DIMLORA: DIMENSION-LEVEL LORA EXPERTS FOR MULTI-TASK LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-efficient fine-tuning (PEFT) has been widely adopted to reduce the computational cost of adapting large language models to downstream tasks. Among existing PEFT methods, Low-Rank Adaptation (LoRA) is widely recognized for its simplicity and strong empirical performance. However, most LoRA-based approaches are designed for single-task adaptation and often suffer from severe inter-task interference in multi-task scenarios. Recent LoRA-MoE methods mitigate this issue by integrating LoRA with mixture-of-experts (MoE) architectures, but introducing multiple full LoRA modules as experts incurs considerable parameter overhead. Through empirical analysis, we identify significant parameter redundancy in LoRA and task-specific activation patterns among feed-forward network neurons. Motivated by these observations, we propose DimLoRA, a fine-grained expert architecture for parameter-efficient multi-task adaptation. Instead of assigning a complete LoRA module to each expert, DimLoRA decomposes the parameter space along the intermediate dimension of feed-forward networks and introduces lightweight dimension-level experts for localized parameter updates. Extensive experiments demonstrate that DimLoRA achieves superior multi-task performance with substantially lower parameter overhead, highlighting the effectiveness of fine-grained experts for efficient multi-task learning. See the code at `https://anonymous.4open.science/r/DimLoRA`.

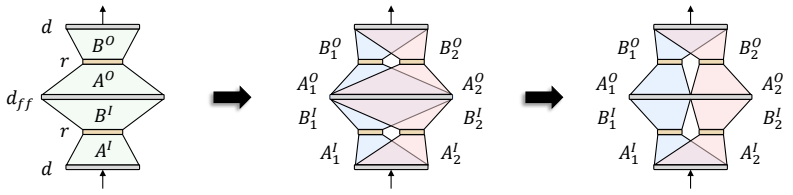
1 INTRODUCTION

Adapting pretrained large language models (LLMs; Grattafiori et al., 2024; Yang et al., 2025a; DeepSeek-AI, 2026) to downstream applications through fine-tuning has become a standard paradigm for meeting domain-specific requirements. However, the large number of model parameters makes full-parameter fine-tuning prohibitively expensive in terms of memory consumption and computational cost, limiting its practical deployment. To address this challenge, parameter-efficient fine-tuning (PEFT) methods (Houlsby et al., 2019; Lester et al., 2021; Hu et al., 2022) have been widely adopted. These methods introduce only a small number of trainable parameters while keeping the backbone model frozen. Among them, Low-Rank Adaptation (LoRA; Hu et al., 2022) has become the most widely used approach due to its simplicity and strong empirical performance.

Despite the success of LoRA across various domains, the increasing task complexity and the cost of separate fine-tuning motivate the exploration of multi-task LoRA adaptation. In this setting, LoRA has to project task-specific features from a sparse high-dimensional space into a shared dense low-dimensional space. This projection can intensify negative interference across tasks, leading to the loss of task-specific information. To alleviate this issue, recent works (Zadouri et al., 2024; Liu et al., 2024a; Tian et al., 2024; Yang et al., 2025b) integrate LoRA with mixture-of-experts (MoE) architectures by introducing multiple LoRA adapters as experts. By enabling different experts to specialize in distinct tasks, these LoRA-MoE methods effectively reduce inter-task interference.

Although effective, constructing each expert as a full LoRA module introduces substantial parameter overhead. Moreover, each LoRA module can modify the entire parameter space, thereby posing a risk of conflicting updates. We empirically find that LoRA is highly overparameterized: masking up to 50% of its parameters causes negligible performance degradation. This observation suggests that a considerable portion of LoRA parameters is redundant and underutilized. Such redundancy is further exacerbated in LoRA-MoE methods, where the total number of trainable parameters increases

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



(a) LoRA (b) LoRA-MoE (c) DimLoRA

Figure 1: Illustration of LoRA feature mappings in a two-layer FFN: (a) standard single LoRA; (b) LoRA-MoE, which uses multiple full LoRA modules as experts; and (c) DimLoRA, in which each LoRA expert adapts a subset of the FFN intermediate dimension. Here, d denotes the hidden dimension, d_{ff} denotes the FFN intermediate dimension, and r denotes the LoRA rank.

linearly with the number of experts, weakening the parameter-efficiency advantage of LoRA. To address these limitations, we revisit the design of LoRA experts and investigate the following fundamental question:

Can LoRA experts specialize at a finer granularity for more targeted adaptation?

To answer this question, we conduct a detailed analysis of feed-forward networks (FFNs) in LLMs. Specifically, we examine activation magnitudes at the output of the FFN input projection and find that different tasks exhibit distinct activation patterns. This observation indicates that different tasks consistently activate distinct subsets of FFN neurons, suggesting that task specialization may naturally emerge at the neuron level rather than across the entire FFN intermediate dimension.

Motivated by these observations, we propose DimLoRA, a fine-grained expert architecture for multi-task parameter-efficient fine-tuning. As shown in Figure 1, instead of assigning a full LoRA module to each expert, DimLoRA decomposes the adaptation space along the FFN intermediate dimension, allowing each expert to operate on only a subset of neurons. This fine-grained design enables the model to selectively adapt different functional components of the network for different tasks, substantially reducing parameter overhead while retaining the benefits of the MoE architecture. Extensive experiments on multi-task learning benchmarks show that DimLoRA outperforms existing LoRA-MoE methods with fewer trainable parameters, demonstrating the effectiveness of fine-grained experts for multi-task parameter-efficient fine-tuning.

Our contributions are as follows: ❶ We identify substantial redundancy in LoRA parameters and reveal task-specific activation patterns in FFN neurons through empirical analysis. ❷ We propose DimLoRA, a fine-grained expert architecture in which each LoRA expert adapts only a subset of the FFN intermediate dimensions. ❸ Extensive experiments demonstrate that DimLoRA outperforms existing LoRA-MoE methods in both parameter efficiency and multi-task performance.

2 Background and Motivation

2.1 Low-Rank Adaptation

LoRA (Hu et al., 2022) assumes that parameter updates during fine-tuning typically lie in a low-rank subspace (Aghajanyan et al., 2021). It parameterizes the update $\Delta \boldsymbol{W} \in \mathbb{R}^{d_1 \times d_2}$ as the product of two low-rank matrices $\boldsymbol{B} \in \mathbb{R}^{d_1 \times r}$ and $\boldsymbol{A} \in \mathbb{R}^{r \times d_2}$, where $r \ll \min(d_1, d_2)$. Given a pretrained weight $\boldsymbol{W}_0 \in \mathbb{R}^{d_1 \times d_2}$, LoRA is formulated as

$$\boldsymbol{W}' = \boldsymbol{W}_0 + \Delta \boldsymbol{W} = \boldsymbol{W}_0 + \boldsymbol{B}\boldsymbol{A}, \tag{1}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are trainable, while $\boldsymbol{W}_0$ remains frozen during fine-tuning. In practice, $\boldsymbol{B}$ is initialized to zero, while $\boldsymbol{A}$ is initialized using Kaiming Uniform initialization (He et al., 2015), ensuring $\Delta \boldsymbol{W} = 0$ at the beginning of fine-tuning.

2.2 Mixture-of-Experts

An MoE layer contains N experts $\{E_i\}_{i=1}^N$ and a router G. Given input x, the router computes expert weights $\{w_i\}_{i=1}^N$, and the layer output is obtained as a weighted combination of expert outputs:

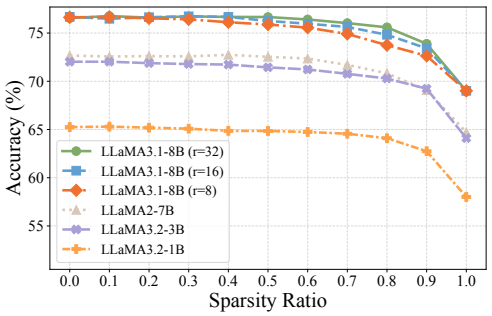


Figure 2: Average accuracy of commonsense reasoning tasks (**y-axis**) under various sparsity ratios (**x-axis**) applied to the trainable parameters. Masking 50% of the LoRA parameters yields negligible performance degradation.

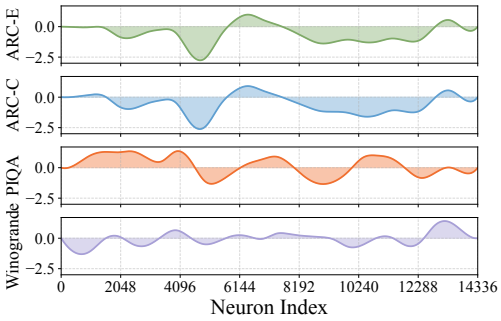


Figure 3: Activation magnitude distributions (10^{-3}) of FFN intermediate hidden states on the ARC-E, ARC-C, PIQA, and Winogrande datasets. The distributions exhibit clear task-specific patterns.

$$y = \sum_{i=1}^{N} w_i E_i(x). \tag{2}$$

LoRA-MoE integrates LoRA into the MoE framework, yielding the following computation:

$$y = \boldsymbol{W}_0 x + \Delta \boldsymbol{W} x = \boldsymbol{W}_0 x + \sum_{i=1}^{N} w_i \boldsymbol{B}_i \boldsymbol{A}_i x, \tag{3}$$

where $\boldsymbol{B}_i \in \mathbb{R}^{d_1 \times r}$ and $\boldsymbol{A}_i \in \mathbb{R}^{r \times d_2}$ denote the parameters of the i-th expert. Each expert adopts the same structure and input-output dimensions as a standard full LoRA module.

2.3 Observations

Existing LoRA-MoE methods (Zadouri et al., 2024; Liu et al., 2024a; Tian et al., 2024; Yang et al., 2025b) improve multi-task learning capability by introducing multiple full LoRA experts, implicitly assuming that the limitations of single-LoRA adaptation stem from insufficient parameter capacity. However, this assumption overlooks whether the parameters within a single LoRA module are fully utilized. To investigate this issue, we conduct a simple sparsification experiment by randomly setting a fixed proportion of LoRA parameters to zero while fine-tuning the remaining parameters. This leads to our first observation:

***Observation I**: A single LoRA module contains substantial parameter redundancy.*

As shown in Figure 2, masking 50% of the LoRA parameters causes negligible performance degradation across different model scales and rank settings. This result suggests that the parameters within a single LoRA module are underutilized, and increasing the number of LoRA experts may therefore introduce additional redundancy rather than fundamentally improving adaptation capability.

Given this redundancy, a natural question arises: can task adaptation be performed at a finer granularity? To answer this question, we examine FFN layers, which are known to encode substantial linguistic and factual knowledge (Geva et al., 2021). Specifically, we analyze the magnitude distributions of intermediate hidden states across different tasks, leading to our second observation:

***Observation II**: FFN neurons exhibit clear task-specific activation patterns.*

As shown in Figure 3, similar tasks, such as the ARC benchmarks, exhibit highly similar activation patterns, whereas different tasks show clearly distinct patterns. This observation indicates that task specialization may naturally emerge within subspaces of the FFN intermediate dimension rather than uniformly across the entire layer. Together, these findings suggest that multi-task adaptation does not necessarily require multiple full LoRA modules. Instead, parameter updates can be localized to task-relevant subspaces within the FFN.

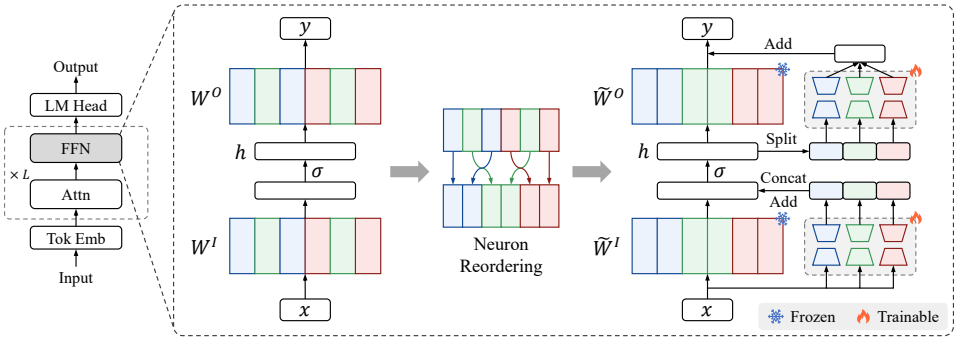


Figure 4: Architecture and workflow of DimLoRA. DimLoRA first reorders the FFN parameters to group neurons with similar functionalities together, and then assigns a dedicated LoRA expert to each neuron group, with each expert adapting only the parameters of its assigned neurons. Combined with sparse routing, this design enables selective updates to different functional components of the network for more targeted fine-tuning. The routing function is omitted from the figure for clarity.

3 DIMLORA

In this section, we present *DimLoRA*, a fine-grained expert architecture that enables localized parameter updates. It consists of four components: neuron reordering, fine-grained experts, task-aware routing, and mutual information regularization.

3.1 NEURON REORDERING

To encourage experts to learn more specialized functions, we first group FFN neurons with similar parameter representations. A standard FFN layer consists of two linear projections. Given an input $x \in \mathbb{R}^d$, the computation is

$$h = \sigma(\boldsymbol{W}^I x), \quad F(x) = \boldsymbol{W}^O h, \tag{4}$$

where $\boldsymbol{W}^I \in \mathbb{R}^{d_{ff} \times d}, \boldsymbol{W}^O \in \mathbb{R}^{d \times d_{ff}}$, d is the hidden dimension, d_{ff} is the intermediate dimension, and σ is the activation function. We define the i-th FFN neuron as $n_i = (\boldsymbol{W}^I_{i,:}, \boldsymbol{W}^O_{:,i})$, corresponding to the parameter vectors of the i-th row of $\boldsymbol{W}^I$ and the i-th column of $\boldsymbol{W}^O$.

Based on the intuition that neurons with similar parameters tend to exhibit similar activation patterns, we follow prior work (Zhang et al., 2022; Szatkowski et al., 2024) and apply balanced K-Means (Malinen & Fränti, 2014) with Euclidean distance to these neuron vectors to obtain N clusters. We then reorder the neurons according to the clustering results, placing neurons within the same cluster together. Let $\boldsymbol{P} \in \mathbb{R}^{d_{ff} \times d_{ff}}$ denote the corresponding permutation matrix. The reordered parameters are $\tilde{\boldsymbol{W}}^I = \boldsymbol{P}\boldsymbol{W}^I, \tilde{\boldsymbol{W}}^O = \boldsymbol{W}^O\boldsymbol{P}^\top$, which do not change the FFN output:

$$\tilde{\boldsymbol{W}}^O \sigma(\tilde{\boldsymbol{W}}^I x) = \boldsymbol{W}^O \boldsymbol{P}^\top \sigma(\boldsymbol{P}\boldsymbol{W}^I x) = \boldsymbol{W}^O \boldsymbol{P}^\top \boldsymbol{P} \sigma(\boldsymbol{W}^I x) = \boldsymbol{W}^O \sigma(\boldsymbol{W}^I x). \tag{5}$$

Finally, the neurons are partitioned into N contiguous groups, each containing $d_e = d_{ff}/N$ neurons. For FFNs with GLU variants (Shazeer, 2020), the same neuron reordering is applied consistently to the corresponding gating projections.

3.2 FINE-GRAINED LORA EXPERTS

After neuron reordering, we introduce a fine-grained LoRA design in which each expert adapts only the neurons within its assigned group rather than the entire layer, as illustrated in Figure 4.

For the input projection, each expert updates only the rows corresponding to its assigned neurons. The hidden representation is computed as:

$$h = \sigma\left(\tilde{\boldsymbol{W}}^I x + \mathrm{Concat}_{i=1}^{N}\left(w_i^I \boldsymbol{B}_i^I \boldsymbol{A}_i^I x\right)\right), \tag{6}$$

where $\boldsymbol{A}_i^I \in \mathbb{R}^{r \times d}$ and $\boldsymbol{B}_i^I \in \mathbb{R}^{d_e \times r}$ are the parameters of the i-th expert.

For the output projection, each expert updates only the columns corresponding to its assigned neuron group and processes only the associated slice of activations:

$$F(x) = \tilde{\boldsymbol{W}}^O h + \sum_{i=1}^{N} w_i^O \boldsymbol{B}_i^O \boldsymbol{A}_i^O h_{[(i-1)d_e:id_e-1]}, \tag{7}$$

where $\boldsymbol{A}_i^O \in \mathbb{R}^{r\times d_e}$, $\boldsymbol{B}_i^O \in \mathbb{R}^{d\times r}$.

This design ensures that each expert updates only a localized subset of FFN parameters. Specifically, $\boldsymbol{B}_i^I \boldsymbol{A}_i^I \in \mathbb{R}^{d_e\times d}$ updates only rows $[(i-1)d_e : id_e - 1]$ of $\tilde{\boldsymbol{W}}^I$, while $\boldsymbol{B}_i^O \boldsymbol{A}_i^O \in \mathbb{R}^{d\times d_e}$ updates only columns $[(i-1)d_e : id_e - 1]$ of $\tilde{\boldsymbol{W}}^O$.

Conventionally, attaching N full LoRA experts to an FFN layer introduces $2Nrd + 2Nrd_{ff}$ additional parameters. In contrast, DimLoRA requires only $2Nrd + 2Nrd_e = 2Nrd + 2rd_{ff}$. The number of additional parameters associated with d_{ff} is reduced by a factor of N. Given the large d_{ff} of modern LLMs, DimLoRA substantially reduces parameter overhead while preserving the benefits of expert specialization.

3.3 Task-Aware Routing

The routing function in MoE determines the weight assigned to each expert. The standard Top-k routing is defined as follows:

$$G(x) = \mathrm{TopK}\left(\mathrm{Softmax}\left(\boldsymbol{W}x\right)\right), \tag{8}$$

where $\boldsymbol{W} \in \mathbb{R}^{N\times d}$. The $\mathrm{TopK}(\cdot)$ operator retains the largest k elements and sets the remaining elements to 0. However, this design is suboptimal in multi-task settings because a single shared router struggles to capture diverse expert preferences across tasks, while assigning a separate router to each task introduces substantial parameter overhead.

To address this issue, we choose to augment the Top-k router with task-specific information. Given M tasks $\{T_1, ..., T_M\}$, we introduce a task embedding matrix $\boldsymbol{Z} \in \mathbb{R}^{M\times d_T}$, where d_T is the embedding dimension and $\boldsymbol{Z}_{j,:} \in \mathbb{R}^{d_T}$ represents task T_j. For input x from task T_j, the router becomes

$$G(x, j) = \mathrm{TopK}(\mathrm{Softmax}((\mathrm{RMSNorm}(\boldsymbol{W}x) + \mathrm{RMSNorm}(\boldsymbol{W}_T\boldsymbol{Z}_{j,:}))/\tau)), \tag{9}$$

where $\boldsymbol{W}_T \in \mathbb{R}^{N\times d_T}$. Here, the shared term $\boldsymbol{W}x$ captures token-level routing patterns, while $\boldsymbol{W}_T\boldsymbol{Z}_{j,:}$ introduces a task-specific routing bias. $\mathrm{RMSNorm}(\cdot)$ (Zhang & Sennrich, 2019) balances the scales of the two terms, while the temperature τ stabilizes expert activation. This design preserves token-level routing flexibility while incorporating task awareness with minimal parameter overhead. The additional parameter count is $d_T(M + N)$, which is negligible when $d_T \ll d$.

3.4 Mutual Information Regularization

In multi-task settings, desirable routing should encourage task-specific expert preferences while maintaining balanced expert utilization. To achieve both objectives, we introduce a mutual information regularizer between tasks and experts.

Assume that the model is deployed to task T_j with a uniform prior $P(T_j)$. For a given MoE layer, let $P(E_i \mid T_j)$ denote the probability that expert E_i is activated for task T_j, estimated by the average routing weights: $P(E_i \mid T_j) = \sum_{x\in T_j} G(x, j)_i/|T_j|$. The joint distribution is $P(E_i, T_j) = P(E_i \mid T_j)P(T_j)$, and the marginal expert distribution is $P(E_i) = \sum_{j=1}^{M} P(E_i, T_j)$. The mutual information between tasks and experts is defined as

$$I(T; E) = \sum_{i=1}^{N}\sum_{j=1}^{M} P(T_j, E_i) \log \frac{P(T_j, E_i)}{P(T_j)P(E_i)}. \tag{10}$$

This expression can be decomposed as

$$I(T; E) = \sum_{i=1}^{N}\sum_{j=1}^{M} P(T_j, E_i) \log P(T_j, E_i) - \sum_{j=1}^{M} P(T_j) \log P(T_j) - \sum_{i=1}^{N} P(E_i) \log P(E_i). \tag{11}$$

Table 1: Comparative performance of DimLoRA and different PEFT methods on single-domain benchmarks. "**Bold**" indicates the best result under each training setting. "%Params" denotes the proportion of trainable parameters.

Method	BoolQ	PIQA	SIQA	ARC-E	ARC-C	OBQA	HS	WG	Avg.	%Params↓
LLaMA3.1-8B	83.00	80.74	48.26	82.41	55.03	45.60	70.66	73.88	67.45	–
Single-Task Learning										
Full Fine-Tuning	89.42	**84.28**	58.65	87.04	**64.59**	57.20	73.63	84.77	74.95	100
Prompt Tuning	86.48	83.03	57.27	79.76	55.03	48.80	71.09	79.32	70.10	0.001
Prefix Tuning	62.17	81.28	57.06	85.19	61.26	51.60	65.57	79.32	67.93	0.843
P-Tuning	62.17	82.37	56.40	71.47	54.52	46.40	69.26	80.82	65.43	0.624
$(IA)^3$	83.39	81.56	53.17	83.38	55.21	47.40	72.23	76.72	69.13	0.007
AdaLoRA	83.61	81.50	58.14	82.74	55.21	46.00	73.34	80.43	70.12	0.264
LoRA	**90.03**	83.95	61.52	**87.84**	63.65	**58.40**	74.89	84.06	**75.54**	0.700
DimLoRA (Ours)	89.60	83.79	**61.67**	87.63	62.71	56.00	**75.14**	**86.43**	75.37	0.329
Multi-Task Learning										
Full Fine-Tuning	**90.15**	82.86	60.18	84.98	59.81	**53.80**	72.71	**84.93**	73.68	100
$\text{LoRA}_{(r=8)}$	89.02	82.64	59.98	85.02	61.18	49.20	72.29	84.14	72.93	0.176
$\text{LoRA}_{(r=16)}$	88.35	82.70	59.83	85.69	61.78	49.40	73.69	83.03	73.06	0.351
$\text{LoRA}_{(r=32)}$	88.87	82.86	59.72	85.73	61.43	51.00	73.14	83.98	73.34	0.700
DimLoRA (Ours)	89.17	**83.24**	**60.39**	**87.67**	**62.46**	51.40	**74.80**	**84.93**	**74.26**	0.329

The first term is the negative entropy of $P(T_j, E_i) = P(E_i \mid T_j)P(T_j)$, encouraging sharper conditional distributions $P(E_i \mid T_j)$ and thereby inducing task-specific expert preferences. The second term is constant under a fixed task prior. The third term corresponds to the entropy of $P(E_i)$, promoting balanced expert utilization. We use negative mutual information as the regularizer:

$$\mathcal{L} = \mathcal{L}_m - w_{MI} I(T; E), \tag{12}$$

where $\mathcal{L}_m$ is the language modeling loss and w_{MI} controls the regularization strength. For efficiency, $P(E \mid T)$ and the mutual information are estimated within each training batch.

4 Experiments

4.1 Experimental Setup

Models and Benchmarks. We evaluate DimLoRA on LLaMA3.1-8B (Grattafiori et al., 2024) and Qwen3-14B-Base (Yang et al., 2025a) in both single-domain and multi-domain settings. • *Single-domain*. We use eight commonsense reasoning tasks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), ARC-Easy (Clark et al., 2018), ARC-Challenge (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), HellaSwag (Zellers et al., 2019), and Winogrande (Sakaguchi et al., 2021). • *Multi-domain*. We construct a heterogeneous multi-domain setting spanning five domains: 1) General: trained on auxiliary multiple-choice data and evaluated on MMLU (Hendrycks et al., 2021); 2) Math: trained on the GSM8K (Cobbe et al., 2021) training set and evaluated on its test set; 3) Code: trained on the Python subset of CodeAlpaca (Chaudhary, 2023) and evaluated on HumanEval (Chen et al., 2021); 4) Law: trained on Legal-Dataset-4.4k-Instructions (Nisaar, 2023) and evaluated on AGIEval-LSAT-RC (Zhong et al., 2024); 5) Medicine: trained on MedMCQA (Pal et al., 2022) and evaluated on medical tasks in MMLU. All evaluations are conducted using the Language Model Evaluation Harness[1]. Additional implementation details are provided in Appendix A.

Baselines. • First, we compare DimLoRA with representative PEFT methods on single-domain datasets: 1) Full Fine-Tuning; 2) Prompt Tuning (Lester et al., 2021); 3) Prefix Tuning (Li & Liang, 2021); 4) P-Tuning (Liu et al., 2024c); 5) $(IA)^3$ (Liu et al., 2022); 6) AdaLoRA (Zhang et al., 2023); and 7) LoRA (Hu et al., 2022). • Second, we compare DimLoRA with representative LoRA-MoE methods on multi-domain datasets: 1) MultiLoRA (Wang et al., 2023), which uses learnable scaling vectors to weight experts; 2) MoLORA (Zadouri et al., 2024), which introduces a routing function for expert selection; 3) MOELoRA (Liu et al., 2024a), which incorporates task-aware signals into routing; 4) HydraLoRA (Tian et al., 2024), which shares the LoRA $\boldsymbol{A}$ matrix across experts; 5)

[1] https://github.com/EleutherAI/lm-evaluation-harness

Table 2: Comparative performance of DimLoRA and various LoRA-MoE methods on multi-domain benchmarks. "#A" and "#B" denote the numbers of $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices.

Method	General	Math	Code	Law	Medicine	Avg.	%Params↓	#A	#B
LLaMA3.1-8B	64.12	50.27	36.59	36.43	69.67	51.41	–	–	–
LoRA (ICLR'22)	64.20	59.06	36.59	41.26	68.15	53.85	0.700	1	1
MultiLoRA (arXiv'23)	65.07	55.19	40.24	43.49	71.58	55.12	0.700	4	4
MoLORA (ICLR'24)	64.72	58.98	42.07	42.75	69.23	55.55	0.736	4	4
MOELoRA (SIGIR'24)	65.33	58.98	42.07	39.41	71.68	55.49	0.700	4	4
HydraLoRA (NeurIPS'24)	65.15	**59.29**	39.63	40.15	71.07	55.06	0.410	1	3
MTL-LoRA (AAAI'25)	64.96	56.56	39.63	42.01	69.55	54.54	0.384	1	3
SMoRA (KDD'26)	65.08	58.61	40.24	41.64	70.25	55.16	0.493	16	16
DimLoRA (Ours)	**65.47**	58.83	**42.68**	**43.87**	**72.84**	**56.74**	**0.329**	4	4
Qwen3-14B-Base	77.07	81.20	37.81	70.63	83.18	69.98	–	–	–
LoRA (ICLR'22)	78.27	85.22	75.61	68.77	83.83	78.34	0.582	1	1
MultiLoRA (arXiv'23)	77.76	82.87	73.17	70.63	83.48	77.58	0.582	4	4
MoLORA (ICLR'24)	78.44	84.46	77.44	66.91	83.87	78.22	0.612	4	4
MOELoRA (SIGIR'24)	78.45	86.35	73.17	71.38	**84.52**	78.77	0.582	4	4
HydraLoRA (NeurIPS'24)	78.44	85.44	76.22	64.68	84.00	77.76	0.341	1	3
MTL-LoRA (AAAI'25)	78.37	84.46	75.61	75.09	84.44	79.59	0.319	1	3
SMoRA (KDD'26)	78.25	86.73	70.12	62.83	84.13	76.41	0.411	16	16
DimLoRA (Ours)	**78.46**	**86.88**	**78.05**	**78.07**	84.09	**81.11**	**0.276**	4	4

Table 3: Ablation changes relative to DimLoRA across multiple benchmarks on LLaMA3.1-8B.

Variant	Gen.	Math	Code	Law	Med.	Avg.
DimLoRA	0	0	0	0	0	0
w/o Reordering	-0.30	-1.44	-1.83	0	-2.06	-1.13
w/o Sparsity	-0.29	-0.08	-4.27	-0.37	-3.56	-1.71
w/o Task-Aware	-0.48	-1.67	-1.83	1.49	-1.72	-0.84
w/o MI Loss	-0.36	-2.12	-1.22	-2.97	0.18	-1.30
Random	0	-1.44	-2.44	-3.35	-3.57	-2.16
Co-Activation	0.16	0	0.61	0.37	-1.50	-0.07

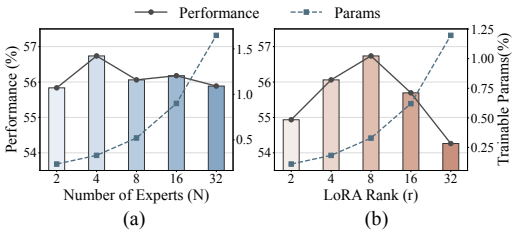


Figure 5: Hyperparameter study results: (a) number of experts N; (b) LoRA rank r.

MTL-LoRA (Yang et al., 2025b), which further introduces task-specific diagonal matrices to decouple the parameter space; and 6) SMoRA (Zhao et al., 2026), which treats each LoRA rank as an independent expert. All methods use the same training data and training budget.

4.2 Overall Performance

We first evaluate single-task learning on commonsense reasoning benchmarks. As shown in Table 1, LoRA achieves the best overall performance among PEFT methods, while DimLoRA incurs only a 0.23% performance drop despite activating just 25.93% of the adapter parameters per token, confirming the substantial parameter redundancy observed in Section 2.3. We then combine the training sets of all eight tasks for multi-task learning. As shown in the lower part of Table 1, DimLoRA consistently outperforms LoRA across all tasks, suggesting that sparse updates effectively alleviate inter-task interference. We further evaluate DimLoRA in a heterogeneous multi-domain setting. As shown in Table 2, DimLoRA achieves the best performance on most tasks with substantially lower parameter overhead, demonstrating that fine-grained subspace decoupling and selective updates provide a better efficiency–performance trade-off than simply increasing the parameter count.

4.3 Ablation Study

Impact of Components. We evaluate each component by removing it individually in Table 3: *w/o Reordering* removes neuron reordering, *w/o Sparsity* activates all experts, *w/o Task-Aware* uses standard Top-k routing, and *w/o MI Loss* removes mutual information regularization. DimLoRA consistently outperforms all ablated variants, confirming the contribution of each component. No-

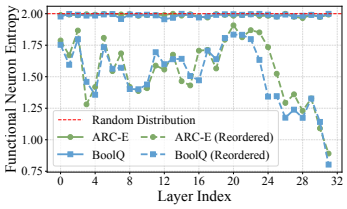


Figure 6: Entropy of functional neuron distributions across different expert groups.

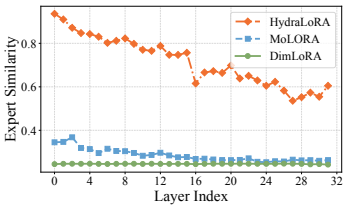


Figure 7: Average expert parameter similarity across various model layers.

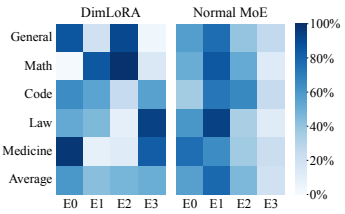


Figure 8: Activation frequency across tasks (**y-axis**) and four experts (**x-axis**).

tably, *w/o Task-Aware* still surpasses other LoRA-MoE baselines in Table 2 on average, suggesting that the dimension-level expert architecture remains effective without explicit task identities.

Impact of Neuron Reordering. We compare parameter-based clustering with random grouping and co-activation-based clustering. Random grouping assigns neurons uniformly at random, while co-activation-based clustering constructs a neuron graph on a small calibration set with edge weight $w_{i,j} = \sum_{\boldsymbol{x}} |a_i(\boldsymbol{x}) \cdot a_j(\boldsymbol{x})|$, where $a_i(\boldsymbol{x})$ denotes the activation of neuron n_i, and partitions it using graph partitioning (Karypis & Kumar, 1998). As shown in the lower part of Table 3, random grouping substantially degrades performance, whereas co-activation-based clustering differs from parameter-based clustering by only 0.07% on average. This confirms the effectiveness of parameter-based clustering without requiring additional calibration data.

Impact of Expert Count and LoRA Rank. We study the effects of the number of experts N and LoRA rank r in Figure 5. As shown in Figure 5a, performance improves from $N = 2$ to $N = 4$ but degrades at $N = 8$, likely due to increased routing complexity and optimization difficulty. We therefore use $N = 4$ by default. Similarly, Figure 5b shows that performance improves from $r = 2$ to $r = 8$ but declines at $r = 16$. Larger ranks also incur additional parameter and computational costs. We therefore use $r = 8$ by default for a favorable performance–efficiency trade-off.

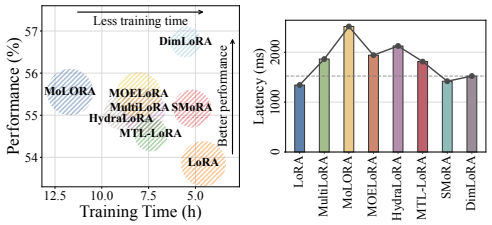
4.4 Analysis

Neuron Reordering Analysis. To further verify whether neuron reordering groups functionally similar neurons together, we measure neuron importance using the first-order loss change criterion (Molchanov et al., 2019) and select the top 1% of neurons for each task as functional neurons. We then compute the entropy of their distribution across neuron groups. As shown in Figure 6, reordering substantially reduces the entropy for both ARC-E and BoolQ, supporting its effectiveness in concentrating functionally similar neurons for subsequent localized adaptation.

Expert Similarity Analysis. To further understand expert specialization, we analyze the average pairwise parameter similarity among experts across layers. As shown in Figure 7, DimLoRA consistently exhibits lower expert similarity than HydraLoRA and MoLORA across layers. This result indicates that parameter sharing in HydraLoRA and full-space adaptation in MoLORA can lead to more redundant expert learning. In contrast, DimLoRA assigns each expert to a targeted parameter subspace, encouraging more distinct specialization and reducing potential parameter conflicts.

Activation Analysis. To demonstrate the effect of mutual information regularization, we visualize expert activation frequency in Figure 8. DimLoRA and Normal MoE use identical expert architectures and settings, differing only in their routing mechanisms. DimLoRA exhibits sharper and sparser activation patterns, indicating stronger task-specific expert preferences. Meanwhile, the average expert utilization remains relatively balanced. In contrast, Normal MoE exhibits similar activation patterns across tasks, with certain experts (e.g., E3) being significantly underutilized.

Efficiency Analysis. We evaluate the training and inference efficiency of DimLoRA. As shown in Figure 9a, DimLoRA achieves a favorable efficiency–performance trade-off among LoRA-MoE methods. Although standard LoRA has lower training latency without routing overhead, it yields inferior performance due to inter-task interference. As shown in Figure 9b, DimLoRA also achieves low inference latency among LoRA-MoE baselines.



(a) (b)

Figure 9: (a) Training time vs. performance (area $\propto$ trainable parameters). (b) Inference latency at batch size 8 and sequence length 1024.

Table 4: Continual learning results under different training orders. DimLoRA consistently outperforms LoRA across all task orders.

Order	Method	Task1	Task2	Task3	Task4	Avg.
1	LoRA	67.64	48.29	79.05	90.00	71.25
	DimLoRA	85.06	60.50	83.84	89.91	**79.83**
2	LoRA	73.43	82.20	60.90	59.47	69.00
	DimLoRA	75.40	86.20	60.65	63.40	**71.41**
3	LoRA	85.10	50.40	75.14	61.18	67.96
	DimLoRA	86.83	50.80	75.17	63.23	**69.01**
4	LoRA	58.19	83.98	80.09	84.44	76.68
	DimLoRA	60.15	82.72	85.56	84.28	**78.18**

Continual Learning. We further evaluate whether sparse updates mitigate catastrophic forgetting in continual learning. Following standard protocols, we train the model sequentially on multiple randomly ordered task sequences and evaluate its final performance on previously seen tasks. As shown in Table 4, DimLoRA consistently outperforms LoRA across all task orders, indicating better retention of prior knowledge. Detailed experimental settings are provided in Appendix D.

5 Related Work

5.1 Parameter-Efficient Fine-Tuning

PEFT reduces the cost of adapting LLMs while achieving performance comparable to full fine-tuning. Existing methods mainly fall into three categories: *additive methods*, which introduce trainable modules such as adapters (Houlsby et al., 2019; Lin et al., 2020) and prompts (Lester et al., 2021; Li & Liang, 2021; Liu et al., 2024c); *selective methods*, which update only a subset of pre-trained parameters (Ben Zaken et al., 2022; Zhao et al., 2020; Sung et al., 2021; Zhang et al., 2025); and *reparameterization methods*, represented by LoRA (Hu et al., 2022), which employ low-rank transformations for efficient adaptation. Recent LoRA variants further improve efficiency and performance through dynamic rank allocation (Valipour et al., 2023; Zhang et al., 2023), quantization (Dettmers et al., 2023) and improved initialization (Meng et al., 2024; Liu et al., 2024b). Despite their success, most PEFT methods are designed for single-task adaptation and suffer from severe inter-task interference in multi-task settings.

5.2 Multi-LoRA Architectures

To mitigate inter-task interference in multi-task settings, recent studies introduce multiple LoRA modules. Early approaches follow a retrieve-and-combine paradigm, where task-specific adapters are independently trained and dynamically aggregated at inference time (Chronopoulou et al., 2023; Huang et al., 2024). More recent approaches integrate LoRA with MoE architectures. MultiLoRA (Wang et al., 2023) combines LoRA experts using learnable weights, while MoLORA (Zadouri et al., 2024) employs routing networks for expert selection. HydraLoRA (Tian et al., 2024) improves parameter efficiency by sharing the $\boldsymbol{A}$ matrix across experts, and MTL-LoRA (Yang et al., 2025b) further decouples task subspaces using task-specific diagonal matrices. Despite these advances, existing methods still operate at the granularity of the entire parameter space. In contrast, our method introduces dimension-level experts for more targeted specialization with lower parameter overhead.

6 Conclusion

In this paper, we propose DimLoRA, a fine-grained LoRA expert architecture for parameter-efficient multi-task adaptation. By decomposing parameter space along the FFN intermediate dimension, DimLoRA introduces dimension-level experts to enable more targeted adaptation. Extensive experiments demonstrate that DimLoRA improves multi-task performance with significantly fewer parameters. We hope this work inspires future research on fine-grained expert architectures and more efficient utilization of LoRA parameters.

AI USE STATEMENT

In this work, we used generative AI tools to search for and identify relevant prior work. We have not used generative AI tools for research ideation, and dataset generation and construction are not applicable to this work. Additionally, we used generative AI tools to assist with language polishing. We have reviewed all AI-assisted work. Specifically, we carefully examined the related work identified with AI assistance and manually revised the AI-polished manuscript sentence by sentence to ensure that the wording accurately reflected our intended meaning. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work focuses on parameter-efficient fine-tuning and mixture-of-experts techniques for improving the efficiency of multi-task adaptation in LLMs. Our experiments are conducted on publicly available models and benchmark datasets and do not involve human-subject studies or the collection of personally identifiable information. The proposed method is intended to reduce parameter and computational overhead during model adaptation and does not introduce new capabilities beyond those of the underlying pretrained models. Nevertheless, as with other LLM adaptation techniques, the resulting models may inherit biases, harmful behaviors, or other limitations from the pretrained models and training data. We therefore encourage responsible use of the method in accordance with applicable licenses, dataset usage policies, and relevant legal and ethical requirements.

REPRODUCIBILITY STATEMENT

We have made extensive efforts to ensure the reproducibility of this work. The experimental setup, including the evaluated models, benchmarks, baseline methods, and evaluation protocols, is described in Section 4.1, while detailed training configurations and implementation settings are provided in Appendix A. All models and datasets used in our experiments are publicly available, and evaluation is conducted with publicly released code. We also provide an anonymous implementation of DimLoRA, including the code required to reproduce the main experiments, at `https://anonymous.4open.science/r/DimLoRA`.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pp. 7319–7328, 2021.

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 (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL `https://aclanthology.org/2022.acl-short.1/`.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, Apr. 2020. doi: 10.1609/aaai.v34i05.6239. URL `https://ojs.aaai.org/index.php/AAAI/article/view/6239`.

Sahil Chaudhary. Code alpaca: an instruction-following llama model for code generation. 2023. URL `https://github.com/sahil280114/codealpaca`.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde 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, Josh 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, 2021. URL https://arxiv.org/abs/2107.03374.

Alexandra Chronopoulou, Matthew Peters, Alexander Fraser, and Jesse Dodge. AdapterSoup: Weight averaging to improve generalization of pretrained language models. In Andreas Vlachos and Isabelle Augenstein (eds.), *Findings of the Association for Computational Linguistics: EACL 2023*, pp. 2054–2063, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.153. URL https://aclanthology.org/2023.findings-eacl.153/.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1feb87871436031bdc0f2beaa62a049b-Paper-Conference.pdf.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

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 (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

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 *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic loRA composition. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=TrloAXEJ2B.

George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998. doi: 10.1137/S1064827595287997. URL https://doi.org/10.1137/S1064827595287997.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. Exploring versatile generative language model via parameter-efficient transfer learning. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 441–459, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.41. URL https://aclanthology.org/2020.findings-emnlp.41/.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022.

Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. When moe meets llms: Parameter efficient fine-tuning for multi-task medical applications. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 1104–1114, New York, NY, USA, 2024a. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657722. URL https://doi.org/10.1145/3626772.3657722.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024b. URL https://openreview.net/forum?id=3d5CIRG1n2.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *AI Open*, 5:208–215, 2024c. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2023.08.012. URL https://www.sciencedirect.com/science/article/pii/S2666651023000141.

Mikko I Malinen and Pasi Fränti. Balanced k-means for clustering. In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*, pp. 32–41. Springer, 2014.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 121038–121072. Curran Associates, Inc., 2024. doi: 10.52202/079017-3846. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/db36f4d603cc9e3a2a5e10b93e6428f2-Paper-Conference.pdf.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang,

Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Agharia Nisaar. Indian law ai: Fine-tuning falcon-7b and llama 2 language models. 2023. URL https://huggingface.co/datasets/nisaar/LLAMA2_Legal_Dataset_4.4k_Instructions.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In Gerardo Flores, George H Chen, Tom Pollard, Joyce C Ho, and Tristan Naumann (eds.), *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pp. 248–260. PMLR, 07–08 Apr 2022. URL https://proceedings.mlr.press/v174/pal22a.html.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Comput. Surv.*, 58(5), November 2025. ISSN 0360-0300. doi: 10.1145/3735633. URL https://doi.org/10.1145/3735633.

Yi-Lin Sung, Varun Nair, and Colin Raffel. Training neural networks with fixed sparse masks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 24193–24205. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/cb2653f548f8709598e8b5156738cc51-Paper.pdf.

Filip Szatkowski, Bartosz Wójcik, Mikoł aj Piórczyński, and Simone Scardapane. Exploiting activation sparsity with dense to dynamic-k mixture-of-experts conversion. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 43245–43273. Curran Associates, Inc., 2024. doi: 10.52202/079017-1370. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4c2092ec0b1370cce3fb5965ab255fae-Paper-Conference.pdf.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 9565–9584. Curran Associates, Inc., 2024. doi: 10.52202/079017-0304. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/123fd8a56501194823c8e0dca00733df-Paper-Conference.pdf.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In Andreas

Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3274–3287, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.239. URL https://aclanthology.org/2023.eacl-main.239/.

Yiming Wang, Yu Lin, Xiaodong Zeng, and Guannan Zhang. Multilora: Democratizing lora for better multi-task learning, 2023. URL https://arxiv.org/abs/2311.11501.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Yaming Yang, Dilxat Muhtar, Yelong Shen, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Weiwei Deng, Feng Sun, Qi Zhang, Weizhu Chen, and Yunhai Tong. Mtl-lora: Low-rank adaptation for multi-task learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(20):22010–22018, Apr. 2025b. doi: 10.1609/aaai.v39i20.35509. URL https://ojs.aaai.org/index.php/AAAI/article/view/35509.

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. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=EvDeiLv7qc.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. LoRI: Reducing cross-task interference in multi-task low-rank adaptation. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=b8cW86QcOD.

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*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. MoEfication: Transformer feed-forward layers are mixtures of experts. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 877–890, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.71. URL https://aclanthology.org/2022.findings-acl.71/.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. Masking as an efficient alternative to finetuning for pretrained language models. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2226–2241, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.174. URL https://aclanthology.org/2020.emnlp-main.174/.

Ziyu Zhao, Yixiao Zhou, Xin Yu, Zhi Zhang, Didi Zhu, Tao Shen, Zexi Li, Jinluan Yang, Xuwu Wang, Jing Su, Kun Kuang, Zhongyu Wei, Fei Wu, and Yu Cheng. Each rank could be an expert: Single-ranked mixture of experts lora for multi-task learning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, KDD '26, pp. 1998–2009, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722585. doi: 10.1145/3770854.3780222. URL https://doi.org/10.1145/3770854.3780222.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. AGIEval: A human-centric benchmark for evaluating foundation models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2299–2314, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.149. URL https://aclanthology.org/2024.findings-naacl.149/.

A Training and Implementation Details

A.1 Datasets

Single-Domain. This domain comprises eight commonsense reasoning benchmarks. We strictly use the official training splits for model training and evaluate on the corresponding test splits to avoid data leakage. We first fine-tune models on each dataset independently to conduct single-task experiments. Subsequently, we construct a multi-task setting by mixing all eight datasets.

1. **BoolQ** is a question answering dataset consisting of yes/no questions, collected from anonymized, voluntary queries issued to the Google search engine. Each instance is a tuple (question, passage, answer).
2. **PIQA** focuses on physical commonsense reasoning in everyday scenarios, often favoring atypical solutions. The dataset is inspired by instructables.com, which provides procedural guidance for building, crafting, or manipulating objects using common materials. Each instance consists of a question, multiple candidate solutions, and a correct label.
3. **SocialIQA (SIQA)** evaluates social commonsense reasoning about human actions and their social implications. Each example includes a context, a question, multiple choices, and the correct label.
4. **ARC-Easy** and **ARC-Challenge** datasets consist of grade-school science questions in a multiple-choice format. Each instance contains a question, answer choices, and the correct label.
5. **OpenBookQA** evaluates advanced question answering requiring multi-step reasoning, integration of external commonsense knowledge, and deep text understanding. Each example consists of a question, answer options, and the correct label.
6. **HellaSwag** is a commonsense natural language inference benchmark. Each instance presents the beginning of a scenario describing everyday activities, followed by multiple candidate endings. The model is required to select the most plausible continuation. Each example consists of a context, answer choices, and the correct label.
7. **Winogrande** is a binary-choice cloze-style dataset that requires commonsense reasoning to resolve ambiguous pronouns or sentence completions. Each instance consists of a sentence with a blank, two candidate options, and the correct label.

Multi-Domain. We further evaluate the generalization ability of DimLoRA under a multi-domain setting by constructing a mixed dataset spanning five distinct domains.

1. **General.** We train on the auxiliary training split of MMLU and evaluate on its test set to assess general-purpose capabilities. MMLU is a large-scale multitask benchmark comprising multiple-choice questions across 57 subjects, including elementary mathematics, U.S. history, computer science, law, and more.
2. **Math.** We adopt GSM8K as the benchmark for mathematical reasoning. GSM8K is a high-quality, linguistically diverse dataset of grade-school math word problems, designed to evaluate multi-step reasoning in question answering on basic mathematical problems.

3. **Code.** We use the Python subset of CodeAlpaca for training and evaluate on HumanEval. CodeAlpaca is an instruction-tuning dataset for code generation, while HumanEval consists of 164 Python programming problems designed to assess functional correctness of generated code.
4. **Law.** We train on the open-source instruction-tuning dataset Legal-Dataset-4.4k-Instructions from Hugging Face and evaluate on AGIEval-LSAT-RC. The latter is derived from the Law School Admission Test (LSAT) and is designed to measure the reasoning and analytical skills of prospective law students.
5. **Medicine.** We use MedMCQA as the training dataset and evaluate on the medical subsets of MMLU, specifically clinical_knowledge, professional_medicine, and college_medicine. MedMCQA is a large-scale multiple-choice question answering dataset derived from real-world medical entrance examinations.

A.2 BASELINES

PEFT Methods

1. **Full Fine-Tuning** is the default strategy for instruction tuning. The model is initialized with pretrained weights, and all parameters are updated via gradient descent.
2. **Prompt Tuning** prepends task-specific continuous prompts to the input while keeping the pretrained model parameters frozen. Only the prompt embeddings are optimized, enabling lightweight adaptation.
3. **Prefix Tuning** introduces a sequence of trainable task-specific vectors that are prepended to the input at every layer of the transformer. The pretrained model remains frozen, and only the prefix parameters are learned.
4. **P-Tuning** augments the input with trainable prompt embeddings generated by a prompt encoder, which automatically learns effective prompt representations, eliminating the need for manual prompt design.
5. **(IA)**^{3} enhances parameter efficiency by introducing learnable scaling vectors into transformer layers to modulate intermediate activations, thereby achieving competitive performance with negligible parameter overhead and minimal impact on inference latency.
6. **AdaLoRA** dynamically allocates the parameter budget across layers and weight matrices based on their importance, in contrast to LoRA, which distributes ranks uniformly across modules.
7. **LoRA** assumes that parameter updates during fine-tuning lie in a low-rank subspace and models these updates as the product of two low-rank matrices. The pretrained weights are frozen, and only the low-rank factors are trained, significantly reducing computational cost and memory footprint.

LoRA-MoE Methods

1. **MultiLoRA** horizontally scales LoRA modules to reduce parameter dependency. Specifically, it divides LoRA along the rank dimension, introduces learnable scaling factors, and modifies the parameter initialization to enhance the expressiveness of LoRA modules.
2. **MoLORA** proposes an extremely parameter-efficient MoE approach by combining an MoE architecture with lightweight LoRA-based experts. Instead of using scaling factors, it employs a routing function to dynamically control the contribution of different experts.
3. **MOELoRA** extends standard LoRA experts by introducing a task-aware gating function for all MOELoRA layers, which dynamically regulates the contribution of each expert and enables the model to produce task-specific parameter adaptations.
4. **HydraLoRA** is motivated by empirical observations that some LoRA parameters learn the commonalities across data, while others focus on the unique aspects of each intrinsic component. Based on this insight, it proposes an asymmetric LoRA structure that eliminates the need for domain expertise. In particular, multiple LoRA experts share a common $\boldsymbol{A}$ matrix, thereby improving parameter efficiency without sacrificing expressiveness.

Algorithm 1 DimLoRA FFN Layer

Input: $x \in \mathbb{R}^d$, task id j
Output: $y \in \mathbb{R}^d$

1: // Input projection
2: $G^I \leftarrow \text{Router}(x, j)$ // $G^I \in \mathbb{R}^N$
3: $h \leftarrow \tilde{W}^I x$ // $h \in \mathbb{R}^{d_{ff}}$
4: **for** $i = 1$ to N **do**
5: $\quad \mathcal{S}_i \leftarrow [(i-1)d_e : id_e - 1]$
6: $\quad h_{\mathcal{S}_i} \leftarrow h_{\mathcal{S}_i} + G_i^I \cdot B_i^I A_i^I x$
7: **end for**
8: $h \leftarrow \sigma(h)$
9:
10: // Output projection
11: $G^O \leftarrow \text{Router}(h, j)$ // $G^O \in \mathbb{R}^N$
12: $y \leftarrow \tilde{W}^O h$
13: **for** $i = 1$ to N **do**
14: $\quad \mathcal{S}_i \leftarrow [(i-1)d_e : id_e - 1]$
15: $\quad y \leftarrow y + G_i^O \cdot B_i^O A_i^O h_{\mathcal{S}_i}$
16: **end for**
17: **return** y

5. **MTL-LoRA** extends LoRA by introducing task-adaptive parameters that differentiate task-specific information and capture shared knowledge within a low-dimensional subspace. Similar to HydraLoRA, it shares the $\boldsymbol{A}$ matrix across experts; however, it further incorporates a learnable diagonal matrix for each task to decouple the parameter space.
6. **SMoRA** embeds MoE into LoRA by treating each rank as an independent expert. With a dynamic rank-wise activation mechanism, it facilitates a flexible composition of knowledge, enabling the model to learn deeper and more diverse features while mitigating task conflicts.

A.3 TESTBED

All experiments with PEFT methods are conducted on a local machine equipped with two NVIDIA RTX 4090 GPUs (24GB VRAM each), connected via PCIe 4.0 ×16 with a peak bandwidth of 32 GB/s. In contrast, Full Fine-Tuning is performed on a cloud server with two NVIDIA A800 GPUs (80GB VRAM each), interconnected via NVLink with a maximum bandwidth of 400 GB/s. The software environment consists of Ubuntu 20.04 LTS, CUDA 12.6, NCCL 2.21, PyTorch 2.6.0, and Transformers 4.51.0.

A.4 HYPERPARAMETERS

We set the number of experts to $N = 4$, with $k = 2$ experts activated per input. Each LoRA expert uses a rank of $r = 8$, with $\alpha = 16$ and a dropout rate of 0.05. For fair comparison, all methods adapt only the FFN layers. The number of tasks M is 8 in the single-domain setting and 5 in the multi-domain setting. The task embedding dimension is set to $d_T = 64$. The temperature parameter τ in the routing function is fixed at 20. We tune the coefficient of the mutual information regularizer w_{MI} via hyperparameter search. The optimal value is 1×10^{-2} for LLaMA3.1-8B and 5×10^{-5} for Qwen3-14B-Base. We employ the AdamW optimizer with its default configuration. The maximum learning rate is set to 3.0×10^{-4}, and it is decayed to 1.0×10^{-6} following a cosine schedule. We apply a linear warmup over the first 5% of the training steps.

B PSEUDOCODE

The pseudo-code for the FFN layer computation in DimLoRA is presented in Algorithm 1. Lines 1–8 correspond to the input projection layer, while Lines 10–17 correspond to the output projection layer. Here, $\mathcal{S}_i$ denotes the subset of intermediate neurons adapted by the i-th expert.

C GRADIENT LOCALITY OF DIMLORA

We show that DimLoRA induces slice-wise gradient locality: when only one expert is activated, gradients are restricted to the LoRA parameters associated with the corresponding FFN neuron slice.

Consider a DimLoRA FFN layer with N fine-grained experts. Let the i-th expert correspond to the neuron slice

$$\mathcal{S}_i = [(i-1)d_e : id_e - 1].$$

Define the effective LoRA updates as

$$\Delta \boldsymbol{W}_i^I = \boldsymbol{B}_i^I \boldsymbol{A}_i^I \in \mathbb{R}^{d_e \times d},$$
$$\Delta \boldsymbol{W}_i^O = \boldsymbol{B}_i^O \boldsymbol{A}_i^O \in \mathbb{R}^{d \times d_e}.$$

Let $\boldsymbol{R}_i \in \mathbb{R}^{d_e \times d_{ff}}$ be the slice-selection matrix that extracts the coordinates indexed by $\mathcal{S}_i$ from a vector in $\mathbb{R}^{d_{ff}}$. Correspondingly, $\boldsymbol{R}_i^\top$ pads a vector in $\mathbb{R}^{d_e}$ back to the full FFN hidden dimension, with zeros outside $\mathcal{S}_i$.

For the input projection, DimLoRA concatenates the expert outputs across neuron slices:

$$\boldsymbol{h} = \sigma\left(\tilde{\boldsymbol{W}}^I \boldsymbol{x} + \mathrm{Concat}_{k=1}^N \left(w_k \Delta \boldsymbol{W}_k^I \boldsymbol{x}\right)\right).$$

Equivalently, this concatenation can be written in slice-padded form as

$$\boldsymbol{h} = \sigma\left(\tilde{\boldsymbol{W}}^I \boldsymbol{x} + \sum_{k=1}^N w_k \boldsymbol{R}_k^\top \Delta \boldsymbol{W}_k^I \boldsymbol{x}\right).$$

This summation is only a compact representation of the concatenation operation, since different $\boldsymbol{R}_k^\top$ write to disjoint neuron slices.

For the output projection, each expert consumes only its corresponding hidden slice:

$$\boldsymbol{y} = \tilde{\boldsymbol{W}}^O \boldsymbol{h} + \sum_{k=1}^N w_k \Delta \boldsymbol{W}_k^O \boldsymbol{R}_k \boldsymbol{h}.$$

Suppose only the i-th expert is activated, i.e.,

$$w_i = 1, \qquad w_k = 0, \quad \forall k \neq i.$$

Then the forward computation reduces to

$$\boldsymbol{h} = \sigma\left(\tilde{\boldsymbol{W}}^I \boldsymbol{x} + \boldsymbol{R}_i^\top \Delta \boldsymbol{W}_i^I \boldsymbol{x}\right),$$
$$\boldsymbol{y} = \tilde{\boldsymbol{W}}^O \boldsymbol{h} + \Delta \boldsymbol{W}_i^O \boldsymbol{R}_i \boldsymbol{h}.$$

Therefore, the output $\boldsymbol{y}$ depends only on the effective updates of the active expert, namely $\Delta \boldsymbol{W}_i^I$ and $\Delta \boldsymbol{W}_i^O$. For any inactive expert $k \neq i$, neither $\Delta \boldsymbol{W}_k^I$ nor $\Delta \boldsymbol{W}_k^O$ participates in the forward computation. Hence,

$$\frac{\partial \boldsymbol{y}}{\partial \Delta \boldsymbol{W}_k^I} = \boldsymbol{0}, \qquad \frac{\partial \boldsymbol{y}}{\partial \Delta \boldsymbol{W}_k^O} = \boldsymbol{0}.$$

By the chain rule, for any loss $\mathcal{L}$ computed from $\boldsymbol{y}$, we have

$$\frac{\partial \mathcal{L}}{\partial \Delta \boldsymbol{W}_k^I} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \Delta \boldsymbol{W}_k^I} = \boldsymbol{0},$$
$$\frac{\partial \mathcal{L}}{\partial \Delta \boldsymbol{W}_k^O} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \Delta \boldsymbol{W}_k^O} = \boldsymbol{0}.$$

Thus, inactive experts receive zero gradients.

For the active expert, the output explicitly depends on both $\Delta \boldsymbol{W}_i^I$ and $\Delta \boldsymbol{W}_i^O$. Therefore,

$$\frac{\partial \mathcal{L}}{\partial \Delta \boldsymbol{W}_i^I} \neq \boldsymbol{0}, \qquad \frac{\partial \mathcal{L}}{\partial \Delta \boldsymbol{W}_i^O} \neq \boldsymbol{0}.$$

This proves that when only the i-th DimLoRA expert is activated, gradients are restricted to the LoRA parameters associated with the i-th neuron slice, while all inactive experts receive zero gradients. Therefore, DimLoRA induces slice-wise localized parameter updates.

Table 5: Four different orders of task sequences used for continual learning experiments.

Order	Task Sequence
1	ARC-E→ARC-C→PIQA→BoolQ
2	HellaSwag→ARC-E→SIQA→ARC-C
3	ARC-E→OBQA→HellaSwag→ARC-C
4	ARC-C→Winogrande→ARC-E→PIQA

D Continual Learning Details

Experimental Setup. Continual learning (Shi et al., 2025) aims to develop learning algorithms to accumulate knowledge on non-stationary data. In supervised continual learning, a sequence of tasks $\{T_1, ..., T_M\}$ arrives in a streaming fashion. Each task $T_j = \{(x_i^j, y_i^j)\}_{i=1}^{|T_j|}$ contains a separate target dataset, where $x_i^j \in \mathcal{X}_j$ and $y_i^j \in \mathcal{Y}_j$. A single model is required to adapt to these tasks sequentially, with access only to T_j at the j-th stage. Formally, given a predictive model h_θ parameterized by θ, continual learning seeks to optimize the following objective across all tasks:

$$\max_{\theta} \sum_{j=1}^{M} \sum_{(x,y)\in T_j} \log p_\theta(y \mid x).$$

We randomly sample four tasks from the commonsense reasoning domain to construct each task sequence. All task orders are listed in Table 5. We compare DimLoRA with standard LoRA.

Evaluation Metric. Let $a_{i,j}$ denote the test accuracy on the i-th task after training on the j-th task. We use average accuracy to evaluate knowledge retention. It is defined as the average accuracy across all tasks after training on the final task: $\frac{1}{T}\sum_{i=1}^{T} a_{i,T}$.

E Neuron Reordering Analysis

E.1 Time Overhead

As described in Section 3.1, DimLoRA first clusters neurons based on their parameter vectors, and then reorders the FFN parameters according to the clustering assignments. Here, we report the preprocessing time of this step. Using LLaMA3.1-8B as an example, the model contains 32 transformer layers, each with 14,336 FFN neurons. Applying balanced K-Means to cluster the neurons in a single layer into four groups takes approximately 164 seconds. Consequently, reordering the entire model requires about 1.46 hours. Notably, this is a one-time preprocessing cost that only needs to be performed once per model.

E.2 Functional Neuron Distribution

We quantify the contribution of each neuron to a given task using an importance score. Following prior work in model pruning (Molchanov et al., 2019), parameter importance is defined as the change in loss after zeroing out the parameter:

$$\begin{aligned} I_{\boldsymbol{w}} = |\mathcal{L}_{\boldsymbol{w}} - \mathcal{L}_{\boldsymbol{w}=\mathbf{0}}| &\approx \left|(\boldsymbol{w}-\mathbf{0})^\top \nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{w}}\right| \\ &= \left|\boldsymbol{w}^\top \nabla_{\boldsymbol{w}} \mathcal{L}_{\boldsymbol{w}}\right|, \end{aligned}$$

where $\mathcal{L}_{\boldsymbol{w}}$ denotes the loss with parameter $\boldsymbol{w}$, and $\mathcal{L}_{\boldsymbol{w}=\mathbf{0}}$ denotes the loss after zeroing it out. Since neuron n_i consists of a row vector $\boldsymbol{W}^I_{i,:}$ and a column vector $\boldsymbol{W}^O_{:,i}$, its importance for task T_j is defined as:

$$I_i = \sum_{(x,y)\in T_j} \left(I_{\boldsymbol{W}^I_{i,:}} + I_{\boldsymbol{W}^O_{:,i}}\right).$$

For each task, we select the top 1% of neurons by importance score as its *functional neurons*. We then measure how these neurons are distributed across expert groups by computing the entropy of the

resulting distribution. Higher entropy indicates a more dispersed distribution, whereas lower entropy indicates stronger concentration. Figure 6 shows the entropy of functional neuron distributions across all layers for ARC-E and BoolQ. We observe a substantial entropy reduction for both tasks after reordering, indicating that neuron reordering effectively groups neurons with similar functional roles together, thereby laying the foundation for subsequent localized parameter updates. In contrast, without reordering, functional neurons exhibit a distribution closer to random scattering.

F Expert Similarity Analysis

The Activation Analysis in Section 4.4 shows that DimLoRA exhibits clear task-specific expert activation patterns. To further understand how such routing behavior affects expert learning, we analyze the parameter similarity among experts. Specifically, we define the similarity between experts i and j as

$$\mathrm{Sim}(\boldsymbol{W}_i, \boldsymbol{W}_j) = \frac{1}{2}\cos(\boldsymbol{W}_i, \boldsymbol{W}_j) + \frac{1}{2}\exp(-D_{\mathrm{rel}}),$$

where

$$D_{\mathrm{rel}} = \frac{\|\boldsymbol{W}_i - \boldsymbol{W}_j\|_F}{\|\boldsymbol{W}_i\|_F + \|\boldsymbol{W}_j\|_F}.$$

Here, $\cos(\cdot,\cdot)$ is the cosine similarity between two matrices, measuring the directional similarity of expert parameters, while D_{rel} is the relative Frobenius distance, measuring their difference in magnitude. In our setting, each expert is parameterized by low-rank matrices, and the effective parameter of the i-th expert is computed as $\boldsymbol{W}_i = \boldsymbol{B}_i\boldsymbol{A}_i$.

We compare DimLoRA with two representative LoRA-MoE methods: MoLORA, a standard LoRA-MoE architecture, and HydraLoRA, which shares the $\boldsymbol{A}$ matrix across experts. We compute the average pairwise similarity among experts for each layer and report the similarities across all layers in Figure 7. We make two observations from this figure. (1) Expert similarity gradually decreases as the layer depth increases, suggesting that experts in lower layers tend to capture shared features, whereas experts in higher layers are more likely to specialize in different directions. (2) HydraLoRA shows the highest expert similarity, followed by MoLORA, while DimLoRA exhibits the lowest similarity. This indicates that although parameter sharing across experts in HydraLoRA improves parameter efficiency, it also introduces the risk of parameter competition. In MoLORA, since each expert updates the entire FFN parameter space, different experts tend to learn redundant knowledge. In contrast, DimLoRA assigns each expert to a more targeted parameter subspace, encouraging experts to specialize in distinct directions and effectively reducing the risk of parameter conflict.