# DISENTANGLED MIXTURE OF RANK-1 EXPERTS FOR LOW-RANK ADAPTATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The uniform rank allocation in Low-Rank Adaptation (LoRA) overlooks heterogeneous learning dynamics across modules and inherent semantic differences across inputs. We propose Disentangled Mixture of Rank-1 Experts (DMoRE), a principled framework for input-dependent rank allocation. To this end, we develop dynamic self-routing to activate a variable number of rank-1 components per input without an external router, along with gradient-derived budget estimation to avoid potentially misaligned manual budget specification. We further propose co-activation-aware rank disentanglement that encourages frequently co-activated components to learn distinct transformations. Extensive experiments on diverse language and vision tasks demonstrate that DMoRE consistently outperforms competitive LoRA baselines with negligible additional parameter overhead. It also exhibits structured rank allocation patterns at multiple granularities and reliably translates additional component activations into effective rank capacity.

## 1 INTRODUCTION

Low-Rank Adaptation (LoRA) has emerged as a leading paradigm for parameter-efficient fine-tuning (PEFT), enabling the efficient adaptation of large language models to diverse downstream tasks (Liu et al., 2024a; Mao et al., 2025; Weyssow et al., 2025). By factorizing weight updates into trainable low-rank matrices, LoRA achieves performance comparable to full-parameter fine-tuning with reduced computational overhead and memory consumption (Hu et al., 2022). Meanwhile, LoRA overlooks the learning dynamics of module weights and inherent semantic differences across inputs, resulting in suboptimal parameter utilization and degraded performance (Zhang et al., 2023; Li et al., 2025a). This module- and input-agnostic behavior stems from its uniform rank allocation, where a predefined rank is assigned to all modules and remains fixed across inputs.

To achieve input-dependent rank allocation, a natural approach is to incorporate sparse conditional computation, as exemplified by the Mixture-of-Experts (MoE) framework (Jacobs et al., 1991; Jordan & Jacobs, 1994). Recent MoE-based LoRA variants (Zhao et al., 2025; Zou et al., 2025; Lu et al., 2026) decompose weight updates into rank-1 components and selectively activate these components through input-dependent routing. Originally developed for multi-task or continual learning scenarios, these methods provide an intrinsic mechanism for input-dependent rank allocation. By varying the number of active rank-1 components conditioned on each input, the routing mechanism can, in principle, modulate the effective rank of the adaptation. Moreover, since routing decisions are learned separately within each adapted module, the same mechanism can yield module-specific rank allocation patterns without additional architectural modifications or manually designed heuristics.

However, directly adopting the aforementioned paradigm for input-dependent rank allocation faces two key limitations. First, existing rank-wise approaches commonly rely on top-$k$ routing, assigning the same number of rank-1 components to all inputs. This prevents input-adaptive variation in activation counts and requires a manually specified activation budget. Second, during training, rank-1 components co-activated for the same input receive gradient signals aligned with the same update direction. Without explicit constraints, repeated co-activation can therefore promote the learning of correlated transformations. Consequently, activating additional rank-1 components may merely reinforce redundant transformation directions rather than genuinely expand the adaptation subspace. Such redundancy, which we refer to as *rank collapse*, can lead to inefficient parameter utilization and undermine the core objective of dynamic rank allocation.



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.

In this paper, we propose Disentangled Mixture of Rank-1 Experts (DMoRE), a principled conditional computation framework for input-dependent rank allocation in low-rank adaptation. To overcome the fixed activation counts imposed by top-$k$ routing, we introduce dynamic self-routing, where rank-1 components are activated when the magnitudes of their down-projected feature responses exceed component-specific thresholds. This allows both the number and identities of active components to vary across inputs without requiring an external router. To avoid manually specifying a target rank budget that may be misaligned with adaptation demand, we estimate it from the effective dimensionality of the gradient signal and calibrate the activation thresholds accordingly. To mitigate rank collapse, we further introduce co-activation-aware rank disentanglement regularization. It encourages frequently co-activated rank-1 components to learn distinct transformations while preserving flexibility for task-driven component specialization.

We conduct extensive experiments on diverse downstream tasks, including natural language understanding, natural language reasoning, and image classification. Across backbones spanning various architectures and scales, DMoRE achieves the highest performance among representative LoRA variants from diverse methodological categories while introducing negligible additional parameter overhead over LoRA. Its performance gains persist across different maximum rank capacities and target-module configurations, demonstrating robustness to adaptation settings. Our analyses reveal structured rank allocation patterns at the task, module, and token levels. We also find that rank disentanglement narrows the gap between activation count and effective rank, indicating that additional component activations more reliably translate into usable rank capacity.

Our main contributions are summarized as follows:

- We present DMoRE, a principled conditional computation framework for input-dependent rank allocation in low-rank adaptation.
- We propose dynamic self-routing with gradient-derived budget estimation, enabling input-dependent rank allocation without an external router or manually specified rank budget.
- We develop co-activation-aware rank disentanglement regularization, mitigating rank collapse without unnecessarily constraining task-driven component specialization.
- We demonstrate that DMoRE consistently outperforms competitive LoRA variants across diverse tasks, while exhibiting structured rank-allocation patterns at multiple granularities.

## 2 Related Work

**Low-Rank Adaptation.** LoRA (Hu et al., 2022) freezes pre-trained weights and introduces trainable low-rank matrices for parameter-efficient fine-tuning. Since the learned updates can be merged into the base weights after training, LoRA preserves the original model architecture and introduces no additional inference latency. Existing work has primarily investigated three directions: (i) narrowing the performance gap to full-parameter fine-tuning (Gu et al., 2025; Li et al., 2026b; Tastan et al., 2026); (ii) developing a deeper understanding of its underlying properties (Zeng & Lee, 2024; Shuttleworth et al., 2025); and (iii) extending its application beyond fine-tuning (Huh et al., 2024; Ge et al., 2025; Li et al., 2025b). These efforts highlight LoRA as a prominent approach for adapting large foundation models to downstream tasks, particularly in resource-constrained settings.

**Rank Allocation in LoRA.** The static rank configuration in LoRA fails to match the heterogeneous adaptation needs across layers and inputs, resulting in suboptimal performance. Prior methods address this by adjusting the rank of each module based on spectral or heuristic importance scores (Zhang et al., 2023; Ke et al., 2024; Yang et al., 2025b; Liu et al., 2026), optimization-informed feedback (Cui et al., 2026; He et al., 2025; Li et al., 2026a; Ye et al., 2026), constraint-aware diagnostics (Kim et al., 2024; Ed-dib et al., 2025; Huang et al., 2025), or hypernetwork-based generation (Zhang et al., 2025). However, these methods adjust ranks at the structural level regardless of inputs. This overlooks nuanced input-dependent variation and can lead to inefficient use of parameter budget, motivating the development of an input-aware rank allocation strategy.

**Input-Dependent Modulation in LoRA.** To overcome the limitations of static adaptation, recent studies introduce input-dependent modulation into LoRA, where the adapter transformation is dynamically adjusted according to each input (Borse et al., 2024; Li et al., 2025a; Ou et al., 2026). While these methods improve flexibility, input-dependent modulation alone does not guarantee ef-

**Algorithm 1** DMoRE: Disentangled Mixture of Rank-1 Experts

**Require:** Frozen weight $\boldsymbol{W}$, scaling coefficient $\alpha$, rank $r$, threshold update rate $\eta_{\boldsymbol{\tau}}$, rank disentanglement strength $\eta_{\text{rank}}$, training data $\mathcal{D}$, and training steps $S$.

**for** $s \leftarrow 1$ to $S$ **do**
  Sample a mini-batch $\mathcal{B} \sim \mathcal{D}$.
  *# Forward pass*
  Compute detached gates: $\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x}) \leftarrow \mathbb{1}_{\text{sg}}(|\boldsymbol{\Lambda Q x}| > \boldsymbol{\tau})\,, \quad \forall \boldsymbol{x} \in \mathcal{B}$.
  Compute adapted outputs: $\boldsymbol{y}_{\text{DMoRE}} \leftarrow \boldsymbol{W x} + \frac{\alpha}{r}\boldsymbol{P}\big(\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x}) \odot \boldsymbol{\Lambda Q x}\big), \quad \forall \boldsymbol{x} \in \mathcal{B}$.
  Compute overall objective: $\mathcal{L}_{\text{DMoRE}} \leftarrow \mathcal{L}_{\text{task}} + \eta_{\text{rank}}\mathcal{L}_{\text{rank}}$. ▷ Eq. (8)
  *# Backward pass*
  Backpropagate $\mathcal{L}_{\text{DMoRE}}$ and cache the task-loss gradient $\nabla_{\boldsymbol{y}_{\text{DMoRE}}}\mathcal{L}_{\text{task}}$.
  *# Parameter update*
  Compute the candidate gradient $\boldsymbol{G_P}$ and target rank budget $R(\mathcal{B})$. ▷ Eq. (7)
  Compute activation counts $\boldsymbol{c} \leftarrow \sum_{\boldsymbol{x}\in\mathcal{B}} \boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})$ and update $\boldsymbol{\tau}$ using $\eta_{\boldsymbol{\tau}}$. ▷ Eq. (6)
  Update $\boldsymbol{P}$, $\boldsymbol{\Lambda}$, and $\boldsymbol{Q}$. ▷ Optimizer step
**end for**
**return** $\boldsymbol{P}$, $\boldsymbol{\Lambda}$, $\boldsymbol{Q}$, and $\boldsymbol{\tau}$.

fective dynamic rank allocation: co-activated rank-1 components can learn directionally redundant transformations if no constraint is provided, leading to poor parameter utilization and rank collapse. This motivates our study of input-dependent rank allocation in LoRA, aiming to improve rank utilization and enable finer-grained control over adapter expressivity.

**Mixture of Experts with LoRA.** MoE models (Jacobs et al., 1991; Jordan & Jacobs, 1994) employ input-conditioned routing to activate a subset of model parameters, enabling substantial model scaling while keeping per-token computation approximately constant. Such efficiency has driven the wide adoption of MoE across diverse domains (Jiang & Shen, 2024; Bai et al., 2025; DeepSeek-AI et al., 2025). Recent work integrates MoE with LoRA to improve downstream adaptation and reduce interference in multi-task learning, treating either full LoRA modules (Li et al., 2024; Luo et al., 2024; Tian et al., 2024; Sun et al., 2025; Liu & Luo, 2024; Huang et al., 2026; Zhuang et al., 2026) or individual rank-1 components (Ning et al., 2025; Zhao et al., 2025; Zou et al., 2025) in LoRA as experts. Nevertheless, these methods either activate a fixed number of experts or rely on additional router networks, which limits their flexibility and efficiency for input-adaptive rank allocation. Instead, we propose a framework that activates a variable number of rank-1 components per input based on latent feature responses without a separate routing network.

# 3 Method

## 3.1 Preliminaries

In this subsection, we present a rank-wise decomposition of LoRA that motivates input-dependent rank allocation and discuss the associated risk of rank collapse.

**Effective Rank.** Given a non-zero matrix $\boldsymbol{M} \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ with maximum possible rank $r_{\max} = \min(d_{\text{out}}, d_{\text{in}})$, its effective rank (Roy & Vetterli, 2007) is defined as:

$$\text{erank}(\boldsymbol{M}) = \exp\left(-\sum_{i=1}^{r_{\max}} p_i \log p_i\right), \quad p_i = \frac{\sigma_i}{\sum_{j=1}^{r_{\max}}\sigma_j}, \tag{1}$$

where $\{\sigma_i\}_{i=1}^{r_{\max}}$ are the singular values of $\boldsymbol{M}$ with $0\log 0 = 0$. It provides a continuous proxy for spectral dimensionality by discounting directions with negligible spectral contributions.

**Rank-Wise Decomposition of LoRA.** LoRA (Hu et al., 2022) parameterizes the update to a frozen weight $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ as the product of two trainable low-dimensional matrices $\boldsymbol{B} \in \mathbb{R}^{d_{\text{out}}\times r}$ and $\boldsymbol{A} \in \mathbb{R}^{r\times d_{\text{in}}}$. Given an input $\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}$, the output $\boldsymbol{y}_{\text{LoRA}} \in \mathbb{R}^{d_{\text{out}}}$ can be expressed as:

$$\boldsymbol{y}_{\text{LoRA}} = \boldsymbol{W x} + \frac{\alpha}{r}\boldsymbol{BAx} = \boldsymbol{W x} + \frac{\alpha}{r}\sum_{i=1}^{r}\boldsymbol{g}_{\text{LoRA}}(\boldsymbol{x})_i \boldsymbol{b}_i \boldsymbol{a}_i^\top \boldsymbol{x}, \tag{2}$$

where $\boldsymbol{g}_{\text{LoRA}}(\boldsymbol{x}) \equiv \mathbf{1}_r \in \mathbb{R}^r$, $\boldsymbol{b}_i = \boldsymbol{B}[:, i] \in \mathbb{R}^{d_{\text{out}}}$ is the $i$-th column of $\boldsymbol{B}$, $\boldsymbol{a}_i = \boldsymbol{A}[i, :]^\top \in \mathbb{R}^{d_{\text{in}}}$ is the transpose of the $i$-th row of $\boldsymbol{A}$, $\alpha$ is a scaling coefficient, and $r$ is the maximum rank capacity. This suggests a natural extension: by replacing $\boldsymbol{g}_{\text{LoRA}}(\boldsymbol{x}) \equiv \mathbf{1}_r$ with a sparse $\boldsymbol{g}(\boldsymbol{x})$, the effective rank of $\Delta\boldsymbol{W}(\boldsymbol{x}) = \frac{\alpha}{r}\sum_{i=1}^{r} \boldsymbol{g}(\boldsymbol{x})_i \boldsymbol{b}_i \boldsymbol{a}_i^\top \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is upper-bounded by the activation count $\|\boldsymbol{g}(\boldsymbol{x})\|_0$.

**Risk of Rank Collapse.** However, the effective rank upper bound is not necessarily tight because $\boldsymbol{g}(\boldsymbol{x})$ does not guarantee that the activated rank-1 components contribute sufficiently independent input and output directions. Here, we focus on the directional redundancy that can arise from component co-activation. Given a differentiable loss $\mathcal{L}$ on the output $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \frac{\alpha}{r}\sum_{i=1}^{r} \boldsymbol{g}(\boldsymbol{x})_i \boldsymbol{b}_i \boldsymbol{a}_i^\top \boldsymbol{x} \in \mathbb{R}^{d_{\text{out}}}$, the gradient with respect to the induced rank-1 component $\boldsymbol{b}_i \boldsymbol{a}_i^\top$ is:

$$\nabla_{(\boldsymbol{b}_i \boldsymbol{a}_i^\top)} \mathcal{L} = \frac{\alpha}{r} \boldsymbol{g}(\boldsymbol{x})_i \nabla_{\boldsymbol{y}} \mathcal{L} \boldsymbol{x}^\top. \tag{3}$$

At the induced component level, rank-1 components co-activated for the same input receive descent signals collinear with the same outer-product direction $\nabla_{\boldsymbol{y}} \mathcal{L} \boldsymbol{x}^\top$, differing by scalar gating values. As training proceeds, repeated co-activation can therefore encourage the learning of correlated transformations, so the activation count can overstate the effective rank of the induced update.

### 3.2 Disentangled Mixture of Rank-1 Experts

Following the rank-wise decomposition of LoRA in Eq. (2), we introduce Disentangled Mixture of Rank-1 Experts (DMoRE), a principled conditional computation framework for input-dependent rank allocation in low-rank adaptation, as shown in Alg. 1. To facilitate conditional rank-wise activation, we adopt the $\boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{Q}$ parameterization inspired by singular value decomposition (Zhang et al., 2023; Liu et al., 2026). Given an input $\boldsymbol{x}$, the output $\boldsymbol{y}_{\text{DMoRE}} \in \mathbb{R}^{d_{\text{out}}}$ is formulated as:

$$\boldsymbol{y}_{\text{DMoRE}} = \boldsymbol{W}\boldsymbol{x} + \frac{\alpha}{r}\sum_{i=1}^{r} \boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i \lambda_i \boldsymbol{p}_i \boldsymbol{q}_i^\top \boldsymbol{x}, \tag{4}$$

where $\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x}) \in \{0, 1\}^r$ denotes the input-dependent binary gating vector, $\boldsymbol{P} \in \mathbb{R}^{d_{\text{out}} \times r}$ and $\boldsymbol{Q} \in \mathbb{R}^{r \times d_{\text{in}}}$ are up- and down-projections, respectively, $\boldsymbol{p}_i = \boldsymbol{P}[:, i] \in \mathbb{R}^{d_{\text{out}}}$ is the $i$-th column of $\boldsymbol{P}$, $\boldsymbol{q}_i = \boldsymbol{Q}[i, :]^\top \in \mathbb{R}^{d_{\text{in}}}$ is the transpose of the $i$-th row of $\boldsymbol{Q}$, and $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \cdots, \lambda_r) \in \mathbb{R}^{r \times r}$ contains learnable scales for each rank-1 component.

**Dynamic Self-Routing.** To activate a variable number of rank-1 components per input, we propose a threshold-based self-routing mechanism that is formally defined as:

$$\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x}) = \mathbb{1}_{\text{sg}}(|\boldsymbol{\Lambda}\boldsymbol{Q}\boldsymbol{x}| - \boldsymbol{\tau} > 0), \tag{5}$$

where $\boldsymbol{\tau} \in \mathbb{R}^r$ contains component-wise activation thresholds, and $\mathbb{1}_{\text{sg}}(\cdot)$ is the element-wise indicator function whose output is detached from gradient computation. Unlike prior methods (Li et al., 2024; Tian et al., 2024; Zhao et al., 2025) that employ an explicit router to activate a fixed number of experts, DMoRE directly leverages the latent magnitudes $|\boldsymbol{\Lambda}\boldsymbol{Q}\boldsymbol{x}|$ as routing signals and activates rank-1 components whose magnitudes exceed the corresponding thresholds. The design follows the insight that a component's internal activation magnitude can serve as an intrinsic proxy for its relevance to a given input (Lv et al., 2025; Zou et al., 2025). We detach the binary gates so that task gradients do not propagate through the routing decisions. As discussed in App. A.2, under a straight-through estimator (STE) (Bengio et al., 2013), the routing gradient with respect to $\boldsymbol{q}_i$ or $\lambda_i$ of an active component is a nonnegative scalar multiple of its expert gradient. When the resulting amplification strengthens the component's latent response, it may reinforce subsequent activation and produce a rich-get-richer effect, potentially leading to inefficient component utilization.

**Load Balancing.** Routing collapse can reduce effective capacity utilization by concentrating parameter updates on a few experts while leaving others under-trained (Shazeer et al., 2017). To mitigate it, we extend the load-balancing strategy from (Wang et al., 2024a) and update $\boldsymbol{\tau}$ via:

$$\boldsymbol{\tau} \leftarrow \boldsymbol{\tau} + \eta_{\boldsymbol{\tau}} \,\text{sign}\left(\boldsymbol{c} - \frac{TR(\mathcal{B})}{r}\mathbf{1}_r\right), \tag{6}$$

where $\eta_{\boldsymbol{\tau}} \in \mathbb{R}_+$ denotes the threshold update rate, $\boldsymbol{c} = \sum_{\boldsymbol{x} \in \mathcal{B}} \boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})$ contains the component-wise activation counts over $T$ inputs in a batch $\mathcal{B}$, and $R(\mathcal{B})$ is the target average number of activated rank-1 components per input. Intuitively, it serves as a feedback mechanism that encourages

balanced component utilization at the target level $R(\mathcal{B})$ by discouraging frequently activated components and promoting less frequently activated ones through adaptive threshold adjustment.

**Budget Control.** Unconstrained dynamic routing can drift toward an overly dense regime, where a large number of rank-1 components are consistently activated regardless of the input. To address this, we estimate the target rank budget from the batch-level candidate gradient $\boldsymbol{G_P}$ as follows:

$$R(\mathcal{B}) = \mathrm{erank}(\boldsymbol{G_P}), \quad \boldsymbol{G_P} = \frac{\alpha}{r} \sum_{\boldsymbol{x} \in \mathcal{B}} (\nabla_{\boldsymbol{y}_{\text{DMoRE}}} \mathcal{L}_{\text{task}})(\boldsymbol{\Lambda Q x})^\top, \tag{7}$$

where $\mathcal{L}_{\text{task}}$ denotes the primary task loss. Inspired by Ye et al. (2026), we use the effective dimensionality of the gradient as a proxy for rank-capacity demand, assigning a larger rank budget when the gradient is distributed across more update directions. Compared with the routed gradient $\nabla_{\boldsymbol{P}} \mathcal{L}_{\text{task}} = \frac{\alpha}{r} \sum_{\boldsymbol{x} \in \mathcal{B}} (\nabla_{\boldsymbol{y}_{\text{DMoRE}}} \mathcal{L}_{\text{task}})(\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x}) \odot \boldsymbol{\Lambda Q x})^\top$, the candidate gradient $\boldsymbol{G_P}$ approximates the update directions under activation of all rank-1 components. This allows currently inactive components to contribute to dimensionality estimation, reducing the risk that the current routing pattern biases the estimated budget downward and reinforces component inactivity. We zero-initialize $\boldsymbol{\tau}$ so that all components can receive initial task gradients before reliable budget estimates are formed.

**Rank Disentanglement.** As discussed in Sec. 3.1, repeated co-activation can encourage rank-1 components to learn correlated transformations, potentially resulting in rank collapse. To mitigate this, we introduce a rank disentanglement loss $\mathcal{L}_{\text{rank}}$ that is defined as:

$$\mathcal{L}_{\text{rank}} = \left\| \boldsymbol{P}^\top \boldsymbol{P} - \boldsymbol{I}_r \right\|_{\boldsymbol{\Omega}}^2 + \left\| \boldsymbol{Q}\boldsymbol{Q}^\top - \boldsymbol{I}_r \right\|_{\boldsymbol{\Omega}}^2, \; \boldsymbol{\Omega}_{i,j} = \begin{cases} 1, & i = j, \\ \frac{1}{T} \sum_{\boldsymbol{x} \in \mathcal{B}} \boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i \boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_j, & i \neq j, \end{cases} \tag{8}$$

with $\|\boldsymbol{M}\|_{\boldsymbol{\Omega}}^2 = \sum_{i,j} \boldsymbol{\Omega}_{i,j} \boldsymbol{M}_{i,j}^2$. Differing from the conventional orthogonality constraint $\mathcal{L}_{\text{ortho}} = \|\boldsymbol{P}^\top \boldsymbol{P} - \boldsymbol{I}_r\|_F^2 + \|\boldsymbol{Q}\boldsymbol{Q}^\top - \boldsymbol{I}_r\|_F^2$ from Zhang et al. (2023); Ke et al. (2024); Liu et al. (2026), $\mathcal{L}_{\text{rank}}$ encourages unit-norm up- and down-projections and penalizes directional overlap between frequently co-activated rank-1 components. This avoids imposing uniform orthogonality pressure across all rank-1 components, preserving the flexibility for task-driven component specialization. Since $\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})$ is detached as defined in Eq. (5), $\mathcal{L}_{\text{rank}}$ does not propagate gradients through the routing pathway and directly regularizes $\boldsymbol{P}$ and $\boldsymbol{Q}$ through their Gram matrices. The overall training objective of DMoRE is defined as:

$$\mathcal{L}_{\text{DMoRE}} = \mathcal{L}_{\text{task}} + \eta_{\text{rank}} \mathcal{L}_{\text{rank}}, \tag{9}$$

where $\eta_{\text{rank}} \in \mathbb{R}_+$ controls the strength of rank disentanglement.

## 4 Experiments

In this section, we first comprehensively validate the effectiveness of DMoRE across diverse model architectures and tasks, including natural language understanding (Sec. 4.1), natural language reasoning (Sec. 4.2), and image classification (Sec. 4.3). We further analyze key properties of DMoRE in Sec. 4.4, followed by ablation studies in Sec. 4.5 to assess the contribution of each component.

**Implementation Details.** We implement all models in PyTorch and train them on NVIDIA A100 80 GB GPUs. Unless otherwise specified, we apply adapters to all linear modules in the backbone and set $r = 8$ and $\alpha = 2r$. For each baseline, we adopt the recommended hyperparameters from the corresponding paper or official codebase whenever available. For DMoRE, we set $\eta_{\text{rank}} = 1 \times 10^{-2}$ and $\eta_{\boldsymbol{\tau}} = 1 \times 10^{-3}$ across all tasks. We repeat every experiment with three different random seeds and report the average performance, with across-seed variability provided in App. B.4.

**Baselines.** We compare DMoRE with representative baselines spanning diverse methodological categories: (i) *fixed-rank adaptation*, including LoRA (Hu et al., 2022) and DoRA (Liu et al., 2024b); (ii) *module-aware rank allocation*, including AdaLoRA (Zhang et al., 2023) and IGU-LoRA (Cui et al., 2026); (iii) *input-dependent modulation*, including TopLoRA (Li et al., 2025a); (iv) *LoRA-based MoE adaptation*, including HydraLoRA (Tian et al., 2024) and LD-MoLE (Zhuang et al., 2026); and (v) *rank-based MoE adaptation*, including SMoRA (Zhao et al., 2025) and FlyLoRA (Zou et al., 2025). Collectively, these baselines cover key design dimensions of low-rank adaptation, enabling a comprehensive evaluation of DMoRE across diverse adaptation paradigms.

Table 1: **Quantitative comparison of LoRA variants on NLU.** We report accuracy (↑) for all tasks except MRPC, QQP, and STS-B, for which the combined scores (↑) are used. #Param. indicates the number of trainable parameters. The **best** results are highlighted.

| Model | Method | #Param. | CoLA | MNLI | MRPC | QNLI | QQP | SST-2 | STS-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|
| RoBERTa-B | LoRA | 1.33M | 77.28 | 85.55 | 78.19 | 90.31 | 86.20 | 93.27 | 81.78 | 84.65 |
| | DoRA | 1.41M | 77.75 | 85.27 | 76.56 | 90.27 | 86.20 | 93.42 | 81.74 | 84.46 |
| | AdaLoRA | 1.33M | 77.95 | 85.44 | 75.96 | 90.94 | 86.60 | 93.46 | 85.39 | 85.11 |
| | IGU-LoRA | 1.33M | 69.13 | 83.57 | 74.98 | 85.98 | 83.70 | 90.63 | 86.57 | 82.08 |
| | TopLoRA | 1.99M | 78.84 | 85.24 | 76.50 | 90.40 | 86.08 | 93.39 | 85.62 | 85.15 |
| | HydraLoRA | 1.58M | 76.38 | 85.27 | 79.46 | 89.97 | 85.88 | 93.27 | 78.56 | 84.11 |
| | LD-MoLE | 1.69M | 71.33 | 85.21 | 75.02 | 89.69 | 85.78 | 92.97 | 72.99 | 81.85 |
| | SMoRA | 1.99M | 69.13 | 85.05 | 74.80 | 89.66 | 85.72 | 92.47 | 82.91 | 82.82 |
| | FlyLoRA | 0.66M | 69.80 | 83.41 | 75.29 | 87.81 | 84.33 | 92.66 | 73.22 | 80.93 |
| | DMoRE | 1.33M | **80.02** | **85.98** | **79.93** | **91.16** | **87.11** | **94.00** | **87.99** | **86.60** |
| DeBERTa-v3-L | LoRA | 3.54M | 86.45 | 91.80 | 74.92 | 95.25 | 89.74 | 95.60 | 86.06 | 88.55 |
| | DoRA | 3.76M | 86.64 | 91.68 | 74.92 | 95.21 | 89.69 | 95.68 | 86.44 | 88.61 |
| | AdaLoRA | 3.54M | 87.41 | 91.75 | 81.29 | 95.31 | 89.87 | 95.72 | 88.64 | 90.00 |
| | IGU-LoRA | 3.54M | 80.89 | 90.33 | 74.90 | 92.15 | 85.06 | 91.40 | 90.09 | 86.40 |
| | TopLoRA | 5.31M | 87.25 | 91.77 | 82.24 | 95.23 | 89.46 | 95.87 | 89.78 | 90.23 |
| | HydraLoRA | 4.20M | 85.94 | 91.62 | 75.30 | 95.08 | 89.60 | 95.91 | 85.22 | 88.38 |
| | LD-MoLE | 4.46M | 85.36 | 91.73 | 75.04 | 95.15 | 89.55 | 95.57 | 83.48 | 87.98 |
| | SMoRA | 5.31M | 84.24 | 91.39 | 74.92 | 94.81 | 89.28 | 95.14 | 81.14 | 87.27 |
| | FlyLoRA | 1.77M | 84.69 | 90.71 | 75.19 | 94.35 | 88.49 | 94.38 | 85.96 | 87.68 |
| | DMoRE | 3.54M | **87.79** | **91.82** | **83.54** | **95.46** | **90.46** | **96.10** | **90.15** | **90.76** |

### 4.1 Natural Language Understanding (NLU)

**Settings.** We fine-tune RoBERTa-B (Liu et al., 2019) and DeBERTa-v3-L (He et al., 2021) on seven selected GLUE (Wang et al., 2018) tasks, covering linguistic acceptability, natural language inference, paraphrase detection, question-answering inference, sentiment classification, and semantic textual similarity. Detailed experimental settings are provided in App. B.1.

**Results.** Tab. 1 shows that DMoRE achieves the best performance on all NLU tasks across both backbones while using the nearly same number of trainable parameters as LoRA. These consistent gains demonstrate the effectiveness of input-dependent rank allocation.

### 4.2 Natural Language Reasoning (NLR)

**Settings.** We fine-tune Llama-3.2-3B (Grattafiori et al., 2024) and Qwen-3-8B (Yang et al., 2025a) on diverse reasoning tasks, including commonsense reasoning, general knowledge understanding, mathematical reasoning, and code generation. Detailed settings are provided in App. B.2.

**Results.** As shown in Tab. 2, DMoRE achieves the highest average performance on both backbones and the best results on five of seven tasks for each backbone. These consistent gains indicate that adaptively allocating rank capacity conditioned on inputs can be more effective than uniformly increasing the parameter budget. This highlights the effectiveness and parameter efficiency of input-dependent rank allocation across a diverse range of reasoning tasks.

### 4.3 Image Classification (IC)

**Settings.** We fine-tune the CLIP-ViT-B/16 (Radford et al., 2021) and SigLIP-ViT-L/16 (Zhai et al., 2023) models on seven image classification benchmarks. Detailed settings are provided in App. B.3.

**Results.** Tab. 3 shows that DMoRE achieves the best performance across all datasets. It improves the average accuracy from 82.92% to 86.44% compared with LoRA and outperforms the next-best baseline, TopLoRA, by 1.77 percentage points despite using fewer trainable parameters. The gains are particularly pronounced on tasks like Cars and DTD. Together with the NLU and NLR results in Tabs. 1 and 2, these findings demonstrate the applicability of DMoRE across language and vision tasks. Additional results on SigLIP-ViT-L/16 are provided in App. B.3.

Table 2: **Quantitative comparison of LoRA variants on NLR.** We report accuracy (↑) for all tasks except HumanEval, for which Pass@1 (↑) is used. #Param. indicates the number of trainable parameters. The **best** results are highlighted. HS = HellaSwag, HE = HumanEval, WG = WinoGrande.

| Model | Method | #Param. | ARC-C | BoolQ | GSM8K | HS | HE | MMLU | WG | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|
| LLaMA-3.2-3B | LoRA | 12.16M | 49.46 | 88.20 | 34.49 | 78.99 | 23.98 | 55.73 | 83.64 | 59.21 |
| | DoRA | 12.93M | 50.11 | 88.16 | 35.03 | 79.02 | 24.19 | 55.67 | 83.80 | 59.42 |
| | AdaLoRA | 12.20M | 46.19 | 81.91 | 18.48 | 77.96 | 25.20 | 55.79 | 78.98 | 54.93 |
| | IGU-LoRA | 12.20M | 50.31 | 86.63 | 27.62 | 78.22 | 19.71 | 56.04 | 80.74 | 57.04 |
| | TopLoRA | 18.12M | 50.97 | 88.84 | 39.83 | **79.42** | 24.19 | **56.25** | 83.58 | 60.44 |
| | HydraLoRA | 14.51M | 47.84 | 87.41 | 33.54 | 78.92 | 23.58 | 55.75 | 83.21 | 58.61 |
| | LD-MoLE | 15.23M | 46.64 | 86.87 | 33.21 | 78.80 | 24.39 | 55.82 | 82.69 | 58.35 |
| | SMoRA | 18.12M | 47.47 | 86.47 | 33.18 | 78.82 | 23.78 | 55.61 | 82.79 | 58.30 |
| | FlyLoRA | 6.19M | 50.32 | 88.18 | 34.24 | 78.62 | 24.39 | 54.89 | 82.48 | 59.02 |
| | DMoRE | 12.16M | **54.49** | **89.31** | **41.98** | 78.70 | **25.41** | 54.24 | **83.82** | **61.13** |
| Qwen-3-8B | LoRA | 21.82M | 59.61 | 90.12 | 81.43 | 79.51 | 59.35 | 74.02 | 82.90 | 75.28 |
| | DoRA | 23.22M | 60.47 | 90.06 | 82.11 | 79.51 | 59.15 | 74.02 | 82.80 | 75.45 |
| | AdaLoRA | 21.82M | 56.91 | 89.61 | 81.58 | 79.50 | 60.98 | 73.83 | 83.40 | 75.11 |
| | IGU-LoRA | 21.82M | 58.76 | 89.65 | 75.08 | 78.47 | 57.72 | 72.14 | 81.74 | 73.37 |
| | TopLoRA | 32.44M | 62.88 | 90.72 | 81.35 | 79.90 | 55.49 | **74.40** | 83.53 | 75.47 |
| | HydraLoRA | 26.10M | 56.40 | 89.67 | 81.81 | 79.25 | 61.18 | 74.01 | 82.37 | 74.96 |
| | LD-MoLE | 27.26M | 57.82 | 89.13 | 81.70 | 79.25 | 59.15 | 74.17 | 82.32 | 74.79 |
| | SMoRA | 32.44M | 59.27 | 89.38 | 81.70 | 79.00 | **64.02** | 74.38 | 81.53 | 75.61 |
| | FlyLoRA | 11.21M | 57.99 | 90.44 | 82.64 | 78.70 | 59.56 | 73.55 | 80.24 | 74.73 |
| | DMoRE | 21.83M | **64.56** | **91.67** | **82.66** | **79.93** | 57.12 | 73.76 | **83.74** | **76.21** |

### 4.4 Analysis

**Rank Allocation.** Fig. 1a shows effective-rank distributions for tokens evenly sampled from the vocabulary based on their rank utilization. Among the sampled tokens, semantically complete words (e.g., "date", "typing", and "executive") receive higher effective ranks, whereas subword fragments or non-lexical tokens (e.g., "re", "axy", and "`:`") receive lower ranks. These results indicate that DMoRE learns heterogeneous, token-dependent rank allocations rather than assigning uniform capacity across tokens. Task- and module-level rank allocation analysis is provided in App. B.4.

**Rank Disentanglement.** We examine the rank collapse discussed in Sec. 3.1 by comparing the effective rank with the number of active rank-1 components. The activation count provides a theoretical upper bound, approached when active components contribute independent and spectrally balanced update directions. Fig. 1b shows that, without $\mathcal{L}_{\text{rank}}$, the effective rank increasingly deviates from the upper bound as more components are activated, indicating that additional activations contribute diminishing spectral dimensionality. This gap is substantially narrowed by $\mathcal{L}_{\text{rank}}$, showing that it translates component activations into realized effective-rank capacity.

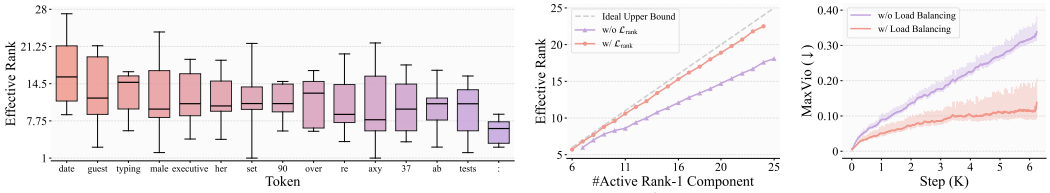
**Component Utilization.** We assess component utilization using MaxVio (Wang et al., 2024a), which quantifies the maximum relative deviation from uniform component load. We compare DMoRE with a variant that retains the same rank budget but removes component-wise load balancing. Fig. 1c shows that the threshold update in Eq. (6) consistently reduces MaxVio throughout training. We further observe a 0.92 percentage-point gain in overall performance over the variant without load balancing. These results highlight the importance of balanced component utilization for effective input-dependent rank allocation.

**Robustness.** We assess the robustness of DMoRE by varying the maximum rank capacity and target module coverage. Fig. 2 shows that DMoRE consistently achieves the highest average performance across all settings, with average performance increasing as rank capacity or target-module coverage expands. Notably, even at the smallest rank capacity, DMoRE outperforms baselines using larger adaptation budgets. These results demonstrate the robustness of DMoRE.

**Computational Overhead.** Tab. 4 shows that DMoRE incurs moderate training-time overhead while maintaining peak memory nearly identical to LoRA. At inference, DMoRE remains more efficient than other evaluated unmergeable methods. Overall, these results indicate that DMoRE introduces controlled overhead in exchange for improved performance.

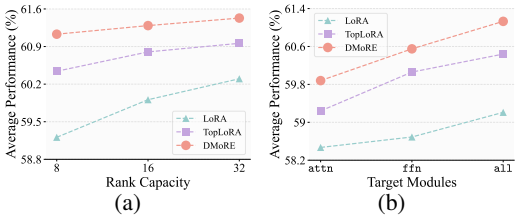
Table 3: **Quantitative comparison of LoRA variants on IC.** We report top-1 classification accuracy (↑) on CLIP-ViT-B/16. #Param. indicates the number of trainable parameters. The **best** results are highlighted. Results on SigLIP-ViT-L/16 are in App. B.3. ESAT = EuroSAT, RE45=RESISC45.

| Method | #Param. | Cars | DTD | ESAT | GTSRB | RE45 | SUN397 | SVHN | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| LoRA | 1.33M | 71.65 | 52.20 | 97.62 | 97.08 | 91.88 | 73.45 | 96.59 | 82.92 |
| DoRA | 1.41M | 71.85 | 52.38 | 97.67 | 97.15 | 92.06 | 73.56 | 96.62 | 83.04 |
| AdaLoRA | 1.33M | 70.54 | 46.24 | 97.59 | 97.30 | 92.13 | 74.32 | 96.58 | 82.10 |
| IGU-LoRA | 1.33M | 68.29 | 51.51 | 96.41 | 94.51 | 85.88 | 64.62 | 94.82 | 79.43 |
| TopLoRA | 1.99M | 75.17 | 56.34 | 97.98 | 97.71 | 93.53 | 75.10 | 96.88 | 84.67 |
| HydraLoRA | 1.58M | 70.32 | 49.65 | 97.50 | 96.82 | 91.26 | 72.48 | 96.36 | 82.06 |
| LD-MoLE | 1.69M | 69.18 | 46.83 | 97.30 | 96.25 | 90.37 | 71.59 | 96.06 | 81.08 |
| SMoRA | 1.99M | 68.62 | 45.18 | 97.25 | 95.91 | 89.60 | 70.95 | 95.97 | 80.50 |
| FlyLoRA | 0.66M | 71.97 | 52.96 | 97.64 | 95.69 | 91.07 | 72.50 | 95.86 | 82.53 |
| DMoRE | 1.33M | **77.98** | **63.32** | **98.18** | **98.07** | **94.29** | **76.09** | **97.18** | **86.44** |




(a) (b) (c)

Figure 1: **Rank utilization analysis of DMoRE on NLR.** (a) Distribution of effective ranks across tokens. Box plots report the median, interquartile range, and minimum/maximum values. (b) Comparison of effective rank against the number of active rank-1 components with and without $\mathcal{L}_{\text{rank}}$. (c) Comparison of module-averaged MaxVio (↓) with and without load balancing. Removing load balancing reduces average NLR performance to 60.21. For (a) and (b), we aggregate measurements within a single layer of Llama-3.2-3B and use $r = 32$ for a wider observable rank range.




(a) (b)

Figure 2: **Sensitivity analysis of LoRA variants on NLR.** We evaluate DMoRE against representative LoRA variants while varying (a) the maximum rank capacity and (b) the target modules. Detailed results are provided in App. B.4.

Table 4: **Computational overhead of LoRA variants on NLR.** We report per-step training time, per-token inference latency, peak GPU memory, and average performance. Detailed profiling settings are provided in App. B.4.

| Method | Training | | Inference | | Avg. |
|---|---|---|---|---|---|
| | Time (s) | Mem. (GB) | Lat. (ms) | Mem. (GB) | |
| LoRA | 0.55 | 20.76 | 0.17 | 6.09 | 59.21 |
| AdaLoRA | 0.61 | 21.11 | 0.17 | 6.09 | 54.93 |
| LD-MoLE | 1.72 | 33.46 | 1.47 | 6.16 | 58.35 |
| SMoRA | 0.61 | 21.12 | 0.48 | 6.16 | 58.30 |
| DMoRE | 0.82 | 20.79 | 0.45 | 6.13 | 61.13 |

### 4.5 Ablation Studies

In this subsection, we perform ablation studies to investigate key design choices in DMoRE. We use Llama-3.2-3B as the backbone and follow the experimental settings described in Sec. 4.2.

**Routing.** Tab. 5a compares four routing strategies: (i) *dense routing*, which activates all eight rank-1 components for every input; (ii) *top-$k$ routing*, which activates the two rank-1 components with the highest latent magnitudes; (iii) *dynamic routing*, as defined in Eq. (5); and (iv) *dynamic routing with a straight-through estimator*, which retains the same hard gating in the forward pass in Eq. (5) but approximates its derivatives with a sigmoid surrogate during backpropagation. Dynamic routing achieves the best overall performance. Unlike dense and top-$k$ routing, it permits both the identities and the number of active rank-1 components to vary across inputs, thereby enabling a richer set of activation patterns. Its improvement over the STE variant is also consistent with the routing-gradient analysis in App. A.2, which shows that detachment prevents routing gradients from amplifying active-component updates and potentially reinforcing parameter utilization imbalances.

Table 5: **Ablation of DMoRE on NLR.** We investigate different design choices for (a) routing, (b) budget control, (c) rank disentanglement, and evaluate sensitivity to (d) $\eta_{\text{rank}}$ and (e) $\eta_{\boldsymbol{\tau}}$, respectively. We report accuracy (↑) for all other tasks and Pass@1 (↑) for HumanEval. The **best** results and default design choices are highlighted. HS = HellaSwag, HE = HumanEval, WG = WinoGrande.

| | Case | ARC-C | BoolQ | GSM8K | HS | HE | MMLU | WG | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| (a) | Dense | 49.46 | 88.20 | 34.49 | 78.99 | 23.98 | 55.73 | 83.64 | 59.21 |
| | Top-$k$ | 52.10 | 89.13 | 40.21 | 78.52 | 24.39 | 53.61 | 82.35 | 60.05 |
| | Dynamic | 54.47 | 89.34 | 41.47 | 78.73 | 25.41 | 54.26 | 83.64 | **61.04** |
| | Dynamic + STE | 54.58 | 89.46 | 40.81 | 78.64 | 24.39 | 54.43 | 83.19 | 60.79 |
| (b) | Fixed | 54.55 | 89.39 | 39.07 | 78.87 | 24.19 | 53.88 | 83.09 | 60.43 |
| | $\text{erank}(\nabla_{\boldsymbol{P}}\mathcal{L}_{\text{task}})$ | 53.92 | 89.24 | 40.86 | 78.58 | 23.78 | 54.66 | 83.19 | 60.60 |
| | $\text{erank}(\boldsymbol{G_P})$ | 54.47 | 89.34 | 41.47 | 78.73 | 25.41 | 54.26 | 83.64 | 61.04 |
| | $\text{EMA}(\text{erank}(\boldsymbol{G_P}))$ | 54.58 | 89.40 | 41.09 | 79.00 | 23.58 | 55.88 | 83.90 | **61.06** |
| (c) | $\boldsymbol{PAQ}$ | 54.47 | 89.34 | 41.47 | 78.73 | 25.41 | 54.26 | 83.64 | 61.04 |
| | $\boldsymbol{PAQ}$ + Stiefel opt. | 46.36 | 87.22 | 30.40 | 77.70 | 24.19 | 54.39 | 81.21 | 57.35 |
| | $\boldsymbol{PAQ} + \mathcal{L}_{\text{ortho}}$ | 54.35 | 89.44 | 41.07 | 78.62 | 23.58 | 54.50 | 83.61 | 60.74 |
| | $\boldsymbol{PAQ} + \mathcal{L}_{\text{rank}}$ | 54.52 | 89.44 | 41.65 | 78.75 | 25.20 | 54.37 | 83.87 | **61.11** |
| | $\boldsymbol{BA} + \mathcal{L}_{\cos}$ | 54.26 | 89.38 | 38.71 | 78.79 | 24.39 | 54.26 | 83.58 | 60.48 |
| (d) | $\eta_{\text{rank}} = 1 \times 10^{0}$ | 54.47 | 89.44 | 41.85 | 78.64 | 25.81 | 54.28 | 83.19 | 61.10 |
| | $\eta_{\text{rank}} = 1 \times 10^{-1}$ | 54.52 | 89.44 | 41.65 | 78.75 | 25.20 | 54.37 | 83.87 | 61.11 |
| | $\eta_{\text{rank}} = 1 \times 10^{-2}$ | 54.49 | 89.31 | 41.98 | 78.70 | 25.41 | 54.24 | 83.82 | **61.13** |
| (e) | $\eta_{\boldsymbol{\tau}} = 1 \times 10^{-1}$ | 54.35 | 89.63 | 40.89 | 78.88 | 25.00 | 54.06 | 83.35 | 60.88 |
| | $\eta_{\boldsymbol{\tau}} = 1 \times 10^{-2}$ | 53.67 | 89.54 | 41.09 | 78.72 | 24.39 | 54.66 | 83.82 | 60.84 |
| | $\eta_{\boldsymbol{\tau}} = 1 \times 10^{-3}$ | 54.49 | 89.31 | 41.98 | 78.70 | 25.41 | 54.24 | 83.82 | **61.13** |

**Budget Control.** Tab. 5b compares different designs for rank budget estimation. Both gradient-derived targets outperform the fixed budget $R(\mathcal{B}) \equiv 2$, supporting the use of gradient complexity for adaptive budget estimation. The candidate gradient performs best among unsmoothed estimators, consistent with its ungated formulation retaining inactive components in budget estimation. We also explore applying exponential moving average (EMA) smoothing to the estimated budget. It provides no clear gains, likely due to the robustness of $\text{sign}(\cdot)$ in Eq. (6) to batch-wise variation.

**Rank Disentanglement.** Tab. 5c compares several rank disentanglement designs: (i) *no disentanglement*; (ii) *Stiefel optimization*, which optimizes $\boldsymbol{P}$ and $\boldsymbol{Q}$ on their Stiefel manifolds through Riemannian updates to ensure strict global orthonormality (Li et al., 2025c); (iii) *orthonormal loss*, as defined by $\mathcal{L}_{\text{ortho}} = \|\boldsymbol{P}^\top\boldsymbol{P} - \boldsymbol{I}_r\|_F^2 + \|\boldsymbol{Q}\boldsymbol{Q}^\top - \boldsymbol{I}_r\|_F^2$; and (iv) $\mathcal{L}_{\text{rank}}$, as defined in Eq. (8). The performance degradation under global orthonormality suggests that such indiscriminate constraints can interfere with task-driven optimization. In contrast, $\mathcal{L}_{\text{rank}}$ preserves task performance while reducing the gap between activation count and effective rank as shown in Fig. 1b. We further compare DMoRE with the $\boldsymbol{BA}$ parameterization that applies only co-activation-aware directional regularization without constraining component norms. The gain over this variant further supports explicitly separating projection directions from component scales.

**Hyperparameter Sensitivity.** Tab. 5d and Tab. 5e evaluate the sensitivity to $\eta_{\text{rank}}$ and $\eta_{\boldsymbol{\tau}}$, respectively, by varying one value at a time. The results show no clear performance variation across the evaluated scales, demonstrating the robustness of DMoRE to both hyperparameters.

## 5 Conclusion

In this work, we present DMoRE, a principled conditional computation framework for input-dependent rank allocation. We introduce dynamic self-routing with gradient-derived budget estimation, enabling input-dependent rank allocation without an external router or manually specified rank budget. We further develop co-activation-aware rank disentanglement, which more reliably translates component activations into effective rank capacity. Experiments across diverse tasks validate the effectiveness and robustness of DMoRE across adaptation settings, with structured rank allocation patterns at multiple granularities. We hope our work provides a foundation for input-dependent rank allocation and inspires the development of more flexible and adaptive PEFT methods.

### AI USE STATEMENT

In this work, we used generative AI tools for assisting in the writing of proofs, designing or providing feedback on research methodology or experiments, and assisting with translation. We have not used generative AI tools for proposing or refining hypotheses, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, implementing methods, interpreting results, or supporting qualitative or thematic data analysis, and the remaining tasks with required disclosure are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures or images, summarizing or analyzing existing literature, brainstorming, and formatting references. We have reviewed all AI-assisted work. The AI-assisted proofs and mathematical derivations were verified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL technical report. *arXiv:2511.21631*, 2025.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*, 2013.

Shubhankar Borse, Shreya Kadambi, Nilesh Prasad Pandey, Kartikeya Bhardwaj, Viswanath Ganapathy, Sweta Priyadarshi, Risheek Garrepalli, Rafael Esteves, Munawar Hayat, and Fatih Porikli. FouRA: Fourier low-rank adaptation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017. doi: 10.1109/JPROC.2017.2675998.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3606–3613, 2014.

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 *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, 2019. doi: 10.18653/v1/N19-1300.

Peter Clark, Isaac Cowhey, Oren Etzioni, et al. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. Training verifiers to solve math word problems. *arXiv:2110.14168*, 2021.

Xuan Cui, HuiYue Li, Run Zeng, Yunfei Zhao, Jinrui Qian, Wei Duan, Bo Liu, and Zhanpeng Zhou. IGU-loRA: Adaptive rank allocation via integrated gradients and uncertainty-aware scoring. In *The Fourteenth International Conference on Learning Representations*, 2026.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, et al. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv:2512.02556*, 2025.

Abdessalam Ed-dib, Zhanibek Datbayev, and Amine M. Aboussalah. GeLoRA: Geometric adaptive ranks for efficient LoRA fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

SooHwan Eom, Hee Suk Yoon, Eunseop Yoon, Mark A. Hasegawa-Johnson, and Chang D. Yoo. Gated loRA: Dual-purpose projections for parameter-efficient mini-expert fine-tuning, 2026.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024.

Chendi Ge, Xin Wang, Zeyang Zhang, et al. Dynamic mixture of curriculum loRA experts for continual multimodal instruction tuning. In *International Conference on Machine Learning*, 2025.

A. Grattafiori, A. Dubey, A. Jauhri, et al. The Llama 3 herd of models. *arXiv:2407.21783*, 2024.

Yuxuan Gu, Wuyang Zhou, Giorgos Iacovides, and Danilo Mandic. TeRA: Vector-based random tensor network for high-rank adaptation of large language models. *arXiv:2509.03234*, 2025.

Haonan He, Peng Ye, Yuchen Ren, Yuan Yuan, Luyang Zhou, Shucun Ju, and Lei Chen. GoRA: Gradient-driven adaptive low rank adaptation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019. doi: 10.1109/JSTARS.2019.2918242.

Dan Hendrycks, Collin Burns, Steven Basart, et al. Measuring massive multitask language understanding. *arXiv:2009.03300*, 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. HiRA: Parameter-efficient hadamard high-rank adaptation for large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yitong Huang, Ziqi Yang, Zihui Wang, Jianzhong Qi, Rongshan Yu, Xiaoliang Fan, and Cheng Wang. Hybrid routing for a mixture of lora experts. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, volume 40, pp. 31211–31219. AAAI Press, 2026. doi: 10.1609/aaai.v40i37.40383.

Minyoung Huh, Brian Cheung, Jeremy Bernstein, et al. Training neural networks from scratch with parallel low-rank adapters. *arXiv:2402.16828*, 2024.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Yufeng Jiang and Yiqing Shen. M4oe: A foundation model for medical multimodal image segmentation with mixture of experts. In *Proceedings of Medical Image Computing and Computer Assisted Intervention*, 2024.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994. doi: 10.1162/neco.1994.6.2.181.

Wenjun Ke, Jiahao Wang, Peng Wang, Jiajun Liu, Dong Nie, Guozheng Li, and Yining Li. Unveiling loRA intrinsic ranks via salience analysis. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Minsoo Kim, Sihwa Lee, Wonyong Sung, and Jungwook Choi. RA-LoRA: Rank-adaptive parameter-efficient fine-tuning for accurate 2-bit quantized large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 15773–15786, 2024. doi: 10.18653/v1/2024.findings-acl.933.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *arXiv:2404.15159*, 2024.

Minglei Li, Peng Ye, Jingqi Ye, haonan he, and Tao Chen. E²loRA: Efficient and effective low-rank adaptation with entropy-guided adaptive sharing. In *The Fourteenth International Conference on Learning Representations*, 2026a.

Shiwei Li, Xiandi Luo, Haozhao Wang, Xing Tang, Ziqiang Cui, Dugang Liu, Yuhua Li, Xiuqiang He, and Ruixuan Li. Beyond higher rank: Token-wise input-output projections for efficient low-rank adaptation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

Shiwei Li, Xiandi Luo, Haozhao Wang, Xing Tang, Ziqiang Cui, Dugang Liu, Yuhua Li, Yichen Li, Xiuqiang He, and Ruixuan Li. BoRA: Towards more expressive low-rank adaptation with block diversity. In *The Fourteenth International Conference on Learning Representations*, 2026b.

Xinlong Li, Weijieying Ren, Wei Qin, Lei Wang, Tianxiang Zhao, and Richang Hong. Analyzing and reducing catastrophic forgetting in parameter efficient tuning. In *ICASSP*, 2025b. doi: 10.1109/ICASSP49660.2025.10889361.

Zhizhong Li, Sina Sajadmanesh, Jingtao Li, and Lingjuan Lyu. StelLA: Subspace learning in low-rank adaptation using stiefel manifold. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025c.

Zongqian Li, Yixuan Su, Han Zhou, Zihao Fu, and Nigel Collier. Flexi-loRA: Efficient loRA fine-tuning with input-adaptive dynamic ranks. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*, 2025d.

Muqing Liu, Chongjie Si, and Yuheng Jia. FlexloRA: Entropy-guided flexible low-rank adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026.

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*, pp. 1104–1114, 2024a. doi: 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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Zefang Liu and Jiahua Luo. AdaMoLE: Fine-tuning large language models with adaptive mixture of low-rank adaptation experts. In *First Conference on Language Modeling*, 2024.

Haodong Lu, Chongyang Zhao, Minhui Xue, Lina Yao, Kristen Moore, and Dong Gong. Little by little: Continual learning via incremental mixture of rank-1 associative memory experts. In *Forty-third International Conference on Machine Learning*, 2026.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv:2402.12851*, 2024.

Ang Lv, Ruobing Xie, Yining Qian, et al. Autonomy-of-experts models. In *International Conference on Machine Learning*, 2025.

Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. A survey on lora of large language models. *Frontiers of Computer Science*, 19(7):197605, 2025. doi: 10.1007/s11704-024-40663-9.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

Lin Ning, Harsh Lara, Meiqi Guo, and Abhinav Rastogi. MoDE: Effective multi-task parameter efficient fine-tuning with a mixture of dyadic experts. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 8248–8261, 2025. doi: 10.18653/v1/2025.findings-naacl.460.

Jie Ou, Shuaihong Jiang, Yingjun Du, and Cees GM Snoek. GateRA: Token-aware modulation for parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32610–32618, 2026. doi: 10.1609/aaai.v40i38.40538.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference*, pp. 606–610, 2007.

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

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

Reece S Shuttleworth, Jacob Andreas, Antonio Torralba, and Pratyusha Sharma. LoRA vs full fine-tuning: An illusion of equivalence. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012. doi: 10.1016/j.neunet.2012.02.016.

Mengyang Sun, Yihao Wang, Tao Feng, et al. A stronger mixture of low-rank experts for fine-tuning foundation models. In *International Conference on Machine Learning*, 2025.

Nurbek Tastan, Stefanos Laskaridis, Martin Takáč, Karthik Nandakumar, and Samuel Horváth. LoFT: Low-rank adaptation that behaves like full fine-tuning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Chunlin Tian, Zhan Shi, Zhijiang Guo, et al. HydraloRA: An asymmetric loRA architecture for efficient fine-tuning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

Lean Wang, Huazuo Gao, Chenggang Zhao, et al. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv:2408.15664*, 2024a.

Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-rank adaptation with gradient approximation. In *Advances in Neural Information Processing Systems*, volume 37, 2024b.

Martin Weyssow, Xin Zhou, Kisub Kim, et al. Exploring parameter-efficient fine-tuning techniques for code generation with large language models. *ACM Transactions on Software Engineering and Methodology*, 34(7), 2025. doi: 10.1145/3714461.

Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3485–3492. IEEE, 2010. doi: 10.1109/CVPR.2010.5539970.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv:2505.09388*, 2025a.

Haodong Yang, Lei Wang, and Md Zakir Hossain. SRLoRA: Subspace recomposition in low-rank adaptation via importance-based fusion and reinitialization. *arXiv:2505.12433*, 2025b.

Jingqi Ye, Haonan He, Minglei Li, Fujun Han, Tao Chen, and Peng Ye. Gradient intrinsic dimensionality alignment: Narrowing the gap between low-rank adaptation and full fine-tuning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, et al. HellaSwag: Can a machine really finish your sentence? *arXiv:1905.07830*, 2019.

Yuchen Zeng and Kangwook Lee. The expressive power of low-rank adaptation. In *International Conference on Learning Representations*, 2024.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11975–11986, 2023.

Hao Zhang, Zhenjia Li, Runfeng Bao, et al. HyperAdaLoRA: Accelerating lora rank allocation during training via hypernetworks without sacrificing performance. *arXiv:2510.02630*, 2025.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, et al. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023.

Ziyu Zhao, Yixiao Zhou, Zhi Zhang, et al. Each rank could be an expert: Single-ranked mixture of experts lora for multi-task learning. *arXiv:2501.15103*, 2025.

Yuan Zhuang, Yi Shen, Yuexin Bian, Qing Su, Shihao Ji, Yuanyuan Shi, and Fei Miao. LD-moLE: Learnable dynamic routing for mixture of loRA experts. In *The Fourteenth International Conference on Learning Representations*, 2026.

Heming Zou, Yunliang Zang, Wutong Xu, et al. FlyloRA: Boosting task decoupling and parameter efficiency via implicit rank-wise mixture-of-experts. In *The Annual Conference on Neural Information Processing Systems*, 2025.

# A METHOD

## A.1 GRADIENT ANALYSIS OF EXTERNAL ROUTING

In this subsection, we analyze the gradients that would arise if an external router were used to activate the rank-1 components. We follow the rank-wise formulation in Eq. (2) and retain the conventional $\boldsymbol{BA}$ parameterization. Given an input $\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}$, the output $\boldsymbol{y}_{\text{ext}} \in \mathbb{R}^{d_{\text{out}}}$ is defined as:

$$\boldsymbol{y}_{\text{ext}} = \boldsymbol{W}\boldsymbol{x} + \frac{\alpha}{r}\sum_{i=1}^{r} \boldsymbol{g}_{\text{ext}}(\boldsymbol{x})_i \boldsymbol{\rho}(\boldsymbol{x})_i \boldsymbol{b}_i \boldsymbol{a}_i^\top \boldsymbol{x}, \tag{10}$$

where $\boldsymbol{b}_i = \boldsymbol{B}[:, i] \in \mathbb{R}^{d_{\text{out}}}$ is the $i$-th column of $\boldsymbol{B}$, and $\boldsymbol{a}_i = \boldsymbol{A}[i, :]^\top \in \mathbb{R}^{d_{\text{in}}}$ is the transpose of the $i$-th row of $\boldsymbol{A}$. We retain threshold-based sparse activation by defining the binary gate as:

$$\boldsymbol{g}_{\text{ext}}(\boldsymbol{x}) = \mathbb{1}_{\text{sg}}(\boldsymbol{\rho}(\boldsymbol{x}) - \boldsymbol{\tau} > 0), \quad \boldsymbol{\rho}(\boldsymbol{x}) = \text{sigmoid}(\boldsymbol{W}_{\text{ext}}\boldsymbol{x}), \tag{11}$$

where $\boldsymbol{W}_{\text{ext}} \in \mathbb{R}^{r \times d_{\text{in}}}$ denotes the weight of the external router, and $\boldsymbol{\tau} \in \mathbb{R}^r$ is the component-wise activation threshold. We denote the transpose of the $i$-th router row by $\boldsymbol{w}_{\text{ext},i} = \boldsymbol{W}_{\text{ext}}[i, :]^\top \in \mathbb{R}^{d_{\text{in}}}$. Herein, the binary gate $\boldsymbol{g}_{\text{ext}}(\boldsymbol{x})$ is detached from gradient computation, while the routing scores $\boldsymbol{\rho}(\boldsymbol{x})$ serve as differentiable multiplicative weights on the selected rank-1 components, allowing task gradients to backpropagate to the external router.

**Routing Gradient.** Since the binary gate is detached from gradient computation, $\nabla_{\boldsymbol{\rho}(\boldsymbol{x})_i} \boldsymbol{g}_{\text{ext}}(\boldsymbol{x})_j = 0$. The gradient of the task loss $\mathcal{L}_{\text{task}}$ with respect to $\boldsymbol{\rho}(\boldsymbol{x})_i$ is:

$$\begin{aligned}
\nabla_{\boldsymbol{\rho}(\boldsymbol{x})_i} \mathcal{L}_{\text{task}} &= \left(\nabla_{\boldsymbol{\rho}(\boldsymbol{x})_i} \boldsymbol{y}_{\text{ext}}\right)^\top \nabla_{\boldsymbol{y}_{\text{ext}}} \mathcal{L}_{\text{task}} \\
&= \frac{\alpha}{r}\sum_{j=1}^{r} \boldsymbol{g}_{\text{ext}}(\boldsymbol{x})_j \underbrace{\nabla_{\boldsymbol{\rho}(\boldsymbol{x})_i} \boldsymbol{\rho}(\boldsymbol{x})_j}_{\mathbb{1}_{i=j}} \left(\boldsymbol{a}_j^\top \boldsymbol{x}\right) \boldsymbol{b}_j^\top \nabla_{\boldsymbol{y}_{\text{ext}}} \mathcal{L}_{\text{task}} \\
&= \frac{\alpha}{r} \boldsymbol{g}_{\text{ext}}(\boldsymbol{x})_i \left(\boldsymbol{a}_i^\top \boldsymbol{x}\right) \boldsymbol{b}_i^\top \nabla_{\boldsymbol{y}_{\text{ext}}} \mathcal{L}_{\text{task}}.
\end{aligned} \tag{12}$$

**Latent Response Gradient.** The gradient of the task loss with respect to the scalar latent response $\boldsymbol{a}_i^\top \boldsymbol{x}$ is given by:

$$\begin{aligned}
\nabla_{\boldsymbol{a}_i^\top \boldsymbol{x}} \mathcal{L}_{\text{task}} &= \left(\nabla_{\boldsymbol{a}_i^\top \boldsymbol{x}} \boldsymbol{y}_{\text{ext}}\right)^\top \nabla_{\boldsymbol{y}_{\text{ext}}} \mathcal{L}_{\text{task}} \\
&= \frac{\alpha}{r}\sum_{j=1}^{r} \boldsymbol{g}_{\text{ext}}(\boldsymbol{x})_j \boldsymbol{\rho}(\boldsymbol{x})_j \underbrace{\nabla_{\boldsymbol{a}_i^\top \boldsymbol{x}} \left(\boldsymbol{a}_j^\top \boldsymbol{x}\right)}_{\mathbb{1}_{i=j}} \boldsymbol{b}_j^\top \nabla_{\boldsymbol{y}_{\text{ext}}} \mathcal{L}_{\text{task}} \\
&= \frac{\alpha}{r} \boldsymbol{g}_{\text{ext}}(\boldsymbol{x})_i \boldsymbol{\rho}(\boldsymbol{x})_i \boldsymbol{b}_i^\top \nabla_{\boldsymbol{y}_{\text{ext}}} \mathcal{L}_{\text{task}}.
\end{aligned} \tag{13}$$

**Routing-Latent Response Alignment.** For a single input $\boldsymbol{x}$, consider a small gradient-descent step on $\mathcal{L}_{\text{task}}$, updating $\boldsymbol{W}_{\text{ext}}$ and $\boldsymbol{A}$ with positive learning rates $\eta_{\text{r}}$ and $\eta_{\text{e}}$, respectively. Let $\Delta$ denote the resulting changes evaluated at unchanged input features. The sigmoid chain rule and linear down-projection give:

$$\begin{aligned}
\Delta\boldsymbol{\rho}(\boldsymbol{x})_i &\simeq \boldsymbol{\rho}(\boldsymbol{x})_i\left[1 - \boldsymbol{\rho}(\boldsymbol{x})_i\right] \left(\Delta \boldsymbol{w}_{\text{ext},i}\right)^\top \boldsymbol{x} \\
&= -\eta_{\text{r}} \boldsymbol{\rho}(\boldsymbol{x})_i\left[1 - \boldsymbol{\rho}(\boldsymbol{x})_i\right] \left(\nabla_{\boldsymbol{w}_{\text{ext},i}} \mathcal{L}_{\text{task}}\right)^\top \boldsymbol{x} \\
&= -\eta_{\text{r}} \boldsymbol{\rho}(\boldsymbol{x})_i\left[1 - \boldsymbol{\rho}(\boldsymbol{x})_i\right] \left(\nabla_{\boldsymbol{\rho}(\boldsymbol{x})_i} \mathcal{L}_{\text{task}}\right) \left(\nabla_{\boldsymbol{w}_{\text{ext},i}} \boldsymbol{\rho}(\boldsymbol{x})_i\right)^\top \boldsymbol{x} \\
&= -\eta_{\text{r}} \left[\boldsymbol{\rho}(\boldsymbol{x})_i\right]^2 \left[1 - \boldsymbol{\rho}(\boldsymbol{x})_i\right]^2 \left(\nabla_{\boldsymbol{\rho}(\boldsymbol{x})_i} \mathcal{L}_{\text{task}}\right) \boldsymbol{x}^\top \boldsymbol{x} \\
&= -\eta_{\text{r}} \left[\boldsymbol{\rho}(\boldsymbol{x})_i\right]^2 \left[1 - \boldsymbol{\rho}(\boldsymbol{x})_i\right]^2 \|\boldsymbol{x}\|_2^2 \nabla_{\boldsymbol{\rho}(\boldsymbol{x})_i} \mathcal{L}_{\text{task}},
\end{aligned} \tag{14}$$

$$\begin{aligned}
\Delta\left(\boldsymbol{a}_i^\top \boldsymbol{x}\right) &= \left(\Delta \boldsymbol{a}_i\right)^\top \boldsymbol{x} \\
&= -\eta_{\text{e}} \left(\nabla_{\boldsymbol{a}_i} \mathcal{L}_{\text{task}}\right)^\top \boldsymbol{x} \\
&= -\eta_{\text{e}} \left[\left(\nabla_{\boldsymbol{a}_i^\top \boldsymbol{x}} \mathcal{L}_{\text{task}}\right) \boldsymbol{x}\right]^\top \boldsymbol{x} \\
&= -\eta_{\text{e}} \|\boldsymbol{x}\|_2^2 \nabla_{\boldsymbol{a}_i^\top \boldsymbol{x}} \mathcal{L}_{\text{task}}.
\end{aligned}$$

Substituting the preceding loss gradients and applying the first-order expansion $\Delta[(\boldsymbol{a}_i^\top \boldsymbol{x})^2] \simeq 2(\boldsymbol{a}_i^\top \boldsymbol{x})\Delta(\boldsymbol{a}_i^\top \boldsymbol{x})$ yields:

$$\Delta\boldsymbol{\rho}(\boldsymbol{x})_i \simeq \underbrace{\frac{\eta_{\mathrm{r}}}{2\eta_{\mathrm{e}}}\boldsymbol{\rho}(\boldsymbol{x})_i\big[1-\boldsymbol{\rho}(\boldsymbol{x})_i\big]^2}_{>0}\,\Delta\Big[\big(\boldsymbol{a}_i^\top \boldsymbol{x}\big)^2\Big]\,. \tag{15}$$

The positive coefficient indicates that, under these local update assumptions, the external router tends to increase its preference for a component when task-driven learning strengthens that component's latent response to the same input, and to decrease its preference when the response is weakened. Thus, the latent-response magnitude is itself shaped by a task signal that also guides external routing, suggesting its use as an intrinsic proxy for component relevance.

### A.2 Gradient Analysis of Self-Routing without Detachment

In this subsection, we analyze the gradient that would arise if the routing branch in Eq. (5) were not detached. Given that the hard indicator has zero derivative almost everywhere, we apply a straight-through estimator (STE) (Bengio et al., 2013), where the binary gating is retained in the forward pass and its backward derivative is approximated by a differentiable non-decreasing surrogate function $\phi$ with $\phi'(\cdot) \geq 0$. We apply the surrogate gradient only to active components by multiplying it by the forward-pass mask $\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i$. Thus, for the $i$-th rank-1 component, the per-input gradient with respect to $\theta_i \in \{\boldsymbol{q}_i, \lambda_i\}$ can be expressed as follows using the product rule:

$$\begin{aligned}\nabla_{\theta_i}\mathcal{L}_{\text{task}} = &\underbrace{\frac{\alpha}{r}\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i\big(\boldsymbol{p}_i^\top \nabla_{\boldsymbol{y}_{\text{DMoRE}}}\mathcal{L}_{\text{task}}\big)\,\nabla_{\theta_i}\big(\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big)}_{\nabla_{\theta_i}^{\text{expert}}\mathcal{L}_{\text{task}}} \\ &+\underbrace{\frac{\alpha}{r}\left(\boldsymbol{p}_i^\top \nabla_{\boldsymbol{y}_{\text{DMoRE}}}\mathcal{L}_{\text{task}}\right)\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\,\nabla_{\theta_i}^{\text{STE}}\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i}_{\nabla_{\theta_i}^{\text{router}}\mathcal{L}_{\text{task}}}\,.\end{aligned} \tag{16}$$

Herein, $\nabla_{\theta_i}^{\text{expert}}\mathcal{L}_{\text{task}}$ propagates through the selected rank-1 transformation, whereas $\nabla_{\theta_i}^{\text{router}}\mathcal{L}_{\text{task}}$ propagates through the routing branch. The gating surrogate derivative is, almost everywhere,

$$\begin{aligned}\nabla_{\theta_i}^{\text{STE}}\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i &= \boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i \nabla_{\theta_i}\phi\big(\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big| - \boldsymbol{\tau}_i\big) \\ &= \boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i \phi'\big(\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big| - \boldsymbol{\tau}_i\big)\,\nabla_{\theta_i}\big(\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big| - \boldsymbol{\tau}_i\big)\,.\end{aligned} \tag{17}$$

Since $\boldsymbol{\tau}$ is updated outside backpropagation, $\nabla_{\theta_i}\boldsymbol{\tau}_i = 0$ and Eq. (17) can be simplified to:

$$\begin{aligned}\nabla_{\theta_i}^{\text{STE}}\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i &= \boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i \phi'\big(\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big| - \boldsymbol{\tau}_i\big)\,\nabla_{\theta_i}\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big| \\ &= \boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i \phi'\big(\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big| - \boldsymbol{\tau}_i\big)\,\operatorname{sign}\big(\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big)\,\nabla_{\theta_i}\big(\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big)\,.\end{aligned} \tag{18}$$

Substituting this into the definition of $\nabla_{\theta_i}^{\text{router}}\mathcal{L}_{\text{task}}$ gives:

$$\begin{aligned}\nabla_{\theta_i}^{\text{router}}\mathcal{L}_{\text{task}} &= \frac{\alpha}{r}\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i\left(\boldsymbol{p}_i^\top \nabla_{\boldsymbol{y}_{\text{DMoRE}}}\mathcal{L}_{\text{task}}\right)\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\,\phi'\big(\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big| - \boldsymbol{\tau}_i\big)\operatorname{sign}\big(\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big)\,\nabla_{\theta_i}\big(\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big) \\ &= \frac{\alpha}{r}\boldsymbol{g}_{\text{DMoRE}}(\boldsymbol{x})_i\,\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big|\,\phi'\big(\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big| - \boldsymbol{\tau}_i\big)\left(\boldsymbol{p}_i^\top \nabla_{\boldsymbol{y}_{\text{DMoRE}}}\mathcal{L}_{\text{task}}\right)\nabla_{\theta_i}\big(\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big) \\ &= \big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big|\,\phi'\big(\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big| - \boldsymbol{\tau}_i\big)\,\nabla_{\theta_i}^{\text{expert}}\mathcal{L}_{\text{task}}.\end{aligned} \tag{19}$$

Since $\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big|\,\phi'\big(\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big| - \boldsymbol{\tau}_i\big) \geq 0$, the router gradient is a non-negative scalar multiple of the expert gradient. For active components, it therefore preserves the per-input expert-gradient direction and, when nonzero, increases its magnitude. For inactive components, the explicit mask makes both the expert and router gradients zero.

*Remark* A.1 (Self-Reinforcement.). Combining Eqs. (16) and (19) yields:

$$\nabla_{\theta_i}\mathcal{L}_{\text{task}} = \big[1 + \big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big|\,\phi'\big(\big|\lambda_i \boldsymbol{q}_i^\top \boldsymbol{x}\big| - \boldsymbol{\tau}_i\big)\big]\,\nabla_{\theta_i}^{\text{expert}}\mathcal{L}_{\text{task}}.$$

This non-negative rescaling can induce self-reinforcing routing dynamics when task-driven updates strengthen the corresponding latent responses. Specifically, active components can receive larger task-aligned updates and may therefore adapt more rapidly, whereas inactive components receive

Table 6: **Additional quantitative comparison of LoRA variants on IC.** We report accuracy (↑) on SigLIP-ViT-L/16. #Param. indicates the number of trainable parameters. The **best** results are highlighted. ESAT = EuroSAT, RE45=RESISC45.

| Method | #Param. | Cars | DTD | ESAT | GTSRB | RE45 | SUN397 | SVHN | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| LoRA | 3.54M | 94.91 | 66.13 | 97.94 | 98.23 | 93.41 | 78.42 | 97.13 | 89.45 |
| DoRA | 3.76M | 95.01 | 66.40 | 97.80 | 98.31 | 93.62 | 78.54 | 97.18 | 89.55 |
| AdaLoRA | 3.54M | 94.81 | 64.70 | 97.92 | 98.62 | 94.13 | 80.01 | 97.18 | 89.62 |
| IGU-LoRA | 3.54M | 94.13 | 66.44 | 96.84 | 96.33 | 87.25 | 73.26 | 96.78 | 87.29 |
| TopLoRA | 5.31M | 95.33 | 68.12 | 98.25 | 98.75 | 94.91 | 80.33 | 97.35 | 90.44 |
| HydraLoRA | 4.20M | 94.72 | 65.13 | 97.52 | 97.78 | 92.64 | 77.58 | 96.95 | 88.90 |
| LD-MoLE | 4.46M | 94.58 | 64.86 | 97.20 | 97.23 | 91.56 | 76.72 | 96.74 | 88.41 |
| SMoRA | 5.31M | 94.51 | 64.73 | 97.05 | 96.65 | 91.26 | 76.32 | 96.53 | 88.15 |
| FlyLoRA | 1.77M | 95.09 | 67.80 | 97.56 | 96.60 | 92.26 | 77.54 | 96.16 | 89.00 |
| DMoRE | 3.54M | **95.72** | **73.19** | **98.37** | **98.99** | **95.41** | **80.99** | **97.57** | **91.46** |

neither expert nor STE routing gradients. Because the same latent magnitudes determine subsequent routing decisions, components with stronger responses are more likely to remain above their activation thresholds, whereas less frequently selected components receive fewer expert updates and may remain under-trained. Over training, this feedback can amplify early utilization asymmetries, yielding a rich-get-richer allocation and inefficient use of the available rank capacity.

*Remark* A.2 (Routing Adaptation under Detachment.). Detaching the binary gate removes direct gradient propagation through the routing branch, but does not render the routing policy static. Routing continues to evolve through two complementary mechanisms. First, *task-driven evolution* arises from updates to the expert parameters under the downstream objective, which reshape the latent responses that determine the routing scores. Second, *utilization-driven evolution* arises from threshold updates based on observed activation frequencies, which promote balanced component usage under the estimated rank budget. This decoupled design allows routing to adapt to both the task objective and utilization statistics while avoiding the self-reinforcing gradient dynamics described above. Empirically, it yields stronger average performance without introducing the additional parameters or computational overhead of a separate routing network.

### A.3 Comparison with Existing Methods

In this subsection, we distinguish DMoRE from LoRA variants that similarly introduce input-dependent control over low-rank adaptation. As official implementations of these methods are not publicly available, we provide a detailed methodological comparison rather than empirical evaluation to avoid potentially inconsistent reimplementations.

**FouRA.** To mitigate data copying and distribution collapse in LoRA-adapted text-to-image diffusion models, FouRA (Borse et al., 2024) transforms input features into the Fourier domain and uses an auxiliary MLP-based gate to dynamically mask low-rank components across inputs and denoising steps. Although the resulting mask cardinality induces input-dependent rank variation, the activated capacity is determined implicitly through downstream optimization. Moreover, while Fourier-domain adaptation promotes the learning of subspaces that are more decorrelated from the base model weights, it does not explicitly address the rank collapse described in Sec. 3.1. In contrast, DMoRE explicitly estimates a target rank budget from the candidate-gradient spectrum and employs co-activation-aware rank disentanglement to encourage activated components to span non-redundant directions. This establishes a controllable connection between batch-level gradient complexity and aggregate allocated capacity. Furthermore, DMoRE derives routing decisions directly from the latent magnitudes, eliminating FouRA's auxiliary gating network and Fourier transformations.

**Flexi-LoRA.** To accommodate inputs of varying difficulty, Flexi-LoRA (Li et al., 2025d) trains a difficulty-aware router using categories annotated from task-specific performance metrics. For each input sequence, the router selects a rank $r$ from a predefined set and applies the first $r$ rank-1 components uniformly across adapted layers. In contrast, DMoRE operates at a different granularity. Without requiring a separate router or auxiliary difficulty categorization, it self-routes arbitrary subsets of rank-1 components for each token and calibrates the average activation based on the intrinsic dimensionality of the gradient. This enables finer-grained capacity allocation without imposing a

Table 7: **Standard deviations for the quantitative comparison of LoRA variants on NLU.** We report the standard deviations corresponding to the results in Tab. 1.

| Model | Method | CoLA | MNLI | MRPC | QNLI | QQP | SST-2 | STS-B | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| RoBERTa-B | LoRA | 0.58 | 0.12 | 0.39 | 0.22 | 0.05 | 0.40 | 0.48 | 0.08 |
| | DoRA | 0.25 | 0.81 | 0.16 | 0.20 | 0.09 | 0.29 | 0.55 | 0.16 |
| | AdaLoRA | 0.59 | 0.17 | 0.18 | 0.32 | 0.01 | 0.31 | 0.13 | 0.12 |
| | IGU-LoRA | 0.27 | 0.08 | 0.32 | 0.28 | 0.20 | 0.35 | 0.94 | 0.19 |
| | TopLoRA | 0.20 | 0.25 | 0.87 | 0.34 | 0.06 | 0.43 | 0.02 | 0.07 |
| | HydraLoRA | 0.72 | 0.06 | 0.85 | 0.34 | 0.09 | 0.13 | 0.12 | 0.13 |
| | LD-MoLE | 0.46 | 0.11 | 0.38 | 0.25 | 0.11 | 0.37 | 0.65 | 0.07 |
| | SMoRA | 0.62 | 0.12 | 0.55 | 0.34 | 0.07 | 0.07 | 0.81 | 0.14 |
| | FlyLoRA | 0.33 | 0.08 | 0.42 | 0.06 | 0.04 | 0.30 | 0.69 | 0.25 |
| | DMoRE | 0.68 | 0.16 | 0.61 | 0.11 | 0.06 | 0.29 | 0.73 | 0.15 |
| DeBERTa-v3-L | LoRA | 0.31 | 0.14 | 0.10 | 0.05 | 0.12 | 0.41 | 0.49 | 0.10 |
| | DoRA | 0.22 | 0.13 | 0.10 | 0.14 | 0.04 | 0.17 | 0.72 | 0.12 |
| | AdaLoRA | 0.06 | 0.09 | 0.84 | 0.05 | 0.10 | 0.17 | 0.49 | 0.15 |
| | IGU-LoRA | 0.65 | 0.49 | 0.18 | 0.89 | 0.51 | 0.80 | 0.06 | 0.22 |
| | TopLoRA | 0.33 | 0.15 | 0.97 | 0.45 | 0.06 | 0.35 | 0.58 | 0.14 |
| | HydraLoRA | 0.72 | 0.17 | 0.87 | 0.15 | 0.01 | 0.07 | 0.72 | 0.27 |
| | LD-MoLE | 0.59 | 0.05 | 0.42 | 0.13 | 0.05 | 0.29 | 0.31 | 0.13 |
| | SMoRA | 0.87 | 0.25 | 0.21 | 0.13 | 0.08 | 0.24 | 0.13 | 0.12 |
| | FlyLoRA | 0.12 | 0.03 | 0.68 | 0.16 | 0.11 | 0.30 | 0.15 | 0.06 |
| | DMoRE | 0.15 | 0.03 | 0.69 | 0.09 | 0.11 | 0.10 | 0.24 | 0.11 |

sequence-wide rank or requiring per-input difficulty labels. Whether local adaptation demand should be associated with *token-specific* semantic complexity, and how such complexity can be reliably defined and quantified, remain open questions for future work. Moreover, DMoRE regularizes both projection directions according to co-activation, mitigating directional redundancy so that additional activations more reliably translate into usable effective-rank capacity.

**Gated LoRA.** To reduce task interference from uniformly activating all LoRA rank directions, Gated LoRA (Eom et al., 2026) reuses down-projected features as self-gating scores and selects rank components via ReLU gating, relying on a manually specified sparsity target. In contrast, DMoRE derives a target rank budget from the gradient spectrum, adapting capacity to task demands without a predefined activation ratio. Alongside this, Gated LoRA regularizes only the up-projection $\boldsymbol{B}$ toward orthogonality, leaving the input projection $\boldsymbol{A}$ unconstrained. Consequently, its activation count may not strictly reflect realized effective rank. DMoRE instead regularizes both $\boldsymbol{P}$ and $\boldsymbol{Q}$ to better preserve usable rank capacity. Moreover, Gated LoRA lacks an explicit mechanism for balancing component utilization. DMoRE instead adaptively adjusts component-wise thresholds under the estimated rank budget to encourage load balancing.

# B Experiments

## B.1 Natural Language Understanding

**Training and Evaluation.** Following Wang et al. (2024b), we use the AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay$= 0$) with a cosine learning-rate schedule (3% warmup). We train each task independently with a batch size of 32 and a maximum sequence length of 128. The peak adapter learning rates are set to 1e-3 for AdaLoRA and IGU-LoRA and 1e-4 for all other methods for meaningful comparisons, while the task-specific classification head is fully fine-tuned with a learning rate of 1e-4. We report accuracy for all tasks except MRPC, QQP, and STS-B; for MRPC and QQP, we report the average of accuracy and F1, and for STS-B, we report the average of Pearson and Spearman correlations.

## B.2 Natural Language Reasoning

**Datasets.** We conduct experiments on diverse reasoning tasks: (i) *commonsense reasoning and reading comprehension* on ARC-Challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), Hel-

Table 8: **Standard deviations for the quantitative comparison of LoRA variants on NLR.** We report the standard deviations corresponding to the results in Tab. 2.

| Model | Method | ARC-C | BoolQ | GSM8K | HS | HE | MMLU | WG | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| LLaMA-3.2-3B | LoRA | 0.13 | 0.25 | 0.33 | 0.12 | 1.27 | 0.31 | 0.32 | 0.23 |
| | DoRA | 0.22 | 0.42 | 0.40 | 0.06 | 1.96 | 0.24 | 0.59 | 0.21 |
| | AdaLoRA | 0.20 | 0.22 | 0.36 | 0.05 | 0.35 | 0.22 | 0.37 | 0.01 |
| | IGU-LoRA | 1.65 | 0.74 | 0.50 | 0.21 | 1.27 | 0.39 | 0.80 | 0.54 |
| | TopLoRA | 0.40 | 0.23 | 0.31 | 0.05 | 0.35 | 0.15 | 0.71 | 0.12 |
| | HydraLoRA | 0.55 | 0.36 | 1.03 | 0.09 | 0.70 | 0.25 | 0.44 | 0.17 |
| | LD-MoLE | 0.38 | 0.37 | 0.27 | 0.09 | 1.06 | 0.19 | 0.09 | 0.24 |
| | SMoRA | 0.34 | 0.30 | 1.07 | 0.08 | 2.11 | 0.37 | 0.32 | 0.13 |
| | FlyLoRA | 0.10 | 0.09 | 0.16 | 0.08 | 1.61 | 0.31 | 0.49 | 0.18 |
| | DMoRE | 0.78 | 0.10 | 0.88 | 0.07 | 1.41 | 0.30 | 0.36 | 0.18 |
| Qwen-3-8B | LoRA | 1.69 | 0.22 | 0.50 | 0.18 | 2.14 | 0.15 | 0.39 | 0.10 |
| | DoRA | 1.83 | 0.18 | 0.33 | 0.01 | 2.66 | 0.13 | 0.55 | 0.32 |
| | AdaLoRA | 1.72 | 0.17 | 0.31 | 0.10 | 2.19 | 0.39 | 0.25 | 0.41 |
| | IGU-LoRA | 1.87 | 0.18 | 0.31 | 0.28 | 4.49 | 0.95 | 0.72 | 0.76 |
| | TopLoRA | 1.54 | 0.14 | 0.42 | 0.13 | 1.06 | 0.11 | 0.59 | 0.26 |
| | HydraLoRA | 1.74 | 0.18 | 0.62 | 0.06 | 1.96 | 0.14 | 0.09 | 0.09 |
| | LD-MoLE | 0.77 | 0.17 | 0.80 | 0.02 | 1.06 | 0.21 | 0.08 | 0.14 |
| | SMoRA | 0.43 | 0.09 | 0.23 | 0.03 | 2.79 | 0.10 | 0.08 | 0.44 |
| | FlyLoRA | 1.18 | 0.17 | 0.55 | 0.04 | 0.70 | 0.37 | 0.24 | 0.07 |
| | DMoRE | 0.79 | 0.28 | 0.37 | 0.01 | 0.70 | 0.27 | 0.14 | 0.36 |

laSwag (Zellers et al., 2019), and WinoGrande (Sakaguchi et al., 2021); (ii) *general knowledge understanding* on MMLU (Hendrycks et al., 2020); (iii) *mathematical reasoning* on GSM8K (Cobbe et al., 2021); and (iv) *code generation*, where models are trained on CodeAlpaca-20K (Chaudhary, 2023) and evaluated on HumanEval (Chen et al., 2021).

**Training and Evaluation.** We independently fine-tune on each task using AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay$= 0$) with a linear learning-rate schedule (3% warmup). The batch size is set to 16, and the peak learning rates are 5e-4 for AdaLoRA and IGU-LoRA and 5e-5 for all other methods. We use maximum sequence lengths of 128 for ARC-Challenge, HellaSwag, and MMLU; 256 for BoolQ and WinoGrande; and 512 for GSM8K and CodeAlpaca-20K. We evaluate models using lm-evaluation-harness (Gao et al., 2024) and report length-normalized accuracy for ARC-Challenge and HellaSwag, exact-match accuracy for GSM8K, pass@1 for HumanEval, and accuracy for all remaining tasks.

### B.3 Image Classification

**Datasets.** We conduct experiments on representative image classification tasks, including Stanford Cars (Krause et al., 2013), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), GTSRB (Stallkamp et al., 2012), RESISC45 (Cheng et al., 2017), SUN397 (Xiao et al., 2010), and SVHN (Netzer et al., 2011). These datasets span diverse visual recognition scenarios, from general object recognition to fine-grained and remote-sensing scene classification.

**Training and Evaluation.** We independently fine-tune on each task using AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay$= 0$) with a cosine learning rate schedule (3% warmup). We apply adapters to all linear modules in the vision transformer. The batch size is set to 64, and the peak learning rates are 1e-3 for AdaLoRA and IGU-LoRA and 1e-4 for all other methods to ensure a fair comparison. Classification results are obtained with the prompt "`a photo of a {class}`". We evaluate the final checkpoint and report top-1 accuracy.

**Additional Results.** Tab. 6 presents an extended comparison of LoRA variants on IC using SigLIP-ViT-L/16 as the backbone. DMoRE consistently achieves the best performance across all datasets, further demonstrating that its dynamic rank allocation mechanism transfers effectively to vision tasks and remains robust across different backbone architectures. These results corroborate the main findings in Sec. 4.3 and further support the broad applicability of DMoRE across modalities.

Table 9: **Standard deviations for the quantitative comparison of LoRA variants on IC.** We report the standard deviations corresponding to the results in Tabs. 3 and 6.

| Model | Method | Cars | DTD | EuroSAT | GTSRB | RESISC45 | SUN397 | SVHN | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| CLIP-ViT-B/16 | LoRA | 0.28 | 0.31 | 0.07 | 0.25 | 0.26 | 0.05 | 0.09 | 0.08 |
| | DoRA | 0.27 | 0.20 | 0.13 | 0.24 | 0.25 | 0.07 | 0.06 | 0.13 |
| | AdaLoRA | 0.15 | 0.16 | 0.09 | 0.03 | 0.15 | 0.15 | 0.07 | 0.01 |
| | IGU-LoRA | 0.64 | 0.27 | 0.28 | 0.87 | 1.49 | 0.38 | 0.98 | 0.23 |
| | TopLoRA | 0.52 | 0.52 | 0.10 | 0.10 | 0.08 | 0.17 | 0.09 | 0.16 |
| | HydraLoRA | 0.27 | 0.58 | 0.05 | 0.06 | 0.01 | 0.10 | 0.08 | 0.07 |
| | LD-MoLE | 0.22 | 0.03 | 0.11 | 0.30 | 0.23 | 0.12 | 0.05 | 0.05 |
| | SMoRA | 0.24 | 0.20 | 0.04 | 0.05 | 0.08 | 0.08 | 0.06 | 0.08 |
| | FlyLoRA | 0.18 | 0.56 | 0.15 | 0.29 | 0.34 | 0.16 | 0.05 | 0.15 |
| | DMoRE | 0.13 | 0.95 | 0.19 | 0.09 | 0.10 | 0.04 | 0.11 | 0.15 |
| SigLIP-ViT-L/16 | LoRA | 0.04 | 0.13 | 0.22 | 0.07 | 0.03 | 0.13 | 0.02 | 0.01 |
| | DoRA | 0.05 | 0.03 | 0.10 | 0.11 | 0.06 | 0.05 | 0.06 | 0.03 |
| | AdaLoRA | 0.09 | 0.06 | 0.17 | 0.10 | 0.12 | 0.04 | 0.09 | 0.02 |
| | IGU-LoRA | 0.15 | 0.09 | 0.15 | 0.41 | 1.24 | 0.02 | 0.14 | 0.13 |
| | TopLoRA | 0.04 | 0.25 | 0.05 | 0.09 | 0.08 | 0.05 | 0.02 | 0.06 |
| | HydraLoRA | 0.06 | 0.03 | 0.08 | 0.04 | 0.15 | 0.05 | 0.03 | 0.01 |
| | LD-MoLE | 0.06 | 0.08 | 0.18 | 0.11 | 0.10 | 0.04 | 0.03 | 0.02 |
| | SMoRA | 0.04 | 0.06 | 0.07 | 0.06 | 0.16 | 0.04 | 0.06 | 0.01 |
| | FlyLoRA | 0.06 | 0.11 | 0.10 | 0.10 | 0.23 | 0.07 | 0.02 | 0.08 |
| | DMoRE | 0.04 | 0.56 | 0.15 | 0.09 | 0.25 | 0.06 | 0.07 | 0.03 |

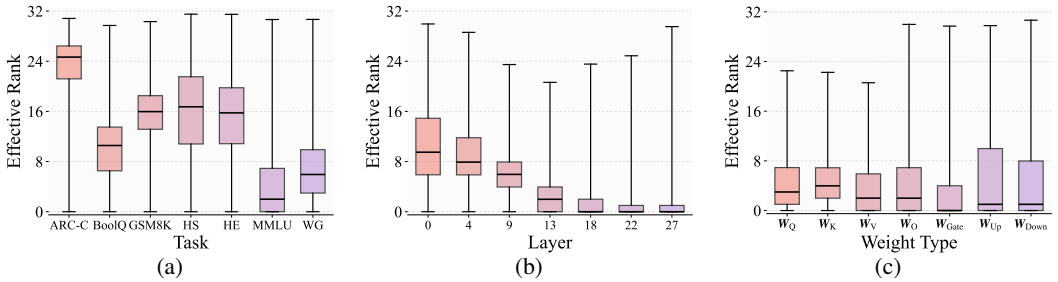



Figure 3: **Additional rank utilization analysis of DMoRE on NLR.** (a) Distribution of effective ranks across tasks aggregated across all adapted modules. (b) Distribution of effective ranks across layers. (c) Distribution of effective ranks across weight types. Box plots report the median, interquartile range, and minimum/maximum values. We fine-tune on Llama-3.2-3B and use $r = 32$ for a wider observable rank range.

### B.4 Analysis

**Across-Seed Variability.** Tabs. 7–9 show the standard deviations for the quantitative comparison of LoRA variants on NLU, NLR, and IC, respectively. It can be seen that the standard deviations remain small and are typically lower than the performance gains achieved by DMoRE. These results indicate that the improvements achieved by DMoRE are consistent across experimental runs.

**Rank Allocation across Tasks.** Fig. 3a shows that DMoRE learns distinct effective-rank distributions across tasks, reflecting task-specific capacity demands. For most tasks, the distributions are concentrated at roughly 25% to 50% of the available rank capacity but span the full effective-rank range. These results suggest that the gradient-induced budget control in Eq. (7) promotes sparse parameter utilization while preserving the flexibility to allocate higher ranks when needed.

**Rank Allocation across Layers.** Fig. 3b shows that DMoRE tends to learn higher effective ranks in earlier layers and lower effective ranks in deeper layers, although deeper layers retain a broad effective-rank range. One possible explanation is that earlier layers use broader adaptation subspaces for feature reshaping and contextual integration, whereas deeper layers use more selective update directions for task-specific or output-oriented refinement. Nevertheless, the observed pattern indicates that DMoRE can learn structured layer-wise rank allocations without explicit per-layer rank schedules or importance supervision.

Table 10: **Quantitative comparison of LoRA variants under different rank capacities on NLR.** We sweep rank capacities and report accuracy (↑) for all other tasks and Pass@1 (↑) for HumanEval. The **best** results are highlighted. HS = HellaSwag, HE = HumanEval, WG = WinoGrande.

| Rank Capacity | Method | ARC-C | BoolQ | GSM8K | HS | HE | MMLU | WG | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| $r = 8$ | LoRA | 49.46 | 88.20 | 34.49 | 78.99 | 23.98 | 55.73 | 83.64 | 59.21 |
| | AdaLoRA | 46.19 | 81.91 | 18.48 | 77.96 | 25.20 | 55.79 | 78.98 | 54.93 |
| | TopLoRA | 50.97 | 88.84 | 39.83 | 79.42 | 24.19 | 56.25 | 83.58 | 60.44 |
| | LD-MoLE | 46.64 | 86.87 | 33.21 | 78.80 | 24.39 | 55.82 | 82.69 | 58.35 |
| | SMoRA | 47.47 | 86.47 | 33.18 | 78.82 | 23.78 | 55.61 | 82.79 | 58.30 |
| | DMoRE | 54.49 | 89.31 | 41.98 | 78.70 | 25.41 | 54.24 | 83.82 | **61.13** |
| $r = 16$ | LoRA | 50.99 | 88.70 | 35.66 | 79.16 | 24.80 | 55.87 | 84.16 | 59.91 |
| | AdaLoRA | 48.07 | 87.78 | 35.76 | 79.09 | 25.41 | 55.62 | 83.79 | 59.36 |
| | TopLoRA | 52.42 | 89.36 | 41.72 | 79.54 | 23.37 | 56.04 | 83.14 | 60.80 |
| | LD-MoLE | 47.78 | 87.44 | 34.52 | 78.98 | 24.59 | 55.71 | 83.56 | 58.94 |
| | SMoRA | 47.24 | 86.37 | 33.97 | 78.84 | 24.39 | 55.88 | 82.80 | 58.50 |
| | DMoRE | 54.38 | 89.61 | 42.31 | 79.09 | 25.00 | 55.41 | 83.24 | **61.29** |
| $r = 32$ | LoRA | 52.22 | 89.00 | 37.30 | 79.24 | 24.19 | 56.25 | 83.90 | 60.30 |
| | AdaLoRA | 47.75 | 87.47 | 35.33 | 78.95 | 25.20 | 55.74 | 83.87 | 59.19 |
| | TopLoRA | 53.70 | 89.33 | 41.92 | 79.36 | 22.97 | 55.99 | 83.48 | 60.96 |
| | LD-MoLE | 49.52 | 88.08 | 36.09 | 79.07 | 25.00 | 55.74 | 84.03 | 59.65 |
| | SMoRA | 47.64 | 86.44 | 34.22 | 78.96 | 23.37 | 55.92 | 82.90 | 58.49 |
| | DMoRE | 54.49 | 89.42 | 41.24 | 79.25 | 26.02 | 55.51 | 84.08 | **61.43** |

**Rank Allocation across Weight Types.** Fig. 3c shows that DMoRE learns broadly similar effective-rank distributions across weight types, with moderate differences in their upper ranges. In particular, the QKV projections ($\boldsymbol{W}_{\text{Q,K,V}}$) exhibit lower maximum effective ranks, whereas the output and MLP projections ($\boldsymbol{W}_{\text{O,Gate,Up,Down}}$) can approach the full rank capacity. This suggests that DMoRE adapts rank allocation to different weight types, favoring lower capacity for QKV projections while retaining the flexibility to allocate substantially higher ranks to other transformations when needed.

**Robustness.** Tabs. 10 and 11 provide the detailed results underlying the robustness analysis presented in Sec. 4.4 and extend the evaluation to additional representative LoRA variants. Across all rank capacities and target-module configurations, DMoRE consistently achieves the highest average performance. These results indicate that DMoRE's improvements persist across the evaluated rank capacities and target-module configurations, supporting its robustness to these adaptation settings.

**Computational Overhead.** We benchmark training and inference efficiency on a single NVIDIA RTX A6000. Following the experimental settings described in Sec. 4.2, we use Llama-3.2-3B as the backbone and set the same maximum rank capacity $r = 8$ for all adapters. We measure training step time and peak CUDA memory over 50 synchronized optimizer steps after 10 warm-up steps using a batch size of 16 and a sequence length of 128. We measure inference per-token latency and peak CUDA memory over 50 synchronized next-token forward passes after 10 warm-up batches. Adapters are merged into the base weights during inference when supported.

## C Discussion

In this section, we discuss DMoRE from several perspectives, including its behavior and broader implications, current limitations, and potential directions for future work.

**Computational Overhead.** Given that DMoRE performs input-dependent routing, its adapter weights cannot be merged into the base weights, requiring the low-rank projections to be evaluated at inference time. Nevertheless, the self-routing mechanism reuses the down-projected features and incurs only lightweight element-wise thresholding and masking, without requiring a separate routing network. As a result, while DMoRE has higher latency than mergeable methods, it remains comparable to SMoRA and is around $3.3\times$ faster than LD-MoLE. Moreover, its peak inference memory is nearly identical to that of LoRA. Overall, these results demonstrate that DMoRE maintains competitive inference efficiency among non-mergeable LoRA variants.

Table 11: **Quantitative comparison of LoRA variants under different target-module sets on NLR.** We report accuracy (↑) for all other tasks and Pass@1 (↑) for HumanEval. `attn` targets $\{\boldsymbol{W}_{\mathrm{Q}}, \boldsymbol{W}_{\mathrm{K}}, \boldsymbol{W}_{\mathrm{V}}, \boldsymbol{W}_{\mathrm{O}}\}$, `ffn` targets $\{\boldsymbol{W}_{\mathrm{Gate}}, \boldsymbol{W}_{\mathrm{Up}}, \boldsymbol{W}_{\mathrm{Down}}\}$, and `all` targets their union. The **best** results are highlighted. HS = HellaSwag, HE = HumanEval, WG = WinoGrande.

| Target Module | Method | ARC-C | BoolQ | GSM8K | HS | HE | MMLU | WG | Avg. |
|---|---|---|---|---|---|---|---|---|---|
| attn | LoRA | 47.52 | 87.24 | 32.80 | 78.51 | 25.81 | 55.08 | 82.32 | 58.47 |
| | AdaLoRA | 47.21 | 87.27 | 32.85 | 78.62 | 26.22 | 54.44 | 82.45 | 58.44 |
| | TopLoRA | 49.63 | 88.43 | 36.11 | 79.09 | 23.58 | 55.22 | 82.58 | 59.24 |
| | LD-MoLE | 48.01 | 85.71 | 30.05 | 78.47 | 25.00 | 55.24 | 81.61 | 57.73 |
| | SMoRA | 48.66 | 85.14 | 31.08 | 78.47 | 25.00 | 55.57 | 81.51 | 57.92 |
| | DMoRE | 52.59 | 89.08 | 37.12 | 78.88 | 24.59 | 53.89 | 83.00 | **59.88** |
| ffn | LoRA | 47.84 | 87.63 | 33.84 | 78.92 | 23.78 | 55.46 | 83.32 | 58.69 |
| | AdaLoRA | 47.18 | 87.44 | 34.04 | 79.01 | 25.00 | 55.60 | 83.90 | 58.88 |
| | TopLoRA | 50.71 | 88.61 | 38.16 | 79.31 | 23.78 | 55.93 | 83.92 | 60.06 |
| | LD-MoLE | 47.04 | 86.56 | 32.02 | 78.77 | 23.58 | 55.88 | 82.53 | 58.05 |
| | SMoRA | 48.09 | 85.99 | 32.55 | 78.81 | 25.00 | 55.99 | 81.98 | 58.35 |
| | DMoRE | 53.36 | 89.43 | 40.23 | 79.00 | 23.17 | 55.10 | 83.56 | **60.55** |
| all | LoRA | 49.46 | 88.20 | 34.49 | 78.99 | 23.98 | 55.73 | 83.64 | 59.21 |
| | AdaLoRA | 46.19 | 81.91 | 18.48 | 77.96 | 25.20 | 55.79 | 78.98 | 54.93 |
| | TopLoRA | 50.97 | 88.84 | 39.83 | 79.42 | 24.19 | 56.25 | 83.58 | 60.44 |
| | LD-MoLE | 46.64 | 86.87 | 33.21 | 78.80 | 24.39 | 55.82 | 82.69 | 58.35 |
| | SMoRA | 47.47 | 86.47 | 33.18 | 78.82 | 23.78 | 55.61 | 82.79 | 58.30 |
| | DMoRE | 54.49 | 89.31 | 41.98 | 78.70 | 25.41 | 54.24 | 83.82 | **61.13** |

**Effective Rank Allocation.** As shown in Fig. 1b, DMoRE reduces the gap between the effective rank of the input-dependent adaptation and the number of active rank-1 components, without necessarily enforcing equality. Achieving equality would require the active components to induce an equal number of nonzero singular values with identical magnitudes, since directional independence alone is insufficient. $\mathcal{L}_{\text{rank}}$ instead promotes complementary directions among co-activated components while preserving learnable relative strengths through $\boldsymbol{\Lambda}$. Imposing spectral uniformity would further constrain these strengths and could limit task-dependent component specialization without general justification from the task objective.

**Emergent Token-Level Rank Allocation.** DMoRE deliberately separates batch-level capacity estimation from token-level rank allocation. This yields a simple threshold-based allocation mechanism in which the batch-level controller calibrates aggregate component utilization while token-level gates remain input-dependent. In contrast, explicit token-level control would require jointly predicting per-token cardinalities and enforcing both token- and component-level capacity constraints. This turns rank allocation into a coupled prediction-and-assignment problem that can be more difficult to optimize. It would also incur additional computational overhead and potentially require auxiliary supervision. Moreover, the current gradient-based criterion cannot directly yield token-specific budgets, since the candidate gradient of a single token is an outer product with rank at most one and thus provides no informative spectral estimate of its rank demand. Whether explicit token-level control is necessary, whether token-specific cardinalities and budgets can be estimated reliably and efficiently, and whether such estimation requires explicit difficulty supervision remain open questions, which we leave for future work.

**Rank Disentanglement.** As shown in Tab. 5c and Fig. 1b, $\mathcal{L}_{\text{rank}}$ primarily acts as a directional regularizer: it narrows the gap between the number of active components and the effective rank of their update while preserving downstream performance better than global orthogonality constraints. Its average performance gain over the unregularized $\boldsymbol{PAQ}$ parameterization is small in the evaluated single-task setting. One possible explanation is that independently trained tasks do not expose the adapter to the competing objectives or sequential distribution shifts that could make redundant update directions more costly. Moreover, directional regularization may sometimes oppose task-driven updates, limiting its immediate benefit even as it improves effective-rank utilization. We hypothesize that disentanglement could lead to higher performance gains in multi-task or continual-learning settings, where interference may be more pronounced. We leave this validation for future work.