efficiency of *HydraLoRA* from two perspectives: training energy consumption and latency. The following experiments were executed on a GPU infrastructure consisting of 4 NVIDIA A40 GPUs and a CPU powered by an Intel(R) Xeon(R) Gold 6330 CPU clocked at 2.00GHz. Power consumption measurements were recorded using CodeCarbon [34]. Figure 5 shows the results of various fine-tuning approaches for GSM-8K using the LLaMA2-7B model. we can see that *HydraLoRA* effectively speeds up the training process $1.96\times$ and reduces 49.6% energy cost compared to LoRA (rank=32). While the energy consumption and latency of LoRA-Split exceeds the LoRA (rank=32). This is for the reason that *HydraLoRA* jointly considers inherent knowledge modularity and collaboration, which utilizes the "Hydra" structure with a shared A matrix and different B matrix. In this way, it only employs rank=16 training overhead but expands to a performance enhancement of more than rank=32. Overall, this experiment demonstrates the parameter effectiveness of *HydraLoRA*.

4.4 Ablation Study

RQ6: What impact do the MoE architecture and the gate function have on the fine-tuning process? To delve deeper into understanding the contributions of each component in *HydraLoRA*. we present the results of our ablation study in Figure 6. The variant *w/o* MoE (essentially reverts to LoRA) excludes the MoE architecture. Similarly, the *w/o* gate variant employs uniform expert weights bypassing the gate function. The *w/o* hydra adopts multiple vanilla LoRAs in a straightforward way. Figure 6 indicates that the full *HydraLoRA* model outperforms its variants, showing that both the MoE architecture and gate function significantly contribute to its effectiveness across various language understanding domains.

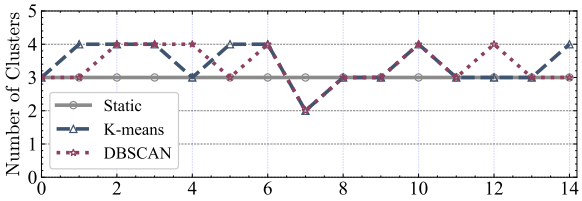


Figure 7: Number of clusters generated by different approaches including developer-specific (static), k-means, and DBSCAN.

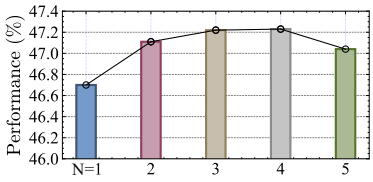


Figure 8: The results of experiments for hyper-parameters number of clusters.

4.5 Hyper-parameter Analysis

RQ7: How do the number of intrinsic component of *HydraLoRA* influence performance outcomes? As Figure 8 shown, we conduct a comprehensive and meticulous analysis by fine-tuning the Dolly-15K model on the LLaMA2-7B dataset and subsequently evaluating its performance on the MMLU benchmark to rigorously examine the impact of variations in the intrinsic component, symbolized by the variable N, on the model's overall performance. *Empirically we find that the number N of clusters is not a sensitive parameter for HydraLoRA*, with a wide range of reasonable number N of clusters (e.g. 2 to 4) performing decently well in all settings in our experiments. Specifically, the performance loss of N = 3 vs. the optimal N = 4 is only 0.42%. Meanwhile, as illustrated in Figure 7, we employ three distinct methods to generate the number of corpus clusters 15-fold, and the results demonstrate that the k-means [30] yields comparable outcomes with DBSCAN [39]. Therefore, based on this observation, we choose k-means because it is simple but effective, more sophisticated hyperparameter search approaches (e.g. DBSCAN, parameter sweep and Bayesian optimization) will be unnecessarily costly. It's noteworthy that *HydraLoRA* is adeptly designed to orchestrate its components in a way that it can automatically calibrate and navigate toward the optimal performance configuration across various parameters. This intelligent auto-tuning is achieved through the application of the k-means clustering algorithm. This strategic component orchestration not only enhances performance but also ensures a more efficient and effective utilization of resources, underpinning the model's capability to adapt and perform efficiently in a dynamic computational environment.

5 Related work

Parameter-Efficient Fine-tuning LLMs are becoming increasingly powerful, but fine-tuning them often requires significant computational resources. This has spurred research on parameter-efficient fine-tuning (PEFT) techniques that reduce memory and storage costs during adaptation. One prominent PEFT approach is adapters [17, 37]. It introduces new, trainable dense layers within the existing model, keeping the original parameters frozen. This concept has proven successful across various domains [35, 42, 43, 56]. Further improvements on adapter compactness involve constructing parameter matrices using Kronecker products of low-rank matrices [31]. Another PEFT strategy directly manipulates activations with learned vectors. This can be achieved through concatenation [29, 25, 24], multiplication (IA^3; [27]), or addition (BitFit; [53]). Prefix-tuning [25] and prompt-tuning [24] are noteworthy examples that fine-tune continuous prompts instead of designing discrete ones [9]. Interestingly, a study suggests that many PEFT methods can be viewed as a form of adapter, providing a unified perspective [14]. Beyond adding new parameters or altering the computational graph, researchers also explore sparse [12, 44, 46] or low-rank updates (LoRA; [18]).

Multi-LoRA Architecture LoRA has notably garnered increasing interest recently, becoming a standard approach for adapting LLMs such as LLaMA [47, 48] under limited computational resources. Recognizing its potential, researchers have delved deeper, exploring the benefits of employing multiple LoRAs. LoraHub [19] takes this multi-LoRA approach by training several adapters and strategically picking combinations based on the domain during inference. Meanwhile, MultiLoRA [50] focuses on horizontal scaling, aiming to reduce LoRA's parameter dependence. This involves splitting LoRA modules along the rank dimension and introducing learnable scaling factors for enhanced expressiveness. Addressing scaling challenges from a different angle, the mixture of LoRA concept is further proposed [52]. This mitigates resource consumption when

https://doi.org/10.52202/079017-0304

scaling instruction-tuned LLMs. Recognizing the potential for conflict during instruction tuning, LoRAMoE [11] leverages the Mixture-of-Experts (MoEs; [20]) structure to safeguard the pre-trained LLM's knowledge from excessive corruption by instruction data. Similarly, MOELoRA [28] incorporates a MoE framework into LLMs, thereby improving their multitasking capabilities in the medical domain. Shifting the focus to the system perspective, S-LoRA [41] provides a framework for efficiently serving multiple LoRA adapters. Unlike previous methods that relied on choosing LoRA combinations based on their training domains, *HydraLoRA* breaks free from the dependence on domain knowledge during inference. Additionally, *HydraLoRA*'s asymmetric structure further enhances parameter efficiency compared to existing symmetric approaches.

6 Conclusion

In this work, we start by conducting exploratory experiments applying the LoRA technique to LLaMA2, aiming to adapt it to a new domain across various tasks. This study unveils the limitations of employing a single LoRA for the entire domain, highlighting the detrimental effects of domain interference. In response, we introduce a novel architecture *HydraLoRA* that features an asymmetric structure with a shared matrix for all samples and distinct matrices for each intrinsic component. This design improves domain adaptation by selectively focusing on distinct components, enhancing both fine-tuning and inference efficiency. Our research highlights the importance of balancing learning capabilities for diverse datasets against the need for a lean model, offering a viable pathway for improving LLMs with minimal parameter growth. More discussion about limitation and broader impacts are available in Appendix D and E.

7 Acknowledgments

This research received support from MYRG-GRG2023-00211-IOTSC-UMDF and the Start-up Research Grant of the University of Macau (SRG2022-00010-IOTSC). Chunlin Tian and Zhan Shi contributed equally to this work. For correspondence, please contact Dr. Li Li (llili@um.edu.mo) or Dr. ChengZhong Xu (czxu@um.edu.mo).

References

[1] Alignment-Lab-AI. Lawyer-instruct, 2024.

[2] Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. Slora: Federated parameter efficient fine-tuning of language models. *CoRR*, abs/2308.06522, 2023.

[3] 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, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[4] Ilias Chalkidis, Nicolas Garneau, Catalina Goanta, Daniel Katz, and Anders Søgaard. LeXFiles and LegalLAMA: Facilitating English multinational legal language model development. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15513–15535, Toronto, Canada, July 2023. Association for Computational Linguistics.

[5] Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`, 2023.

[6] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul

Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

[8] Mike Conover, Matt Hayes, Ankit Mathur, Xiangrui Meng, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, et al. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

[9] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 3369–3391. Association for Computational Linguistics, 2022.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[11] Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *CoRR*, abs/2312.09979, 2023.

[12] Demi Guo, Alexander M. Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4884–4896. Association for Computational Linguistics, 2021.

[13] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[14] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034. IEEE Computer Society, 2015.

[16] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *CoRR*, abs/2009.03300, 2020.

 https://doi.org/10.52202/079017-0304

[17] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019.

[18] 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. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[19] Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition. *CoRR*, abs/2307.13269, 2023.

[20] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Comput.*, 3(1):79–87, 1991.

[21] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Comput.*, 6(2):181–214, 1994.

[22] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[23] David J Ketchen and Christopher L Shook. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, 17(6):441–458, 1996.

[24] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics, 2021.

[25] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics, 2021.

[26] Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6), 2023.

[27] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[28] Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *CoRR*, abs/2310.18339, 2023.

[29] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *CoRR*, abs/2103.10385, 2021.

[30] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982.

[31] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1022–1035, 2021.

[32] OpenAI. ChatGPT, 2022.

[33] OpenAI. GPT-4 Technical Report. *CoRR*, abs/2303.08774, 2023.

[34] David A. Patterson, Joseph Gonzalez, Quoc V. Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *CoRR*, abs/2104.10350, 2021.

[35] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 487–503. Association for Computational Linguistics, 2021.

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[37] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 506–516, 2017.

[38] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.*, 24(5):513–523, 1988.

[39] Erich Schubert, Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Trans. Database Syst.*, 42(3):19:1–19:21, 2017.

[40] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[41] Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. S-lora: Serving thousands of concurrent lora adapters. *CoRR*, abs/2311.03285, 2023.

[42] Asa Cooper Stickland and Iain Murray. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR, 2019.

[43] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. VL-ADAPTER: parameter-efficient transfer learning for vision-and-language tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5217–5227. IEEE, 2022.

[44] Yi-Lin Sung, Varun Nair, and Colin Raffel. Training neural networks with fixed sparse masks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 24193–24205, 2021.

 https://doi.org/10.52202/079017-0304

[45] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics, 2023.

[46] Chunlin Tian, Xinpeng Qin, and Li Li. Greenllm: Towards efficient large language model via energy-aware pruning. In *32nd IEEE/ACM International Symposium on Quality of Service, IWQoS 2024, Guangzhou, China, June 19-21, 2024*, pages 1–2. IEEE, 2024.

[47] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[48] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

[49] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[50] Yiming Wang, Yu Lin, Xiaodong Zeng, and Guannan Zhang. Multilora: Democratizing lora for better multi-task learning. *CoRR*, abs/2311.11501, 2023.

[51] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[52] Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermis, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *CoRR*, abs/2309.05444, 2023.

[53] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1–9. Association for Computational Linguistics, 2022.

[54] Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning. *CoRR*, abs/2308.03303, 2023.

[55] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[56] Han Zhou, Xingchen Wan, Ivan Vulic, and Anna Korhonen. Autopeft: Automatic configuration search for parameter-efficient fine-tuning. *CoRR*, abs/2301.12132, 2023.

A Datasets and Baselines

A.1 Datasets

Single Domain

1. **General**: we fine-tune with the general instruction tuning databricks-dolly-15k for generic language capability and evaluate with MMLU.
2. **Medical**: we fine-tune with GenMedGPT and clinic-10k from ChatDoctor for medicine applications and evaluate medical tasks in MMLU including three related tasks: "clinical knowledge", "professional medicine" and "college medicine".
3. **Law**: we fine-tune with two legal instruction tuning datasets Lawyer-Instruct and US-Terms then evaluate with law tasks in MMLU including two related tasks: "professional law" and "international law".
4. **Math**: we fine-tune with the training split of GSM8K for mathematical reasoning and evaluate with test set of GSM8K.
5. **Code**: we fine-tune with CodeAlpaca for code generation and evaluate with HumanEval.

Multi-task Domain As well for complex mixed multi-task/domain, we select a portion of the Flanv2 datasets covering Natural Language Understanding (NLU) and Natural Language Generation (NLG), which can be grouped into 10 distinct task clusters. Then we evaluate it with the Big-Bench Hard (BBH) benchmark.

We summarize the details of the used datasets as follows:

1. **Struct-to-Text Conversion**: This task evaluates the capability to generate natural language descriptions from structured data inputs. We use the following datasets: (1) CommonGen; (2) DART; (3) E2ENLG; (4) WebNLG;
2. **Translation**: Translation involves converting text from one language to another, maintaining the original meaning and nuances. We use the following datasets: (1) En-Fr from WMT'14; EnDe, En-Tr, En-Ru, En-Fi, En-Ro from WMT'16; (3) En-Es from Paracrawl.
3. **Commonsense Reasoning**: This involves assessing the ability to apply physical or scientific principles alongside common sense in reasoning tasks. We use the following datasets: (1) COPA, (2) HellaSwag, (3) PiQA, and (4) StoryCloze.
4. **Sentiment Analysis**: A fundamental task in natural language processing (NLP) that determines the sentiment polarity (positive or negative) of a given text. We use the following datasets: (1) IMDB, (2) Sentiment140, (3) SST-2, and (4) Yelp. information sources. We use the following datasets: (1) ARC, (2) NQ, and (3) TriviaQA.
5. **Paraphrase Detection**: This task requires models to ascertain whether two sentences convey the same meaning, indicating semantic equivalence. We use the following datasets: (1) MRPC, (2) QQP, and (3) Paws Wiki.
6. **Coreference Resolution**: Involves identifying instances within a text that refer to the same entity, demonstrating an understanding of textual context. We use the following datasets: (1) DPR and (2) WSC273.
7. **Reading comprehension**: Assesses the capability to derive answers to questions from a provided text containing relevant information. We use the following datasets: (1) BoolQ, (2) DROP, (3) MultiRC, (4) OBQA, (5) SQuADv1, (6) SQuADv2.
8. **Reading Comprehension with Commonsense**: Merges traditional reading comprehension skills with commonsense reasoning, requiring understanding beyond the explicit text. We use the following datasets: (1) CosmosQA; (2) ReCoRD.
9. **Natural Language Inference**: Focuses on deducing the relationship between two sentences, determining if the second sentence logically follows from, contradicts, or is unrelated to the first sentence. We use the following datasets: (1) ANLI, (2) CB; (3) MNLI; (4) QNLI; (5) SNLI; (6) WNLI; (7) RTE.
10. **Closed-Book Question Answering**: This task challenges models to answer questions about general knowledge without direct access to external

https://doi.org/10.52202/079017-0304

A.2 Baselines

PEFT methods

1. **Full Fine-tuning** is the default strategy for adaptation. During fine-tuning, the model is initialized with pretrained weights and biases, and all model parameters undergo gradient updates.
2. **Prompt Tuning** adds task-specific prompts to the input, and these prompt parameters are updated independently of the pretrained model parameters which are frozen.
3. **P-Tuning** adds trainable prompt embeddings to the input that is optimized by a prompt encoder to find a better prompt, eliminating the need to manually design prompts. The prompt tokens can be added anywhere in the input sequence, and P-Tuning also introduces anchor tokens for improving performance.
4. **Prefix Tuning** prefixes a series of task-specific vectors to the input sequence that can be learned while keeping the pretrained model frozen. The prefix parameters are inserted in all of the model layers.
5. **IA3** enhances efficiency by infusing learned vectors into transformer architectures, drastically cutting trainable parameters while preserving performance and minimizing inference latency.
6. **AdaLoRA** is a method for optimizing the number of trainable parameters to assign to weight matrices and layers, unlike LoRA, which distributes parameters evenly across all modules. More parameters are budgeted for important weight matrices and layers while less important ones receive fewer parameters.

Multiple LoRA weighted average methods

1. **LoRA MoE**. A collection of n parameterized experts, denoted as $E_1, ..., E_n$, is orchestrated by a router network R. This network features a dense layer with adjustable weights W_g from $\mathbb{R}^{d_m \times n}$. A softmax function then processes an intermediate token representation x, yielding gating scores $s_1, ..., s_n$ that determine the weighted contribution of each expert's output:

$$s_i = R(x)_i = \text{softmax}(W_g^T x) \quad \text{(Router)} \tag{5}$$

   Subsequently, the overall output y is synthesized by aggregating the experts' outputs, each modulated by its respective gating score:

$$y = \sum_{i=1}^{n} s_i \cdot E_i(x) \quad \text{(MoE)} \tag{6}$$

   This results in a dynamic allocation of the model's capacity, enabling specialized processing by experts as directed by the router's gating mechanism.
2. **LoraHub** aggregates 20 LoRAs at random for new downstream tasks. To master the weight of each LoRA, it utilizes a black-box optimization technique, bypassing the need for gradient calculations of the large model. This process involves weighted averaging at the parameter level. Mirroring the MoE training approach, we select 20 random samples for each task, creating a cohesive training dataset optimized through this black-box method.

B Initialization via k-means

In the case of considering heterogeneous corpora, it is crucial to select the appropriate number N of matrix B, to ensure consistent performance and minimize unnecessary computational overhead. This choice is usually closely related to the training corpus. In this work, we propose initializing *HydraLoRA* modules via k-means [30] algorithm for adaptive initialization. Specifically, k-means is utilized to process the heterogeneous corpus to identify the best-fit taxonomy of the corpus, i.e., the optimal N. First, we extract key features from the corpus by applying the Term Frequency-Inverse Document Frequency (TF-IDF; [38]) algorithm and transform the textual information into numerical

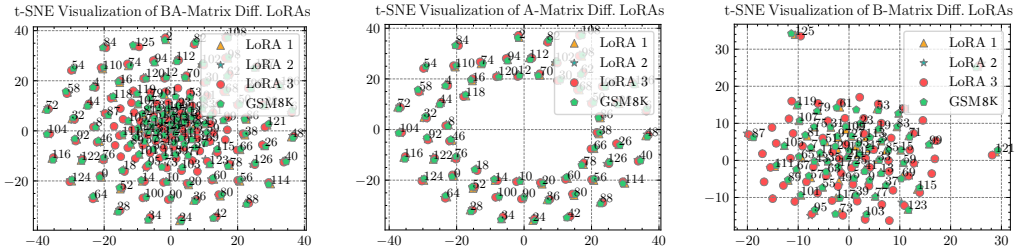
feature vectors. We integrate the elbow method [23] to determine the optimal value of N. Initially, N cluster centers are randomly selected for preliminary clustering as Eq. 7, followed by updating the cluster centers to accurately reflect the data within each cluster as Eq. 8. where C_j is the cluster center to which data point X_i is assigned and $d(\cdot, \cdot)$ is the Euclidean distance function. S_j is the set of data points in the j-th cluster.

$$C_j = \underset{C_j}{\operatorname{argmin}}\, d(X_i, C_j) \tag{7}$$

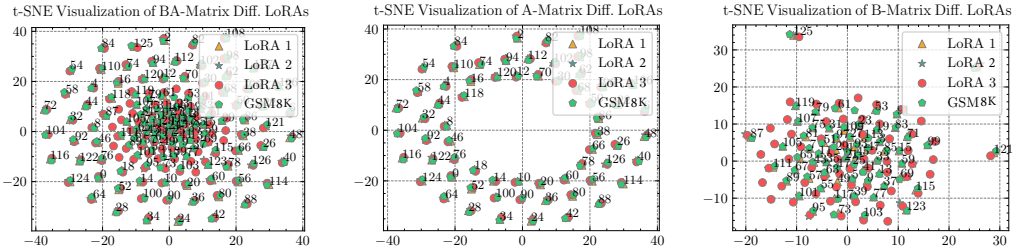
$$C_j = \frac{1}{|S_j|} \sum_{X_i \in S_j} X_i \tag{8}$$

By analyzing the relationship between the sum of squares of errors (SSE) and different N values, we observe that SSE decreases as N increases. Identifying the elbow point on the SSE curve — where the rate of decrease in SSE slows down — is crucial. The elbow point represents the optimal N value, beyond which increasing the number of clusters does not significantly enhance performance, thereby achieving an ideal balance between model complexity and performance.

C LoRA Breakdown



(a) Compare fine-tuned LoRA modules of GSM8K [7] with its subsets using T-SNE. We employ the Independent and Identically Distributed (IID) segmentation scheme to divide GSM8K into three subsets and fine-tune them using different LoRAs.



(b) Specific tasks.

Figure 9: Breakdown analysis of LoRA modules. Compare fine-tuned LoRA modules of GSM-8K [7] with its subsets using T-SNE. We employ the Independent and Identically Distributed (IID) segmentation scheme to divide GSM8K into three subsets and fine-tune them using different LoRAs. Consider LLaMA2-7B (random seed=42), which contains 32 decoder layers, corresponding to 32 adaptive modules. Each module consists of {0: q_proj_A, 1: q_proj_B, 2: v_proj_A, 3: v_proj_B} submodules. This makes a total of 32×4 submodules. (a,b) left displays all submodules. (a,b) center shows all even submodules, i.e. the A-matrix. (a,b) right represents all odd submodules, i.e. the B-matrix. It can be seen that the differences in the fine-tuned LoRA modules for different tasks arise mainly from the B matrix.

D Limitation

In this study, we concentrate on well-established PEFT techniques, particularly LoRA. However, we have not yet investigated other configurations, such as prompt-tuning and adapters, which are

https://doi.org/10.52202/079017-0304

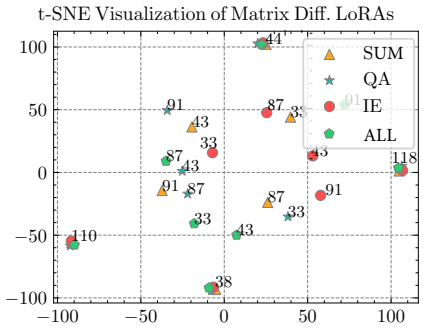


Figure 10: Breakdown analysis of LoRA modules (Dolly-15K) with its subsets using T-SNE on different layer.

reserved for future exploration. Our current evaluation is limited to fine-tuning, while an examination of the efficacy of these techniques during the pre-training phase remains a subject for subsequent research. Another potential limitation stems from the training data itself. In multi-task scenarios, extreme conditions—such as contaminated or adversarial data—can significantly impair performance due to aggregation issues. The heterogeneity across tasks, including variations in language, task type, and domain, may reduce the effectiveness of shared knowledge, thereby diminishing its overall impact. This challenge is not exclusive to HydraLoRA but is common to all multi-task frameworks. Addressing these concerns may involve implementing robustness-enhancing measures (e.g., data sanitization, robust aggregation, and anomaly detection) and integrating privacy-preserving technologies, such as homomorphic encryption, differential privacy, and blockchain.

E Broader Impacts

Positive Societal Impacts The proposed *HydraLoRA* framework, with its asymmetric structure and parameter-efficient fine-tuning approach, has the potential to make LLMs more accessible and efficient. This could democratize AI, enabling more researchers, developers, and organizations to leverage the power of LLMs for various applications, ultimately driving innovation and progress. Moreover, by effectively addressing the challenge of domain or task interference, *HydraLoRA* could significantly enhance the performance of LLMs in complex, multi-task domains. This could lead to more accurate and reliable AI-powered tools and services in areas like healthcare, education, and finance, ultimately improving the quality of life for many people. Lastly, the parameter-efficient approach of *HydraLoRA* could help reduce the computational resources required for training and fine-tuning LLMs, thereby lessening the environmental impact of AI.

Negative Societal Impacts As with any AI technology, there are potential negative societal impacts to consider. The risk of misuse is a significant concern, as *HydraLoRA* could be used for malicious purposes, such as creating more sophisticated and convincing deepfakes or spreading misinformation and propaganda. Additionally, the increased efficiency and accessibility of AI brought about by *HydraLoRA* could lead to job displacement in certain sectors, as AI-powered tools and services become capable of performing tasks traditionally done by humans. Lastly, the use of LLMs in various applications could potentially lead to privacy and security issues, especially if these models are used to process or generate sensitive information. The proposed *HydraLoRA* framework, while not directly related to these issues, could inadvertently contribute to them by making it easier to deploy LLMs in various applications.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the abstract and introduction sections, we articulate the motivation behind *HydraLoRA*, highlight its differences from previous work, and outline the contributions of this paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in the Appendix D.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide a detailed explanation of the parameter usage and will release the source code to ensure reproducibility.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide a detailed explanation of parameter usage and will release the source code.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The full details are provided in the appendix and supplemental material (code).

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Experimental results are tested multiple times to ensure stability and reliability.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

https://doi.org/10.52202/079017-0304

Justification: Compute resources are thoroughly described and evaluated in both the experimental setup and the experimental results sections.

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in this paper fully conforms to the NeurIPS Code of Ethics in every respect.

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the potential positive societal impacts and negative societal impacts of the work performed in Appendix E.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original paper that produced the code package or dataset.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: This paper relates the details of the code as part of the submission.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.