FlyLoRA: Boosting Task Decoupling and Parameter Efficiency via Implicit Rank-Wise Mixture-of-Experts

Heming Zou[1]* **Yunliang Zang**[2]* **Wutong Xu**[1] **Yao Zhu**[1] **Xiangyang Ji**[1]†
[1]Department of Automation, Tsinghua University
[2]Academy of Medical Engineering and Translational Medicine, Tianjin University
{zouhm24, xwt22}@mails.tsinghua.edu.cn
yunliangzang@tju.edu.cn, ee_zhuy@zju.edu.cn
xyji@tsinghua.edu.cn

Abstract

Low-Rank Adaptation (LoRA) is a widely used parameter-efficient fine-tuning method for foundation models, but it suffers from parameter interference, resulting in suboptimal performance. Although Mixture-of-Experts (MoE)-based LoRA variants show promise in mitigating intra-task correlations in single-task instruction tuning, they introduce additional router parameters and remain ineffective in multi-task model merging where inter-task interference arises. Inspired by the fly olfactory circuit, we propose FlyLoRA, an implicit MoE-based LoRA variant that introduces: (1) rank-wise expert activation in the up-projection matrix, and (2) an implicit router that unifies expert routing and down-projection, where a frozen sparse random projection matrix replaces the traditional dense trainable version. This design resolves the trade-off between intra-task decorrelation and computational efficiency by eliminating the need for an explicit router, while inherently mitigating inter-task interference due to the orthogonality property of random matrices. Extensive experiments across four domains—general knowledge understanding, scientific question answering, mathematical reasoning, and code generation—demonstrate consistent performance improvements over existing methods. Beyond empirical gains, FlyLoRA highlights how biological structures can inspire innovations in AI technologies. Code is available at https://github.com/gfyddha/FlyLoRA.

1 Introduction

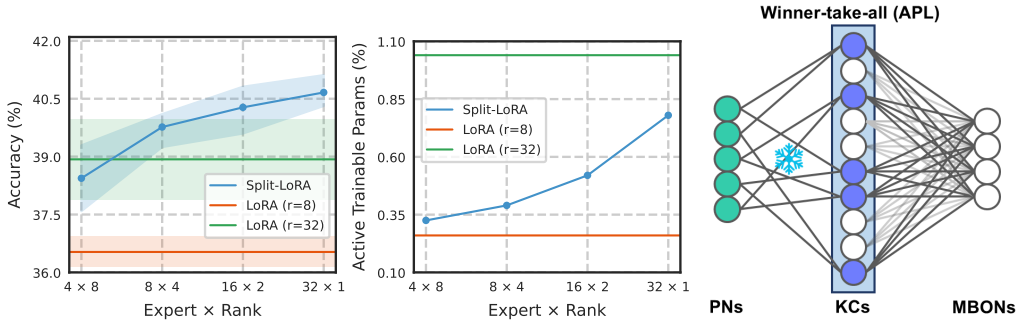
Foundation models have demonstrated remarkable cross-domain capabilities with the scaling of model parameters [1, 4, 15, 43, 57, 63, 64]. To enhance their performance on downstream tasks, Supervised Fine-Tuning (SFT) has become a typical post-training approach. However, full-parameter fine-tuning (Full FT) incurs prohibitive computational overhead and storage costs, making customized deployment impractical for most individual users. To address this issue, Parameter-Efficient Fine-Tuning (PEFT) [26, 27, 35, 40, 44, 47, 84, 90] has emerged as a widely adopted technique that significantly reduces resource consumption by keeping pre-trained weights frozen while fine-tuning only a small set of additional injected parameters.

Low-Rank Adaptation (LoRA) [27] is one of the most prominent PEFT methods. By leveraging the intrinsic low-dimensional properties of large language models [2, 36], LoRA approximates the parameter matrix update $\Delta \boldsymbol{W} \in \mathbb{R}^{m\times n}$ as the product of two low-rank matrices, $\boldsymbol{B} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{A} \in \mathbb{R}^{r\times n}$, where $r \ll \min(m, n)$. This method preserves much of the capability of Full FT across most tasks while substantially reducing both memory requirements and computational overhead.

*Equal contribution
†Corresponding author

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

 https://doi.org/10.52202/085713-0349



(a) (b) (c)

Figure 1: **(a)** Accuracy comparison under a fixed total rank $r = 32$ and activation rank $k = 8$. Finer-grained rank allocation (from 4 experts $\times$ 8 rank to 32 experts $\times$ 1 rank) yields consistent performance gains. **(b)** Activated trainable parameters (relative to Full FT) under the same budget. Increasing expert granularity leads to a monotonic rise in activated parameters due to router overhead. **(c)** Schematic of the fly olfactory circuit. Odor signals in projection neurons (PNs) are randomly projected to Kenyon cells (KCs), with each KC connecting to a fixed number of PNs (but not all), forming sparse connections. These signals are then selectively projected to mushroom body output neurons (MBONs), while lateral inhibition from an anterior paired lateral (APL) neuron suppresses weak KC-MBON connections, implementing a winner-take-all strategy. Thus, the number of activated KCs is much smaller than the total dimension of the KC layer.

However, to achieve strong performance on complex tasks, LoRA typically requires much higher ranks, which contradicts PEFT's core goal of efficiency [31, 45]. Moreover, interference within LoRA's ranks can impair training [76], leading to issues such as hallucination [22] and gradient explosion [58], thereby largely limiting its potential. We refer to this challenge as **intra-task interference**. Meanwhile, foundation models often need to integrate multiple capabilities to handle complex downstream tasks, but retraining on multi-domain corpora is expensive [71], particularly when several specialized models are already available. Consequently, model merging [11, 28, 29, 54] is widely used to combine LoRA components trained on different domains in a training-free manner. Arising from conflicts between different components, this process introduces another challenge: **inter-task interference**.

To address intra-task interference, several studies have incorporated the Mixture-of-Experts (MoE) architecture [19, 30, 34, 70, 78, 81] into LoRA [17, 20, 21, 37, 55, 76, 82, 89], where each expert learns specialized knowledge to partially achieve task decoupling. We refer to these approaches as **MoE-based LoRA methods**. They replace the original low-rank matrices with multiple experts and use a dynamic router to selectively activate them. By leveraging redundant parameters and sparse activations, these methods keep the computational budget comparable to LoRA with fewer total ranks. However, they may still suffer from interference within each expert. To investigate this issue, we conduct pilot studies on MMLU using Llama-3.1-8B, adopting Split-LoRA as a representative MoE-based method. As shown in Figure 1(a), finer-grained rank allocation yields consistent performance improvements under a fixed budget. However, as illustrated in Figure 1(b), pushing expert granularity to extremes also increases the number of activated trainable parameters due to additional router overhead. This trade-off makes it difficult to achieve both high performance and efficiency. Meanwhile, resolving inter-task interference has received limited emphasis in existing MoE-based LoRA methods.

Therefore, we seek to design an improved MoE-based LoRA variant that simultaneously achieves:

- *Reduced parameter interference among different ranks within a single LoRA component;*
- *Reduced parameter interference between different LoRA components;*
- *Reduced activated trainable parameters in routers.*

Inspired by the fly olfactory circuit [6, 13, 38, 42, 72, 100], which shows strong similarity to MoE-based LoRA, we introduce an implicit router to mitigate the trade-off between intra-task interference and efficiency. As shown in Figure 1(c), this leads to the design of **FlyLoRA**, which (1) treats matrix $\boldsymbol{A}$ as a frozen sparse random projection that maps inputs into a higher-rank space (e.g., $r = 32$ vs.

$r = 8$); and (2) simulates the bio-inspired "winner-take-all" mechanism by activating k rank-1 experts in $\boldsymbol{B}$ linked to the top-k magnitudes after projection by $\boldsymbol{A}$. This unifies the roles of $\boldsymbol{A}$ and router $\boldsymbol{G}$ into a single frozen projection, jointly performing down-projection and expert selection. Without explicit router parameters, the resulting implicit rank-wise MoE structure maintains computational efficiency while reducing intra-task interference. Moreover, we theoretically show that distinct random projections $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$ from different LoRA components naturally map task updates into approximately orthogonal subspaces, thereby alleviating inter-task interference.

In summary, the core contributions of our proposed FlyLoRA framework are:

• **Efficient Intra-task Decoupling:** By using implicit rank-wise MoE, we enable finer expert allocation with reduced parameter interference in single-task scenarios. Additionally, FlyLoRA surpasses MoE-based LoRA in efficiency by eliminating the need for router parameters.

• **Efficient Inter-task Decoupling:** In multi-task model merging scenarios, different random projections $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$ naturally form approximately orthogonal subspaces. This inherent property ensures different LoRA components operate in uncorrelated subspaces, thus achieves decoupling.

• **Neuroscience-Inspired Design:** The efficacy of our algorithm, combined with its structural alignment with the fly olfactory circuit, establishes a promising bridge between neuroscience and artificial intelligence.

2 Revisiting MoE-based LoRA Methods

2.1 Preliminaries

LoRA (visualized in Figure 2(a)) simulates weight updates during fine-tuning by decomposing the update matrix into two learnable low-rank matrices. Given a pretrained weight matrix $\boldsymbol{W}_0 \in \mathbb{R}^{m\times n}$, the parameter update is computed as:

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

where $\boldsymbol{B} \in \mathbb{R}^{m\times r}$, $\boldsymbol{A} \in \mathbb{R}^{r\times n}$, and the rank $r \ll \min(m, n)$. The scaling factor α is typically set to $2r$. For an input embedding $\boldsymbol{x} \in \mathbb{R}^n$, the forward pass becomes:

$$f_{\text{LoRA}}(\boldsymbol{x}) = \boldsymbol{W}'\boldsymbol{x} = \boldsymbol{W}_0\boldsymbol{x} + \frac{\alpha}{r}\boldsymbol{B}\boldsymbol{A}\boldsymbol{x}. \tag{2}$$

Here, $\boldsymbol{W}_0$ remains frozen during training, while only $\{\boldsymbol{A}, \boldsymbol{B}\}$ are updated. This approach reduces the number of trainable parameters from $\mathcal{O}(mn)$ to $\mathcal{O}(r(m + n))$, thereby achieving higher parameter efficiency. The low-rank structure allows LoRA to maintain stable performance while significantly reducing both computational overhead and GPU memory requirements during fine-tuning.

2.2 MoE-based LoRA Framework

The MoE paradigm (visualized in Figure 2(b)) extends LoRA by decomposing the low-rank adaptation into N specialized experts. Each expert $\boldsymbol{E}_i$ is parameterized by a pair of matrices $\{\boldsymbol{B}_i \in \mathbb{R}^{m\times r_i}, \boldsymbol{A}_i \in \mathbb{R}^{r_i\times n}\}$, where r_i denotes the expert-specific rank. The forward pass incorporates a gating mechanism $\boldsymbol{G}(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^N$ that dynamically routes inputs to activate the most relevant experts. Formally, the output combines the frozen pretrained weights $\boldsymbol{W}_0$ with a sparse combination of expert contributions:

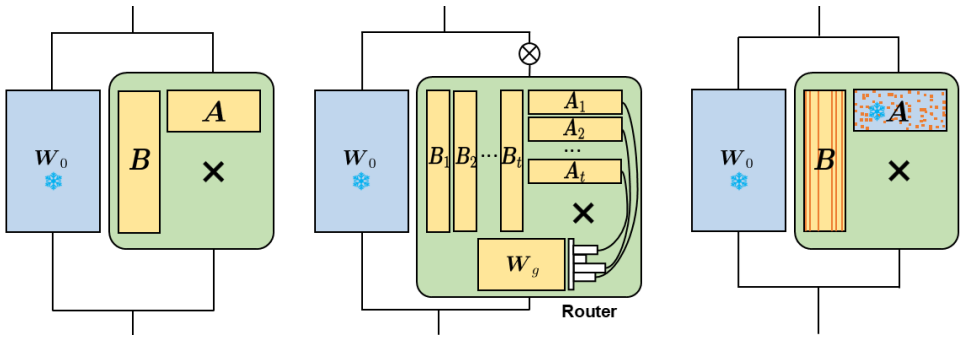
$$f_{\text{MoE-LoRA}}(\boldsymbol{x}) = \boldsymbol{W}_0\boldsymbol{x} + \frac{\alpha}{r}\sum_{i=1}^{N}\boldsymbol{G}(\boldsymbol{x})_i \cdot \underbrace{\boldsymbol{B}_i\boldsymbol{A}_i\boldsymbol{x}}_{\boldsymbol{E}_i(\boldsymbol{x})}, \tag{3}$$

where the router $\boldsymbol{G}(\boldsymbol{x})$ typically follows a top-k selection policy via a trainable projection $\boldsymbol{W}_g \in \mathbb{R}^{N\times n}$. For simplicity, we omit the activation function, formulating the router as:

$$\boldsymbol{G}(\boldsymbol{x}) = \text{top-}k(\boldsymbol{W}_g\boldsymbol{x}). \tag{4}$$

By activating only k experts per input, this design maintains computational efficiency. The sparse routing strategy enables conditional computation, which expands the model's representational capacity without incurring a proportional increase in computational cost. In our work, we implement Split-LoRA under this framework as a minimal yet representative instantiation of MoE-based LoRA. Further implementation details are provided in Appendix C.3.

https://doi.org/10.52202/085713-0349



(a) LoRA (b) MoE-based LoRA (c) FlyLoRA

Figure 2: **Schematic illustrations of different LoRA variants. (a)** LoRA employs low-rank matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ to simulate weight updates, where each row of $\boldsymbol{A}$ is fully connected to the corresponding column of $\boldsymbol{B}$. **(b)** MoE-based LoRA decomposes the updates into multiple small experts $\{\boldsymbol{A}_i, \boldsymbol{B}_i\}_{i=1}^{N}$ and uses a router to determine which experts should be activated. **(c)** FlyLoRA unifies the down-projection and router into a frozen matrix $\boldsymbol{A}$ and selectively activates only the ranks in $\boldsymbol{B}$ linked to the top-k magnitude activations after projection through $\boldsymbol{A}$.

2.3 Pushing MoE-based LoRA Architecture to the Extreme

Comparing Eq. 2 and Eq. 3 reveals that MoE-based LoRA can be viewed as a finer-grained, sparsely activated variant of LoRA, where the separation of experts mitigates task conflicts. Taking this decomposition to the extreme motivates our **rank-wise expert** design, where each expert governs a single rank, achieving the best decorrelating effect (see Figure 1(a)). Formally, for a rank-r LoRA, the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be decomposed into r rank-1 components:

$$f_{\text{rank-wise-LoRA}}(\boldsymbol{x}) = \boldsymbol{W}_0\boldsymbol{x} + \frac{\alpha}{r}\sum_{i=1}^{r} \boldsymbol{G}(\boldsymbol{x})_i \cdot \underbrace{\boldsymbol{b}_i\boldsymbol{a}_i\boldsymbol{x}}_{\boldsymbol{E}_i(\boldsymbol{x})}, \tag{5}$$

with $\boldsymbol{a}_i = \boldsymbol{A}[i,:] \in \mathbb{R}^{1\times n}$ and $\boldsymbol{b}_i = \boldsymbol{B}[:,i] \in \mathbb{R}^{m\times 1}$.

However, this approach introduces a scalability challenge: the router's linear layer $\boldsymbol{W}_g \in \mathbb{R}^{N\times n}$ grows linearly with the number of experts N (see Figure 1(b)). Under a fixed total rank, finer-grained experts with larger N make the explicit routing mechanism computationally prohibitive, undermining the efficiency gains of sparse activation.

To overcome this limitation, we seek an **implicit routing mechanism** that eliminates the need for the explicit router parameter $\boldsymbol{W}_g$ entirely. This entails finding a proxy that leverages intrinsic signals within the model to select the top-k experts, effectively approximating the function of the original router $\boldsymbol{G}$. To address this, we draw inspiration from the perspective of Singular Value Decomposition (SVD), which can also be viewed as a rank-wise decomposition. In SVD, the low-rank update matrix $\Delta\boldsymbol{W}$ can be decomposed as $\Delta\boldsymbol{W} = \sum_{i=1}^{r}\sigma_i\boldsymbol{u}_i\boldsymbol{v}_i^\top$, where σ_i denotes the i-th singular value (indicating the importance of the corresponding component), $\boldsymbol{u}_i$ is the i-th left-singular vector, and $\boldsymbol{v}_i$ is the i-th right-singular vector. Each component $\sigma_i\boldsymbol{u}_i\boldsymbol{v}_i^\top$ is a rank-1 update. The Eckart-Young-Mirsky theorem [18] guarantees that the top-k components, selected based on the magnitude of σ_i, provide the best rank-k approximation to the original rank-r matrix in terms of Frobenius norm, thereby capturing the most salient features with minimal reconstruction error. While exact SVD is computationally prohibitive and thus impractical in our framework, this insight naturally suggests that the magnitude of each rank-1 term, $\|\boldsymbol{b}_i\boldsymbol{a}_i\boldsymbol{x}\|$ in Eq. 6, approximately reflects its importance:

$$f_{\text{LoRA}}(\boldsymbol{x}) = \boldsymbol{W}_0\boldsymbol{x} + \frac{\alpha}{r}\sum_{i=1}^{r}\boldsymbol{b}_i\boldsymbol{a}_i\boldsymbol{x}. \tag{6}$$

Nevertheless, a naive approach of first computing all r terms $\boldsymbol{b}_i\boldsymbol{a}_i\boldsymbol{x}$ and then selecting the top-k would also forfeit the computational benefits of sparse activation, as the cost of computing all terms remains $\mathcal{O}(rmn)$. This necessitates a routing strategy that can identify the most important experts *before* fully computing their outputs. Furthermore, beyond efficient routing, another critical limitation of existing MoE-based LoRA methods is their lack of inherent support for multi-task deployment.

When merging models already fine-tuned on different tasks, interference between LoRA adapters often leads to significant performance degradation, as the underlying architecture does not structurally encourage task-specific updates to reside in orthogonal or non-overlapping parameter subspaces.

These dual challenges motivate the following two key design requirements for an improved MoE-based LoRA framework:

• *Implicit magnitude-based router for top-k activation, without explicit router parameters, enabling expert selection prior to full computation;*

• *Native support for training-free model merging through architectural properties that promote inter-task interference mitigation.*

3 FlyLoRA

Inspired by the fly olfactory circuit (Figure 1(c)), whose neural architecture inherently meets our requirements for MoE-based LoRA variants, we propose FlyLoRA (visualized in Figure 2(c)). Section 3.1 presents its formal design, while subsequent sections analyze its key advantages: Section 3.2 shows how a fixed $\boldsymbol{A}$ acts as an implicit router, Section 3.3 demonstrates intra-task decoupling, and Section 3.4 establishes inherent support for inter-task decoupling in model merging.

3.1 Formulation of FlyLoRA

In FlyLoRA, the matrix $\boldsymbol{A} \in \mathbb{R}^{r\times n}$ is sparse and frozen. It is randomly initialized at the beginning and remains frozen during training, implementing an intrinsic top-k operation in the projection space $\mathbb{R}^r$ for implicit routing. Given an input token $\boldsymbol{x} \in \mathbb{R}^n$, this process is formulated as:

$$\boldsymbol{y}' = \text{top-}k(\boldsymbol{y}) = \text{top-}k\left(\boldsymbol{A}\boldsymbol{x}\right), \tag{7}$$

where each row of $\boldsymbol{A}$ contains exactly p ($p < n$) non-zero entries independently sampled from $\mathcal{N}(0, \frac{1}{r^2})$ (a widely used standard initialization). We define the sparsity ratio as $\rho = \frac{p}{n}$. After projection through $\boldsymbol{A}$, only the columns $\boldsymbol{b}_i \in \mathbb{R}^m$ ($i \in \{1, \ldots, r\}$) in the up-projection matrix $\boldsymbol{B} \in \mathbb{R}^{m\times r}$ linked to dimensions with top-k ($k < r$) magnitudes in $\boldsymbol{A}\boldsymbol{x} \in \mathbb{R}^r$ are activated. Formally:

$$[\boldsymbol{B}\boldsymbol{y}']_i = \begin{cases} [\boldsymbol{B}\boldsymbol{y}]_i & \text{if the magnitude of } [\boldsymbol{y}]_i \text{ is among the top-}k \text{ values of } \boldsymbol{y}, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

To enhance training stability in this MoE structure, we incorporate a simple expert-wise bias term $\boldsymbol{d} \in \mathbb{R}^r$ for loss-free load balancing, following [43]. This auxiliary term is updated manually via:

$$\boldsymbol{d}_i \leftarrow \boldsymbol{d}_i + u \cdot \text{sign}(\bar{\boldsymbol{c}}_i - \boldsymbol{c}_i), \tag{9}$$

where u is a small learning rate, $\bar{\boldsymbol{c}}_i$ represents the expected assignment frequency for expert i, $\boldsymbol{c}_i$ tracks the actual assignment count, and $\text{sign}(\cdot)$ denotes the sign function. This bias term $\boldsymbol{d}$ is added to $\boldsymbol{A}\boldsymbol{x}$ in expert selection to promote the activation of under-activated experts and suppress over-activated experts, thereby achieving load balancing. Thus, the activated experts are selected by:

$$\mathcal{I}_{\text{top}k} = \left\{i_1, \ldots, i_k\right\} \quad \text{where} \quad i_j = \underset{i\notin\{i_1,\ldots,i_{j-1}\}}{\arg\max} \left(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}\right)_i. \tag{10}$$

The forward pass is then computed as:

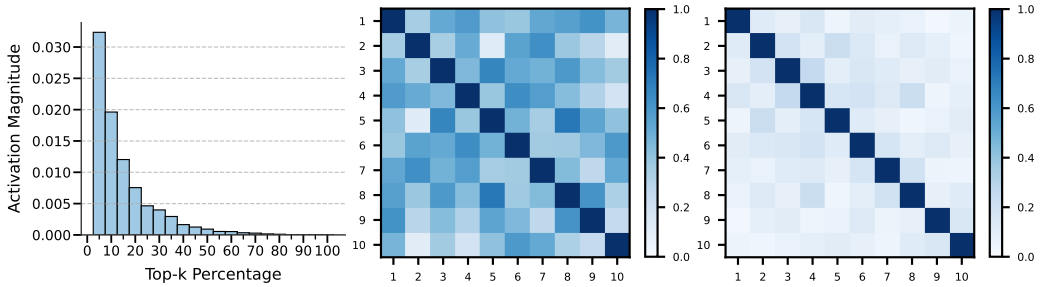
$$f_{\text{FlyLoRA}}(\boldsymbol{x}) = \boldsymbol{W}_0\boldsymbol{x} + \Delta\boldsymbol{W}\boldsymbol{x} = \boldsymbol{W}_0\boldsymbol{x} + \frac{\alpha}{r}\sum_{i=1}^{r} \mathbb{I}(i \in \mathcal{I}_{\text{top}k}) \cdot \boldsymbol{b}_i\boldsymbol{a}_i\boldsymbol{x}, \tag{11}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function.

3.2 Fixed Sparse Random Projection as Implicit Router

The core objective of the MoE router is to select k out of r experts that best approximate the effect of using all experts. However, as established in Sec. 2.3, it is computationally impractical to determine the selection based on $\|\boldsymbol{b}_i\boldsymbol{a}_i\boldsymbol{x}\|$. We therefore seek to perform the selection using only $\boldsymbol{a}_i\boldsymbol{x} \in \mathbb{R}$ as a surrogate. In FlyLoRA, since we fix $\boldsymbol{A}$ as a sparse random projection, computing $\boldsymbol{a}_i\boldsymbol{x}$ is as efficient as in standard $\text{LoRA}_{(r=k)}$. We theoretically prove that this projection preserves pairwise distances (see Theorem 3.1), and thus can serve as an effective implicit router.

https://doi.org/10.52202/085713-0349



(a) Activation distribution (b) LoRA-FA gradient correlation (c) FlyLoRA gradient correlation

Figure 3: **(a)** Activation value magnitude distribution across dimensions, showing the mean activation strength at different top-k selection percentages. **(b-c)** Gradient correlation matrices of (b) LoRA-FA$_{(r=32)}$ versus (c) FlyLoRA$_{(k=8)}$'s $\boldsymbol{B}$ matrices (10 randomly sampled columns). For a simplified illustration, we use the LoRA module of q_proj in the middle layer of Llama-3.1-8B on MMLU.

Theorem 3.1. *Given the matrix $\boldsymbol{A} \in \mathbb{R}^{r\times n}$ with each row having exactly p non-zero entries randomly sampled from $\mathcal{N}(0, \frac{1}{r^2})$, for any $\epsilon > 0$,*

$$\mathbb{P}\left((1-\epsilon)\|\boldsymbol{x}-\boldsymbol{y}\|^2 \le \frac{1}{r\sigma^2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{A}\boldsymbol{y}\|^2 \le (1+\epsilon)\|\boldsymbol{x}-\boldsymbol{y}\|^2\right) \ge 1 - e^{-(\epsilon^2-\epsilon^3)\frac{r}{4}} - e^{-\frac{(\epsilon^2-\epsilon^3)r}{2(\frac{3p}{n}+1)}},$$

for any input embeddings $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, where $\sigma^2 = \frac{p}{nr^2}$. A detailed proof is provided in Appendix A.1.

This theorem establishes a probabilistic guarantee for the approximate preservation of Euclidean distances under a sparse random projection. Notably, the concentration bound tightens with reduced sparsity (p/n) and larger rank (r), which aligns with intuitive expectation. We further present its robustness empirically via hyperparameter sensitivity analysis in Section 4.4.

Building on this property, we posit that FlyLoRA can self-select experts based on the values of $\boldsymbol{a}_i\boldsymbol{x}$, which aligns with the finding in [49] that "an expert is aware of its own capacity to effectively process a token, an awareness reflected in the scale of its internal activations." Unlike traditional trainable MoE routers that explicitly learn routing weights, FlyLoRA leverages the fixed geometry of $\boldsymbol{A}$ to perform implicit, activation-driven routing. This design not only eliminates the difficulty of learning routing parameters and avoids the separation between the router's decision-making and the experts' execution, but also reduces training instability by removing the stochasticity in router optimization. Because the projection $\boldsymbol{A}$ preserves pairwise distances, two semantically similar inputs $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are mapped to nearby low-dimensional representations $\boldsymbol{A}\boldsymbol{x}_i$ and $\boldsymbol{A}\boldsymbol{x}_j$, and therefore routed to similar experts, while dissimilar inputs are routed to different experts. This geometry-induced consistency helps mitigate expert representation homogenization in MoE models, enabling each expert to focus on specialized knowledge, reduce internal conflicts, and improve sample efficiency (as supported by previous studies, e.g., [10]). In this sense, FlyLoRA resembles a variant of the hash router [66], which also achieves lightweight and stable expert assignment through a fixed mapping. Consequently, the top-k operation naturally selects the most important experts according to the magnitudes of $\boldsymbol{a}_i\boldsymbol{x}$. In Figure 3(a), empirically around top-25% of dimensions account for more than 80% "energy". Thus, it typically does not cause a large performance drop.

3.3 Gradient Decoupling via Top-k Sparsity

In the FlyLoRA framework, only matrix $\boldsymbol{B}$ requires updating. We theoretically demonstrate that our rank-wise expert allocation strategy, induced by top-k selection, inherently reduces gradient covariance between distinct experts, thus mitigating intra-task interference. Our analysis begins with Assumption 3.2 describing the sparsity pattern of activations. For analytical convenience, we consider a simplified condition where the top-k operation randomly selects k out of r columns for activation.

Assumption 3.2 (Uniform Sparse Activation)**.** *During the top-k operation, each training sample activates exactly k columns of the parameter matrix $\boldsymbol{B} \in \mathbb{R}^{m\times r}$, with uniform selection probability $p = \frac{k}{r}$ per column.*

Based on Assumption 3.2, we derive Theorem 3.3 (proof in Appendix A.2).

Theorem 3.3 (Covariance Reduction Under top-k)**.** *Let $\tilde{\mathbf{\Sigma}}$ and $\mathbf{\Sigma}$ denote the gradient covariance matrices with and without top-k activation. When $r > k$, off-diagonal entries scale as:*

$$\mathbb{E}[\tilde{\mathbf{\Sigma}}_{(i,j)}] \approx \mathbb{E}[\mathbf{\Sigma}_{(i,j)}] \cdot \frac{k^2}{r^2}, \quad \forall i \neq j.$$

This $\mathcal{O}(k^2/r^2)$ reduction factor quantifies how top-k sparsity promotes parameter decoupling by suppressing interference terms. When $k = 1$ (only one rank is activated), the off-diagonal covariance almost vanishes, achieving full decoupling; when $k = r$ (all ranks are activated, degenerating to LoRA-FA [94]), it recovers the dense training regime. Theorem 3.3 is proved in Appendix A.2. To empirically validate this theoretical result, we visualize the gradient correlation patterns of LoRA-FA and FlyLoRA in Figure 3(b) and (c), where correlations are computed using 10 randomly selected gradient columns (see heatmap visualization). The observed sparsity pattern strongly supports our theoretical prediction of reduced off-diagonal covariance under top-k selection.

3.4 Inter-Task Orthogonality in Model Merging

Traditional LoRA model merging often suffers from parameter interference when combining task-specific components through weight averaging:

$$\boldsymbol{W}' = \boldsymbol{W}_0 + \sum_{i=1}^{t} w_i \boldsymbol{B}_i \boldsymbol{A}_i. \tag{12}$$

We analyze how FlyLoRA's inherent subspace orthogonality enables effective multi-task model merging. We derive Theorem 3.4 (proof in Appendix A.3).

Theorem 3.4 (Approximate Subspace Orthogonality)**.** *For independent random matrices $\boldsymbol{A}_i, \boldsymbol{A}_j \in \mathbb{R}^{r\times n}$ with sparse Gaussian entries ($\mathcal{N}(0, \frac{1}{r^2})$ for $p < n$ randomly selected entries per row), the following holds,*

1. ***Exact mean orthogonality**: $\mathbb{E}[\boldsymbol{A}_i \boldsymbol{A}_j^\top] = \mathbf{0}_{r\times r}$*
2. ***Polynomially decaying correlations**: $\mathbb{P}(\|\boldsymbol{A}_i \boldsymbol{A}_j^\top\|_2 \geq \epsilon r) \leq \frac{p^2}{nr^2\epsilon^2}$*

This theorem establishes that sparse random projections naturally induce nearly orthogonal subspaces. The residual correlation bound of order $\mathcal{O}(\frac{p^2}{nr^2})$ indicates that interference becomes negligible under practical parameter scales. This property directly leads to Corollary 3.5 (proof in Appendix A.3).

Corollary 3.5. *Let $\boldsymbol{A}_i, \boldsymbol{A}_j \in \mathbb{R}^{r\times n}$ be fixed sparse random projections after initialization. Then for any learned matrices $\boldsymbol{B}_i\boldsymbol{A}_i$ and $\boldsymbol{B}_j\boldsymbol{A}_j$, they satisfy the pairwise orthogonality property:*

$$\langle \boldsymbol{B}_i\boldsymbol{A}_i, \boldsymbol{B}_j\boldsymbol{A}_j \rangle_F \approx 0 \quad \textit{for} \quad i \neq j.$$

This orthogonal decomposition provides a key advantage for model merging: task-specific updates $\boldsymbol{B}_i\boldsymbol{A}_i$ occupy nearly orthogonal subspaces, thereby preventing destructive interference. The "Pairwise Orthogonality" property captures FlyLoRA's behavior during multi-task aggregation. According to geometric intuition that orthogonality facilitates model merging [29, 56, 75], and consistent with theoretical analyses in [39, 92], FlyLoRA's fixed sparse projection design aligns with this principle. Empirically (see Section 4.3), the random projection $\boldsymbol{A}$ enables FlyLoRA to preserve task-specific performance after merging, whereas the learnable $\boldsymbol{A}$ in conventional LoRA exhibits significantly higher interference. A similar analysis of LoRA merging was conducted in a concurrent study [93].

4 Experiments

4.1 Experimental Setup

Datasets and Backbones: We evaluate FlyLoRA's performance across four key domains: (1) *general knowledge understanding* using the MMLU [25] benchmark with auxiliary training datasets for fine-tuning and test set for evaluation, (2) *scientific question answering* using the ScienceQA [48] dataset for fine-tuning and evaluation, (3) *mathematical reasoning* on GSM8K [12] problems for

https://doi.org/10.52202/085713-0349

Table 1: **Performance Comparison of LoRA Variants in Single-task Evaluation.** We evaluate various methods across four benchmarks: MMLU, ScienceQA, GSM8K (accuracy), and HumanEval (Pass@k), with all metrics reported in percentage (%). Param(%) indicates the percentage of activated trainable parameters relative to Full FT. The best results are highlighted in **bold**.

Model	Method	Param(%)	MMLU	ScienceQA	GSM8K	HumanEval		
						Pass@1	Pass@5	Pass@10
Llama-3.1-8B	LoRA$_{(r=8)}$	0.26	36.53±0.40	91.39±0.55	55.34±0.24	29.13±0.56	52.28±1.24	61.67±0.61
	LoRA$_{(r=32)}$	1.03	38.93±1.04	94.01±0.17	56.25±0.29	30.37±1.06	54.37±0.39	64.02±0.94
	Split-LoRA$_{(4\times8)}$	0.33	38.44±0.69	92.41±0.54	55.65±0.47	31.28±1.52	54.16±1.12	63.94±0.89
	FlyLoRA$_{(k=8)}$	**0.13**	**40.88**±1.61	**94.15**±0.36	**58.76**±0.74	**36.88**±1.91	**62.40**±1.82	**73.34**±1.24
Qwen-2.5-7B	LoRA$_{(r=8)}$	0.26	49.84±0.56	92.84±0.13	77.01±0.32	47.20±1.54	78.89±0.36	85.94±0.64
	LoRA$_{(r=32)}$	1.05	52.07±0.31	95.01±0.21	79.23±0.22	52.87±1.79	81.67±1.14	87.80±0.72
	Split-LoRA$_{(4\times8)}$	0.33	50.68±1.06	93.08±0.41	77.12±0.76	48.65±1.18	79.30±0.91	86.05±0.44
	FlyLoRA$_{(k=8)}$	**0.13**	**53.68**±0.47	**95.55**±0.18	**80.82**±0.56	**54.34**±2.13	**82.85**±0.52	**89.63**±0.55

fine-tuning and evaluation, and (4) *code generation* assessed via CodeAlpaca-20k [7] for training and HumanEval [9] for evaluation. Except for HumanEval, which is evaluated via pass@k metrics, others are evaluated via accuracy. All benchmarks are evaluated in a zero-shot manner. We examine our framework in both single-task configurations, training with these four datasets individually, and multi-task settings, where the LoRA components trained in single-task setup for each dataset are merged together in a training-free manner. Most experiments are conducted using Llama-3.1-8B [23] and Qwen-2.5-7B [86], respectively. See Appendix C for implementation details.

Baselines: For the single-task setup, we compare FlyLoRA against: (1) vanilla LoRA with identical activation ranks (LoRA$_{(r=8)}$) and total ranks (LoRA$_{(r=32)}$), and (2) representative MoE-based LoRA variants Split-LoRA$_{(4\times8)}$ (abbreviated for 4 expert×8 rank). For the multi-task setup, we benchmark FlyLoRA against them using weight averaging fusion, and several advanced merging techniques. Across all datasets, FlyLoRA$_{(k=8)}$ uses total rank $r = 32$ but activates only $k = 8$ ranks after the top-k operation between $\boldsymbol{A}$ and $\boldsymbol{B}$, with the fixed sparse random $\boldsymbol{A}$'s sparsity ratio ρ set to $8/32$.

4.2 Single Task Performance

The single-task results are presented in Table 1. Despite operating under a lower computational budget, FlyLoRA$_{(k=8)}$ outperforms LoRA variants with the same rank (LoRA$_{(r=8)}$) across all datasets. This improvement can be attributed to its broader parameter space. Notably, FlyLoRA$_{(k=8)}$ also achieves slightly better performance than LoRA variants with the same total rank (LoRA$_{(r=32)}$), suggesting that a significant portion of LoRA's parameters are redundant and may introduce interference. Additionally, FlyLoRA$_{(k=8)}$ demonstrates superior performance over Split-LoRA$_{(4\times8)}$, highlighting the benefits of its finer expert allocation strategy within the MoE framework, which enables **intra-task decoupling**. The reduction in activated trainable parameters compared to these baselines shows FlyLoRA's efficiency. Extended results with larger models and further baselines are in Appendix B.

4.3 Multi-task Performance

For simplicity, we first employ the widely used weight averaging technique for model merging. Specifically, this corresponds to setting $w_i = \frac{1}{t}$ in Eq. 12, yielding the merged weights $\boldsymbol{W}' = \boldsymbol{W}_0 + \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{B}_i\boldsymbol{A}_i$. The multi-task results are presented in Table 2, where LoRA components from different domains are merged. Compared to both LoRA variants ($r = 8$ and $r = 32$) and Split-LoRA$_{(4\times8)}$, FlyLoRA achieves higher accuracy both before and after merging, with significantly smaller performance degradation. This robustness stems from its **inter-task decoupling** enabled by approximate orthogonal random projection, as theoretically analyzed in Section 3.4. Additional results with advanced fusion techniques are provided in Appendix B.

4.4 Ablation Study and Hyperparameter Sensitivity Analysis

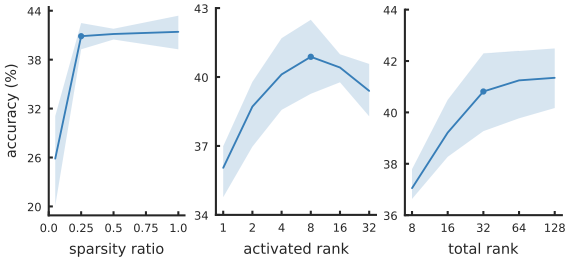
We conduct an ablation study to analyze key properties of FlyLoRA by evaluating two critical modifications: (1) removing load-balancing strategies and (2) replacing the frozen matrix $\boldsymbol{A}$ with an

Table 2: **Multi-task Performance Comparison Before and After Parameter Merging.** We evaluate LoRA variants across MMLU, ScienceQA, GSM8K (accuracy), and HumanEval (Pass@k) benchmarks. The table shows performance before merging, after merging, and the relative performance drop (Δ%). The best results are highlighted in **bold**.

Model	Method	Merge Status	MMLU	ScienceQA	GSM8K	HumanEval		
						Pass@1	Pass@5	Pass@10
Llama-3.1-8B	LoRA$_{(r=8)}$	Before	36.53±0.40	91.39±0.55	55.34±0.24	29.13±0.56	52.28±1.24	61.67±0.61
		After	30.05±0.82	31.05±2.38	25.19±2.36	16.09±3.15	45.38±1.62	56.49±2.13
		Δ (%)	-6.48	-60.34	-30.15	-13.04	-6.90	-5.18
	LoRA$_{(r=32)}$	Before	38.93±1.04	94.01±0.17	56.25±0.29	30.37±1.06	54.37±0.39	64.02±0.94
		After	34.02±1.32	34.35±1.42	24.77±0.94	18.94±1.48	46.39±1.74	59.27±1.19
		Δ (%)	-4.91	-59.66	-31.48	-11.43	-7.98	-4.75
	Split-LoRA$_{(4\times8)}$	Before	38.44±0.69	92.41±0.54	55.65±0.47	31.28±1.52	54.16±1.12	63.94±0.89
		After	33.58±1.16	37.67±1.06	27.35±1.10	21.36±1.08	46.01±0.87	59.52±0.95
		Δ (%)	-4.86	-54.74	-28.30	-9.92	-8.15	-4.42
	FlyLoRA$_{(k=8)}$	Before	40.88±1.61	94.15±0.36	58.76±0.74	36.88±1.91	62.40±1.82	73.34±1.24
		After	38.86±1.46	51.10±0.71	36.95±1.37	32.61±1.45	56.59±2.66	69.76±0.75
		Δ (%)	**-2.02**	**-43.05**	**-21.81**	**-4.27**	**-5.81**	**-3.58**
Qwen-2.5-7B	LoRA$_{(r=8)}$	Before	49.84±0.56	92.84±0.13	77.01±0.32	47.20±1.54	78.89±0.36	85.94±0.64
		After	44.62±1.23	60.07±2.18	81.56±1.48	22.09±2.68	68.38±0.86	80.49±0.77
		Δ (%)	-5.22	-32.77	+4.55	-25.21	-10.51	-5.45
	LoRA$_{(r=32)}$	Before	52.07±0.31	95.01±0.21	79.23±0.22	52.87±1.79	81.67±1.14	87.80±0.72
		After	32.86±1.06	55.58±0.76	83.91±0.70	23.84±2.13	66.23±1.65	79.27±1.05
		Δ (%)	-19.21	-39.43	+4.68	-29.03	-15.44	-8.53
	Split-LoRA$_{(4\times8)}$	Before	50.68±1.06	93.08±0.41	77.12±0.76	48.65±1.18	79.30±0.91	86.05±0.44
		After	41.83±1.92	59.37±0.59	81.70±0.52	22.98±1.35	67.02±0.59	81.01±1.34
		Δ (%)	-8.85	-33.71	+4.58	-25.67	-12.28	-5.04
	FlyLoRA$_{(k=8)}$	Before	53.68±0.47	95.55±0.18	80.82±0.56	54.34±2.13	82.85±0.52	89.63±0.55
		After	60.23±0.95	71.78±0.41	85.62±0.43	33.11±1.61	75.28±1.72	87.15±1.26
		Δ (%)	**+6.55**	**-23.77**	**+4.80**	**-21.23**	**-7.57**	**-2.48**

Table 3: Ablation study of FlyLoRA variants analyzing: **(a)** Load balancing in single-task (ST) setting, **(b)** Matrix $\boldsymbol{A}$ freezing in both ST and multi-task merging (MT) settings, where LB=Load Balancing, Frz=$\boldsymbol{A}$ Frozen, Trn=$\boldsymbol{A}$ Trainable.

Load Balancing		**Frozen** A	
Variant	Acc (%)	Variant	Acc (%)
w/ LB	**40.88±1.61**	ST+Frz	**40.88±1.61**
w/o LB	37.56±2.87	ST+Trn	40.64±1.35
		MT+Frz	**38.86±1.46**
		MT+Trn	34.43±2.24



(a) (b) (c)

Figure 4: Accuracy comparison for: **(a)** Sparsity ratio in $\boldsymbol{A}$, **(b)** Activated rank (with fixed total rank $r = 32$), **(c)** Total rank (with fixed activated rank $k = 8$).

updatable version, as shown in Table 3 (MMLU and Llama-3.1-8B). Our results demonstrate that load-balancing significantly improves MoE's training stability and boosts accuracy by 3.32%. For matrix $\boldsymbol{A}$, we observe minimal performance differences between frozen and updatable versions in single-task settings. However, in multi-task merging scenarios, using an updatable matrix leads to a 4.43% performance degradation, as the updatable $\boldsymbol{A}$ does not satisfy the approximately orthogonal property. More ablation studies are provided in Appendix B.

Further sensitivity analysis on single-task performance (Figure 4) reveals three key insights. First, model accuracy increases monotonically with the sparsity ratio of $\boldsymbol{A}$ before saturating, exhibiting only marginal degradation unless the sparsity ratio becomes extremely small. Second, under a fixed total rank budget, performance peaks at intermediate activation ranks: insufficient rank fails to capture task-specific features, while excessive rank induces parameter interference. Third, increasing the total rank while holding the activation rank constant consistently yields performance gains.

https://doi.org/10.52202/085713-0349

5 Discussion

5.1 Interference in Model Merging

In scenarios requiring training, gradient orthogonalization techniques are commonly employed to reduce task interference, as seen in multi-task learning [88] and continual learning [8, 91]. In our training-free model merging setting, all components are derived from the same base model through domain-specific SFT. Task interference can be quantified by measuring the orthogonality of parameter updates (relative to the base model) across different tasks [29]. For our LoRA components merging, these parameter updates correspond to $\boldsymbol{B}_i\boldsymbol{A}_i$. We formally prove the near-orthogonality of FlyLoRA in Appendix A.3, which inherently reduces inter-task correlations.

5.2 FlyLoRA's Connection to Other Orthogonality-Based Designs in PEFT

Representative orthogonality-based PEFT methods like OFT [46, 59] and LoReFT [83] both operate on single tasks, and their orthogonal matrix $\boldsymbol{R}$ multiplies the pre-trained weight matrix $\boldsymbol{W}_0$, differing from LoRA variants (including FlyLoRA) that add $\Delta\boldsymbol{W}$ to $\boldsymbol{W}_0$. The multiplication scheme rotates the entire weight parameter space, and [46, 59] demonstrate this better adjusts semantic information compared to changing magnitude, explaining its success. In contrast, the additive scheme lacks this property since $\boldsymbol{W}_0$ cannot be rotated. In single-task settings, removing the MoE part with only random $\boldsymbol{A}$ reduces FlyLoRA to LoRA-FA [94] or Asymmetry LoRA [97]. These variants can save resources but cannot improve performance. Thus, although all methods use orthogonality, FlyLoRA succeeds differently. We think the orthogonality design in LoRA excels in *multi-task* scenarios, such as model merging (this work and LoRI [93]) and continual learning (O-LoRA [80]), because it decouples parameter interference across multiple downstream tasks when fine-tuning from the base model.

6 Related Work

Low-Rank Adaptation LoRA [27] is a widely used PEFT strategy for fine-tuning LLMs. To enhance its expressive power, several improvements [45, 95] have been proposed. Recently, to address parameter interference in settings like multi-task and continual learning, several MoE-based LoRA variants [17, 20, 21, 37, 55, 76, 82, 89] have proven effective by forcing each expert to specialize in specific areas. Several works [73, 75, 93] also aim to reduce interference during LoRA merging. In this article, we further develop the MoE-based LoRA structure. To improve LoRA's efficiency, LoRA-FA [94] and AsymmetryLoRA [97] show that fixing the down-projection matrix $\boldsymbol{A}$ saves memory for input activations without performance degradation. In our work, we reconsider freezing $\boldsymbol{A}$ from a new perspective by showing that its orthogonality and distance-preserving properties can be utilized to design an improved intra-/inter-task decoupling mechanism.

Fly Olfactory Circuit The fly olfactory circuit [6, 13, 38, 42, 72] serves as an exemplary model in bio-inspired AI due to its structural simplicity and functional completeness. Its core mechanism—random projection followed by sparse selection—effectively transforms high-dimensional inputs into separable representations. This biological principle has inspired algorithmic innovations across multiple AI domains, including locality-sensitive hashing [14, 67, 69], word embedding [41], federated learning [65], and continual learning [100, 101]. The circuit's enduring relevance highlights the value of cross-disciplinary inspiration in advancing computational methods.

7 Conclusion

In summary, this work provides a comprehensive revisit of the MoE-based structure for LoRA and analyzes its drawbacks regarding parameter interference and efficiency. Inspired by the fly olfactory circuit, we introduce FlyLoRA, a novel MoE-based LoRA variant that employs rank-wise expert activation in matrix $\boldsymbol{B}$ and a fixed sparse random projection for matrix $\boldsymbol{A}$ as an implicit router. Through the theoretical properties of these components, FlyLoRA achieves both intra-task and inter-task decoupling, significantly improving decorrelation in single-domain instruction tuning and LoRA component fusion in multi-task settings. Additionally, the implicit routing strategy and inherent sparsity ensure computational efficiency.

8 Acknowledgments

We thank Cheems Wang for his valuable suggestions on the manuscript. We also thank the anonymous reviewers for their positive feedback and constructive comments. This work was supported by the National Key R&D Program of China under Grant 2018AAA0102801.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

[3] Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] Nina Dekoninck Bruhin and Bryn Davies. Bioinspired random projections for robust, sparse classification. *SIAM Journal on Imaging Sciences*, 15(4):1833–1850, 2022.

[6] Sophie JC Caron, Vanessa Ruta, Larry F Abbott, and Richard Axel. Random convergence of olfactory inputs in the drosophila mushroom body. *Nature*, 497(7447):113–117, 2013.

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

[8] Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr. Continual learning in low-rank orthogonal subspaces. *Advances in Neural Information Processing Systems*, 33: 9900–9911, 2020.

[9] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[10] Mohammed Nowaz Rabbani Chowdhury, Shuai Zhang, Meng Wang, Sijia Liu, and Pin-Yu Chen. Patch-level routing in mixture-of-experts is provably sample-efficient for convolutional neural networks. In *International Conference on Machine Learning*, pages 6074–6114. PMLR, 2023.

[11] Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *arXiv preprint arXiv:2302.07027*, 2023.

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

 https://doi.org/10.52202/085713-0349

[13] Sanjoy Dasgupta, Charles F Stevens, and Saket Navlakha. A neural algorithm for a fundamental computing problem. *Science*, 358(6364):793–796, 2017.

[14] Sanjoy Dasgupta, Charles F. Stevens, and Saket Navlakha. A neural algorithm for a fundamental computing problem. *Science*, 358(6364):793–796, 2017. doi: 10.1126/science.aam9868.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186, 2019.

[16] Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. *arXiv preprint arXiv:2311.11696*, 2023.

[17] Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin. *arXiv preprint arXiv:2312.09979*, 2023.

[18] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[19] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[20] Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. Mixture-of-loras: An efficient multitask tuning for large language models. *arXiv preprint arXiv:2403.03432*, 2024.

[21] Chongyang Gao, Kezhen Chen, Jinmeng Rao, Baochen Sun, Ruibo Liu, Daiyi Peng, Yawen Zhang, Xiaoyuan Guo, Jie Yang, and VS Subrahmanian. Higher layers need more lora experts. *arXiv preprint arXiv:2402.08562*, 2024.

[22] Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning llms on new knowledge encourage hallucinations? *arXiv preprint arXiv:2405.05904*, 2024.

[23] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[24] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[25] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[26] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[27] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1 (2):3, 2022.

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

[29] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[30] 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.

[31] Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, et al. Mora: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*, 2024.

[32] William B Johnson, Joram Lindenstrauss, et al. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[33] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMlR, 2019.

[34] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[35] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[36] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

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

[38] Haiyang Li, Liao Yu, Qiang Yu, and Yunliang Zang. Seemingly redundant modules enhance robust odor learning in fruit flies. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[39] Hongkang Li, Yihua Zhang, Shuai Zhang, Meng Wang, Sijia Liu, and Pin-Yu Chen. When is task vector provably effective for model editing? a generalization analysis of nonlinear transformers. *arXiv preprint arXiv:2504.10957*, 2025.

[40] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[41] Yuchen Liang, Chaitanya K Ryali, Benjamin Hoover, Leopold Grinberg, Saket Navlakha, Mohammed J Zaki, and Dmitry Krotov. Can a fruit fly learn word embeddings? *arXiv preprint arXiv:2101.06887*, 2021.

[42] Andrew C Lin, Alexei M Bygrave, Alix De Calignon, Tzumin Lee, and Gero Miesenböck. Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination. *Nature neuroscience*, 17(4):559–568, 2014.

[43] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

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

[45] 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*, 2024.

[46] Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, et al. Parameter-efficient orthogonal finetuning via butterfly factorization. *arXiv preprint arXiv:2311.06243*, 2023.

[47] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *AI Open*, 5:208–215, 2024.

 https://doi.org/10.52202/085713-0349

[48] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[49] Ang Lv, Ruobing Xie, Yining Qian, Songhao Wu, Xingwu Sun, Zhanhui Kang, Di Wang, and Rui Yan. Autonomy-of-experts models. *arXiv preprint arXiv:2501.13074*, 2025.

[50] Yixiu Mao, Hongchang Zhang, Chen Chen, Yi Xu, and Xiangyang Ji. Supported trust region optimization for offline reinforcement learning. In *International Conference on Machine Learning*, pages 23829–23851. PMLR, 2023.

[51] Yixiu Mao, Hongchang Zhang, Chen Chen, Yi Xu, and Xiangyang Ji. Supported value regularization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:40587–40609, 2023.

[52] Yixiu Mao, Qi Wang, Chen Chen, Yun Qu, and Xiangyang Ji. Offline reinforcement learning with ood state correction and ood action suppression. *Advances in Neural Information Processing Systems*, 37:93568–93601, 2024.

[53] Yixiu Mao, Qi Wang, Yun Qu, Yuhang Jiang, and Xiangyang Ji. Doubly mild generalization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37: 51436–51473, 2024.

[54] Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

[55] Mohammed Muqeeth, Haokun Liu, and Colin Raffel. Soft merging of experts with adaptive routing. *arXiv preprint arXiv:2306.03745*, 2023.

[56] Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36:66727–66754, 2023.

[57] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[58] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *Advances in neural information processing systems*, 30, 2017.

[59] Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. *Advances in Neural Information Processing Systems*, 36:79320–79362, 2023.

[60] Yun Qu, Boyuan Wang, Jianzhun Shao, Yuhang Jiang, Chen Chen, Zhenbin Ye, Liu Linc, Yang Feng, Lin Lai, Hongyang Qin, et al. Hokoff: Real game dataset from honor of kings and its offline reinforcement learning benchmarks. *Advances in Neural Information Processing Systems*, 36:22166–22190, 2023.

[61] Yun Qu, Qi Wang, Yixiu Mao, Vincent Tao Hu, Björn Ommer, and Xiangyang Ji. Can prompt difficulty be online predicted for accelerating rl finetuning of reasoning models? *arXiv preprint arXiv:2507.04632*, 2025.

[62] Yun Qu, Qi Cheems Wang, Yixiu Mao, Yiqin Lv, and Xiangyang Ji. Fast and robust: Task sampling with posterior and diversity synergies for adaptive decision-makers in randomized environments. *arXiv preprint arXiv:2504.19139*, 2025.

[63] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[64] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[65] Parikshit Ram and Kaushik Sinha. Federated nearest neighbor classification with a colony of fruit-flies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8036–8044, 2022.

[66] Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *advances in neural information processing systems*, 34:17555–17566, 2021.

[67] Chaitanya Ryali, John Hopfield, Leopold Grinberg, and Dmitry Krotov. Bio-inspired hashing for unsupervised similarity search. In *International conference on machine learning*, pages 8295–8306. PMLR, 2020.

[68] Jianzhun Shao, Yun Qu, Chen Chen, Hongchang Zhang, and Xiangyang Ji. Counterfactual conservative q learning for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36:77290–77312, 2023.

[69] Jaiyam Sharma and Saket Navlakha. Improving similarity search with high-dimensional locality-sensitive hashing. *arXiv preprint arXiv:1812.01844*, 2018.

[70] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[71] Jiayi Shen, Qi Wang, Zehao Xiao, Nanne Van Noord, and Marcel Worring. Go4align: Group optimization for multi-task alignment. *Advances in Neural Information Processing Systems*, 37:111382–111405, 2024.

[72] Charles F Stevens. What the fly's nose tells the fly's brain. *Proceedings of the National Academy of Sciences*, 112(30):9460–9465, 2015.

[73] George Stoica, Pratik Ramesh, Boglarka Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. *arXiv preprint arXiv:2410.19735*, 2024.

[74] Peng Sun, Yao Zhu, Yunjian Zhang, Xiu Yan, Zizhe Wang, and Xiangyang Ji. Unleashing the potential of large language models through spectral modulation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3892–3911, 2024.

[75] Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. Parameter efficient multi-task model fusion with partial linearization. *arXiv preprint arXiv:2310.04742*, 2023.

[76] Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Cheng-Zhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37:9565–9584, 2024.

[77] Jiachen Tianhao Wang, Tong Wu, Dawn Song, Prateek Mittal, and Ruoxi Jia. Greats: Online selection of high-quality data for llm training in every iteration. *Advances in Neural Information Processing Systems*, 37:131197–131223, 2024.

[78] Qi Wang and Herke Van Hoof. Learning expressive meta-representations with mixture of expert neural processes. *Advances in neural information processing systems*, 35:26242–26255, 2022.

[79] Qi Cheems Wang, Zehao Xiao, Yixiu Mao, Yun Qu, Jiayi Shen, Yiqin Lv, and Xiangyang Ji. Model predictive task sampling for efficient and robust adaptation. *arXiv preprint arXiv:2501.11039*, 2025.

[80] Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. *arXiv preprint arXiv:2310.14152*, 2023.

 https://doi.org/10.52202/085713-0349

[81] Ziqi Wang, Chang Che, Qi Wang, Yangyang Li, Zenglin Shi, and Meng Wang. Separable mixture of low-rank adaptation for continual visual instruction tuning. *arXiv preprint arXiv:2411.13949*, 2024.

[82] Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts. *arXiv preprint arXiv:2404.13628*, 2024.

[83] Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Reft: Representation finetuning for language models. *Advances in Neural Information Processing Systems*, 37:63908–63962, 2024.

[84] Zehao Xiao, Shilin Yan, Jack Hong, Jiayin Cai, Xiaolong Jiang, Yao Hu, Jiayi Shen, Qi Wang, and Cees GM Snoek. Dynaprompt: Dynamic test-time prompt tuning. *arXiv preprint arXiv:2501.16404*, 2025.

[85] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.

[86] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[87] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

[88] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33:5824–5836, 2020.

[89] Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*, 2023.

[90] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

[91] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

[92] Siqi Zeng, Yifei He, Weiqiu You, Yifan Hao, Yao-Hung Hubert Tsai, Makoto Yamada, and Han Zhao. Efficient model editing with task vector bases: A theoretical framework and scalable approach. *arXiv preprint arXiv:2502.01015*, 2025.

[93] Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. Lori: Reducing cross-task interference in multi-task low-rank adaptation. *arXiv preprint arXiv:2504.07448*, 2025.

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

[95] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

[96] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

[97] Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Saez De Ocariz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in low-rank adapters of foundation models. *arXiv preprint arXiv:2402.16842*, 2024.

[98] Yao Zhu, Yunjian Zhang, Zizhe Wang, Xiu Yan, Peng Sun, and Xiangyang Ji. Patchwise cooperative game-based interpretability method for large vision-language models. *Transactions of the Association for Computational Linguistics*, 13:744–759, 2025.

[99] Heming Zou, Yixiu Mao, Yun Qu, Qi Wang, and Xiangyang Ji. Utility-diversity aware online batch selection for llm supervised fine-tuning. *arXiv preprint arXiv:2510.16882*, 2025.

[100] Heming Zou, Yunliang Zang, and Xiangyang Ji. Structural features of the fly olfactory circuit mitigate the stability-plasticity dilemma in continual learning. *arXiv preprint arXiv:2502.01427*, 2025.

[101] Heming Zou, Yunliang Zang, Wutong Xu, and Xiangyang Ji. Fly-cl: A fly-inspired framework for enhancing efficient decorrelation and reduced training time in pre-trained model-based continual representation learning. *arXiv preprint arXiv:2510.16877*, 2025.

 https://doi.org/10.52202/085713-0349

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We have summarized the contribution and scope in the abstract, introduction, and conclusion. Especially, we list our contribution in the last paragraph of the introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: We have discussed the possible limitations in Appendix D.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: We have listed theorems in Section 3 of the main text, with complete proofs presented in Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We have listed the training details in Appendix C, from which people can reproduce the main experimental results based on it. We also provide code in the supplementary materials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

https://doi.org/10.52202/085713-0349

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

Answer: [Yes]

Justification: We provide code in the supplementary materials. The open-source datasets and models are listed in Appendix C.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We discuss the experimental setup and training details in Section 4 and Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report the error bar over three random seeds.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We summarize our computational resources in Appendix C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have read the Code of Ethics and make sure to preserve anonymity.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We have discuss potential societal impacts in Appendix E.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

https://doi.org/10.52202/085713-0349

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: Our work poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have used open-source code and datasets, and have appropriately cited them.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: Our paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

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

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

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

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

https://doi.org/10.52202/085713-0349

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

A Theoretical Analysis

A.1 Distance Preserving Property for Sparse Random Projection

In this section, we prove that the fixed sparse random projection matrix $\boldsymbol{A}$ satisfies the distance-preserving property. This demonstrates that a fixed projection $\boldsymbol{A}$ can function like a hash router without an explicit router. Our result extends the well-known Johnson-Lindenstrauss Lemma [32]. First, [5, Corollary 3.5] provides the following bound:

Theorem A.1. *Let $\boldsymbol{A} \in \mathbb{R}^{r\times n}$ be a random matrix whose entries $\boldsymbol{A}_{ij}$ are sampled independently and randomly from a distribution that is symmetric around the origin with $\mathbb{E}(\boldsymbol{A}_{ij}^2) = \sigma^2 > 0$.*

1. *Suppose $C = \mathbb{E}(\boldsymbol{A}_{ij}^4) < \infty$. Then, for any $\epsilon > 0$,*

$$\mathbb{P}\left(\|\frac{1}{\sqrt{r}}\boldsymbol{A}\boldsymbol{x}\|^2 \le \sigma^2(1-\epsilon)\|\boldsymbol{x}\|^2\right) \le \exp\left(-\frac{(\epsilon^2-\epsilon^3)r}{2(\frac{1}{\sigma^4}C+1)}\right), \quad \textit{for all } \boldsymbol{x}\in\mathbb{R}^n.$$

2. *Suppose $\exists L > 0$ such that for any integer $k > 0$, $\mathbb{E}(\boldsymbol{A}_{ij}^{2k}) \le \sigma^{2k}\frac{(2k)!}{2^k k!}L^{2k}$. Then, for any $\epsilon > 0$,*

$$\mathbb{P}\left(\|\frac{1}{\sqrt{r}}\boldsymbol{A}\boldsymbol{x}\|^2 \ge \sigma^2(1+\epsilon)L^2\|\boldsymbol{x}\|^2\right) \le \exp\left(-(\epsilon^2-\epsilon^3)\frac{r}{4}\right), \quad \textit{for all } \boldsymbol{x}\in\mathbb{R}^n.$$

According to the definition of $\boldsymbol{A}$ mentioned in Section 3.1, the second moment of $\boldsymbol{A}_{ij}$ satisfies $\mathbb{E}(\boldsymbol{A}_{ij}^2) = \frac{p}{nr^2} > 0$. So, the $2k$-th moment of $\boldsymbol{A}_{ij}$ is given by $\mathbb{E}[\boldsymbol{A}_{ij}^{2k}] = \frac{p}{n}\cdot\mathbb{E}[\boldsymbol{X}_{ij}^{2k}] + \left(1-\frac{p}{n}\right)\cdot 0 = \frac{p}{n}\cdot\mathbb{E}[\boldsymbol{X}_{ij}^{2k}]$, where $\boldsymbol{X}_{ij}\sim\mathcal{N}(0,\frac{1}{r^2})$. From the property of Gaussian distribution, we derive $\mathbb{E}[\boldsymbol{X}_{ij}^{2k}] = (2k-1)!!\cdot\left(\frac{1}{r^2}\right)^{2k} = \frac{(2k)!}{2^k k!}\left(\frac{1}{r^2}\right)^{2k}$, where !! denotes the double factorial. This leads to $\mathbb{E}[\boldsymbol{A}_{ij}^4] = \frac{3p}{nr^4}$, which is clearly finite. To satisfy the inequality $\mathbb{E}[\boldsymbol{A}_{ij}^{2k}] = \frac{(2k)!}{2^k k!}\sigma^{2k} \le \sigma^{2k}\frac{(2k)!}{2^k k!}L^{2k}$, we require $L \ge \log_{2k}\frac{p}{r}$. Due to monotonicity, simply choosing $L = \log_2\frac{p}{r}$ always satisfies the condition specified in Theorem A.1. Combining both bounds from Theorem A.1, we summarize the final result in Theorem A.2.

$$\begin{aligned}&\mathbb{P}\left((1-\epsilon)\|\boldsymbol{x}-\boldsymbol{y}\|^2 \le \frac{1}{r\sigma^2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{A}\boldsymbol{y}\|^2 \le (1+\epsilon)\|\boldsymbol{x}-\boldsymbol{y}\|^2\right) = \\ &1-\mathbb{P}\left(\frac{1}{r}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{A}\boldsymbol{y}\|^2 < (1-\epsilon)\sigma^2\|\boldsymbol{x}-\boldsymbol{y}\|^2\right) - \\ &\mathbb{P}\left(\frac{1}{r}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{A}\boldsymbol{y}\|^2 > (1+\epsilon)\sigma^2\|\boldsymbol{x}-\boldsymbol{y}\|^2\right),\end{aligned} \tag{13}$$

Theorem A.2. *Given the matrix $\boldsymbol{A} \in \mathbb{R}^{r\times n}$ with each entry i.i.d. from $\mathcal{N}(0,\frac{1}{r^2})$, and set to 0 otherwise, for any $\epsilon > 0$,*

$$\mathbb{P}\left((1-\epsilon)\|\boldsymbol{x}-\boldsymbol{y}\|^2 \le \frac{1}{r\sigma^2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{A}\boldsymbol{y}\|^2 \le (1+\epsilon)\|\boldsymbol{x}-\boldsymbol{y}\|^2\right) \ge 1 - e^{-(\epsilon^2-\epsilon^3)\frac{r}{4}} - e^{-\frac{(\epsilon^2-\epsilon^3)r}{2(\frac{3p}{n}+1)}},$$

for any input embeddings $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^n$, where $\sigma^2 = \frac{p}{nr^2}$.

Since this construction does not differ significantly from our desired construction for $\boldsymbol{A}$, which includes an additional constraint on non-zero entries per row and is more consistent with biological observations in the fly olfactory circuit, we use this more analytically tractable form as a surrogate to study the distance-preserving property. In practice, these two construction methods show negligible performance differences.

A.2 Top-k Activation Promotes Rank-Wise Decoupling

Let $\boldsymbol{\Lambda} \in \mathrm{diag}(\lambda_1,\ldots,\lambda_r) \in \mathbb{R}^{r\times r}$ denote a binary mask where $\lambda_i = 1$ if column i is activated by top-k. The relation of the masked gradient between top-k version ($\frac{\partial\mathcal{L}}{\partial\tilde{\boldsymbol{B}}}$) and the dense version ($\frac{\partial\mathcal{L}}{\partial\boldsymbol{B}}$) is:

$$\frac{\partial\mathcal{L}}{\partial\tilde{\boldsymbol{B}}} = \frac{\partial\mathcal{L}}{\partial\boldsymbol{B}}\boldsymbol{\Lambda} \in \mathbb{R}^{m\times r}. \tag{14}$$

https://doi.org/10.52202/085713-0349

Let $\tilde{\boldsymbol{g}}_i$ and $\boldsymbol{g}_i \in \mathbb{R}^m$ denote the gradient vectors for the i-th column of $\frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{B}}}$ and $\frac{\partial \mathcal{L}}{\partial \boldsymbol{B}}$, respectively. Their corresponding cross-column covariance are:

$$\boldsymbol{\Sigma}_{(i,j)} = \mathbb{E}\left[\boldsymbol{g}_i^\top \boldsymbol{g}_j\right] - \mathbb{E}\left[\boldsymbol{g}_i\right]^\top \mathbb{E}\left[\boldsymbol{g}_j\right], \tag{15}$$

$$\tilde{\boldsymbol{\Sigma}}_{(i,j)} = \mathbb{E}\left[\tilde{\boldsymbol{g}}_i^\top \tilde{\boldsymbol{g}}_j\right] - \mathbb{E}\left[\tilde{\boldsymbol{g}}_i\right]^\top \mathbb{E}\left[\tilde{\boldsymbol{g}}_j\right], \tag{16}$$

We can assume $\mathbb{E}[\tilde{\boldsymbol{g}}_i] = \mathbb{E}[\tilde{\boldsymbol{g}}_j] = \mathbb{E}[\boldsymbol{g}_i] = \mathbb{E}[\boldsymbol{g}_j] = \mathbf{0}_m$, these simplify to:

$$\boldsymbol{\Sigma}_{(i,j)} = \mathbb{E}\left[\boldsymbol{g}_i^\top \boldsymbol{g}_j\right], \tag{17}$$

$$\tilde{\boldsymbol{\Sigma}}_{(i,j)} = \mathbb{E}\left[\tilde{\boldsymbol{g}}_i^\top \tilde{\boldsymbol{g}}_j\right]. \tag{18}$$

Thus, the expected covariance depends only on the expected inner product of gradient vectors. From Assumption 3.2, the difference between $\tilde{\boldsymbol{\Sigma}}_{(i,j)}$, $\boldsymbol{\Sigma}_{(i,j)}$ is factored by the co-activation probability of columns i and j:

$$\mathbb{P}(\lambda_i = 1 \cap \lambda_j = 1) = \frac{\binom{r-2}{k-2}}{\binom{r}{k}} = \frac{k(k-1)}{r(r-1)} \approx \frac{k^2}{r^2}. \tag{19}$$

This leads to the following theorem:

Theorem A.3 (Covariance Reduction Under top-k)**.** *Let $\tilde{\boldsymbol{\Sigma}}$ and $\boldsymbol{\Sigma}$ denote the gradient covariance matrices with and without top-k activation. When $r > k$, the off-diagonal entries scale as:*

$$\mathbb{E}[\tilde{\boldsymbol{\Sigma}}_{(i,j)}] \approx \mathbb{E}[\boldsymbol{\Sigma}_{(i,j)}] \cdot \frac{k^2}{r^2}, \quad \forall i \neq j. \tag{20}$$

A.3 Random Projection Induces Approximate Subspace Orthogonality

The orthogonality properties of random projections form the theoretical foundation for FlyLoRA's effectiveness in model merging. Consider two independent random projection matrices $\boldsymbol{A}_i, \boldsymbol{A}_j \in \mathbb{R}^{r\times n}$ with entries distributed as specified in Section 3.1. The expectation of their product reveals:

$$\mathbb{E}[\boldsymbol{A}_i \boldsymbol{A}_j^\top] = \mathbb{E}\left[\sum_{k=1}^{n} (\boldsymbol{A}_i)_{mk} (\boldsymbol{A}_j)_{lk}\right] = \sum_{k=1}^{n} \mathbb{E}[(\boldsymbol{A}_i)_{mk}]\mathbb{E}[(\boldsymbol{A}_j)_{lk}] = \mathbf{0}_{r\times r} \tag{21}$$

This zero-expectation result follows from the independence and zero-mean property of the random matrices. To quantify how tightly the product concentrates around zero, we analyze its variance:

$$\begin{aligned}
\operatorname{Var}\left((\boldsymbol{A}_i \boldsymbol{A}_j^T)_{ml}\right) &= \sum_{k=1}^{n} \Big[\operatorname{Var}\big((\boldsymbol{A}_i)_{mk}\big)\operatorname{Var}\big((\boldsymbol{A}_j)_{lk}\big) + \operatorname{Var}\big((\boldsymbol{A}_i)_{mk}\big)\big(\mathbb{E}(\boldsymbol{A}_j)_{lk}\big)^2 \\
&\quad + \operatorname{Var}\big((\boldsymbol{A}_j)_{lk}\big)\big(\mathbb{E}(\boldsymbol{A}_i)_{mk}\big)^2\Big] \\
&= \sum_{k=1}^{n} \operatorname{Var}\big((\boldsymbol{A}_i)_{mk}\big)\operatorname{Var}\big((\boldsymbol{A}_j)_{lk}\big) \\
&= n\sigma^4 = \frac{p^2}{nr^4}
\end{aligned} \tag{22}$$

Applying Chebyshev's inequality, we bound the probability of large deviations for each entry:

$$\mathbb{P}\left(|(\boldsymbol{A}_i \boldsymbol{A}_j^\top)_{ml}| \geq \epsilon\right) \leq \frac{\operatorname{Var}\left((\boldsymbol{A}_i \boldsymbol{A}_j^\top)_{ml}\right)}{\epsilon^2} = \frac{p^2}{nr^4\epsilon^2} \tag{23}$$

The Frobenius norm characterization gives us:$\|\boldsymbol{A}_i \boldsymbol{A}_j^\top\|_F^2 = \sum_{m=1}^{r}\sum_{l=1}^{r} (\boldsymbol{A}_i \boldsymbol{A}_j^\top)_{ml}^2$. To analyze its probabilistic behavior, we employ the union bound principle: for any events E_{ij} defined as $|(\boldsymbol{A}_i \boldsymbol{A}_j^\top)_{ml}| \geq \epsilon$, we have $\mathbb{P}\left(\bigcup_{m,l} E_{ml}\right) \leq \sum_{m,l} \mathbb{P}(E_{ml})$. Using this union bound over all r^2 entries yields:

$$\mathbb{P}\left(\|\boldsymbol{A}_i \boldsymbol{A}_j^\top\|_F \geq \epsilon r\right) \leq \sum_{m=1}^{r}\sum_{l=1}^{r} \mathbb{P}\left(|(\boldsymbol{A}_i \boldsymbol{A}_j^\top)_{ml}| \geq \epsilon\right) \leq r^2 \cdot \frac{p^2}{nr^4\epsilon^2} = \frac{p^2}{nr^2\epsilon^2}. \tag{24}$$

Since the spectral norm is bounded by the Frobenius norm ($\|\boldsymbol{A}_i^\top \boldsymbol{A}_j\|_2 \leq \|\boldsymbol{A}_i^\top \boldsymbol{A}_j\|_F$), we conclude that when $n \gg \frac{p^2}{r^2\epsilon^2}$, the subspaces spanned by $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$ are approximately orthogonal with high probability.

Theorem A.4 (Approximate Subspace Orthogonality)**.** *For independent random matrices $\boldsymbol{A}_i, \boldsymbol{A}_j \in \mathbb{R}^{r\times n}$ with sparse Gaussian entries ($\mathcal{N}(0, \frac{1}{r^2})$ for $p < n$ randomly selected entries per row), the following holds:*

1. ***Exact mean orthogonality**: $\mathbb{E}[\boldsymbol{A}_i\boldsymbol{A}_j^\top] = \mathbf{0}_{r\times r}$*
2. ***Polynomially decaying correlations**: $\mathbb{P}(\|\boldsymbol{A}_i\boldsymbol{A}_j^\top\|_2 \geq \epsilon r) \leq \frac{p^2}{nr^2\epsilon^2}$*

We demonstrate the approximate orthogonality between distinct LoRA components $\boldsymbol{B}_i\boldsymbol{A}_i$ and $\boldsymbol{B}_j\boldsymbol{A}_j$ through Frobenius inner product analysis following [75]:

$$\begin{aligned}\langle \boldsymbol{B}_j\boldsymbol{A}_j, \boldsymbol{B}_i\boldsymbol{A}_i\rangle_F &= \operatorname{tr}\left(\left(\boldsymbol{B}_j\boldsymbol{A}_j\right)^\top\left(\boldsymbol{B}_i\boldsymbol{A}_i\right)\right)\\ &= \operatorname{tr}\left(\boldsymbol{A}_j^\top\boldsymbol{B}_j^\top\boldsymbol{B}_i\boldsymbol{A}_i\right)\\ &= \operatorname{tr}\left(\boldsymbol{B}_j^\top\boldsymbol{B}_i\boldsymbol{A}_i\boldsymbol{A}_j^\top\right)\\ &\approx \operatorname{tr}\left(\boldsymbol{B}_j^\top\boldsymbol{B}_i\cdot\mathbf{0}_{r\times r}\right)\\ &\approx 0\end{aligned} \tag{25}$$

These analysis demonstrates that random projections naturally create nearly orthogonal subspaces, which helps prevent interference between different experts in FlyLoRA. The small residual correlation becomes negligible when input dimension $n \to \infty$, leading to the effectiveness of our sparse random projection approach for model merging.

Formally, considering the conventional LoRA merging scheme:

$$\boldsymbol{W}' = \boldsymbol{W}_0 + \sum_{i=1}^{t} w_i\boldsymbol{B}_i\boldsymbol{A}_i \tag{26}$$

When multiple FlyLoRA modules ($\boldsymbol{B}_i\boldsymbol{A}_i$) are approximately orthogonal, the squared Frobenius norm of the merged weight matrix can be decomposed into the weighted sum of individual module norms.

$$\left\|\sum_{i=1}^{t} w_i\boldsymbol{B}_i\boldsymbol{A}_i\right\|_F^2 = \sum_{i=1}^{t} w_i^2\|\boldsymbol{B}_i\boldsymbol{A}_i\|_F^2 + \sum_{i\neq j} w_iw_j\langle \boldsymbol{B}_i\boldsymbol{A}_i, \boldsymbol{B}_j\boldsymbol{A}_j\rangle_F \tag{27}$$

$$\approx \sum_{i=1}^{t} w_i^2\|\boldsymbol{B}_i\boldsymbol{A}_i\|_F^2, \tag{28}$$

which aligns with the "Weight disentanglement" property for task arithmetic [56]. We summarize these findings in the following corollary:

Corollary A.5. *Let $\boldsymbol{A}_i, \boldsymbol{A}_j \in \mathbb{R}^{r\times n}$ be fixed sparse random projections after initialization. Then for any learned matrices $\boldsymbol{B}_i\boldsymbol{A}_i$ and $\boldsymbol{B}_j\boldsymbol{A}_j$ the following properties hold:*

1. ***Pairwise Orthogonality**:*

$$\langle \boldsymbol{B}_i\boldsymbol{A}_i, \boldsymbol{B}_j\boldsymbol{A}_j\rangle_F \approx 0 \quad \textit{for} \quad i \neq j$$

2. ***Orthogonality's Outcome in Merging**:*

$$\left\|\sum_{i=1}^{t} w_i\boldsymbol{B}_i\boldsymbol{A}_i\right\|_F^2 \approx \sum_{i=1}^{t} w_i^2\|\boldsymbol{B}_i\boldsymbol{A}_i\|_F^2$$

Through sparse random projections, FlyLoRA inherently constructs nearly orthogonal subspaces for parameter merging from different tasks.

B Additional Results

B.1 Evaluation on Larger Models

We further conducted experiments using the Qwen-2.5-14B model, as shown in Tables 4 and 5, following the settings of Tables 1 and 2. The results show that FlyLoRA remains superior in accuracy

https://doi.org/10.52202/085713-0349

(for both single-task and multi-task settings) and is more parameter-efficient. We encountered no memory or convergence bottlenecks when training FlyLoRA on the 14B model, which consistently outperforms LoRA and Split-LoRA. This scalability confirms that our method's benefits extend effectively to larger model architectures without compromising training stability.

Table 4: **Performance Comparison of LoRA Variants in Single-task Evaluation using Qwen-2.5-14B.** We evaluate various methods across four benchmarks: MMLU, ScienceQA, GSM8K (accuracy), and HumanEval (Pass@k), with all metrics reported in percentage (%). Param(%) indicates the percentage of activated trainable parameters relative to Full FT. The best results are highlighted in **bold**.

Method	Param(%)	MMLU	ScienceQA	GSM8K	HumanEval
LoRA$_{(r=8)}$	0.23	56.74±0.56	95.62±0.18	83.08±0.74	51.69±1.48
LoRA$_{(r=32)}$	0.93	59.35±0.79	97.05±0.22	85.31±0.25	54.80±0.76
Split-LoRA$_{(4\times8)}$	0.29	58.26±1.13	96.85±0.35	84.88±0.51	54.65±0.59
FlyLoRA	**0.12**	**60.17**±1.08	**97.37**±0.32	**85.96**±0.89	**56.42**±1.16

Table 5: **Multi-task Performance Comparison using Qwen-2.5-14B.** We evaluate LoRA variants across MMLU, ScienceQA, GSM8K (accuracy), and HumanEval (Pass@k) benchmarks. The table shows the relative performance drop (Δ%) before and after merging. The best results are highlighted in **bold**.

Method	MMLU	ScienceQA	GSM8K	HumanEval
LoRA$_{(r=8)}$	-13.75	-25.20	-11.43	-18.60
LoRA$_{(r=32)}$	-8.91	-20.45	-7.62	-16.34
Split-LoRA$_{(4\times8)}$	-7.48	-21.97	-6.05	-14.87
FlyLoRA	**-4.35**	**-17.89**	**-2.18**	**-11.72**

B.2 More Baseline into Comparison

We further compare several strong and widely used baselines—AdaLoRA [95] (adaptive rank allocation), SoRA [16] (sparse adaptation), and HydraLoRA [76] (MoE-based)—using Qwen-2.5-7B, as shown in Tables 6 and 7. The results demonstrate that FlyLoRA remains superior in terms of accuracy and efficiency in both single-task learning and multi-task merging scenarios. Notably, FlyLoRA achieves this performance while maintaining a simpler training pipeline, as it avoids the additional hyperparameter tuning required by adaptive and sparse methods.

Table 6: **Performance Comparison with More LoRA Variants in Single-task Evaluation using Qwen-2.5-7B.** We evaluate various methods across four benchmarks: MMLU, ScienceQA, GSM8K (accuracy), and HumanEval (Pass@k), with all metrics reported in percentage (%). Param(%) indicates the percentage of activated trainable parameters relative to Full FT. The best results are highlighted in **bold**.

Method	Param(%)	MMLU	ScienceQA	GSM8K	HumanEval
AdaLoRA$_{(r=8)}$	0.26	51.22±0.21	93.48±0.28	77.65±0.14	47.96±1.34
SoRA$_{(r=8)}$	0.19	50.89±0.42	93.25±0.20	78.46±0.82	47.83±0.94
HydraLoRA$_{(r=8,A=1,B=3)}$	0.52	53.05±0.16	94.69±0.34	79.31±0.49	52.98±1.57
FlyLoRA$_{(k=8)}$	**0.13**	**53.68**±0.47	**95.55**±0.18	**80.82**±0.56	**54.34**±2.13

B.3 Training Time and Memory Consumption

We further conduct experiments comparing the training time and memory consumption of the LoRA variants discussed in Section 4. The results, presented in Table 8, demonstrate that LoRA$_{(r=32)}$ requires more training time and memory than LoRA$_{(r=8)}$ due to its higher rank. Split-LoRA$_{(4\times8)}$ shows intermediate values between these two approaches. Notably, FlyLoRA achieves both the fastest training times (resulting from having the fewest activated parameters) and the lowest memory consumption (mainly attributed to its frozen matrix $\boldsymbol{A}$ that significantly reduces memory for activation

Table 7: **Multi-task Performance Comparison with More Baselines using Qwen-2.5-7B.** We evaluate LoRA variants across MMLU, ScienceQA, GSM8K (accuracy), and HumanEval (Pass@k) benchmarks. The table shows the relative performance drop ($\Delta\%$) before and after merging. The best results are highlighted in **bold**.

Method	MMLU	ScienceQA	GSM8K	HumanEval
AdaLoRA$_{(r=8)}$	-10.90	-33.15	+4.52	-25.46
SoRA$_{(r=8)}$	-9.45	-34.81	+4.05	-26.50
HydraLoRA$_{(r=8,A=1,B=3)}$	-6.22	-34.67	+4.21	-24.94
FlyLoRA$_{(k=8)}$	**+6.55**	**-23.77**	**+4.80**	**-21.23**

values [94]). We also illustrate the theoretical memory consumption for these LoRA variants in Table 9, which tightly aligns with the experimental results. It's clear to see that under a fixed total rank r, a larger number of experts N causes Split-LoRA to require significantly more activated trainable parameters and memory consumption, degrading the efficiency of MoE-based LoRA methods. These results and analysis demonstrate the robustness of FlyLoRA's parameter efficiency compared to MoE-based LoRA methods.

Table 8: **Training Time (Hours) and Memory Usage (GB) of LoRA Variants on Different Datasets and Architectures.** Comparison of LoRA$_{(r=8)}$, LoRA$_{(r=32)}$, Split-LoRA$_{(4\times8)}$, and FlyLoRA$_{(k=8)}$ fine-tuning Llama-3.1-8B and Qwen-2.5-7B on MMLU, ScienceQA, GSM8K, and CodeAlpaca-20k. The best results are highlighted in **bold**.

Metric	Llama-3.1-8B				Qwen-2.5-7B			
	MMLU	ScienceQA	GSM8K	CodeAlpaca	MMLU	ScienceQA	GSM8K	CodeAlpaca
LoRA$_{(r=8)}$								
Training Time	4.79h	5.30h	0.52h	1.85h	4.45h	5.07h	0.54h	1.75h
Memory Usage	12.5GB	14.8GB	20.1GB	20.2GB	14.7GB	16.8GB	22.9GB	22.9GB
LoRA$_{(r=32)}$								
Training Time	5.09h	5.61h	0.58h	1.95h	4.76h	5.39h	0.60h	1.87h
Memory Usage	13.2GB	15.3GB	20.7GB	20.7GB	15.4GB	17.3GB	23.4GB	23.4GB
Split-LoRA$_{(4\times8)}$								
Training Time	4.94h	5.46h	0.56h	1.92h	4.60h	5.23h	0.57h	1.79h
Memory Usage	12.8GB	15.0GB	20.4GB	20.4GB	15.0GB	17.1GB	23.2GB	23.2GB
FlyLoRA$_{(k=8)}$								
Training Time	**4.73h**	**5.23h**	**0.51h**	**1.82h**	**4.39h**	**4.99h**	**0.52h**	**1.70h**
Memory Usage	**10.6GB**	**10.7GB**	**10.9GB**	**10.9GB**	**12.1GB**	**12.2GB**	**12.4GB**	**12.4GB**

Table 9: **Theoretical Memory Consumption Comparison of Different LoRA Variants for a Single Linear Layer.** Param indicates the number of activated trainable parameters. Variables d, r, k, b, s, and N represent hidden dimension, total rank, activation rank, batch size, sequence length, and number of experts, respectively. We record memory usage for weights, gradients, optimizer states, and activations in bytes. These results are calculated under 16-bit mixed-precision training settings.

Method	Param	Weight	Gradient	Optimizer	Activation
LoRA	$2dr$	$2(d^2+2dr)$	$4dr$	$24dr$	$2bsd+2bsr$
Split-LoRA	$2dk+dN$	$2(d^2+2dr+dN)$	$4dk+2dN$	$24dk+12dN$	$2bsd+2bsk+2bsN$
FlyLoRA	dk	$2(d^2+2dr)$	$2dk$	$12dk$	$2bsk$

B.4 Multi-task Performance for Advanced Model Merging Techniques

Extending the results in Section 4.3, we also evaluate two advanced model merging techniques, TIES-MERGING [85] and DARE [87], for merging LoRA components from different domains. The results are listed in Tables 10 and 11, respectively. Overall, both TIES-MERGING and DARE outperform naive weight averaging by resolving parameter conflicts through intelligent selection (TIES' sign consensus and trimming) and selective rescaling (DARE's dropout-based redundancy elimination). Similar to weight averaging, these results demonstrate that FlyLoRA consistently surpasses LoRA$_{(r=8)}$, LoRA$_{(r=32)}$, and Split-LoRA$_{(4\times8)}$ across all comparisons, highlighting the

https://doi.org/10.52202/085713-0349

Table 10: **Multi-task Performance Comparison Before and After Parameter Merging Using TIES-MERGING.** We evaluate LoRA variants across MMLU, ScienceQA, GSM8K (accuracy), and HumanEval (Pass@k) benchmarks. The table shows performance before merging, after merging, and the relative performance drop ($\Delta\%$). The best results are highlighted in **bold**.

Model	Method	Merge Status	MMLU	ScienceQA	GSM8K	HumanEval		
						Pass@1	Pass@5	Pass@10
Llama-3.1-8B	$\text{LoRA}_{(r=8)}$	Before	36.53±0.40	91.39±0.55	55.34±0.24	29.13±0.56	52.28±1.24	61.67±0.61
		After	31.83±0.34	35.96±1.46	27.53±1.08	18.45±1.47	45.52±0.84	56.63±1.24
		Δ (%)	-4.70	-55.43	-27.81	-10.68	-6.76	-5.04
	$\text{LoRA}_{(r=32)}$	Before	38.93±1.04	94.01±0.17	56.25±0.29	30.37±1.06	54.37±0.39	64.02±0.94
		After	34.45±1.73	36.63±0.79	26.59±0.47	20.98±1.13	47.15±0.82	59.97±1.04
		Δ (%)	-4.48	-57.38	-29.66	-9.39	-7.22	-4.05
	$\text{Split-LoRA}_{(4\times8)}$	Before	38.44±0.69	92.41±0.54	55.65±0.47	31.28±1.52	54.16±1.12	63.94±0.89
		After	33.92±0.57	40.81±1.34	29.02±0.82	22.02±0.24	46.32±0.72	59.72±0.25
		Δ (%)	-4.52	-51.60	-26.63	-9.26	-7.84	-4.22
	$\text{FlyLoRA}_{(k=8)}$	Before	40.88±1.61	94.15±0.36	58.76±0.74	36.88±1.91	62.40±1.82	73.34±1.24
		After	39.03±1.31	54.82±0.46	39.12±1.02	33.37±0.62	57.36±1.41	70.35±0.39
		Δ (%)	**-1.85**	**-39.33**	**-19.64**	**-3.51**	**-5.04**	**-2.99**
Qwen-2.5-7B	$\text{LoRA}_{(r=8)}$	Before	49.84±0.56	92.84±0.13	77.01±0.32	47.20±1.54	78.89±0.36	85.94±0.64
		After	44.98±0.72	61.69±1.34	81.89±1.04	23.37±1.24	68.96±1.22	81.25±0.35
		Δ (%)	-4.86	-31.15	+4.88	-23.83	-9.93	-4.69
	$\text{LoRA}_{(r=32)}$	Before	52.07±0.31	95.01±0.21	79.23±0.22	52.87±1.79	81.67±1.14	87.80±0.72
		After	35.92±0.18	58.02±0.27	83.98±0.62	24.79±1.08	67.50±1.13	80.35±1.46
		Δ (%)	-16.15	-36.99	+4.75	-28.08	-14.17	-7.45
	$\text{Split-LoRA}_{(4\times8)}$	Before	50.68±1.06	93.08±0.41	77.12±0.76	48.65±1.18	79.30±0.91	86.05±0.44
		After	44.96±0.65	60.59±0.26	81.82±0.20	23.53±0.74	67.59±0.14	82.34±0.69
		Δ (%)	-5.72	-32.49	+4.70	-25.12	-11.71	-3.71
	$\text{FlyLoRA}_{(k=8)}$	Before	53.68±0.47	95.55±0.18	80.82±0.56	54.34±2.13	82.85±0.52	89.63±0.55
		After	60.51±0.37	73.46±0.80	86.24±0.31	35.37±0.47	76.05±1.25	87.97±1.09
		Δ (%)	**+6.83**	**-22.09**	**+5.42**	**-18.97**	**-6.80**	**-1.66**

robustness of FlyLoRA's near-orthogonality property in reducing inter-task decoupling and further enhancing model merging performance.

We also include comparison with more advanced model merging techniques in Table 12. KnOTS [73] and L-LoRA [75] are both built upon $\text{LoRA}_{(r=32)}$. The results suggest that they achieve comparable performance to FlyLoRA, with each method excelling on different datasets. It is noteworthy that FlyLoRA is not a competitor to these methods; rather, they can be used in a plug-and-play manner with FlyLoRA to further improve performance after merging.

B.5 Additional Ablation Studies on Load-Balancing Strategies

We compare experimental results using different load-balancing strategies. In Section 3.1, we employ an easy-to-implement loss-free balancing strategy. This loss-agnostic approach effectively achieves load balancing with negligible computational overhead and memory footprint. Simultaneously, other loss-controlled load-balancing strategies like [19] are widely used in MoE-like structures. A comparison of them is shown in Table 13. Our results show that different routing strategies achieve similar effects in performance. While FlyLoRA requires load-balancing strategies, it is not sensitive to specific methods.

B.6 Additional Ablation Studies on K-Selection Strategies

To evaluate the impact of activation selection, we analyze different K-selection approaches in our experiments. In neuroscience, the fly olfactory circuit implements a "winner-take-all" strategy, simulated through top-k selection based on activation values across dimensions. To validate top-k's effectiveness, we test random-k selection and full activation (without selection) as baselines. Results in Table 14 show that both top-k and random-k outperform full activation, confirming sparse activation mitigates intra-task interference. Crucially, top-k surpasses random-k because random selection cannot prioritize the most informative dimensions, leading to suboptimal performance.

Table 11: **Multi-task Performance Comparison Before and After Parameter Merging Using DARE.** We evaluate LoRA variants across MMLU, ScienceQA, GSM8K (accuracy), and HumanEval (Pass@k) benchmarks. The table shows performance before merging, after merging, and the relative performance drop (Δ%). The best results are highlighted in **bold**.

Model	Method	Merge Status	MMLU	ScienceQA	GSM8K	HumanEval		
						Pass@1	Pass@5	Pass@10
Llama-3.1-8B	$\text{LoRA}_{(r=8)}$	Before	36.53±0.40	91.39±0.55	55.34±0.24	29.13±0.56	52.28±1.24	61.67±0.61
		After	31.24±0.40	34.37±1.25	26.76±1.10	17.35±1.32	45.49±0.96	57.24±1.36
		Δ (%)	-5.29	-57.02	-28.58	-11.78	-6.79	-4.43
	$\text{LoRA}_{(r=32)}$	Before	38.93±1.04	94.01±0.17	56.25±0.29	30.37±1.06	54.37±0.39	64.02±0.94
		After	34.75±0.83	37.56±0.59	26.89±0.78	19.36±1.38	46.67±0.86	59.85±0.67
		Δ (%)	-4.18	-56.45	-29.36	-11.01	-7.70	-4.17
	$\text{Split-LoRA}_{(4\times8)}$	Before	38.44±0.69	92.41±0.54	55.65±0.47	31.28±1.52	54.16±1.12	63.94±0.89
		After	34.02±0.24	38.58±0.48	28.63±0.72	23.52±1.43	46.84±0.30	60.30±0.41
		Δ (%)	-4.42	-53.83	-27.02	-7.76	-7.32	-3.64
	$\text{FlyLoRA}_{(k=8)}$	Before	40.88±1.61	94.15±0.36	58.76±0.74	36.88±1.91	62.40±1.82	73.34±1.24
		After	39.37±1.02	52.34±0.35	38.20±1.52	33.34±0.83	57.14±1.37	70.24±0.42
		Δ (%)	**-1.51**	**-41.81**	**-20.56**	**-3.54**	**-5.26**	**-3.10**
Qwen-2.5-7B	$\text{LoRA}_{(r=8)}$	Before	49.84±0.56	92.84±0.13	77.01±0.32	47.20±1.54	78.89±0.36	85.94±0.64
		After	45.20±0.40	61.39±1.32	81.98±0.86	23.49±1.02	69.04±0.17	81.22±0.27
		Δ (%)	-4.64	-31.45	+4.97	-23.71	-9.85	-4.72
	$\text{LoRA}_{(r=32)}$	Before	52.07±0.31	95.01±0.21	79.23±0.22	52.87±1.79	81.67±1.14	87.80±0.72
		After	35.27±1.68	56.79±1.34	84.07±0.16	24.35±1.07	67.74±0.70	80.12±0.32
		Δ (%)	-16.80	-38.22	+4.80	-28.52	-13.93	-7.68
	$\text{Split-LoRA}_{(4\times8)}$	Before	50.68±1.06	93.08±0.41	77.12±0.76	48.65±1.18	79.30±0.91	86.05±0.44
		After	43.56±0.84	62.15±0.26	81.82±0.20	23.79±0.52	68.74±0.96	81.53±1.07
		Δ (%)	-7.12	-30.93	+4.70	-24.86	-10.56	-4.52
	$\text{FlyLoRA}_{(k=8)}$	Before	53.68±0.47	95.55±0.18	80.82±0.56	54.34±2.13	82.85±0.52	89.63±0.55
		After	61.35±0.47	72.94±0.21	86.34±0.36	34.47±0.44	75.97±0.70	87.64±1.04
		Δ (%)	**+7.67**	**-22.61**	**+5.52**	**-19.87**	**-6.88**	**-1.99**

Table 12: **Multi-task Performance Comparison with More Advanced Merging Techniques using Qwen-2.5-7B.** We evaluate different methods across MMLU, ScienceQA, GSM8K (accuracy), and HumanEval (Pass@k) benchmarks. The table shows the relative performance drop (Δ%) before and after merging. The best results are highlighted in **bold**.

Method	MMLU	ScienceQA	GSM8K	HumanEval
FlyLoRA	+6.55	-23.77	+4.80	-21.23
KnOTS	+10.76	-26.85	+4.68	-23.37
L-LoRA	+4.51	-22.48	+4.74	-20.85
KnOTS+FlyLoRA	**+11.47**	-23.41	**+5.25**	-20.69
L-LoRA+FlyLoRA	+7.65	**-21.42**	+5.02	**-19.85**

These findings collectively demonstrate that biologically inspired top-k activation optimally balances efficiency and task-specific feature selection.

B.7 Additional Ablation Studies on Matrix A initialization Schemes

We further compare three methods for generating the sparse projection $\boldsymbol{A}$ with $\frac{p}{r}$ sparsity:

1. Gaussian (our default): Each non-zero entry is drawn from $\mathcal{N}(0, \frac{1}{r^2})$.
2. Rademacher (non-Gaussian): Each non-zero entry is $\pm\frac{1}{r}$ with equal probability.
3. FJLT [3] (structured projection): $\boldsymbol{A} = \boldsymbol{PHD}$, where $\boldsymbol{D}$ is a random diagonal matrix with independent Rademacher variables on its diagonal, $\boldsymbol{H}$ is a normalized Hadamard matrix, and $\boldsymbol{P}$ enforces the $\frac{p}{r}$ sparsity.
4. Two-Phase (briefly-learned): The non-zero entries of $\boldsymbol{A}$ are trainable for 5% of total steps as warm-up, then frozen for the remainder.

The results, shown in Table 15, indicate that almost all variants perform similarly. Non-Gaussian, structured, or briefly-learned initializations have little impact, except that the briefly-learned scheme

https://doi.org/10.52202/085713-0349

Table 13: **Performance Comparison of Different Load-Balancing Strategies.** Evaluation on MMLU benchmark using Llama-3.1-8B.

Load-Balancing Strategy	Accuracy (%)
Loss-Free	**40.88**$\pm$**1.61**
Loss-Controlled	40.59$\pm$0.51
No Load-Balancing	37.56$\pm$2.87

Table 14: **Performance Comparison of Different K-Selection Strategies.** Evaluation on MMLU benchmark using Llama-3.1-8B.

K-Selection Strategy	Accuracy (%)
top-k	**40.88**$\pm$**1.61**
random-k	40.02$\pm$0.26
full activation	39.40$\pm$1.14

shows a noticeable drop after merging. This demonstrates that FlyLoRA is robust to the choice of initialization scheme for matrix $\boldsymbol{A}$, and that learning may break the approximate orthogonality of the random matrix, making it unsuitable.

B.8 Analysis of the Performance Gap Between Merged and Non-merged Scenarios

In Tables 2, 10, and 11, we can see that ScienceQA usually demonstrates a large performance drop after merging for all LoRA variants. Intuitively, the four tasks—general knowledge understanding (MMLU), scientific question answering (ScienceQA), mathematical reasoning (GSM8K), and code generation (HumanEval)—represent significantly different distributions, so merging their adapters is prone to substantial conflicts.

Empirically, following [73], we use centered kernel alignment (CKA) [33] to quantify the alignment between the output representations of each single-task adapter and the merged adapter. A higher CKA indicates better output alignment, which is likely the inherent reason, and therefore, results in a smaller accuracy drop after merging. Table 16 reports both CKA and accuracy drop (Δ) on Llama-3.1-8B. Since there is no apparent difference between LoRA$_{(r=8)}$, LoRA$_{(r=32)}$, and Split-LoRA$_{(4\times8)}$ in model merging, we use LoRA$_{(r=8)}$ as a representative to compare with FlyLoRA$_{(k=8)}$. We observe that tasks with lower CKA (especially ScienceQA and GSM8K) suffer the largest accuracy drops. FlyLoRA consistently yields higher CKA than LoRA, which aligns with its consistently smaller Δ. This micro-level analysis corroborates why FlyLoRA outperforms LoRA in heterogeneous-task merging.

C Detailed Experimental Setting

C.1 Datasets

To comprehensively evaluate the effectiveness of our proposed method across diverse domains and task types, we conduct extensive experiments on five carefully selected benchmarks. These datasets span critical capabilities including general knowledge reasoning, scientific understanding, mathematical problem solving, and code generation. Table 17 summarizes the key characteristics of each dataset, while detailed descriptions are provided below:

- **MMLU** [25] serves as a comprehensive benchmark for evaluating broad knowledge understanding and reasoning capabilities. It comprises multiple-choice questions spanning 57 distinct academic subjects, ranging from elementary mathematics and US history to computer science and professional law. The diversity of domains makes it particularly suitable for assessing model generalization across different knowledge types.
- **ScienceQA** [48] provides a multimodal framework for science question answering, with content derived from elementary and high school curricula aligned with California Common Core Content Standards. The questions originate from IXL Learning's expert-curated educational resources. Following the established practice in [37], we utilize only the textual components to focus on linguistic understanding.

Table 15: **Performance Comparison of Different A Initialization Strategies.** Single-task and multi-task accuracy comparison on MMLU using Llama-3.1-8B.

A Initialization Strategy	Before	Δafter merging
Gaussian	$40.88_{\pm 1.61}$	-2.02
Rademacher	$40.42_{\pm 0.23}$	-2.35
FJLT	$40.57_{\pm 1.34}$	-2.50
Two-Phase	$40.76_{\pm 1.04}$	-4.86

Table 16: **CKA and Corresponding Accuracy Drop (△) Between Single-Task Adapter and Merged Model.** Evaluating using Llama-3.1-8B.

Method	Task	MMLU	ScienceQA	GSM8K	HumanEval
$\text{LoRA}_{(r=8)}$	CKA	0.78	0.39	0.58	0.75
	Δ	-6.48	-60.34	-30.15	-13.04
$\text{FlyLoRA}_{(k=8)}$	CKA	**0.85**	**0.53**	**0.71**	**0.84**
	Δ	**-2.02**	**-43.05**	**-21.81**	**-4.27**

- **GSM8K** [12] offers 8,500 high-quality grade school mathematics word problems that demand multi-step arithmetic reasoning. Each problem is accompanied by a detailed, step-by-step solution, making it ideal for evaluating logical reasoning and procedural accuracy in mathematical contexts.
- **CodeAlpaca-20k** [7] contains 20,022 synthetically generated instruction-response pairs specifically designed for code-related tasks. This dataset facilitates effective instruction tuning for programming applications by providing diverse coding prompts paired with corresponding solutions.
- **HumanEval** [9] consists of 164 hand-crafted Python programming problems developed to assess functional correctness in code generation. Crucially, these problems were manually created to prevent data contamination, ensuring they do not appear in the training corpora of existing code generation models.

Table 17: **Details of MMLU, ScienceQA, GSM8K, CodeAlpaca and HumanEval Datasets.** We list the number of training and testing samples and task types for the following datasets used in our experiments.

Dataset	Training Samples	Testing Samples	Task Types
MMLU [25]	99,842	14,042	Multiple Choice
ScienceQA [48]	12,726	4,241	Multiple Choice
GSM8K [12]	7,473	1,319	Math Problems
CodeAlpaca-20k [7]	20,022	–	Code Instruction
HumanEval [9]	–	164	Code Generation

C.2 Training Configuration

This section elaborates on the experimental setup and hyperparameter configurations employed throughout our study. Table 18 documents the shared training parameters applied consistently across all backbone models and datasets. To address the specific requirements of different tasks and model architectures, we additionally provide dataset-specific and model-specific configurations in Table 19, including learning rate schedules, batch size adjustments, and task-specific optimization strategies.

C.3 Split-LoRA

We implement Split-LoRA as a representative MoE-based LoRA method, following the general framework described in Section 2.2. In our experiments, we incorporate a sigmoid activation function in the router to normalize expert selection scores. Thus, the gating function operates as $\boldsymbol{G}(\boldsymbol{x}) = \text{sigmoid}(\text{top-}k(\boldsymbol{W}_g\boldsymbol{x}))$, which ensures differentiable routing while maintaining the sparsity of expert activation. This configuration allows Split-LoRA to serve as a representative baseline for evaluating the effectiveness of MoE structures in LoRA.

https://doi.org/10.52202/085713-0349

Table 18: **General Training Hyperparameters for FlyLoRA.** Shared configuration across all experiments, including rank settings, optimizer details, and architectural choices.

Parameter	Value
Total rank (r)	32
Scaling factor (α)	64
Activated rank	8
Target modules	`{q,k,v,o,gate,down,up}_proj`
Optimizer	AdamW
Warmup ratio	0.01
Gradient accumulated batch	128
Dropout rate	0.00

Table 19: **Dataset-Specific and Model-Specific Training Configurations for FlyLoRA.** Task-optimized settings for Llama-3.1-8B and Qwen-2.5-7B across four benchmarks, showing variations in epoch counts, learning rates, and sequence lengths based on dataset characteristics and model requirements.

Model	Parameter	MMLU	ScienceQA	GSM8K	CodeAlpaca
Llama-3.1-8B	Epochs	1	20	1	2
	Learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
	Max sequence length	128	256	512	512
	micro batch size	8	8	8	8
Qwen-2.5-7B	Epochs	1	20	1	2
	Learning rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	6×10^{-4}
	Max sequence length	128	256	512	512
	micro batch size	8	8	8	8

C.4 Environments

Most experiments were conducted on a Linux server running Ubuntu 20.04.4 LTS, equipped with an Intel(R) Xeon(R) Platinum 8358P CPU at 2.60GHz and 8 NVIDIA GeForce RTX 3090 GPUs, using CUDA version 11.7. Experiments with Qwen-2.5-14B were conducted on a machine with 8 NVIDIA A100 GPUs.

D Limitations and Future Work

In FlyLoRA, matrix $\boldsymbol{A}$ is randomly initialized and frozen during training, but there may still be room for improvement. Recent neuroscience studies [13] suggest that $\boldsymbol{A}$ need not be entirely frozen and random, indicating potential for more bio-inspired mechanisms to enhance task decoupling through an adaptable version of $\boldsymbol{A}$. Moreover, recent works suggest that component-wise interpretability [98] and spectral modulation [74] could inspire adaptive or frequency-aware modifications of $\boldsymbol{A}$ in FlyLoRA to improve efficiency, robustness, and task decoupling.

Recently, RL fine-tuning for LLMs has emerged as a promising approach that significantly enhances their reasoning ability [24]. However, stabilizing MoE RL training remains an open question [96], and further exploration will focus on the integration of FlyLoRA with RL training [61] and potentially extending it to offline policy optimization [50–53, 60, 68]. Additionally, integrating active data selection methods could be a promising direction to further improve data efficiency [62, 77, 79, 99].

E Broader Impact

Our proposed FlyLoRA resolves the trade-off between parameter interference and efficiency in MoE-based LoRA approaches. Additionally, this efficient decoupling mechanism, which is inspired by fly olfactory circuits, can be applied across various domains, helping researchers and developers leverage more powerful LoRA fine-tuning strategies. On the other hand, FlyLoRA could potentially be misused to fine-tune LLMs that exhibit biases or generate harmful content. We recommend implementing model access controls and bias-monitoring frameworks when deploying this technique.