HydraOpt: Navigating the Efficiency-Performance Trade-off of Adapter Merging

Taha Ceritli[1], Ondrej Bohdal[1], Mete Ozay[1],
Jijoong Moon[2], Kyeng-Hun Lee[2], Hyeonmok Ko[2], Umberto Michieli[1]
[1]Samsung R&D Institute UK, United Kingdom, [2]Samsung Research, South Korea
Correspondence: t.ceritli@samsung.com

Abstract

Large language models (LLMs) often leverage adapters, such as low-rank-based adapters, to achieve strong performance on downstream tasks. However, storing a separate adapter for each task significantly increases memory requirements, posing a challenge for resource-constrained environments such as mobile devices. Although model merging techniques can reduce storage costs, they typically result in substantial performance degradation. In this work, we introduce HydraOpt, a new model merging technique that capitalizes on the inherent similarities between the matrices of low-rank adapters. Unlike existing methods that produce a fixed trade-off between storage size and performance, HydraOpt allows us to navigate this spectrum of efficiency and performance. Our experiments show that HydraOpt significantly reduces storage size (48% reduction) compared to storing all adapters, while achieving competitive performance (0.2-1.8% drop). Furthermore, it outperforms existing merging techniques in terms of performance at the same or slightly worse storage efficiency.

1 Introduction

Large language models (LLMs) have become a driving force behind many natural language processing tasks today, including text summarization (Liu et al., 2024b), smart-reply (Bastola et al., 2023) and question-answering (Sticha et al., 2024). While modern LLMs are pre-trained to perform a diverse set of tasks, their performance on specific tasks can be improved by updating its parameters on task-specific datasets. However, this fine-tuning process becomes computationally impractical due to the growing size of LLMs, especially for resource-constrained environments such as mobile devices.

One approach to reduce the computational complexity of the fine-tuning process is parameter-efficient fine-tuning (PEFT) (Hu et al., 2022; Xu et al., 2023; Lialin et al., 2023), where only a small set of parameters is updated while keeping the parameters of the LLMs frozen. For example, low-rank-based adapters such as LoRA (Hu et al., 2022) and VeRA (Kopiczko et al., 2024) have facilitated the use of LLMs for on-device applications, as one can store separate adapters for different tasks and switch to the corresponding parameters when the user wishes to perform a specific task (Gunter et al., 2024). However, storing separate adapters becomes costly for on-device settings where the memory is limited. Model merging techniques (Wortsman et al., 2022; Ilharco et al., 2023; Yadav et al., 2024; Yu et al., 2024) address this issue by combining multiple adapters into one adapter used for all tasks. However, such techniques significantly disrupt the performance with no control.

In this work, we propose a new model merging method (HydraOpt) that allows controllable efficiency-performance trade-off, unlike existing methods that result in a single storage size and

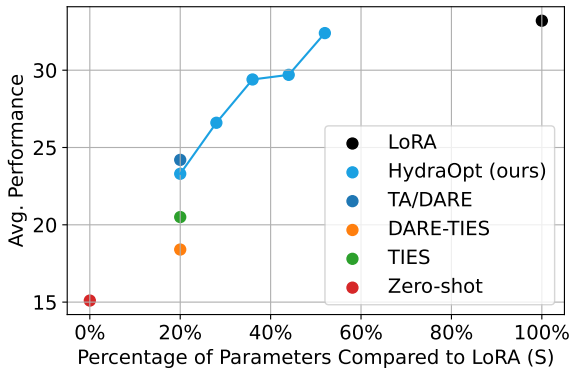

Figure 1: **Performance and storage efficiency trade-off.** Average performance over 5 applications and 8 languages. Existing merging techniques reduce storage costs at significant performance drops. Our method performs similarly at the same efficiency level and improves if more storage is available, achieving performance similar to LoRAs.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 26887–26909
November 4-9, 2025

performance. HydraOpt achieves competitive performance compared to storing all adapters but with reduced storage size and improves over the performance of existing merging techniques at a slightly worse storage efficiency (Fig. 1). Our contributions are three-fold:

- Building on the similarity between low-rank-based adapters, we introduce a new model merging strategy called HydraOpt that can navigate the efficiency-performance trade-off of model merging.
- We design a comprehensive evaluation framework that consists of 40 tasks derived from 5 applications and 8 languages. We conduct experiments to assess the impact of merging adapters across applications, languages, and tasks.
- Our experiments demonstrate that HydraOpt finds a better trade-off between efficiency and performance across different low-rank-based adapters and LLMs. While maintaining comparable performance to existing model merging methods at similar storage levels, HydraOpt consistently outperforms them when a modest increase in storage is permitted.

2 Related Work

Parameter-efficient Fine-tuning (PEFT) techniques adapt models efficiently via training relatively few parameters, making them especially suitable for fine-tuning large language models (Ding et al., 2022; Han et al., 2024). Low-rank-based adapters (Hu et al., 2022; Liu et al., 2024a; Kopiczko et al., 2024; Malinovsky et al., 2024; Ceritli et al., 2024), in particular, have become widely adopted, having small additional storage requirements thanks to their compact size, which makes them suitable for deployment to mobile devices (Gunter et al., 2024). LoRA (Hu et al., 2022) introduces two low-dimensional trainable parameters $A \in \mathbb{R}^{r\times k}$ and $B \in \mathbb{R}^{d\times r}$ which are used to approximate the weight updates ΔW, *i.e.*, $\Delta W = BA$ where rank $r << \min(d, k)$. Then, the LLM parameters can be updated such that $W = W + \Delta W$. Performance of LoRA has been further improved in its many extensions, such as AdaLoRA (Zhang et al., 2023) and DoRA (Liu et al., 2024a).

Various approaches to improve efficiency of LoRA have also been proposed (Kopiczko et al., 2024; Renduchintala et al., 2024). In particular, VeRA (Kopiczko et al., 2024) has become popular for improving storage and parameter efficiency of LoRA, while maintaining competitive performance. VeRA introduces the following model update: $\Delta W = \Lambda_b B \Lambda_d A$ where the parameters $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ are shared across the layers while the parameters $\Lambda_b \in \mathbb{R}^d$ and $\Lambda_d \in \mathbb{R}^r$ are defined per each layer. The resulting method reduces the number of trainable parameters, as the layer-specific parameters are defined as vectors rather than matrices.

Model Merging: Multiple task-specific models can be combined into a single model capable of multi-tasking via a process called model merging. Task Arithmetic (Wortsman et al., 2022; Ilharco et al., 2023) represents the simplest option, and it combines the weights of multiple models as a weighted average. Various more advanced techniques have been developed, including TIES (Yadav et al., 2024) and DARE (Yu et al., 2024). TIES first resets the values of parameters that changed little, then elects the sign in case of conflicts, and merges only sign-aligned parameters. DARE drops part of the weight changes and then rescales the remaining ones accordingly. Other methods (Xiao et al., 2024; Huang et al., 2024; Hammoud et al., 2024; Shenaj et al., 2025) use data to improve merging, however, that is beyond the scope of the present work.

On-device LLMs: LLMs typically include billions of parameters, which requires significant resources, such as high-end GPUs, even for inference only (Borzunov et al., 2024). However, in many use cases, it is desirable to perform computations locally without transferring data to remote servers (Dhar et al., 2021), for example, when using sensitive data stored on resource-constrained devices. Real-world examples include generating personalized replies or summarizing private conversations, where maintaining data privacy is paramount. As a solution, smaller LLMs (*e.g.*, 1–3 billion parameters) have been developed for on-device deployment. These models utilize model compression strategies paired with a smaller size to support efficient on-device inference. Prominent examples include Llama 3.2 1B (Dubey et al., 2024), StableLM2 1.6B (Bellagente et al., 2024) and Qwen2.5 1.5B (Yang et al., 2024; Qwen Team, 2024). Due to their relatively small size, it is stan-

dard practice to include single-task adapters on the device to enable the small LLMs to perform the individual tasks, instead of relying on instruction following (Gunter et al., 2024; Dong et al., 2024).

3 Proposed Method

3.1 Motivation

Our work stems from the analysis of the asymmetric behaviour of low-rank adaptation matrices. Zhu et al. (2024) demonstrate that the B parameters in LoRA exhibit distinct values when fine-tuned across different tasks, while the A parameters remain relatively similar when initialized identically, despite being fine-tuned on diverse tasks. Similarly, Tian et al. (2024) observe that when multiple LoRA adapters are trained on separate datasets, the A parameters tend to converge to similar values, whereas the B parameters become more differentiated.

We observe similar patterns when fine-tuning Llama-3.2-3B-Instruct on five distinct text generation tasks in English. Fig. 2 illustrates the similarity between these LoRA adapters computed via Mean Absolute Error. The plot indicates that the A parameters are more similar to each other compared to the B parameters. We report in Fig. 10 an analysis using Canonical Correlation Analysis (CCA), as in Zhu et al. (2024). We further confirm these results with t-SNE visualizations in Fig. 11 following the approach of Tian et al. (2024).

These findings suggest that the A parameters capture cross-domain commonalities, while the B parameters adapt to task-specific knowledge. This behavior may stem from the initialization schemes for low-rank-based adapters, where B is typically initialized as a zero matrix, while A is sampled from a Gaussian distribution. These behaviors can also be observed in VeRA as Λ_b is initialized as zero vector and Λ_d is sampled from a Gaussian distributions.

3.2 Our Method: HydraOpt

We propose HydraOpt for merging a set of low-rank-based adapters parameters (*e.g.* LoRA) $\{B_i, A_i\}_{i=1}^{K}$. As shown in Fig. 3, HydraOpt approximates the given set of parameters by learning a shared A' parameter and a set of B' parameters $\{B'_i\}_{i=1}^{M}$. The approximation to the original model updates is driven by the following loss function:

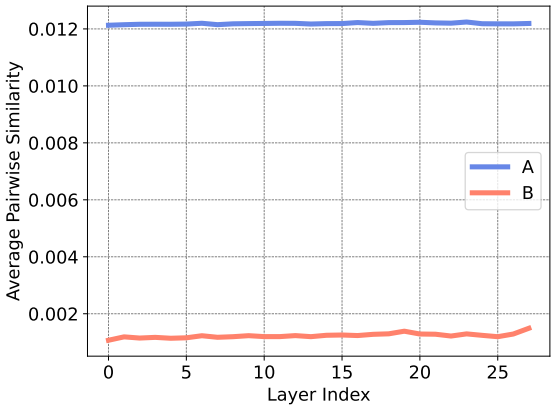

Figure 2: **Similarity between A and B matrices of LoRAs** measured using Mean Absolute Error on query matrices of Llama-3.2-3B-Instruct fine-tuned on 5 applications in English.

$$\ell = \sum_{i=1}^{K} f\left(B_i A_i, \sum_{j=1}^{M} \sigma(\mathbf{C}'_i/T)(j) B'_j A'\right), \quad (1)$$

where f is a distance function that measures the similarity between two model updates $\Delta W_i := B_i A_i$ and $\Delta W'_j := B'_j A'$. Here, σ denotes the softmax function with the temperature term T, and $\mathbf{C}'_i \in \mathbb{R}^M$ is a trainable vector of coefficients with the coefficient $C'_{i,j}$ representing how likely it is to use B'_j for the i^{th} task. The softmax function approximates categorical one-hot encoded vectors for small values, hence guiding the model to use mostly one B'_j parameter for a given task.

Given the sparsity of adapter parameters, we choose Mean Absolute Error as the distance function f to induce sparsity (Bach et al., 2012). We then calculate the gradients of the objective function Eq. (1) to update the parameters $A', \{B'_i\}_{i=1}^{M}, \{\mathbf{C}'_i\}_{i=1}^{K}$ using an iterative optimization algorithm. For instance, the update rule for A' using Gradient Descent becomes $A' = A' - \eta \nabla \ell$ where η is the learning rate and ℓ is the loss.

The coefficients $\mathbf{C}'_i$ are initialized using a Gaussian distribution and updated during training. We remark that this only brings a minimal increase in memory footprint during training, since the number of coefficients is much smaller than the number of LoRA parameters. Moreover, these coefficients are discarded once the training is over, after we associate each task with a B' parameter. Therefore, the inference stage is unaltered.

HydraOpt allows us to walk the efficiency-

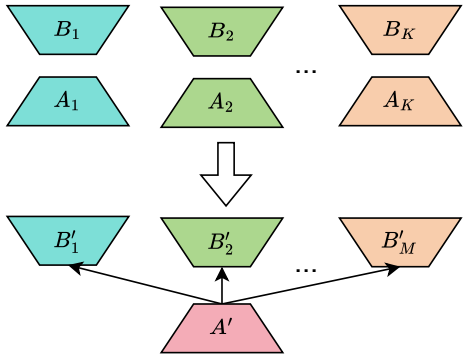

Figure 3: **An overview of HydraOpt.** We approximate K sets of LoRA parameters by learning a shared A' parameter and a set of task-specific parameters $\{B'_i\}_{i=1}^{M}$.

performance trade-off of model merging. In the most aggressive parameter sharing scheme, HydraOpt constructs one set of LoRA parameters $\{B', A'\}$, which causes performance drops due to the reduced flexibility of the approximation similarly to existing model merging techniques. However, by adjusting the level of parameter sharing, we obtain both efficient and accurate model merging solutions.

In the special case of $K = M$, we omit the coefficients $\mathbf{C}'_i$ by learning a separate B' parameter for each task. Specifically, we modify the loss function in Eq. (1) to be:

$$\ell = \sum_{i=1}^{K} f(B_i A_i, B'_i A'). \tag{2}$$

The reduction in parameter size using HydraOpt depends on the number of LoRAs to merge and the size of LoRA parameters. For one layer of an LLM, the number of parameters required by LoRA becomes $K \times r \times (d + k)$ for K tasks, whereas HydraOpt requires $M \times r \times d + r \times k$ parameters where M denotes the number of B' parameters. Assuming that the A and B parameters are the same size, the total number of parameters reduces to 60% when merging 5 pairs of LoRA parameters (Fig. 12 in Appendix A.3). The reduction rate asymptotically reaches 50% as the number of LoRAs increases, while it can be greater than 50% if the A is larger than B.

Notice that the objective functions in Eq. (1)-(2) do not utilize any external task-specific samples. Instead, they treat the given set of LoRA parameters $\{B_i, A_i\}_{i=1}^{K}$ as the target pseudo-labels

Algorithm 1 HydraOpt

Require: Adapter parameters $\{A_i, B_i\}_{i=1}^{K}$, target number of B' parameters M, number of epochs E, temperature T, optimizer

Initialize $A', \{B'_i\}_{i=1}^{M}, \{\mathbf{C}'_i\}_{i=1}^{K}$
for e in E **do**
  if $K \neq M$ **then**
    Calculate the loss ℓ using Eq. (1)
  else
    Calculate the loss ℓ using Eq. (2)
  Parameter update using the optimizer:
    $A' \leftarrow \arg\min_{A'} \ell$
    $\{B'_i\}_{i=1}^{M} \leftarrow \arg\min_{\{B'_i\}_{i=1}^{M}} \ell$
    $\{\mathbf{C}'_i\}_{i=1}^{K} \leftarrow \arg\min_{\{\mathbf{C}'_i\}_{i=1}^{K}} \ell$
Use A' for all tasks
for i-th task in $1, 2, ..., K$ **do**
  if $K \neq M$ **then**
    $j \leftarrow \arg\max_{j \in \{1,2,...,M\}} C'_{i,j}$
    Use B'_j rather than B_i
  else
    Use B'_i rather than B_i

and updates the trainable HydraOpt parameters $A', \{B'_i\}_{i=1}^{M}, \{\mathbf{C}'_i\}_{i=1}^{K}$ during the training. Therefore, our technique is classified as a data-free model merging method. Algorithm 1 provides a brief description of HydraOpt.

Note that HydraOpt differs from HydraLoRA (Tian et al., 2024) in two main aspects. First, HydraLoRA simultaneously trains multiple adapters for multiple tasks where the A parameters are shared across tasks. In contrast, we tackle the problem of merging LoRA adapters that have been trained independently for individual tasks. Secondly, HydraOpt allows choosing the number of B' parameters lower than the number of tasks to navigate the performance-efficiency trade-off, whereas HydraLoRA learns a separate B' parameter for each task, leading to a static and predefined trade-off choice.

4 Experiments

In this section, we describe our experimental setup (Sec. 4.1) and discuss the main results (Sec. 4.2), the ablation studies (Sec. 4.3) and empirical computational complexity analysis (Sec. 4.4).

4.1 Setup

Tasks: We conduct experiments on 40 downstream tasks in total, each of which is a text generation application in a specific language. We tackle 5 applications. (i) Grammar Correction (GC): to generate the correct form of a given input containing grammar errors. (ii) Smart Reply (SR): to generate a response to a given textual message. (iii) Text Summarization (TS): to generate a shorter version of a given sentence. (iv) Tone Adjustment (TA): to re-write a given text in a specific style. (iv) Question Answering (QA): to answer a given question. Each application is considered in 8 different languages: EN, DE, ES, FR, IT, JA, KO, ZH[1].

Datasets: We use Cambridge English Write & Improve (W&I) (Bryant et al., 2019) for GC, Persona-Chat Synthetic (Jandaghi et al., 2024) for SR, SAMSum (Gliwa et al., 2019) for TS, Sound Natural (Einolghozati et al., 2020) rephrased using the fine-tuned RedPajama-INCITE-Base-3B-v1 model (Utsav, 2023) for TA, and SQuAD (Rajpurkar et al., 2016) for QA. As these datasets are collected in English, we utilize machine-translation for the remaining languages. Specifically, we use OPUS-MT (Tiedemann and Thottingal, 2020) for translation to French, German, Italian and Spanish, and M2M100 (Fan et al., 2021) for translation to Chinese, Japanese and Korean. However, the translation process fixes the grammar errors in the input text (as also mentioned in Luhtaru et al., 2024). Therefore, we used datasets collected in the original languages for Grammar Correction, namely EC-Spell (Lv et al., 2023) for Chinese, Merlin (Boyd et al., 2014) for Italian, and GitHub Typo Corpus (Hagiwara and Mita, 2020) for the remaining languages. Table 7 presents a summary of the employed datasets, including their links and number of samples. Table 8 lists the prompts that we have used for each case.

Evaluation Metrics: Following the literature, we report F05 (↑) (Bryant et al., 2017) for GC[2], Weighted Rouge (↑) for SR, RougeL (↑) for TS and TA, and F1 (↑) for QA. Given the page limit, we report the average of these metrics as aggregated overview when needed, with individual results reported in Appendix A.3 for completeness. Additionally, we report the storage (S, %, ↓) as the percentage of the parameters compared to storing all individual adapters.

Table 1: **Performance on 5 English applications using Llama-3.2-1B-Instruct LoRA-finetuned.** S represents the percentage of the parameters compared to storing 5 LoRAs.

Method	S (%)	GC	SR	TS	TA	QA	Avg
Zero-shot	0	13.1	5.1	23.4	27.6	15.8	17.0
LoRA	100	35.1	23.0	38.2	58.1	61.5	43.2
TA	20	25.9	11.4	32.3	51.7	28.7	30.0
TIES	20	25.1	13.2	31.1	51.7	26.9	29.6
DARE	20	21.6	7.4	27.1	39.0	19.7	23.0
DARE-TIES	20	22.5	7.9	27.4	44.0	20.9	24.5
HydraOpt(M=1)	20	26.3	9.0	32.3	50.4	27.6	29.1
HydraOpt(M=2)	28	26.9	17.6	31.7	53.0	28.2	31.5
HydraOpt(M=5)	52	33.1	21.7	37.1	57.1	59.9	41.8

Models: We use Llama-3.2-1B-Instruct (Grattafiori et al., 2024) for our main experiments due to its size suitable for on-device deployment. However, we also present analyses with different model sizes (2B, 3B, and 3.5B) and architectures (Gemma2, Gemma Team, 2024, and Phi-3, Abdin et al., 2024) to prove the generalization of our method.

Baseline Methods: For fair comparison, we compare our method with existing data-free model merging techniques such as Task Arithmetic (TA, Ilharco et al. 2023), TIES (Yadav et al., 2024), DARE (Yu et al., 2024), and DARE-TIES[3].

Implementation Details: We set the LoRA rank to 32, α to 128 and dropout to 0.05 throughout the experiments. We utilize the AdamW optimizer with a learning rate of 5e-5 and a batch size of 3. Please see Appendix A.1 for further details.

4.2 Main Results

Merging 5 LoRA adapters in a language: We first consider a typical model merging scenario where multiple LoRA adapters are obtained by fine-tuning the same model (Llama-3.2-1B-Instruct in this case) across different applications. Table 1 presents a comparison of the methods in terms of storage efficiency and performance. The highest-performing state-of-the-art method is TA that reduces average performance by 13.2% at the storage occupancy of 20%.

HydraOpt exhibits similar performance at the same storage and starts outperforming TA as the extent of storage size reduction is sacrificed. Specif-

[1]Abbreviated for English, German, Spanish, French, Italian, Japanese, Korean, Chinese, respectively

[2]We use ChERRANT (Zhang et al., 2022) for Chinese.

[3]See the dare_ties function at https://github.com/huggingface/peft/blob/main/src/peft/utils/merge_utils.py [Accessed on 17 May 2025]

Figure 4: **Average performance on 5 applications in different languages using Llama-3.2-1B-Instruct.** In this figure, we report the relative average score compared to LoRA.

ically, we obtain a 1.5% gain over TA in average performance when an additional 8% storage size is used. If more storage is available, we can approach the LoRA upper bound: namely, HydraOpt(M=5) is only 1.4% lower than LoRA in terms of average performance, while saving ∼50% storage.

We note that HydraOpt introduces additional runtime; however, the merging operation is still reasonably fast with relatively small GPU memory overhead (please see Table 6 for a comparison of the methods in terms of runtime and memory).

Merging 5 LoRA adapters (multiple languages): We performed the same 5-way merging experiment in multiple languages and we observe similar improvements across the board as shown in Fig. 4.

Different LLMs: Next, we test the generalization of our approach across different LLM sizes and architectures (Llama 1B, Llama 3B, Gemma 2B, Phi3.5B). In Table 2, we observe a consistent trend across model types and sizes.

HydraOpt(M=5) consistently improves average performance over the baselines at a small storage size cost.

Different low-rank-based adapter types: In this experiment, we demonstrate how HydraOpt can be extended to other low-rank-based adapters. In particular, we consider VeRA (Kopiczko et al., 2024) for its efficiency and competitive performance compared to LoRA.

Similarly to the application of HydraOpt to a set of LoRA parameters, we merge a set of VeRA parameters by learning a new set of parameters Λ'_b and Λ'_d with the latter shared across multiple tasks. Note that for simplicity, we discard the merging of the parameters A and B as they are initialized similarly across the tasks and kept frozen during fine-tuning.

Table 2: **Average performance on 5 English applications for different LoRA-finetuned LLMs.** We use Llama-3.2-1B-Instruct (L1B), Llama-3.2-3B-Instruct (L3B), Gemma-2-2B-it (G2B), and Phi-3.5-mini-instruct (P3.5B), and report the average of individual metrics for each model.

Method	S (%)	L1B	L3B	G2B	P3.5
Zero-shot	0	17.0	20.1	20.8	14.9
LoRA	100	43.2	47.8	47.9	45.4
TA	20	30.0	37.4	37.7	33.3
TIES	20	29.6	35.0	35.1	34.1
DARE	20	23.0	38.6	24.6	20.1
DARE-TIES	20	24.5	27.4	25.7	20.1
HydraOpt(M=1)	20	29.1	36.5	36.0	30.6
HydraOpt(M=2)	28	31.5	41.3	40.3	40.8
HydraOpt(M=5)	52	41.8	46.0	47.5	45.2

Table 3: **Average performance on 5 English applications using Llama-3.2-1B-Instruct VeRA-finetuned.** S represents the percentage of the parameters compared to storing 5 VeRAs.

Method	S (%)	Avg
Zero-shot	0.0	17.2
VeRA	100.0	39.0
TA	20.0	0.3
TIES	20.0	27.8
DARE	20.0	28.6
DARE-TIES	20.0	27.7
HydraOpt(M=1)	20.0	26.9
HydraOpt(M=2)	20.7	27.5
HydraOpt(M=3)	21.4	29.9
HydraOpt(M=4)	22.1	35.8
HydraOpt(M=5)	22.7	36.4

Table 3 presents the results. The best state-of-the-art approach is TIES in this case, while TA, which worked well on LoRA, achieves a significantly lower score here. Even though TIES performs better than our method at 20% storage efficiency, we remark once again that our approach allows us to achieve much higher performance at the minimal cost of 2.7% additional storage.

Evaluation across languages and merging setups: We extend our setup to multiple languages in Table 4. Firstly, we merge 5 LoRA adapters in each language and report the average performance across 8 languages (*application* block). Secondly, we apply merging across languages rather than applications, *i.e.*, merging 8 LoRA adapters for each application (*languages* block). Finally, we merge all 40 LoRA adapters, each of which corresponds to an application in a given language (*task* block). De-

Table 4: **Average performance on 40 tasks using Llama-3.2-1B-Instruct (L1B) and Llama-3.2-3B-Instruct (L3B) LoRA-finetuned.** S represents the percentage of parameters compared to storing all 40 LoRAs.

	Method	S (%)	L1B	L3B	Avg
	Zero-shot	0.0	12.3	15.1	13.7
	LoRA	100.0	28.1	33.2	30.7
applications	TA	20.0	19.3	24.1	21.7
applications	TIES	20.0	17.9	20.5	19.2
applications	DARE	20.0	15.0	24.2	19.6
applications	DARE-TIES	20.0	15.6	18.4	17.0
applications	HydraOpt(M=1)	20.0	18.9	23.3	21.1
applications	HydraOpt(M=2)	28.0	19.5	26.6	23.1
applications	HydraOpt(M=5)	52.0	26.9	32.4	29.6
languages	TA	12.5	24.6	29.1	26.9
languages	TIES	12.5	24.7	25.3	25.0
languages	DARE	12.5	15.8	19.0	17.4
languages	DARE-TIES	12.5	17.1	21.1	19.1
languages	HydraOpt(M=1)	12.5	23.9	27.4	25.6
languages	HydraOpt(M=3)	22.5	24.7	29.7	27.2
languages	HydraOpt(M=8)	47.5	26.6	32.1	29.4
tasks	TA	2.5	18.0	21.8	19.9
tasks	TIES	2.5	17.4	18.7	18.0
tasks	DARE	2.5	14.0	16.7	15.3
tasks	DARE-TIES	2.5	14.2	16.6	15.4
tasks	HydraOpt(M=1)	2.5	17.5	22.0	19.8
tasks	HydraOpt(M=40)	41.5	21.9	25.2	23.5

tailed individual results are reported in Appendix A.3.

The results highlight that our method is in line with existing state-of-the-art merging methods for the highest storage efficiency levels, however, it enables large accuracy gains when small additional storage is available.

Merging Across Applications: Our approach performs comparably to the best state-of-the-art method (TA), with only a 0.6% drop in performance. However, by utilizing 8% more storage, HydraOpt achieves a 1.4% performance gain over TA. Additionally, when a 48% reduction in storage size is acceptable, the average performance drop compared to the upper bound individual LoRA adapters can be reduced to 1.1%.

Merging Across Languages: In this setting, TA achieves the best performance at 12.5% storage efficiency, with an average performance drop of 3.8% compared to individual LoRA adapters. HydraOpt surpasses TA with just a 10% increase in storage, and the performance gap continues to widen as storage size increases. At 47.5% storage efficiency, HydraOpt incurs only a 1.3% drop compared to individual LoRA adapters.

Merging Across Tasks: In this most challenging setup, TA and HydraOpt exhibit similar performance at the most constrained storage efficiency of 2.5%. However, as storage size increases to 41.5%, our approach narrows the gap with individual LoRA adapters to as little as 7.2%.

Table 5: **Impact of LoRA rank on average performance for 5 English applications using Llama-3.2-1B-Instruct LoRA-finetuned**. We report the percentage of the parameters compared to storing all 5 LoRA parameters (denoted by S) and the average score.

Method	S (%)	$r = 8$	$r = 16$	$r = 32$
Zero-shot	0	17.0	17.0	17.0
LoRA	100	39.2	41.7	43.2
TA	20	28.4	28.9	30.0
TIES	20	28.1	29.1	29.6
DARE	20	20.7	21.8	23.0
DARE-TIES	20	22.4	23.8	24.5
HydraOpt(M=1)	20	27.1	28.3	29.1
HydraOpt(M=2)	28	30.0	31.0	31.5
HydraOpt(M=5)	52	38.5	40.0	41.8

4.3 Ablation Studies

We fine-tune Llama-3.2-1B-Instruct using the English data and perform several ablation studies when merging 5 LoRA adapters.

Impact of LoRA rank: First, we investigate whether the benefits of HydraOpt remain the same across the LoRA rank $r \in \{8, 16, 32\}$. As shown in Table 5, HydraOpt begins to improve over the performance of the leading competitor method when the storage is increased by 8%. Moreover, a 48% reduction in storage size leads to competitive performance with storing all individual LoRA adapters.

Impact of the distance function: Next, we analyze the choice of distance function f used for calculating the loss during training. In particular, we compare mean absolute error (MAE) with three alternative loss functions based on cosine similarity (CS), Frobenius norm (FRO), and mean squared error (MSE). Fig. 5 shows that MAE and FRO lead to better performance than CS and MSE. This result is not surprising as the optimization is done over the sparse LoRA parameters, which can be better learned using sparsity-inducing penalties (Bach et al., 2012).

Analysis of existing merging methods at higher storage sizes: We apply existing merging methods only on the A parameters for comparison with our method at a reduced storage efficiency level. The existing merging methods lead to a performance drop as shown in Fig. 6, which can be explained by the lack of adaptation on the B parameters after

Table 6: **Comparison of the methods in terms of runtime and GPU memory.** Merging is applied to 5 English applications using Llama-3.2-3B-Instruct LoRA-finetuned.

Metric	TA	TIES	DARE	DARE-TIES	HydraOpt(M=1)	HydraOpt(M=2)	HydraOpt(M=3)	HydraOpt(M=4)	HydraOpt(M=5)
Runtime (mins)	0.7	0.6	0.7	0.7	10.6	13.9	17.2	20.6	8.6
GPU (GB)	19.6	19.6	19.6	19.6	20.5	20.7	21.0	21.3	20.7

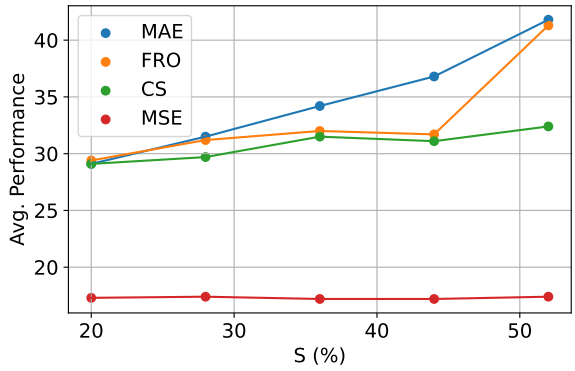

Figure 5: **Impact of distance function used during training.** We report average performance on 5 English applications using Llama-3.2-1B-Instruct LoRA-finetuned.

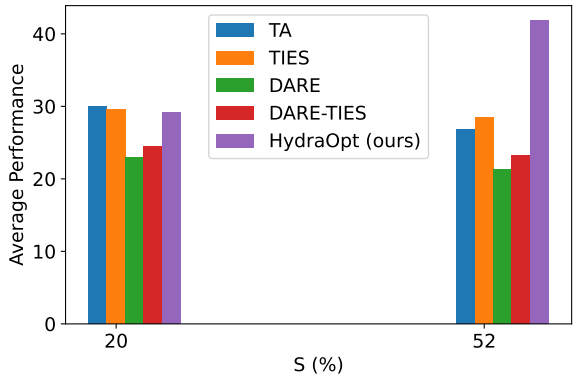

Figure 6: **Impact of storage efficiency level.** We report average performance on 5 English applications using Llama-3.2-1B-Instruct LoRA-finetuned.

obtaining the new shared A parameters. HydraOpt, on the other hand, iteratively adapts the B' parameters to the shared A' parameter to approximate the original model updates ΔW.

4.4 Runtime and GPU Memory Analysis

In this section, we analyze the time and GPU memory required to merge the LoRA parameters. Specifically, we use the LoRA parameters obtained by fine-tuning Llama-3.2-3B-Instruct using the English data. Table 6 compares the methods, which show that HydraOpt introduces additional runtime; however, the merging operation is still reasonably fast with relatively small GPU memory overhead. Given that the merging operation is going to be performed at the server side, HydraOpt provides a practical solution for real-world scenarios.

5 Conclusion

On-device applications of LLMs often leverage parameter-efficient fine-tuning methods, such as low-rank-based adapters, for downstream tasks. However, the need to deploy a separate adapter for each task results in substantial storage overhead, a critical challenge for resource-constrained environments such as mobile devices. While model merging techniques offer a potential solution by reducing the storage size, they often come at the cost of significant performance degradation on downstream tasks, making them impractical for real-world deployment.

In this work, we introduce HydraOpt, a new model merging technique that effectively addresses the trade-off. HydraOpt achieves competitive performance (0.2-1.8% drop) compared to storing all LoRA adapters, while significantly reducing parameter size (48% reduction). Furthermore, it consistently outperforms the performance of existing model merging methods at slightly worse efficiency levels. HydraOpt thus enables efficient storage utilization without compromising task-specific performance for deploying LLMs on-device.

Limitations

Despite the encouraging results obtained using HydraOpt, there are certain limitations in our current study that are worth acknowledging. For instance, this paper considers the generic case of data-free model merging where there is no assumption on the presence of data. Therefore, its performance is upper bounded by LoRA parameters. An interesting research direction may be considering data-driven merging scenarios, for which we expect similar gains in terms of efficiency and performance. Moreover, we tested HydraOpt with low-rank-based adapters such as LoRA and VeRA due to their popularity and efficiency-performance

trade-offs. Exploring its applications to other types of adapters would be a valuable direction for future work. Finally, we believe that our proposed method can be used in more general cross-modal and multi-modal tasks, which are getting more and more attention in the literature.

Potential Risks

This work investigates the efficiency-performance trade-off of serving LLMs for text generation applications on mass accessible devices. Even though our work can reduce the storage costs of LLMs, it does not change the inference complexity and its impact on the environment. Moreover, we did not evaluate how our method impacts LLMs in terms of fairness across different population subgroups, which needs to be verified using additional safeguarding tools.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, and 1 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, and 1 others. 2012. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106.

Ashish Bastola, Hao Wang, Judsen Hembree, Pooja Yadav, Zihao Gong, Emma Dixon, Abolfazl Razi, and Nathan McNeese. 2023. LLM-based smart-reply (LSR): Enhancing collaborative performance with ChatGPT-mediated smart reply system. *arXiv preprint arXiv:2306.11980*.

Marco Bellagente, Jonathan Tow, Dakota Mahan, Duy Phung, Maksym Zhuravinskyi, Reshinth Adithyan, James Baicoianu, Ben Brooks, Nathan Cooper, Ashish Datta, and 1 others. 2024. Stable LM 2 1.6 B technical report. *arXiv preprint arXiv:2402.17834*.

Alexander Borzunov, Max Ryabinin, Artem Chumachenko, Dmitry Baranchuk, Tim Dettmers, Younes Belkada, Pavel Samygin, and Colin A Raffel. 2024. Distributed inference and fine-tuning of large language models over the internet. In *NeurIPS*.

Adriane Boyd, Jirka Hana, Lionel Nicolas, Detmar Meurers, Katrin Wisniewski, Andrea Abel, Karin Schöne, Barbora Stindlová, and Chiara Vettori. 2014. The MERLIN corpus: Learner language and the CEFR. In *LREC*.

Christopher Bryant, Mariano Felice, Øistein E Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Workshop on innovative use of NLP for building educational applications*.

CJ Bryant, Mariano Felice, and Edward Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *ACL*.

Taha Ceritli, Savas Ozkan, Jeongwon Min, Eunchung Noh, Cho Jung Min, and Mete Ozay. 2024. A study of parameter efficient fine-tuning by learning to efficiently fine-tune. In *EMNLP Findings*, pages 15819–15836.

Sauptik Dhar, Junyao Guo, Jiayi (Jason) Liu, Samarth Tripathi, Unmesh Kurup, and Mohak Shah. 2021. A survey of on-device machine learning: An algorithms and learning theory perspective. *ACM Trans. Internet Things*, 2(3).

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, and 1 others. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. In *EMNLP*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Arash Einolghozati, Anchit Gupta, Keith Diedrick, and Sonal Gupta. 2020. Sound natural: Content rephrasing in dialog systems. In *EMNLP*.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, and 1 others. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48.

Gemma Gemma Team. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *ACL*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, and 1 others. 2024. Apple intelligence foundation language models. *arXiv preprint arXiv:2407.21075*.

Masato Hagiwara and Masato Mita. 2020. GitHub typo corpus: A large-scale multilingual dataset of misspellings and grammatical errors. In *LREC*.

Hasan Hammoud, Umberto Michieli, Fabio Pizzati, Philip Torr, Adel Bibi, Bernard Ghanem, and Mete Ozay. 2024. Model merging and safety alignment: One bad model spoils the bunch. In *EMNLP Findings*.

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

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *ICLR*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2024. Lorahub: Efficient cross-task generalization via dynamic lora composition. In *COLM*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *ICLR*.

Pegah Jandaghi, XiangHai Sheng, Xinyi Bai, Jay Pujara, and Hakim Sidahmed. 2024. Faithful persona-based conversational dataset generation with large language models. In *NLP4ConvAI*.

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

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

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

Yixin Liu, Kejian Shi, Katherine S He, Longtian Ye, Alexander R Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. 2024b. On learning to summarize with large language models as references. In *NAACL*.

Agnes Luhtaru, Elizaveta Korotkova, and Mark Fishel. 2024. No error left behind: Multilingual grammatical error correction with pre-trained translation models. In *EACL*.

Qi Lv, Ziqiang Cao, Lei Geng, Chunhui Ai, Xu Yan, and Guohong Fu. 2023. General and domain-adaptive chinese spelling check with error-consistent pretraining. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(5):1–18.

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

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. 2024. Tied-LoRA: Enhancing parameter efficiency of LoRA with weight tying. In *NAACL*.

Donald Shenaj, Ondrej Bohdal, Mete Ozay, Pietro Zanuttigh, and Umberto Michieli. 2025. Lora.rar: Learning to merge loras via hypernetworks for subject-style conditioned image generation. In *ICCV*.

Abigail Sticha, Norbert Braunschweiler, Rama Sanand Doddipatla, and Kate M Knill. 2024. Advancing faithfulness of large language models in goal-oriented dialogue question answering. In *ACM Conference on Conversational User Interfaces*.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. 2024. HydraLoRA: An Asymmetric LoRA Architecture for Efficient Fine-Tuning. In *NeurIPS*.

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT — Building open translation services for the World. In *EAMT*.

Kumar Utsav. 2023. RedPajama-INCITE-Base-3B-v1 model finetuned for Paraphrasing and Changing the Tone. `https://huggingface.co/llm-toys`.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and 1 others. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Xingrun Xing. 2024. LM-Cocktail: Resilient tuning of language models via model merging. In *ACL Findings*.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. 2023. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment. *arXiv preprint arXiv:2312.12148*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. TIES-merging: Resolving interference when merging models. In *NeurIPS*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *ICML*.

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

Yue Zhang, Zhenghua Li, Zuyi Bao, Jiacheng Li, Bo Zhang, Chen Li, Fei Huang, and Min Zhang. 2022. MuCGEC: a multi-reference multi-source evaluation dataset for chinese grammatical error correction. In *NAACL*.

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. 2024. Asymmetry in low-rank adapters of foundation models. In *ICML*.

A Appendix

We provide information about the datasets in Sec. A.1, report analyses about the similarity between LoRA parameters in Sec. A.2, and give more detailed results in Sec. A.3.

A.1 Implementation Details

Table 7 provides a summary of the datasets used in our experiments and additional information about our implementation. Table 8 describes the prompts that we have used. For tone adjustment, we consider four tones, namely professional, casual, witty and (neutral) paraphrasing, which are combined to create a larger dataset. We utilize NVIDIA A40 for our experiments. Adapters are applied to the query, key, value, and output matrices for Llama-3.2-3B-Instruct, Llama-3.2-1B-Instruct, and Phi-3.5-mini-instruct, and to the query and value for Gemma-2-2B-it. Uniform merging coefficients are used for TA and DARE, whereas unary coefficients are used for TIES and DARE-TIES. We provide an implementation of our method in Figures 7, 8, 9.

A.2 Similarities Between LoRA Parameters

We perform Canonical Correlation Analysis (CCA) goodness of fit following Zhu et al. (2024) using LoRA matrices A and B obtained by fine-tuning Llama-3.2-3B-Instruct on English data. Fig. 10 indicates that A parameters result in higher similarity scores than B parameters. Similarly, Fig. 11 illustrates a similar trend using t-SNE plots where A parameters tend to converge to similar values.

A.3 Additional Results

Fig. 12 illustrates how the number of parameters to store changes using HydraOpt. We also present detailed evaluations for Llama-3.2-3B-Instruct when merging across applications (Tables 9-10), languages (Tables 11-12) and tasks (Table 13). Moreover, the average scores are reported in Table 14. Similarly, we present the detailed results under different ranks for Llama-3.2-3B-Instruct in Table 15, and the results for Gemma-2-2B-it LoRA-finetuned (Table 16) and for Phi-3.5-mini-instruct LoRA-finetuned (Table 17). Finally, we investigated the impact of the machine-translated prompts on the performance. Specifically, we used native speakers to make the prompts more natural (see Table 18) and repeated a subset of the experiments (Table 19).

```python
class HydraOpt(nn.Module):
    def __init__(
        self,
        lora_parameters,
        lora_names,
        M,
        T,
    ) -> None:
        super().__init__()

        self.K = len(lora_names)
        self.M = M
        self.T = T

        # trainable parameters
        self.A_prime = nn.ModuleDict({})
        self.B_primes = nn.ModuleDict({})
        self.C_primes = None if self.K == self.M else nn.ParameterDict({})

        keys = lora_parameters[lora_names[0]].keys()
        for i in range(M):
            peft_model_id = lora_names[i]
            for key in keys:
                # '.' can't be used in module_dict
                key_ = key.replace(".", "_")

                # initialize the shared A_prime
                if i == 0 and "lora_A" in key:
                    r, in_features = lora_parameters[peft_model_id][key].shape
                    self.A_prime[key_] = nn.Linear(in_features, r, bias=False)
                    nn.init.kaiming_uniform_(
                            self.A_prime[key_].weight, a=math.sqrt(5)
                        )

                # initialize the B_primes
                if "lora_B" in key:
                    out_features, r = lora_parameters[peft_model_id][key].shape
                    self.B_primes[f"{key_}_{i}"] = nn.Linear(
                        r, out_features, bias=False
                    )
                    nn.init.zeros_(self.B_primes[f"{key_}_{i}"].weight)

        # define coefficients if needed
        if self.K != self.M:
            for i in range(self.K):
                for key in keys:
                    key_ = key.replace(".", "_")

                    if "lora_B" in key:
                        self.C_primes[f"{key_}_{i}"] = nn.Parameter(
                            nn.functional.softmax(
                                torch.randn(M) / T, dim=-1
                                )
                            )

    def forward(self):
        return self.A_prime, self.B_primes, self.C
```

Figure 7: **Implementation of HydraOpt module.**

Table 7: **Dataset statistics.** Information about the datasets.

Task	Dataset	Language	# Training Samples	# Validation Samples	# Test Samples
Grammar Error Correction	ECSpell	Chinese	6,680	750	750
	Cambridge English Write & Improve (W&I)	English	23,523	2,526	2,639
	Merlin	Italian	572	79	81
	GitHub Typo Corpus	French	616	240	227
		German	412	119	132
		Japanese	1,043	325	321
		Korean	255	75	93
		Spanish	348	137	116
Smart Reply	Persona-Chat Synthetic	English	225,061	25,847	24,479
Text Summarization	SAMSum	English	14,732	818	819
Tone Adjustment	Sound Natural	All	2,245	321	642
Question Answering	SQuAD	English	65,699	21,900	10,570

```python
def hydra_loss(
    A_prime, B_primes, C_primes, M, lora_parameters, lora_names, T
):

    keys = list(lora_parameters[lora_names[0]].keys())
    K = len(lora_names)
    J = len(keys)

    for i, peft_model_id in enumerate(lora_names):
        for j in range(J):
            if j % 2 == 0:
                A_key = keys[j]
                B_key = keys[j + 1]
                A_key_ = A_key.replace(".", "_")
                B_key_ = B_key.replace(".", "_")

                if "lora" not in A_key or "lora" not in B_key:
                    continue

                # calculate the original model update
                delta_W = lora_parameters[peft_model_id][B_key] @ lora_parameters[
                    peft_model_id][A_key]

                if K == M:
                    delta_W_prime = (
                        B_primes[f"{B_key_}_{i}"].weight @ A_prime[A_key_].weight
                    )
                else:
                    delta_W_prime = torch.stack(
                        [
                            nn.functional.softmax(
                                C_primes[f"{B_key_}_{i}"] * T, dim=-1
                                )[l]
                            * B_primes[f"{B_key_}_{l}"].weight
                            @ A_prime[A_key_].weight
                            for l in range(M)
                        ]
                    ).sum(dim=0)
                if i == 0 and j == 0:
                    total_loss = nn.functional.l1_loss(delta_W, delta_W_prime)
                else:
                    total_loss += nn.functional.l1_loss(delta_W, delta_W_prime)

    return total_loss / (K * J)
```

Figure 8: **Implementation of HydraOpt loss function.**

```python
def hydraopt_merging(lora_parameters, lora_names, M, lr=0.01, epochs=1000, T=5.0):

    hydraopt = HydraOpt(lora_parameters, lora_names, M=M, T=T)
    hydraopt = hydraopt.to("cuda:0")

    keys = list(lora_parameters[lora_names[0]].keys())

    # move each pair of LoRA parameters to GPU
    for i, peft_model_id in enumerate(lora_names):
        for j in range(len(keys)):
            if j % 2 == 0:
                A_key = keys[j]
                B_key = keys[j + 1]
                if "lora" not in A_key or "lora" not in B_key:
                    continue
                else:
                    lora_parameters[peft_model_id][B_key] = lora_parameters[
                        peft_model_id][B_key].to("cuda:0")
                    lora_parameters[peft_model_id][A_key] = lora_parameters[
                        peft_model_id][A_key].to("cuda:0")


    optimizer = torch.optim.AdamW(hydraopt.parameters(), lr=lr)
    for epoch in range(epochs):
        optimizer.zero_grad()

        # get the current parameters
        A_prime, B_primes, C_primes = hydraopt()

        # calculate the loss based on the distance to the original LoRA updates
        loss = hydra_loss(
            A_prime,
            B_primes,
            C_primes,
            M,
            lora_parameters,
            lora_names
        )

        # backward pass
        loss.backward()

        # update weights
        optimizer.step()

    # obtain the mapping between the lora_Bs and B_primes parameters
    B_mapping = {}
    for task_index in range(len(lora_names)):
        B_mapping[task_index] = {}
        for key in keys:
            key_ = key.replace(".", "_")
            if "lora_B" in key:
                if hydraopt.K == hydraopt.M:
                    selected_B_prime = task_index
                else:
                    selected_B_prime = nn.functional.softmax(
                        hydraopt.C_primes[f"{key_}_{task_index}"] / T, dim=-1
                        ).argmax()
                B_mapping[task_index][key] = selected_B_prime

    return A_prime, B_primes, B_mapping
```

Figure 9: **Implementation of HydraOpt merging.**

Problem Type	Language	Prompt
Grammar Error Correction	English	Remove all grammatical errors from this text:
	Spanish	Quita todos los errores gramaticales de este texto:
	French	Supprimez tous les erreurs grammaticales de ce texte:
	German	Verbessere alle grammatischen Fehler in diesem Text:
	Italian	Rimuovi tutti gli errori grammaticali da questo testo:
	Chinese	除文本中的所有法:
	Korean	주어진 사용자의 입력에 오타나 문법 오류가 있으면 고친다:
	Japanese	このテキストからすべての文法エラを削除する:
Smart Reply	English	Suggest a reply for the following text:
	Spanish	Sugiera una respuesta para el texto siguiente:
	French	Propose une réponse pour le texte suivant:
	German	Schlagen Sie eine Antwort für den folgenden Text vor:
	Italian	Suggerisci una risposta per il seguente testo:
	Chinese	建以下文本行回:
	Korean	다음 텍스트에 대한 답변을 제안하시오:
	Japanese	次のテキストにする返信を提案します:
Text Summarization	English	Summarize the following text:
	Spanish	Resume el siguiente texto:
	French	Résume le texte suivant:
	German	Zusammenfassen Sie den folgenden Text:
	Italian	Riassumi il seguente testo:
	Chinese	一下下面的文字:
	Korean	다음 텍스트를 요약하시오:
	Japanese	次の文章を要約します:
Tone Adj. (Professional)	English	Changes a given user's input sentence or text to the Professional style:
	Spanish	Cambia la oración o el texto introducido por un usuario al estilo Profesional:
	French	Transforme la phrase ou le texte saisi par un utilisateur en style Professionnel:
	German	ändert die Eingabe eines bestimmten Benutzers in einen Professionellen Stil:
	Italian	Cambia la frase o il testo immesso da un utente in stile Professionale:
	Chinese	定用的入句子或文本更改格:
	Korean	주어진 사용자의 입력을 전문적인 문체로 변경한다:
	Japanese	指定されたユザの入力文またはテキストを プロフェッショナル スタイルに更する:
Tone Adj. (Casual)	English	Changes a given user's input sentence or text to the Casual style:
	Spanish	Cambia la oración o el texto introducido por un usuario al estilo Informal:
	French	Transforme la phrase ou le texte saisi par un utilisateur en style Informel:
	German	ändert die Eingabe eines bestimmten Benutzers in einen Freundlichen Stil:
	Italian	Cambia la frase o il testo immesso da un utente in stile Informal:
	Chinese	定用的入句子或文本更改日常格:
	Korean	주어진 사용자의 입력을 평범한 문체로 변경한다:
	Japanese	指定されたユザの入力文またはテキストを カジュアル スタイルに更する:
Tone Adj. (Witty)	English	Changes a given user's input sentence or text to the Witty style:
	Spanish	Cambia la oración o el texto introducido por un usuario al estilo Ingenioso:
	French	Transforme la phrase ou le texte saisi par un utilisateur en style Spirituel:
	German	ändert die Eingabe eines bestimmten Benutzers in einen Witziger Stil:
	Italian	Cambia la frase o il testo immesso da un utente in stile Spiritoso:
	Chinese	定用的入句子或文本更改机智格:
	Korean	주어진 사용자의 입력을 재치있는 문체로 변경한다:
	Japanese	指定されたユザの入力文またはテキストを ウィットに富んだ スタイルに更する:
Tone Adj. (Paraphrase)	English	Paraphrase the following text:
	Spanish	Parafrasea el siguiente texto:
	French	Paraphraser le texte suivant:
	German	Fassen Sie den folgenden Text zusammen:
	Italian	Parafrasare il testo seguente:
	Chinese	解以下文字:
	Korean	다음 텍스트를 의역하세요:
	Japanese	次のテキストを言い換えてください:
Question Answering	English	Answer the following question:
	Spanish	Responde a la siguiente pregunta:
	French	Réponds à la question suivante:
	German	Beantworten Sie die folgende Frage:
	Italian	Rispondi alla seguente domanda:
	Chinese	回答以下:
	Korean	다음 질문에 답하시오:
	Japanese	次の質問に答えましょう:

Table 8: **Prompts for each application and language.** Details about the prompts we have used.

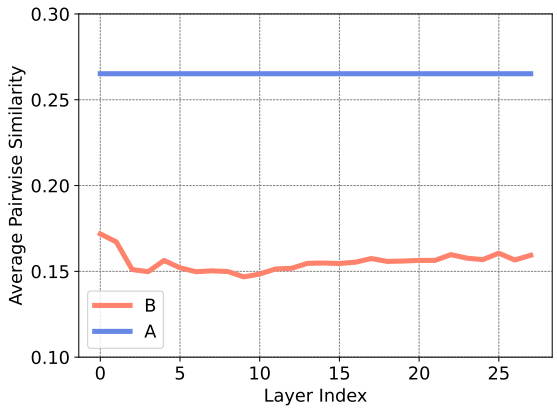

Figure 10: **Similarity between A and B matrices of LoRAs** measured using Canonical Correlation Analysis (CCA) goodness of fit as conducted by Zhu et al. (2024) on query matrices of Llama-3.2-3B-Instruct fine-tuned on English data.

Table 9: **Performance when merging across 5 applications using Llama-3.2-3B-Instruct for En (English), De (German), Es (Spanish), and Fr (French)**. Results are reported per each language (L) and application separately.

Method	L	S (%)	GC	SR	TS	TA	QA	Avg
Zero-shot		0	14.0	5.4	26.9	30.1	24.2	20.1
LoRA		100	39.2	24.5	41.4	59.1	74.7	47.8
TA		20	27.0	13.6	34.1	53.7	58.8	37.4
TIES		20	26.2	14.9	32.4	53.3	48.2	35.0
DARE		20	27.4	14.9	34.5	54.2	61.9	38.6
DARE TIES	En	20	21.6	8.3	29.2	45.2	33.0	27.4
HydraOpt(M=1)		20	26.6	10.2	33.9	52.2	59.6	36.5
HydraOpt(M=2)		30	27.4	22.9	34.6	54.8	67.0	41.3
HydraOpt(M=3)		40	28.9	24.6	36.0	58.8	73.2	44.3
HydraOpt(M=4)		50	36.0	24.7	39.6	57.6	72.3	46.0
HydraOpt(M=5)		60	36.5	24.7	40.9	58.8	73.3	46.8
Zero-shot		0	9.4	2.8	17.7	20.3	10.9	12.2
LoRA		100	41.2	13.8	32.4	44.8	47.2	35.9
TA		20	22.9	6.7	23.6	41.2	22.6	23.4
TIES		20	19.5	6.8	22.9	40.9	18.7	21.8
DARE		20	26.4	6.6	20.5	42.7	29.4	25.1
DARE TIES	De	20	12.1	4.1	20.9	30.0	15.6	16.5
HydraOpt(M=1)		20	23.2	5.6	25.2	37.1	22.8	22.8
HydraOpt(M=2)		30	25.8	9.6	26.2	43.4	34.8	28.0
HydraOpt(M=3)		40	24.4	12.4	27.2	43.4	43.1	30.1
HydraOpt(M=4)		50	24.6	13.4	29.7	43.7	45.3	31.3
HydraOpt(M=5)		60	31.1	13.8	31.7	44.8	46.4	33.5
Zero-shot		0	7.7	2.8	22.4	33.6	16.6	16.6
LoRA		100	34.3	15.8	34.9	46.3	53.8	37.0
TA		20	22.6	7.9	28.4	44.2	29.9	26.6
TIES		20	16.9	8.7	27.0	44.3	23.7	24.1
DARE		20	23.3	7.9	28.5	45.2	30.0	27.0
DARE TIES	Es	20	8.4	4.8	24.1	37.9	21.2	19.3
HydraOpt(M=1)		20	20.8	6.3	29.2	42.5	28.6	25.5
HydraOpt(M=2)		30	24.7	11.7	30.3	44.2	35.2	29.2
HydraOpt(M=3)		40	25.3	15.1	31.5	45.4	46.8	32.8
HydraOpt(M=4)		50	22.9	13.6	32.8	40.8	52.4	32.5
HydraOpt(M=5)		60	31.2	15.1	34.5	46.1	53.1	36.0
Zero-shot		0	10.2	3.6	20.8	28.5	11.3	14.9
LoRA		100	30.2	15.0	34.0	47.8	42.8	34.0
TA		20	20.8	7.6	28.1	46.0	29.0	26.3
TIES		20	16.4	8.8	27.6	46.1	25.8	24.9
DARE		20	19.9	8.3	25.9	47.1	29.2	26.1
DARE TIES	Fr	20	13.1	5.5	23.5	36.9	14.5	18.7
HydraOpt(M=1)		20	19.3	6.8	28.7	43.3	27.4	25.1
HydraOpt(M=2)		30	20.4	11.3	30.2	45.9	32.7	28.1
HydraOpt(M=3)		40	22.4	13.9	30.6	46.8	37.7	30.3
HydraOpt(M=4)		50	24.9	14.1	31.6	29.9	41.3	28.4
HydraOpt(M=5)		60	30.8	14.9	33.8	47.8	42.4	34.0

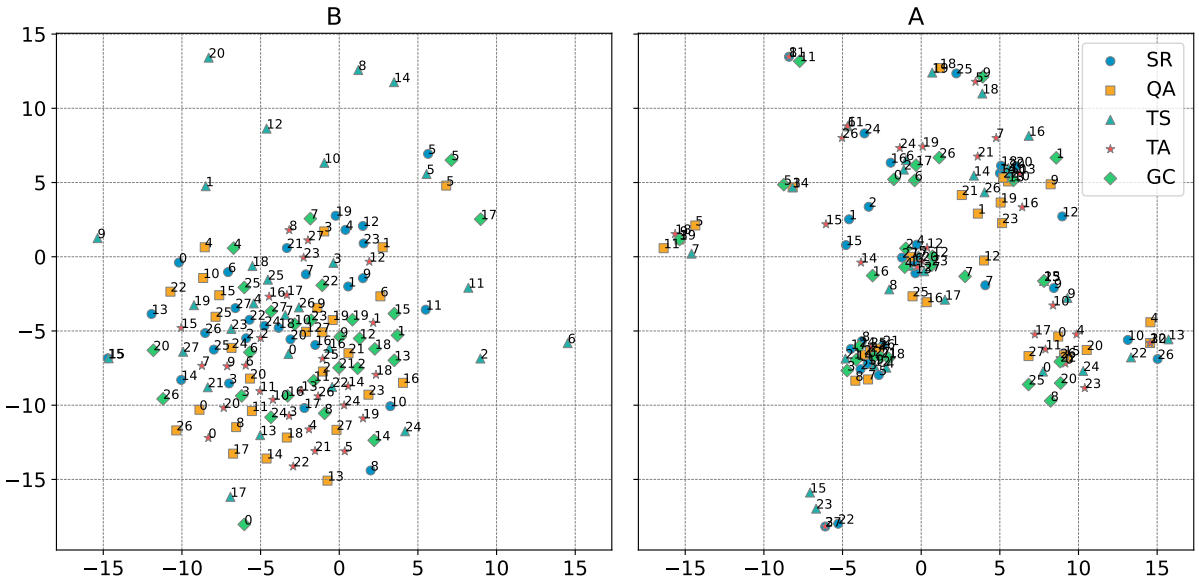

Figure 11: **t-SNE plots using LoRA parameters** for query matrices of Llama-3.2-3B-Instruct fine-tuned across 5 applications in English. Numbers indicate which layer the parameter comes from. Shapes/colors indicate the application the model is fine-tuned for. The differences in the LoRA parameters for different tasks arise mainly from the B matrices.

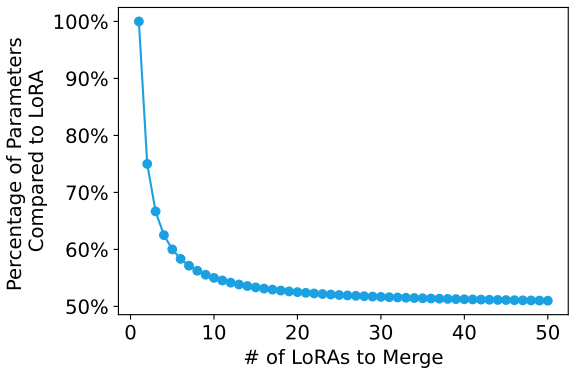

Figure 12: **The reduction in parameter size when using HydraOpt(M=K)**, assuming that the parameters A and B are the same size.

Table 10: **Performance when merging across 5 applications using Llama-3.2-3B-Instruct for It (Italian), Ja (Japanese), Ko (Korean), and Zh (Chinese)**. Results are reported per each language (L) and application separately.

Method	L	S (%)	GC	SR	TS	TA	QA	Avg
Zero-shot		0	26.9	2.3	18.9	29.5	13.5	18.2
LoRA		100	36.0	12.7	32.3	44.9	47.3	34.6
TA		20	34.0	6.0	25.3	43.4	21.0	26.0
TIES		20	31.1	6.7	24.7	43.4	18.4	24.9
DARE		20	29.7	6.0	24.9	44.5	23.8	25.8
DARE TIES	It	20	27.6	3.7	20.1	36.6	15.9	20.8
HydraOpt(M=1)		20	33.3	5.0	26.1	41.1	19.2	24.9
HydraOpt(M=2)		30	33.2	8.5	26.8	43.0	31.9	28.7
HydraOpt(M=3)		40	35.9	12.9	27.4	43.6	42.1	32.4
HydraOpt(M=4)		50	33.5	12.2	29.8	35.0	46.2	31.3
HydraOpt(M=5)		60	35.1	13.1	31.8	44.7	46.7	34.3
Zero-shot		0	4.1	4.7	19.5	35.6	12.6	15.3
LoRA		100	23.3	11.9	29.2	40.1	31.1	27.1
TA		20	10.9	6.0	25.0	40.2	20.7	20.6
TIES		20	6.3	7.0	24.5	39.9	18.6	19.2
DARE		20	9.4	5.8	25.2	40.0	22.0	20.5
DARE TIES	Ja	20	7.1	5.7	22.0	38.0	14.0	17.3
HydraOpt(M=1)		20	11.9	6.1	25.1	39.8	18.4	20.3
HydraOpt(M=2)		30	13.3	8.2	25.2	40.2	22.7	21.9
HydraOpt(M=3)		40	15.0	9.6	24.5	40.0	28.2	23.5
HydraOpt(M=4)		50	18.1	11.2	26.4	37.7	26.3	24.0
HydraOpt(M=5)		60	22.3	11.6	28.4	40.1	30.3	26.5
Zero-shot		0	7.7	0.9	9.4	25.7	11.5	11.0
LoRA		100	16.6	5.8	15.6	31.6	24.8	18.9
TA		20	6.9	2.1	13.0	31.2	11.8	13.0
TIES		20	8.6	2.4	12.4	31.0	8.3	12.5
DARE		20	7.1	2.2	13.2	31.2	9.9	12.7
DARE TIES	Ko	20	7.9	1.7	10.7	28.0	12.3	12.1
HydraOpt(M=1)		20	7.8	1.9	12.4	30.6	12.3	13.0
HydraOpt(M=2)		30	9.0	3.0	12.4	31.2	15.7	14.2
HydraOpt(M=3)		40	8.8	4.2	13.2	25.9	22.5	14.9
HydraOpt(M=4)		50	9.4	5.1	13.2	29.7	22.2	15.9
HydraOpt(M=5)		60	15.4	5.7	15.7	31.4	24.5	18.5
Zero-shot		0	3.7	1.6	20.2	24.8	12.1	12.5
LoRA		100	48.6	7.9	27.6	37.3	30.1	30.3
TA		20	10.6	3.7	24.1	37.8	19.9	19.2
TIES		20	10.3	3.9	23.6	37.0	18.4	18.6
DARE		20	6.3	3.7	23.4	37.9	19.3	18.1
DARE TIES	Zh	20	6.2	2.4	21.0	30.2	14.6	14.9
HydraOpt(M=1)		20	11.7	3.4	24.2	36.2	18.0	18.7
HydraOpt(M=2)		30	17.0	4.9	24.9	38.0	23.2	21.6
HydraOpt(M=3)		40	38.1	7.0	24.7	37.9	27.8	27.1
HydraOpt(M=4)		50	44.0	6.4	24.9	35.7	28.3	27.8
HydraOpt(M=5)		60	46.4	7.3	27.1	37.1	29.6	29.5

Table 11: **Performance when merging across 8 languages using Llama-3.2-3B-Instruct**. Results are reported per each language (L) and application separately.

Method	L	S (%)	GC	SR	TS	TA	QA	Avg
Zero-shot		0	14.0	5.4	26.9	30.1	24.2	20.1
LoRA		100.0	39.2	24.5	41.4	59.1	74.7	47.8
TA		12.5	27.6	15.3	35.8	49.6	75.5	40.8
TIES		12.5	28.2	16.3	36.0	49.7	76.2	41.3
DARE		12.5	17.2	7.6	29.4	35.5	34.5	24.9
DARE TIES		12.5	19.4	8.6	30.5	40.0	41.6	28.1
HydraOpt(M=1)	En	12.5	26.1	14.1	34.4	48.1	73.4	39.2
HydraOpt(M=2)		17.5	27.1	16.7	35.3	50.3	72.6	40.4
HydraOpt(M=3)		22.5	29.5	17.9	36.9	51.6	71.2	41.4
HydraOpt(M=4)		27.5	29.3	19.4	37.1	53.4	69.8	41.8
HydraOpt(M=5)		32.5	36.2	21.2	41.2	57.0	63.0	43.7
HydraOpt(M=6)		37.5	38.9	23.4	40.5	58.2	67.0	45.6
HydraOpt(M=7)		42.5	39.1	24.1	41.0	57.9	72.4	46.9
HydraOpt(M=8)		47.5	36.9	23.9	41.1	58.5	73.3	46.7
Zero-shot		0	9.4	2.8	17.7	20.3	10.9	12.2
LoRA		100.0	41.2	13.8	32.4	44.8	47.2	35.9
TA		12.5	30.5	7.7	31.5	41.2	46.2	31.4
TIES		12.5	29.4	7.7	31.7	41.5	46.5	31.3
DARE		12.5	12.4	4.2	21.9	28.5	17.3	16.9
DARE TIES		12.5	14.1	4.9	24.2	31.3	21.8	19.2
HydraOpt(M=1)	De	12.5	25.8	7.2	29.9	39.7	47.1	30.0
HydraOpt(M=2)		17.5	26.8	7.8	30.6	41.6	46.9	30.7
HydraOpt(M=3)		22.5	27.1	8.0	30.6	43.0	48.0	31.3
HydraOpt(M=4)		27.5	25.7	8.2	30.6	42.4	45.9	30.5
HydraOpt(M=5)		32.5	32.3	9.7	31.4	43.5	48.7	33.1
HydraOpt(M=6)		37.5	26.9	8.6	31.2	43.9	45.7	31.3
HydraOpt(M=7)		42.5	31.4	9.3	31.5	44.7	44.8	32.4
HydraOpt(M=8)		47.5	31.0	12.1	31.6	44.5	46.3	33.1
Zero-shot		0	7.7	2.8	22.4	33.6	16.6	16.6
LoRA		100.0	34.3	15.8	34.9	46.3	53.8	37.0
TA		12.5	32.6	9.4	34.6	34.0	48.5	31.9
TIES		12.5	33.1	9.0	34.8	29.7	49.3	31.2
DARE		12.5	9.8	4.9	26.8	37.8	24.9	20.8
DARE TIES		12.5	12.0	5.8	28.8	39.2	31.7	23.5
HydraOpt(M=1)	Es	12.5	26.1	9.0	33.1	32.6	45.6	29.3
HydraOpt(M=2)		17.5	28.6	9.4	33.5	43.6	48.0	32.6
HydraOpt(M=3)		22.5	28.4	10.3	33.9	44.2	49.3	33.2
HydraOpt(M=4)		27.5	26.6	11.4	34.6	45.7	53.1	34.3
HydraOpt(M=5)		32.5	29.2	10.4	34.7	45.7	51.5	34.3
HydraOpt(M=6)		37.5	28.6	12.0	34.1	45.1	53.3	34.6
HydraOpt(M=7)		42.5	27.3	11.3	33.7	37.8	52.9	32.6
HydraOpt(M=8)		47.5	31.7	13.5	34.5	46.1	52.5	35.7
Zero-shot		0	10.2	3.6	20.8	28.5	11.3	14.9
LoRA		100.0	30.2	15.0	34.0	47.8	42.8	34.0
TA		12.5	29.3	9.4	34.1	45.3	41.0	31.8
TIES		12.5	30.4	9.6	34.4	45.0	42.2	32.3
DARE		12.5	13.5	5.5	24.8	35.7	18.0	19.5
DARE TIES		12.5	14.3	6.4	26.5	37.6	26.3	22.2
HydraOpt(M=1)	Fr	12.5	24.2	9.2	32.6	44.0	38.5	29.7
HydraOpt(M=2)		17.5	29.8	9.1	33.5	45.2	39.1	31.3
HydraOpt(M=3)		22.5	26.7	9.7	33.8	45.9	40.8	31.4
HydraOpt(M=4)		27.5	29.5	9.4	33.6	46.9	41.9	32.3
HydraOpt(M=5)		32.5	31.6	9.6	33.7	47.4	40.2	32.5
HydraOpt(M=6)		37.5	31.1	10.3	33.8	47.8	41.8	32.9
HydraOpt(M=7)		42.5	30.3	10.2	33.0	46.8	39.0	31.9
HydraOpt(M=8)		47.5	31.4	13.6	33.9	47.6	41.9	33.7

Table 12: **Performance when merging across 8 languages using Llama-3.2-3B-Instruct**. Results are reported per each language (L) and application separately.

Method	L	S (%)	GC	SR	TS	TA	QA	Avg
Zero-shot		0	26.9	2.3	18.9	29.5	13.5	18.2
LoRA		100.0	36.0	12.7	32.3	44.9	47.3	34.6
TA		12.5	34.0	6.3	32.4	42.8	45.3	32.2
TIES		12.5	34.2	6.1	32.6	42.4	46.3	32.3
DARE		12.5	28.6	3.6	21.6	35.9	18.9	21.7
DARE TIES		12.5	30.3	4.1	23.7	37.1	23.3	23.7
HydraOpt(M=1)	It	12.5	33.1	5.9	31.1	41.8	43.1	31.0
HydraOpt(M=2)		17.5	33.8	6.6	30.9	42.1	45.0	31.7
HydraOpt(M=3)		22.5	35.0	6.3	32.1	42.6	44.3	32.1
HydraOpt(M=4)		27.5	35.3	7.4	31.0	43.1	45.5	32.4
HydraOpt(M=5)		32.5	35.5	7.7	31.0	43.4	46.1	32.7
HydraOpt(M=6)		37.5	34.7	9.7	31.8	43.8	45.5	33.1
HydraOpt(M=7)		42.5	35.8	6.8	31.5	42.6	45.6	32.5
HydraOpt(M=8)		47.5	35.0	11.4	31.7	44.5	46.3	33.8
Zero-shot		0	4.1	4.7	19.5	35.6	12.6	15.3
LoRA		100.0	23.3	11.9	29.2	40.1	31.1	27.1
TA		12.5	18.8	6.1	27.3	40.4	29.6	24.5
TIES		12.5	21.5	5.9	27.5	39.7	30.4	25.0
DARE		12.5	8.9	5.4	22.5	37.9	17.6	18.5
DARE TIES		12.5	11.1	5.9	23.6	38.5	22.3	20.3
HydraOpt(M=1)	Ja	12.5	15.0	6.1	26.3	40.2	28.6	23.2
HydraOpt(M=2)		17.5	16.6	6.4	26.7	40.8	28.7	23.8
HydraOpt(M=3)		22.5	18.3	6.9	26.4	40.4	28.7	24.1
HydraOpt(M=4)		27.5	19.1	7.3	27.4	40.9	29.5	24.8
HydraOpt(M=5)		32.5	20.8	8.0	28.4	40.8	29.1	25.4
HydraOpt(M=6)		37.5	21.3	7.9	28.7	38.9	28.5	25.0
HydraOpt(M=7)		42.5	22.2	5.0	28.3	37.5	29.9	24.6
HydraOpt(M=8)		47.5	21.6	9.8	28.1	40.2	30.0	25.9
Zero-shot		0	7.7	9.4	25.7	4.2	11.5	11.0
LoRA		100.0	16.6	5.8	15.6	31.6	24.8	18.9
TA		12.5	12.8	2.0	14.0	31.3	19.7	16.0
TIES		12.5	13.8	1.8	14.7	30.9	20.2	16.3
DARE		12.5	12.8	1.6	11.3	28.0	17.4	14.2
DARE TIES		12.5	13.9	1.6	11.8	28.7	19.2	15.0
HydraOpt(M=1)	Ko	12.5	14.1	2.1	13.3	30.6	19.4	15.9
HydraOpt(M=2)		17.5	13.8	2.4	13.7	31.0	20.8	16.3
HydraOpt(M=3)		22.5	16.1	2.5	14.3	30.9	22.8	17.3
HydraOpt(M=4)		27.5	13.4	2.9	14.5	31.8	21.4	16.8
HydraOpt(M=5)		32.5	16.5	3.8	15.3	30.5	22.6	17.7
HydraOpt(M=6)		37.5	11.2	3.1	15.0	31.1	25.3	17.1
HydraOpt(M=7)		42.5	13.3	2.5	15.4	29.2	23.2	16.7
HydraOpt(M=8)		47.5	15.2	4.5	15.2	31.3	24.5	18.2
Zero-shot		0	3.7	1.6	20.2	24.8	12.1	12.5
LoRA		100.0	48.6	7.9	27.6	37.3	30.1	30.3
TA		12.5	24.6	4.0	27.1	37.3	28.6	24.3
TIES		12.5	28.4	4.0	27.4	37.4	29.2	25.3
DARE		12.5	6.5	2.2	21.9	30.1	16.5	15.4
DARE TIES		12.5	7.8	2.6	23.0	31.4	20.3	17.0
HydraOpt(M=1)	Zh	12.5	12.6	3.6	26.1	36.4	26.5	21.0
HydraOpt(M=2)		17.5	19.6	3.9	26.7	36.8	28.0	23.0
HydraOpt(M=3)		22.5	38.3	4.4	26.6	38.2	28.2	27.2
HydraOpt(M=4)		27.5	41.9	5.0	27.0	37.5	29.2	28.1
HydraOpt(M=5)		32.5	43.1	5.2	27.1	37.6	27.7	28.1
HydraOpt(M=6)		37.5	43.8	4.9	27.3	35.1	29.2	28.1
HydraOpt(M=7)		42.5	43.4	4.7	24.3	36.1	26.2	26.9
HydraOpt(M=8)		47.5	46.2	6.9	27.0	37.3	29.6	29.4

Table 13: **Performance when merging across 40 tasks using Llama-3.2-3B-Instruct**. Results are reported per each language (L) and application separately.

Method	L	S (%)	GC	SR	TS	TA	QA	Avg
Zero-shot	En	0	14.0	5.4	26.9	30.1	24.2	20.1
LoRA		100.0	39.2	24.5	41.4	59.1	74.7	47.8
TA		2.5	23.9	8.7	30.6	49.0	40.5	30.5
TIES		2.5	23.3	8.7	30.3	48.2	35.9	29.3
DARE		2.5	15.7	6.6	28.2	33.7	25.9	22.0
DARE-TIES		2.5	15.6	6.6	28.2	33.7	25.8	22.0
HydraOpt(M=1)		2.5	24.2	7.8	30.3	49.1	42.3	30.8
HydraOpt(M=40)		41.5	28.9	22.1	35.6	57.0	71.3	43.0
Zero-shot	De	0	9.4	2.8	17.7	20.3	10.9	12.2
LoRA		100.0	41.2	13.8	32.4	44.8	47.2	35.9
TA		2.5	20.6	6.0	24.9	35.5	19.4	21.3
TIES		2.5	17.5	6.0	24.2	34.9	18.1	20.2
DARE		2.5	11.5	3.2	19.3	25.3	12.6	14.4
DARE-TIES		2.5	11.8	3.3	19.1	25.4	12.5	14.4
HydraOpt(M=1)		2.5	22.7	5.6	25.6	34.3	21.3	21.9
HydraOpt(M=40)		41.5	20.3	7.2	27.7	42.4	31.2	25.8
Zero-shot	Es	0	7.7	2.8	22.4	33.6	16.6	16.6
LoRA		100.0	34.3	15.8	34.9	46.3	53.8	37.0
TA		2.5	19.2	6.8	28.9	41.5	25.4	24.4
TIES		2.5	15.5	6.5	28.3	40.0	22.8	22.6
DARE		2.5	7.9	3.7	24.0	36.6	19.7	18.4
DARE-TIES		2.5	7.4	3.8	23.6	36.2	19.1	18.0
HydraOpt(M=1)		2.5	20.3	6.2	29.2	42.3	27.6	25.1
HydraOpt(M=40)		41.5	17.7	8.8	31.5	42.6	24.4	25.0
Zero-shot	Fr	0	10.2	3.6	20.8	28.5	11.3	14.9
LoRA		100.0	30.2	15.0	34.0	47.8	42.8	34.0
TA		2.5	17.5	7.2	27.5	43.2	23.3	23.8
TIES		2.5	15.8	7.2	26.7	42.1	20.4	22.4
DARE		2.5	12.4	4.5	22.1	33.5	12.1	16.9
DARE-TIES		2.5	12.3	4.6	22.0	33.7	11.8	16.9
HydraOpt(M=1)		2.5	19.2	6.8	28.9	42.1	24.6	24.3
HydraOpt(M=40)		41.5	23.3	8.4	31.2	45.2	27.1	27.0
Zero-shot	It	0	26.9	2.3	18.9	29.5	13.5	18.2
LoRA		100.0	36.0	12.7	32.3	44.9	47.3	34.6
TA		2.5	33.0	4.9	26.7	41.7	19.7	25.2
TIES		2.5	30.5	4.8	26.0	40.6	18.1	24.0
DARE		2.5	26.7	2.7	19.7	34.6	15.2	19.8
DARE-TIES		2.5	26.6	2.7	19.6	34.7	14.9	19.7
HydraOpt(M=1)		2.5	33.5	5.1	27.2	41.2	19.1	25.2
HydraOpt(M=40)		41.5	33.2	5.9	29.2	41.4	20.7	26.1
Zero-shot	Ja	0	4.1	4.7	19.5	35.6	12.6	15.3
LoRA		100.0	23.3	11.9	29.2	40.1	31.1	27.1
TA		2.5	9.8	6.2	22.5	39.3	18.9	19.3
TIES		2.5	9.4	6.0	22.3	39.2	18.0	19.0
DARE		2.5	6.3	4.9	20.7	37.2	13.0	16.4
DARE-TIES		2.5	6.1	5.0	20.8	37.3	12.9	16.4
HydraOpt(M=1)		2.5	11.6	6.2	23.3	39.3	17.7	19.6
HydraOpt(M=40)		41.5	13.8	6.4	23.3	39.1	24.5	21.4
Zero-shot	Ko	0	7.7	0.9	9.4	25.7	11.5	11.0
LoRA		100.0	16.6	5.8	15.6	31.6	24.8	18.9
TA		2.5	9.0	1.6	10.6	30.7	13.1	13.0
TIES		2.5	8.1	1.8	10.6	30.6	12.3	12.7
DARE		2.5	9.1	1.3	10.1	27.4	12.1	12.0
DARE-TIES		2.5	8.9	1.4	10.0	27.5	11.9	11.9
HydraOpt(M=1)		2.5	8.4	1.6	11.2	30.3	13.5	13.0
HydraOpt(M=40)		41.5	13.8	1.8	10.7	30.8	15.5	14.5
Zero-shot	Zh	0	3.7	1.6	20.2	24.8	12.1	12.5
LoRA		100.0	48.6	7.9	27.6	37.3	30.1	30.3
TA		2.5	7.9	3.1	22.6	35.2	15.9	16.9
TIES		2.5	6.7	2.9	22.7	34.9	15.2	16.5
DARE		2.5	4.2	1.8	20.4	29.1	13.1	13.7
DARE-TIES		2.5	4.5	1.9	20.4	29.5	12.8	13.8
HydraOpt(M=1)		2.5	7.9	2.6	22.9	33.3	15.2	16.4
HydraOpt(M=40)		41.5	12.5	3.7	24.7	35.6	16.3	18.5

Table 14: **Performance on 40 tasks using Llama-3.2-3B-Instruct LoRA-finetuned** after merging LoRAs across applications, languages and tasks. S represents the percentage of parameters compared to storing all 40 LoRAs.

	Method	S (%)	GC	SR	TS	TA	QA	Avg
	Zero-shot	0.0	10.5	3.0	19.5	28.5	14.1	15.1
	LoRA	100.0	33.7	13.4	30.9	44.0	44.0	33.2
applications	TA	20.0	19.5	6.7	25.2	42.2	26.7	24.1
	TIES	20.0	14.9	6.0	22.9	38.7	20.1	20.5
	DARE	20.0	18.7	6.9	24.5	42.9	28.2	24.2
	DARE-TIES	20.0	13.0	4.5	21.5	35.3	17.6	18.4
	HydraOpt(M=1)	20.0	19.3	5.7	25.6	40.4	25.8	23.3
	HydraOpt(M=2)	28.0	21.3	10.0	26.3	42.6	32.9	26.6
	HydraOpt(M=3)	36.0	24.8	12.5	26.9	42.7	40.2	29.4
	HydraOpt(M=4)	44.0	26.7	12.6	28.5	38.8	41.8	29.7
	HydraOpt(M=5)	52.0	31.1	13.3	30.5	43.9	43.3	32.4
languages	TA	12.5	26.3	7.5	29.6	40.2	41.8	29.1
	TIES	12.5	21.4	6.3	26.9	37.5	34.2	25.3
	DARE	12.5	13.7	4.4	22.5	33.7	20.6	19.0
	DARE-TIES	12.5	15.4	5.0	24.0	35.5	25.8	21.1
	HydraOpt(M=1)	12.5	22.1	7.1	28.4	39.2	40.3	27.4
	HydraOpt(M=2)	17.5	24.5	7.8	28.8	41.4	41.1	28.7
	HydraOpt(M=3)	22.5	27.4	8.2	29.3	42.1	41.6	29.7
	HydraOpt(M=4)	27.5	27.6	8.9	29.5	42.7	42.0	30.1
	HydraOpt(M=5)	32.5	30.6	9.4	30.4	43.2	41.1	31.0
	HydraOpt(M=6)	37.5	29.6	10.0	30.3	43.0	42.0	31.0
	HydraOpt(M=7)	42.5	30.3	9.2	29.8	41.6	41.8	30.6
	HydraOpt(M=8)	47.5	31.1	12.0	30.4	43.7	43.1	32.1
tasks	TA	2.5	17.6	5.6	24.3	39.5	22.0	21.8
	TIES	2.5	13.7	4.6	22.2	35.5	17.7	18.7
	DARE	2.5	11.7	3.6	20.5	32.2	15.4	16.7
	DARE-TIES	2.5	11.6	3.6	20.4	32.3	15.2	16.6
	HydraOpt(M=1)	2.5	18.5	5.2	24.8	39.0	22.7	22.0
	HydraOpt(M=40)	41.5	20.4	8.0	26.7	41.8	28.9	25.2

Table 15: **Performance on 5 English applications using Llama-3.2-1B-Instruct LoRA-finetuned at variable rank.** S represents the percentage of the parameters compared to storing 5 LoRAs. Results are reported per each application separately.

Rank	Method	S (%)	GC	SR	TS	TA	QA	Avg
	Zero-shot	0.0	13.1	5.1	23.4	27.6	15.8	17.0
8	LoRA	100.0	26.9	20.4	35.4	56.3	57.2	39.2
	TA	20.0	25.4	10.6	30.1	51.1	24.6	28.4
	TIES	20.0	23.9	11.8	28.8	51.6	24.6	28.1
	DARE	20.0	17.8	6.7	25.2	35.0	18.9	20.7
	DARE TIES	20.0	19.9	7.5	26.0	38.6	19.8	22.4
	HydraOpt(M=1)	20.0	25.1	8.5	29.9	48.0	23.8	27.1
	HydraOpt(M=2)	28.0	24.6	15.5	30.0	50.6	29.5	30.0
	HydraOpt(M=3)	36.0	25.5	18.7	30.4	53.5	41.4	33.9
	HydraOpt(M=4)	44.0	26.3	19.1	30.1	55.9	40.2	34.3
	HydraOpt(M=5)	52.0	26.5	20.0	32.8	56.7	56.4	38.5
16	LoRA	100.0	32.6	21.6	36.9	57.2	60.4	41.7
	TA	20.0	25.6	10.6	31.2	50.7	26.5	28.9
	TIES	20.0	24.6	13.1	29.6	52.6	26.0	29.2
	DARE	20.0	18.7	7.0	26.1	37.8	19.6	21.8
	DARE TIES	20.0	20.8	7.7	27.0	42.6	20.9	23.8
	HydraOpt(M=1)	20.0	26.1	8.9	31.6	49.6	25.2	28.3
	HydraOpt(M=2)	28.0	25.6	18.4	31.3	51.9	28.0	31.0
	HydraOpt(M=3)	36.0	26.7	20.7	31.1	56.1	43.5	35.6
	HydraOpt(M=4)	44.0	26.4	20.1	34.0	56.0	57.1	38.7
	HydraOpt(M=5)	52.0	29.1	20.3	36.2	56.3	58.0	40.0

Table 16: **Performance on 5 English applications using Gemma-2-2B-it LoRA-finetuned.** S represents the percentage of the parameters compared to storing 5 LoRAs. Results are reported per each application separately.

Method	S (%)	GC	SR	TS	TA	QA	Avg
Zero-shot	0.0	19.4	4.3	27.4	31.0	21.7	20.8
LoRA	100.0	40.0	24.4	41.1	58.0	75.8	47.9
TA	20.0	29.5	14.9	34.9	51.4	57.9	37.7
TIES	20.0	28.6	14.9	33.0	51.4	47.7	35.1
DARE	20.0	23.1	4.5	28.9	38.6	27.8	24.6
DARE TIES	20.0	24.0	5.0	29.0	41.1	29.5	25.7
HydraOpt(M=1)	20.0	29.3	12.2	35.9	49.4	53.5	36.0
HydraOpt(M=2)	28.0	28.7	20.7	36.1	53.6	62.5	40.3
HydraOpt(M=3)	36.0	28.5	23.1	37.2	57.1	67.3	42.6
HydraOpt(M=4)	44.0	26.5	23.2	39.2	56.5	72.8	43.7
HydraOpt(M=5)	52.0	39.2	23.7	40.8	58.1	75.8	47.5

Table 17: **Performance on 5 English applications using Phi-3.5-mini-instruct LoRA-finetuned.** S represents the percentage of the parameters compared to storing 5 LoRAs. Results are reported per each application separately.

Method	S (%)	GC	SR	TS	TA	QA	Avg
Zero-shot	0.0	19.1	2.6	23.6	21.5	7.5	14.9
LoRA	100.0	33.3	25.1	39.9	58.0	71.0	45.4
TA	20.0	26.8	10.3	33.1	49.4	47.0	33.3
TIES	20.0	26.7	12.9	32.1	52.3	46.4	34.1
DARE	20.0	22.3	3.7	25.9	38.1	10.8	20.1
DARE TIES	20.0	22.3	3.7	25.9	38.1	10.8	20.1
HydraOpt(M=1)	20.0	26.7	7.4	33.7	45.6	39.5	30.6
HydraOpt(M=2)	28.0	27.4	18.4	34.7	52.2	71.1	40.8
HydraOpt(M=3)	36.0	28.0	23.3	36.7	57.5	71.0	43.3
HydraOpt(M=4)	44.0	29.7	21.0	36.8	56.8	67.7	42.4
HydraOpt(M=5)	52.0	32.7	25.0	40.0	57.7	70.7	45.2

Problem Type	Language	Prompt
Grammar Error Correction	English	Remove all grammatical errors from this text:
	Japanese	このテキストからすべての文法エラを削除します:
Smart Reply	English	Suggest a reply for the following text:
	Japanese	次のテキストにする返信を提案します:
Text Summarization	English	Summarize the following text:
	Japanese	次の文章を要約します:
Tone Adj. (Professional)	English	Changes a given user's input sentence or text to the Professional style:
	Japanese	指定されたユザの入力文またはテキストをプロフェッショナルスタイルに更します:
Tone Adj. (Casual)	English	Changes a given user's input sentence or text to the Casual style:
	Japanese	指定されたユザの入力文またはテキストをカジュアルスタイルに更します:
Tone Adj. (Witty)	English	Changes a given user's input sentence or text to the Witty style:
	Japanese	指定されたユザの入力文またはテキストをウィットに富んだスタイルに更します:
Tone Adj. (Paraphrase)	English	Paraphrase the following text:
	Japanese	次のテキストを言い換えます:
Question Answering	English	Answer the following question:
	Japanese	次の質問に答えます:

Table 18: **New prompts for each application in Japanese.** These are used for comparison with the prompts in Table 7. Please see Table 19.

Table 19: **Performance when merging across 5 applications in Japanese using Llama-3.2-1B-Instruct**. The results are consistent with our initial findings that HydraOpt continues to provide a strong balance between performance and efficiency.

Method	L	S (%)	GC	SR	TS	TA	QA	Avg
Zero-shot		0	3.7	4.3	18.4	34.6	6.7	13.5
LoRA		100.0	12.9	9.1	27.5	39.4	23.1	22.4
TA		20.0	6.0	6.0	23.8	39.5	10.4	17.1
TIES		20.0	3.1	5.7	22.3	39.4	9.6	16.0
DARE		20.0	6.3	5.2	20.0	37.1	8.2	15.4
DARE TIES	Old Prompt	20.0	5.8	5.6	20.0	38.2	8.7	15.7
HydraOpt(M=1)		20.0	8.6	6.2	22.0	39.8	10.1	17.3
HydraOpt(M=2)		28.0	9.2	7.0	19.7	39.7	10.1	17.1
HydraOpt(M=3)		36.0	10.4	6.5	19.0	40.1	11.6	17.5
HydraOpt(M=4)		44.0	6.0	3.9	20.5	38.0	15.6	16.8
HydraOpt(M=5)		52.0	12.4	8.4	25.6	39.5	21.9	21.6
Zero-shot		0	3.4	4.8	19.4	34.4	8.4	14.1
LoRA		100.0	11.5	8.7	27.4	39.2	23.4	22.0
TA		20.0	2.9	5.6	20.0	37.7	13.4	15.9
TIES		20.0	2.2	6.1	20.1	37.0	11.6	15.4
DARE		20.0	5.4	5.6	20.6	36.8	9.9	15.7
DARE TIES	New Prompt	20.0	6.1	5.5	21.4	36.3	10.1	15.9
HydraOpt(M=1)		20.0	8.2	6.4	22.1	39.4	12.3	17.7
HydraOpt(M=2)		28.0	8.9	7.5	19.2	38.6	11.3	17.1
HydraOpt(M=3)		36.0	9.5	8.4	19.4	39.5	12.8	17.9
HydraOpt(M=4)		44.0	6.2	4.0	13.8	38.3	17.0	15.9
HydraOpt(M=5)		52.0	13.9	7.6	23.6	39.2	21.5	21.2