HyRA: Head-wise Reflective Adaptation via Input-Dependent Householder Transformations

Anonymous authors
Paper under double-blind review

Abstract

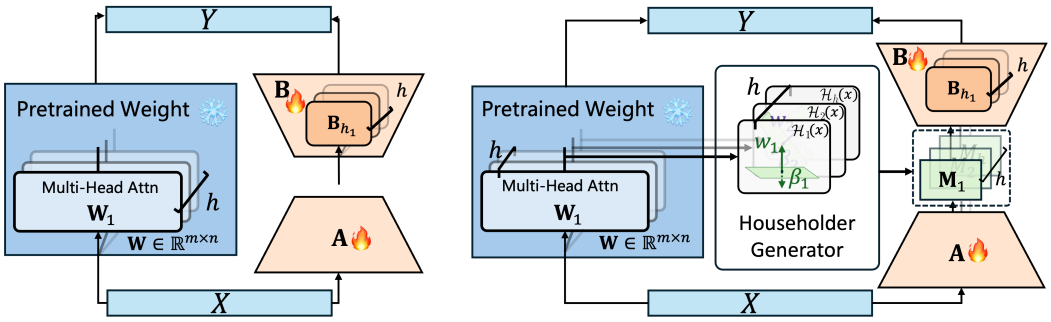
Low-rank adaptation (LoRA) has become a dominant approach to parameter-efficient fine-tuning, which is widely used in large foundation models. LoRA can be conceptualized as projecting the input space into a low-dimensional latent adaptation space, where the dimensionality is determined by the rank of LoRA. However, under multi-head attention, standard LoRA constrains all heads to share a single latent adaptation space, despite attention heads typically encoding functionally distinct roles in pretrained models. We identify this mismatch between the shared adaptation and head-wise specialization as a structural bottleneck that limits LoRA's expressiveness. In this paper, we propose **Head-wise Reflective Adaptation (HyRA)**, which dynamically generates an input-dependent Householder transformation for each head. This Householder generator enables compact head-wise adaptation by modulating feature components along an input-dependent direction while preserving the orthogonal residual. Formally, the head-wise HyRA update can be expressed as $\Delta\mathbf{W}^h = \mathbf{B}_h\mathcal{H}_h(x)\mathbf{A}$, where $\mathbf{A}$ and $\mathbf{B}_h$ are low-rank matrices and $\mathcal{H}_h(x)$ is a head-wise, input-dependent Householder transformation that breaks the shared subspace bottleneck. Notably, HyRA does not increase the LoRA rank, but enables each head to learn its own input-dependent adaptation. Extensive experiments across both language and vision architectures demonstrate that HyRA consistently outperforms LoRA and its competitive variants across diverse downstream benchmarks.

1 Introduction

Large foundation models have achieved remarkable success across domains, including natural language processing (NLP) (Grattafiori et al., 2024; Liu et al., 2024a; Abdin et al., 2024; Bai et al., 2023; Achiam et al., 2023; Yang et al., 2025a) and computer vision (CV) (Dosovitskiy et al., 2021; Kirillov et al., 2023; Ravi et al., 2025). Building on this, Parameter-Efficient Fine-Tuning (PEFT) (Mangrulkar et al., 2022) presents a new paradigm that leverages their strong prior knowledge, adapting them to diverse downstream tasks by updating only a small set of parameters. Among these methods, low-rank adaptation (LoRA) (Hu et al., 2021) has been widely adopted. As illustrated in Figure 1a, LoRA reparameterizes the weight update of a frozen linear layer $\mathbf{W} \in \mathbb{R}^{m\times n}$ as $\Delta\mathbf{W} = \mathbf{BA}$, where $\mathbf{B} \in \mathbb{R}^{m\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times n}$, with r being the LoRA rank (i.e., $r \ll \min\{m, n\}$).

Despite its effectiveness, LoRA typically underperforms full fine-tuning (FFT), a gap often attributed to the limited number of trainable parameters (Liu et al., 2024b). Previous studies have shown that increasing the LoRA rank (Zhang et al., 2023; Lialin et al., 2024) and expanding the adaptation to a broader set of target modules (Hu et al., 2021; Dettmers et al., 2023) generally improve fine-tuning performance. Beyond increasing the number of trainable parameters, a natural yet underexplored question is how effectively LoRA accommodates the distinct adaptation needs of different attention heads under a limited parameter budget. The intuition behind this question is that different heads in Multi-Head Attention (MHA) (Vaswani et al., 2017; Ainslie et al., 2023) may capture distinct patterns and thus benefit from different forms of adaptation. Although standard LoRA allows different heads to learn different updates, it parameterizes the overall weight update through a single low-rank factorization spanning all heads. We therefore revisit LoRA from a head-wise perspective in Section 2 to examine how its shared adapta-

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



(a) LoRA (b) HyRA

Figure 1: **An illustration of HyRA in comparison with LoRA. Left:** LoRA updates $\Delta\mathbf{W} = \mathbf{BA}$ with a shared $\mathbf{A}$ and a single dense $\mathbf{B}$, whose row-block partition $\mathbf{B}_h$ corresponds to each attention head. All heads pass through the same latent space defined by $\mathbf{A}$, without consideration the pretrained weight structure. **Right:** HyRA introduces a head-wise, input-dependent Householder transformation $\mathcal{H}_h(x)$ between the shared $\mathbf{A}$ and the head-wise factors $\mathbf{B}_h$. The Householder Generator takes each head's frozen output as input and produces a per-head modulation $\mathcal{H}_h$, breaking LoRA's shared latent adaptation space across heads.

tion capacity serves individual heads. Our analysis shows that LoRA shares a low-dimensional adaptation representation across heads, while head-specific variation is modeled through input-independent linear projections. This design efficiently reuses shared features, but may become restrictive when different heads require input-dependent transformations of those features.

Recently, the mixture of LoRA experts (MoLE) (Gao et al., 2025; Zhuang et al., 2025) improves adaptation flexibility by routing inputs across multiple LoRA experts, but it introduces substantial parameter overhead, fails to fully utilize the parameter space across experts (Zhao et al., 2026; Back et al., 2026), and does not account for the adaptation requirements of individual attention heads. Token-wise approaches (Li et al., 2026) also introduce input dependence, such as TopLoRA (Li et al., 2025), but apply a shared transformation across heads. Figure 2 further shows that LoRA obtains lower adaptation gains on inputs with higher head-wise heterogeneity, consistently across ranks. MoLE and TopLoRA exhibit the same trend despite their slightly better performance. These observations motivate improving adaptation effectiveness by better accommodating the heterogeneous needs of individual attention heads, rather than relying solely on increasing the number of trainable parameters.

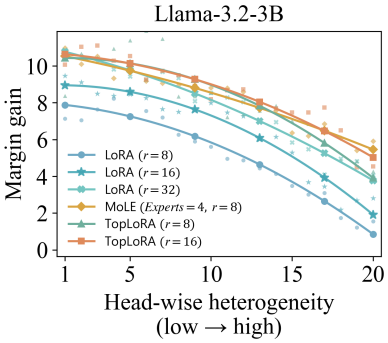


Figure 2: **Adaptation gain under varying head-wise heterogeneity.** Heterogeneity captures head-wise gradient dispersion; margin gain reflects the improvement in prediction margin over the frozen model (Appendix D).

To this end, we propose Head-wise Reflective Adaptation (HyRA) as shown in Figure 1b, which enables head-wise, input-dependent transformations on LoRA's shared adaptation representation. Specifically, HyRA adopts a lightweight identity-plus-rank-one Householder-like transformation (Golub and Van Loan, 2013) for each head to modulate the shared representation. By decomposing the representation into a direction-aligned component and an orthogonal residual, HyRA selectively adjusts the former while preserving the latter, without constructing a full transformation matrix. Our contributions are summarized as follows:

- We identify the shared adaptation representation of LoRA as a structural bottleneck under multi-head attention. This perspective shifts the focus from parameter scaling alone to how adaptation is shared across heads.

- We propose Head-wise Reflective Adaptation (HyRA) to address this structural mismatch. HyRA uses lightweight, input-dependent Householder transformations to selectively modulate components along a learned direction while preserving orthogonal residuals, enabling finer-grained adaptation across heads and avoiding dense transformation matrices.
- We conduct comprehensive experiments on Llama3.2-3B, Llama3.1-8B, Qwen3-1.7B, ViT-Base, ViT-Large, including extensive ablations and detailed analysis, to validate the effectiveness of HyRA.

2 MOTIVATIONS

The mechanism underlying head-specific adaptation in LoRA remains largely unexplored in existing work. To this end, we analyze the head-wise adaptation process of LoRA and identify a structural constraint imposed by its shared latent representation. We characterize this process through the polar decomposition of LoRA factors. Specifically, the matrix $\mathbf{B}$ and $\mathbf{A}$ can be expressed using the right polar decomposition $\mathbf{B} = \mathbf{U}_B\mathbf{H}_B$ and the left polar decomposition $\mathbf{A} = \mathbf{H}_A\mathbf{V}_A^\top$, respectively, where $\mathbf{U}_B \in \mathbb{R}^{m\times r}$, $\mathbf{V}_A \in \mathbb{R}^{n\times r}$ are semi-orthogonal matrix, $\mathbf{H}_B$, $\mathbf{H}_A \in \mathbb{R}^{r\times r}$ are positive semi-definite matrix. At this point, the LoRA weights can be represented as follows:

$$\Delta\mathbf{W} = \mathbf{B}\mathbf{A} = \mathbf{U}_B\mathbf{H}_B\mathbf{H}_A\mathbf{V}_A^\top = \mathbf{U}_B(\mathbf{H}_B\mathbf{H}_A)\mathbf{V}_A^\top = \mathbf{U}_B\mathbf{M}\mathbf{V}_A^\top. \tag{1}$$

The pretrained weight $\mathbf{W}$ can be partitioned into H heads as $\mathbf{W} = [\mathbf{W}^1, \mathbf{W}^2 \ldots, \mathbf{W}^H]^\top$, and the corresponding $\mathbf{B}\mathbf{A} = [\mathbf{U}_{B_1}, \mathbf{U}_{B_2}, \ldots, \mathbf{U}_{B_H}]^\top\mathbf{M}\mathbf{V}_A^\top$, where $\mathbf{U}_{B^i} \in \mathbb{R}^{\frac{m}{H}\times r}$ are partitioned from $\mathbf{U}_B$. For head h, the weight matrix $\mathbf{W}^h$ is adapted by $\Delta\mathbf{W}^h = \mathbf{U}_{B^h}\mathbf{M}\mathbf{V}_A^\top$.

Given an input hidden state $x \in \mathbb{R}^n$, $\mathbf{V}_A^\top$ projects x to a low-dimensional subspace, then $\mathbf{M}$ performs a transformation within this subspace, modulating both the magnitude and direction of the latent representation. Each head subsequently maps this shared representation to its output space through $\mathbf{U}_{\mathbf{B}^h}$. Although $\mathbf{U}_{\mathbf{B}^h}$ provides a head-specific linear readout, its fixed weights may not realize all input-dependent feature combinations required for adaptation. Meanwhile, expanding $\mathbf{V}_A$ by incorporating multiple $\mathbf{A}_i$ offers limited additional expressivity, as these components have been shown to converge to similar subspaces (Tian et al., 2024). The above analysis indicates that the key limitation lies in using a single shared transformation $\mathbf{M} = \mathbf{H}_B\mathbf{H}_A$ in the latent space. When different heads require divergent transformations, this shared $\mathbf{M}$ cannot fully accommodate their distinct requirements, limiting the effectiveness of LoRA.

3 METHOD

To enable head-wise adaptation within the shared latent space, we propose **Head-wise Reflective Adaptation (HyRA)**. HyRA introduces a lightweight input-dependent Householder-like transformation before head-specific reconstruction, allowing each head to selectively modulate latent features along a head- and input-dependent direction with adaptive strength. In this section, we describe the HyRA in detail.

Overview. Figure 1b illustrates HyRA. The shared low-rank projection $\mathbf{A} \in \mathbb{R}^{r\times n}$ maps the input x into a latent space, and $\{\mathbf{B}_h \in \mathbb{R}^{(m/H)\times r}\}_{h=1}^H$ obtained by partitioning $\mathbf{B}$ according to attention heads, map the latent back to each head output. Between $\mathbf{A}$ and $\{\mathbf{B}_h\}$, HyRA inserts an input-dependent Householder transformation $\mathcal{H}_h(x) \in \mathbb{R}^{r\times r}$ that modulates the latent on a per-head basis. The pipeline has three stages: (i) the shared projection produces a latent $u = \mathbf{A}x$; (ii) for each head, a lightweight generator generates the parameters $(\mathbf{w}_h(x), \beta_h(x))$ that define $\mathcal{H}_h(x)$; (iii) each $\mathbf{B}_h$ maps the modulated latent $\mathcal{H}_h(x)\,u$ into the output space of head.

Shared Low-Dimensional Projection. Given an input x, we retain the shared down-projection of LoRA and obtain the compact latent representation $u = \mathbf{A}x$. To further understand the role of this shared projection, we rewrite it with left polar decomposition: $\mathbf{A} = \mathbf{H}_A\mathbf{V}_A^\top$. $\mathbf{V}_A^\top$ maps x onto a shared r-dimensional subspace and the positive semi-definite operator $\mathbf{H}_A$ further encodes the resulting latent components within this space. This decomposition shows that the shared $\mathbf{A}$ provides a common latent basis on which all attention heads operate. A straightforward way to introduce

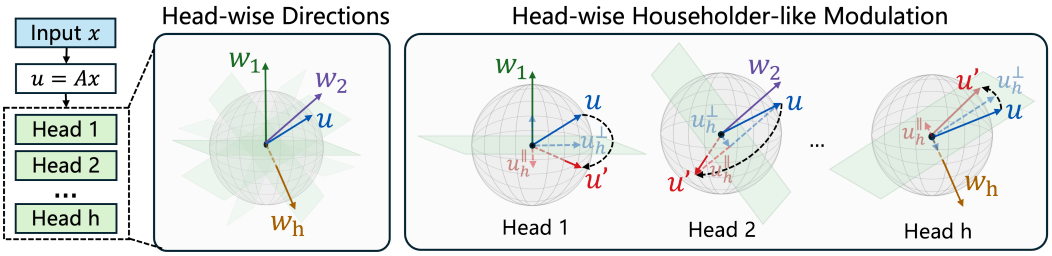


Figure 3: **Head-Wise Householder-like adaptation.** Left: directions w_h define head-wise hyperplanes for u in the shared latent space. Right: each head independently transforms the same latent representation (blue) into its adapted representation (red), modulating the component parallel to w_h while preserving the orthogonal residual.

head-wise adaptation is to assign an independent down-projection $\mathbf{A}_h$ to each attention head. However, this increases the parameter cost with the number of heads and may lead to redundant subspace learning. Meanwhile, different $\mathbf{A}_h$ captures substantially overlapping low-dimensional structures from the same input (Tian et al., 2024).

Head-Wise Householder-Like Adaptation. To enable dynamic head-wise adaptation, we introduce a lightweight generator based on Householder-like transformation [1] . A standard Householder transformation $\mathcal{R}(w)$ reflects a vector across a hyperplane through the origin with unit normal w. It reverses the component aligned with w while leaving the orthogonal residual unchanged. In HyRA, we keep this identity-plus-rank-one structure and learn an input-dependent reflection direction and coefficient for each head. Specifically, we process $\mathbf{W}^h x$ with a linear layer followed by GELU activation to obtain a head-wise hidden state $\mathbf{r}_h \in \mathbb{R}^{d_{\text{head}}}$, from which two outputs produce:

$$\mathbf{w}_h \;=\; \mathrm{L2Norm}(\mathbf{W}_w\,\mathbf{r}_h), \qquad \beta_h \;=\; 2 \times \mathrm{sigmoid}(\mathbf{W}_\beta\,\mathbf{r}_h + b), \tag{2}$$

where $\mathbf{w}_h(x) \in \mathbb{R}^r$, $\beta_h(x) \in (0, 2)$ [2]. This parameterization gives each head a structured and geometrically interpretable mechanism for adapting the shared latent representation. Then, Householder-like transformation is constructed:

$$\mathcal{H}_h(x) = \mathbf{I} - \beta_h(x)\mathbf{w}_h(x)\mathbf{w}_h(x)^\top. \tag{3}$$

For each head, u is decomposed into the component parallel to $\mathbf{w}_h(x)$ and its orthogonal residual:

$$\mathbf{u} = \underbrace{\mathbf{w}_h(x)\mathbf{w}_h(x)^\top\mathbf{u}}_{\mathbf{u}_h^{\parallel}} + \underbrace{(\mathbf{I} - \mathbf{w}_h(x)\mathbf{w}_h(x)^\top)\mathbf{u}}_{\mathbf{u}_h^{\perp}}. \tag{4}$$

Accordingly, $\mathcal{H}_h(x)$ transforms u into $\mathcal{H}_h(x)u = \big(\mathbf{I} - \beta_h(x)\mathbf{w}_h(x)\mathbf{w}_h(x)^\top\big)\mathbf{u} = (1 - \beta_h(x))\mathbf{u}_h^{\parallel} + \mathbf{u}_h^{\perp}$. Each transformation selectively modulates the component $\mathbf{u}_h^{\parallel}$ aligned with $\mathbf{w}_h(x)$ by strength $\beta_h(x)$, which captures information aligned with the head- and input-dependent direction. Figure 3 visualizes this decomposition and the resulting head-wise transformations. The left panel illustrates the head-wise directions, and the right panels show how our transformations independently act on the same latent vector. In this way, HyRA extends adaptation beyond fixed linear readouts, enabling each head to dynamically recombine shared latent features according to the current input.

Head-Wise Reconstruction. Finally, the adapted representation is mapped into the output space of head h through B_h. Consequently, HyRA forward propagation for the head h is

$$\Delta\mathbf{W}^h\,x \;=\; \mathbf{B}_h\,\mathcal{H}_h(x)\,\mathbf{A}\,x \;=\; \mathbf{B}_h\,\big(\mathbf{I} - \beta_h(x)\,\mathbf{w}_h(x)\,\mathbf{w}_h(x)^\top\big)\,\mathbf{A}\,x. \tag{5}$$

We initialize $\mathbf{B}_h$ to zero so that $\Delta\mathbf{W}^h = 0$ at the start of training and the adapted model exactly recovers the frozen backbone, and we initialize $\mathbf{W}_\beta$ to zero so that $\beta_h = 1$ at initialization. The

[1] Householder matrices follow the form $\mathbf{I} - \frac{\mathbf{2}}{\|\mathbf{w}\|_\mathbf{2}^\mathbf{2}}\mathbf{w}\mathbf{w}^\top$ and therefore our matrix is Householder-like.

[2] Here we set $\beta_h(x) \in (0, 2)$ following the conclusion of (Grazzi et al., 2025), because this allows the transition matrix to admit negative eigenvalues, which has been shown to enhance expressivity and performance in rank-one modulations (Yang et al., 2025b; Siems et al., 2025; Jelassi et al., 2024)

remaining parameters, $\mathbf{A}$, $\mathbf{W}_w$ and the linear layer follow Kaiming initialization. To interpret this reconstruction, write $\mathbf{B}_h = [b_{h,1}, \ldots, b_{h,r}]$ and let $c_h(x) = \mathcal{H}_h(x)\mathbf{A}x$. The head-wise update then be expressed as

$$\Delta y_h = \mathbf{B}_h c_h(x) = \sum_{j=1}^{r} c_{h,j}(x) b_{h,j}. \tag{6}$$

Each column $b_{h,j}$ defines a learned head-wise direction in the output space, while $c_{h,j}(x)$ determines its input-dependent contribution to the current update. Thus, $\mathbf{B}_h$ provides the head-wise output directions, and $\mathcal{H}_h(x)$ dynamically determines how the shared latent features are combined for each input. It is important to note that the objective of HyRA is neither to increase the rank of the LoRA weights nor to expand adaptation to a broader set of target modules, but rather to find the optimal projection for each head, even under limited rank and targeted module settings.

4 EXPERIMENTS

In this section, we evaluate HyRA through a comprehensive set of experiments and compare it against state-of-the-art PEFT methods. We conduct experiments on both natural language processing and computer vision benchmarks. For natural language processing benchmarks, we consider Llama-3.2-3B, Llama-3.1-8B, and Qwen3-1.7B, all of which adopt Grouped Query Attention (GQA). For computer vision benchmarks, we use ViT-Base and ViT-Large, which employ standard Multi-Head Attention (MHA). We apply HyRA to all target modules within the GQA and MHA blocks. By evaluating HyRA across models with different attention architectures and across both language and vision tasks, these experiments demonstrate its strong generalization capability across model architectures and task domains. All our experiments are conducted on a server equipped with 8 NVIDIA A100 80GB GPUs.

4.1 NLP BENCHMARKS

We evaluate the overall accuracy of our method against several standard NLP benchmarks. We adopt the same evaluation metric as (Zhuang et al., 2025). Please refer to Appendix C for more details of datasets and setup. We test baseline approaches on Llama-3.2-3B, Llama-3.1-8B (Grattafiori et al., 2024), Qwen3-1.7B (Yang et al., 2025a). The effectiveness of HyRA is demonstrated by comparison with several baseline methods, including LoRA-style and MoELoRA-style methods. Each of these methods introduces distinct modifications to the original LoRA framework. For LoRA-style methods, we compare with LoRA (Hu et al., 2022) and its variants DoRA (Liu et al., 2024b) and TopLoRA (Li et al., 2025). For MoELoRA-style methods, We compare our method primarily against ReMoLE (Wang et al., 2024b), which adapts the ReLU-based routing from ReMoE to the LoRA experts setting. We also include LD-MoLE (Zhuang et al., 2025), which adapts Sparsegen (Laha et al., 2018) to decide the routing probability and ensures differentiability. By comparing with these techniques, we can clearly demonstrate the superiority of HyRA.

In the main experiments, we evaluate the accuracy LoRA across rank 8, 16, and 32 to analyze the trade-off between the number of parameters and model performance. We follow the experimental setup in (Zhuang et al., 2025). The learning rate is chosen from the set $\{1e^{-4}, 2e^{-4}, 4e^{-4}, 5e^{-4}\}$. We adapt a cosine annealing scheduler to adjust the learning rate across training steps, which facilitates a smooth decrease in the learning rate. All methods are trained for the same number of optimization steps on each task. For Qwen3-1.7B and Llama-3.2-3B, we train all methods with AdamW for 10 epochs per task; for Llama-3.1-8B, we run the experiments for 4 epochs since we observe a convergence at the early stage of training. The mean and standard deviation are reported after hyperparameter grid-search for all experiments. Each experiment is repeated five times. Further details can be found in Appendix E.

Results. Table 1 shows the relative ranking of the methods is consistent across LLaMA-3.1-8B, LLaMA-3.2-3B, and Qwen3-1.7B, spanning two model families and three parameter scales from 1.7B to 8B. HyRA achieves the highest average accuracy in every setting. This indicates that the benefit of head-wise input-dependent Householder modulation generalizes across architectures rather than being tied to any particular base model. Increasing the LoRA rank improves fine-tuning performance. Specifically, raising the rank from 8 to 32 increases the average accuracy of LoRA by $+1.51$, $+1.79$, $+1.46$ in Llama3.1-8B, Llama3.2-3B and Qwen3-1.7B, respectively. In all tested

Table 1: Comparison between methods across language downstream tasks with various pretrained models. The best results in each group are shown in **bold**. We report the mean performance over 5 runs with different random seeds.

	Method	Trainable Para.	MMLU-P	ARC-C	ARC-E	OBQA	CSQA	SWAG	Hella	CoLA	RTE	Avg.
Llama3.1-8B	LoRA ($r = 8$)	0.26%	53.31±0.6	82.24±0.1	90.62±0.2	87.48±0.3	83.41±0.1	91.94±0.1	94.73±0.5	84.69±1.0	88.19±0.8	84.07
	LoRA ($r = 16$)	0.52%	53.89±0.9	82.89±0.2	90.80±0.2	88.00±0.2	83.25±0.1	92.08±0.1	95.66±0.4	86.93±0.8	87.81±0.5	84.59
	LoRA ($r = 32$)	1.03%	56.73±1.5	83.34±0.1	91.67±0.1	88.57±0.2	83.58±0.4	92.94±0.3	95.78±0.5	87.87±1.3	88.69±0.5	85.58
	DoRA($r = 16$)	0.55%	54.64±0.6	83.88±0.3	92.19±0.3	90.00±0.4	83.90±0.3	91.99±0.2	95.65±0.3	86.70±0.5	89.27±0.8	85.36
	TopLoRA($r = 8$)	0.38%	54.55±0.7	83.22±0.4	92.01±0.2	88.27±0.2	84.56±0.3	92.66±0.7	95.90±0.7	86.37±0.6	88.67±0.4	85.13
	TopLoRA($r = 16$)	0.77%	55.13±0.6	82.89±0.2	92.01±0.1	89.73±0.1	84.23±0.2	92.82±0.4	96.24±0.3	86.70±0.5	88.93±0.6	85.41
	ReMoLE	2.17%	54.98±0.8	82.00±0.2	91.61±0.3	87.60±0.2	83.17±0.2	92.22±0.4	94.95±0.1	86.21±0.3	88.43±0.5	84.57
	LD-MoLE	2.33%	55.98±1.1	83.67±0.5	92.31±0.3	88.27±0.2	83.01±0.1	92.29±0.3	95.45±0.2	86.32±0.6	91.32±1.3	85.40
	HyRA ($r = 8$)	0.31%	55.86±0.6	84.28±0.2	92.46±0.2	**90.18±0.1**	84.48±0.4	92.55±0.5	96.22±0.1	87.93±0.3	91.03±1.0	86.11
	HyRA ($r = 16$)	0.59%	**58.43±1.2**	**85.45±0.4**	**93.30±0.2**	89.89±0.3	**84.81±0.3**	**93.08±0.3**	**96.42±0.1**	**88.36±0.5**	**93.53±1.2**	**87.03**
Llama3.2-3B	LoRA($r = 8$)	0.38%	44.13±0.7	73.68±0.2	87.01±0.4	83.60±0.3	79.49±0.4	89.34±0.1	92.55±0.6	84.20±0.8	87.18±0.9	80.13
	LoRA($r = 16$)	0.75%	45.69±0.7	74.01±0.1	87.85±0.5	84.07±0.1	80.56±0.7	89.79±0.3	93.54±0.3	85.02±1.3	87.81±0.7	80.93
	LoRA($r = 32$)	1.50%	47.18±1.0	75.32±0.2	89.51±0.5	84.86±0.3	81.39±0.3	91.05±0.2	93.65±0.2	85.69±1.0	88.65±1.0	81.92
	DoRA($r = 16$)	0.77%	47.10±0.8	74.04±0.1	88.67±0.3	84.27±0.2	81.72±0.1	90.97±0.2	93.58±0.3	85.92±0.8	89.19±0.6	81.72
	TopLoRA($r = 8$)	0.57%	47.36±0.6	74.36±0.4	88.42±0.2	84.20±0.1	80.88±0.2	91.21±0.4	93.81±0.3	85.18±0.4	88.17±0.5	81.51
	TopLoRA($r = 16$)	1.12%	47.93±0.8	74.34±0.4	89.23±0.3	84.27±0.1	81.94±0.2	90.94±0.1	93.89±0.1	85.65±0.9	88.81±0.5	81.89
	ReMoLE	3.11%	48.01±0.8	75.25±0.3	89.30±0.4	83.40±0.2	79.52±0.5	90.45±0.2	93.44±0.5	83.95±1.2	89.46±0.8	81.42
	LD-MoLE	3.28%	**49.58±1.3**	74.58±0.3	89.47±0.2	84.00±0.4	81.42±0.6	91.37±0.8	93.60±0.3	86.02±0.7	88.38±1.2	82.05
	HyRA ($r = 8$)	0.48%	47.52±0.4	75.54±0.5	89.78±0.4	84.53±0.3	82.72±0.3	**91.51±0.1**	93.99±0.2	85.31±0.7	90.82±1.3	82.42
	HyRA ($r = 16$)	0.86%	48.63±0.6	**76.95±0.4**	**90.23±0.2**	**85.20±0.5**	**82.74±0.2**	91.34±0.1	**94.50±0.3**	**86.17±0.5**	**92.24±1.0**	**83.13**
Qwen3-1.7B	LoRA ($r = 8$)	0.50%	49.83±1.1	76.97±0.2	90.62±0.2	83.27±0.2	76.24±0.3	86.28±0.3	87.91±0.4	81.71±0.9	84.58±0.6	79.71
	LoRA ($r = 16$)	1.00%	50.44±1.0	77.63±0.2	90.97±0.1	84.00±0.2	76.98±0.4	86.88±0.3	88.02±0.1	82.63±1.0	85.73±0.7	80.36
	LoRA ($r = 32$)	1.99%	51.96±0.7	78.29±0.1	92.18±0.2	84.27±0.2	77.55±0.4	87.04±0.1	88.25±0.1	83.45±0.7	87.50±0.5	81.17
	DoRA ($r = 16$)	1.04%	51.08±0.4	78.64±0.2	91.67±0.2	84.00±0.2	78.75±0.2	87.38±0.2	88.71±0.2	82.26±0.5	87.29±0.7	81.08
	TopLoRA($r = 8$)	0.74%	51.32±0.6	78.29±0.4	91.84±0.3	82.60±0.5	79.41±0.1	87.01±0.1	88.63±0.2	82.17±0.5	86.25±0.3	80.84
	TopLoRA($r = 16$)	1.47%	52.48±0.5	78.62±0.2	**92.54±0.5**	84.13±0.1	78.92±0.3	87.03±0.2	88.85±0.7	83.05±1.1	87.52±0.9	81.46
	ReMoLE	4.12%	53.46±0.6	79.60±0.4	91.75±0.2	84.80±0.2	79.44±0.6	86.37±0.2	88.00±0.1	82.12±0.4	83.74±0.4	81.03
	LD-MoLE	4.23%	**53.82±0.8**	78.67±0.4	92.11±0.3	85.00±0.2	79.30±0.4	86.72±0.4	88.71±0.2	82.61±0.7	87.72±1.4	81.63
	HyRA ($r = 8$)	0.70%	51.91±0.4	79.56±0.1	92.37±0.1	84.60±0.4	79.21±0.2	87.28±0.1	89.33±0.2	84.09±1.0	88.75±0.6	81.90
	HyRA ($r = 16$)	1.21%	53.48±0.7	**80.67±0.2**	92.46±0.3	**85.47±0.3**	**79.58±0.2**	**87.58±0.3**	**88.96±0.4**	**84.39±0.8**	**89.04±1.1**	**82.40**

configurations, HyRA ($r = 16$) achieves the highest average scores: 87.03 on LLaMA-3.1-8B, 83.13 on LLaMA-3.2-3B, and 82.40 on Qwen3-1.7B. HyRA ($r = 8$) is the second best on every base model. The advantage over MoELoRA-style methods is more pronounced. HyRA ($r = 16$) outperforms LD-MoLE by up to $+1.63$ with 3–4$\times$ fewer parameters. This indicates that adding capacity through input-dependent head-wise Householder modulation is more parameter-efficient than expanding LoRA capacity through multiple expert adapters with token-level routing.

The most direct conceptual baseline for HyRA is TopLoRA, which also introduces input-dependent modulation, but it does at the token level by diagonal scaling. At the same rank, HyRA outperforms TopLoRA in all three base models. Notably, HyRA ($r = 16$) achieves higher averages than TopLoRA ($r = 16$), including $+1.62$, $+1.24$ and $+0.94$ in LLaMA-3.1-8B, LLaMA-3.2-3B, and Qwen3-1.7B respectively. This suggests that respecting the multi-head structure is more effective for input-dependent LoRA modulation.

4.2 Computer Vision benchmarks

To further assess the effectiveness of hyRA on more complex vision tasks, we follow the experimental setup in FourierFT (Gao et al., 2024b) and conduct additional evaluations on ViT-Base and ViT-Large backbones. Specifically, we use a fixed rank $r = 16$ for LoRA and HyRA, and spectral coefficients $n = 40000$ for FourierFT. A grid search is conducted over $\{1e^{-4}, 2e^{-4}, 5e^{-4}, 1e^{-3}, 2e^{-3}\}$. The number of training epochs is fixed at 20. All experiments are repeated five times, with the mean and standard deviation across the runs reported in Table 2.

Results. Experiments show that HyRA consistently outperforms FourierFT and LoRA across computer vision benchmarks on both ViT-Base and ViT-Large. HyRA reaches 86.05% and 89.87% average accuracy while training only 0.86% and 0.59% of the parameters, outperforming FourierFT by $+3.77$ and $+2.02$, respectively. Moreover, HyRA approaches comparable results over FFT, demonstrating a strong accuracy–parameter efficiency trade-off.

Table 2: Comparison of HyRA and baseline methods on eight computer vision datasets using ViT-Base and ViT-Large backbones. Full Fine-tuning (FFT) and Linear Probing (LP) are taken from the original FourierFT paper (Gao et al., 2024b). In LP, only the classification head is fine-tuned.

Model	Method	Trainable Para.	OxfordPets	StanfordCars	CIFAR10	DTD	EuroSAT	FGVC	RESISC45	CIFAR100	Avg.
ViT-Base	LP	-	90.28±0.43	25.76±0.28	96.41±0.02	69.77±0.67	88.72±0.13	17.44±0.43	74.22±0.10	84.28±0.11	68.36
	FFT	100%	93.14±0.40	79.78±1.15	98.92±0.05	77.68±1.21	99.05±0.09	54.84±1.23	96.13±0.13	92.38±0.13	86.49
	LoRA	0.69%	93.21±0.21	46.68±0.38	98.68±0.17	75.15±0.33	98.41±0.18	28.16±0.64	92.70±0.21	92.31±0.29	78.16
	FourierFT	1.10%	93.57±0.42	60.83±0.49	98.38±0.10	76.90±0.41	98.68±0.17	43.04±0.97	94.30±0.31	**92.58±0.28**	82.28
	HyRA	0.86%	**94.00±0.20**	**78.87±0.51**	**98.86±0.05**	**78.24±0.42**	**98.91±0.04**	**52.20±0.98**	**94.89±0.22**	92.44±0.42	**86.05**
ViT-Large	LP	-	91.11±0.30	37.91±0.27	97.78±0.04	73.33±0.26	92.64±0.08	24.62±0.24	82.02±0.11	84.28±0.11	72.96
	FFT	100%	94.43±0.56	88.90±0.26	99.15±0.04	81.79±1.01	99.04±0.08	68.25±1.63	96.43±0.07	93.58±0.19	90.20
	LoRA	0.53%	94.52±0.51	74.22±0.32	99.01±0.07	81.65±0.41	98.59±0.12	43.18±0.81	94.91±0.11	**94.81±0.12**	85.11
	FourierFT	0.63%	**94.88±0.12**	83.64±0.60	98.98±0.03	**81.88±0.41**	98.46±0.03	56.28±0.43	95.33±0.11	93.41±0.13	87.85
	HyRA	0.59%	94.56±0.13	**87.34±0.52**	**99.24±0.04**	81.23±0.89	**98.87±0.09**	**67.57±1.53**	**95.86±0.18**	94.27±0.17	**89.87**

5 Ablation

To evaluate the effectiveness of our method, we conducted ablation studies on key components of HyRA. We also visualize the directions learned by HyRA to better understand its behavior. In addition, we assess the scalability of HyRA by varying the LoRA rank over {2,4,8,16,32,64}. Moreover, we evaluate the scalability of HyRA under different tuning granularity. We show that the head structure of multi-head attention provides the most natural substrate for HyRA's head-wise Householder modulation. Detailed ablation results on scalability are provided in the Appendix B.

5.1 Component Isolation Ablation

We isolate two key design choices of HyRA, namely the introduction of input-dependent $\mathcal{H}(x)$ in the LoRA latent space, and making this modulation head-wise. We conduct a 2×2 ablation in Table 3 to compare HyRA with vanilla LoRA, LoRA augmented with a single shared $\mathcal{H}(x)$, and a head-wise variant with static per-head transformations T_h replacing $\mathcal{H}_h(x)$. Each component alone provides a modest improvement over LoRA, as indicated by $+0.78$ from shared $\mathcal{H}(x)$, and $+0.57$ from head-wise static T_h, while combining both improves $+2.36$. This suggests that the two components are *complementary*, with head-wise structure providing the architectural granularity needed to capture heterogeneous adaptation and input dependence allowing the modulation to vary across tokens. Taken together, these results indicate that neither component alone is sufficient to fully realize HyRA's performance advantage.

5.2 Gating Alternative Ablation

To isolate the effect of Householder modulation, Table 4 compares the Householder transformation of HyRA with three coordinate affine alternatives, including scalar gating ($\alpha_h u$), diagonal gating ($g_h \odot u$), and FiLM-style affine modulation ($\gamma_h \odot u + b_h$) (Perez et al., 2018). Notably, All variants share head-wise B_h and an identical input-dependent generator and differ only in the choice of latent modulation operator. The trainable parameters are matched within 2% in all variants. The average accuracy of HyRA consistently outperforms other modulation forms on LLaMA-3.2-3B and Qwen3-1.7B. Moreover, we observe that augmenting diagonal gating with a bias term does not improve performance. In contrast, FiLM slightly underperforms diagonal gating. This suggests that the limitation of coordinate-wise affine modulation is structural as it lacks the ability to model directional transformations. It can confirm that our HyRA captures a more expressive form of head-specific adaptation than coordinate-wise scaling alone.

5.3 Visualization Analysis

To better understand the behavior of HyRA, we analyze the directions $w_h(x)$ learned by the generator and investigate whether they exhibit the head-wise differentiation that motivated our design in Section 1. We extract the Householder directions $w_h \in \mathbb{R}^r$ produced by HyRA on the ARC-Challenge validation set using LLaMA-3.1-8B. For each layer, we compute token-wise head-to-head similarity matrices and average them across all collected validation tokens. For comparison, we also apply the same procedure to the frozen per-head outputs $\bar{y}_h = W^h x$, which reflect the head structure

Table 3: Component ablation isolating two design choices in HyRA: (1) head-wise latent transformation, and (2) input-dependence of the modulation. The variant HyRA w/o $\mathcal{H}_h$ replaces the input-dependent Householder $\mathcal{H}_h(x)$ with a learnable static transformation $T_h \in \mathbb{R}^{r\times r}$ per head initialized to identity. Δ values are computed relative to LoRA based on Avg accuracy. Results are accuracy on ARC-C, OBQA, CSQA, SWAG, and RTE with LLaMA-3.2-3B.

Variant	Head-wise	Input-dep	ARC-C	OBQA	CSQA	SWAG	RTE	Avg.
LoRA	✗	✗	73.68	83.60	79.49	89.34	87.18	82.66
LoRA + shared $\mathcal{H}(x)$	✗	✓	74.34	83.33	80.39	91.11	88.02	83.44
HyRA w/o $\mathcal{H}_h$ (static T_h)	✓	✗	74.37	82.94	81.01	90.37	87.48	83.23
HyRA	✓	✓	75.54	84.53	82.72	91.51	90.82	85.02

Table 4: Gating alternative ablation isolating the design choice of the latent modulation operator. All variants share the same head-wise structure and input-dependent generator; they differ only in how the latent $u = Ax$ is modulated per head. Results are accuracy on ARC-C, OBQA, CSQA, SWAG and RTE with LLaMA-3.2-3B and Qwen3-1.7B.

Variant	Modulation Form	ARC-C	OBQA	CSQA	SWAG	RTE	Avg.
LLaMA-3.2-3B							
Scalar gating	$\alpha_h(x)\, u$	74.34	82.94	81.07	89.94	87.19	83.10
Diagonal gating	$g_h(x) \odot u$	74.05	84.13	81.33	91.16	88.10	83.75
FiLM	$\gamma_h(x) \odot u + b_h(x)$	73.68	83.53	80.80	90.93	88.29	83.45
HyRA	$u - \beta_h(x)\langle u, w_h(x)\rangle\, w_h(x)$	75.54	84.53	82.72	91.51	90.82	85.02
Qwen3-1.7B							
Scalar gating	$\alpha_h(x)\, u$	77.30	83.14	78.35	86.90	84.52	82.05
Diagonal gating	$g_h(x) \odot u$	78.46	83.40	78.35	87.06	86.14	82.68
FiLM	$\gamma_h(x) \odot u + b_h(x)$	78.12	83.33	78.07	87.32	85.21	82.41
HyRA	$u - \beta_h(x)\langle u, w_h(x)\rangle\, w_h(x)$	79.56	84.60	79.21	87.28	88.75	83.88

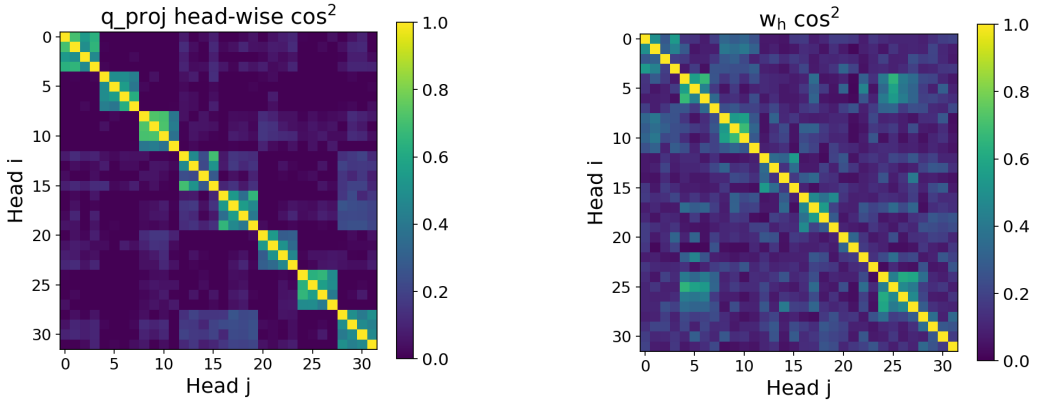
inherited from pretraining. We use squared cosine similarity, $\cos^2(v_i, v_j) = (v_i^\top v_j)^2/(\|v_i\|^2\|v_j\|^2)$, which measures alignment up to sign.

Figure 4(a) visualizes the head-to-head similarity of frozen outputs $\bar{y}_h$ at q-projection of the last Transformer layer. The heatmap exhibits a pronounced 4×4 block-diagonal structure: heads within each block are highly correlated, while heads across blocks are nearly orthogonal. This pattern is an artifact of Grouped Query Attention (Ainslie et al., 2023), where four query heads share a single key-value head and thus produce strongly correlated attention outputs.

In contrast, Figure 4(b) shows the corresponding similarity of learned directions w_h on the same layer for our HyRA. The block structure is preserved coarsely but noticeably less rigid within blocks. Heads share key-value computations in the frozen architecture are assigned distinct adaptation directions in latent space. Importantly, the learned directions do not collapse to a single globally shared axis. This shows that adaptation is head-aware. At the same time, they are not fully independent across heads, which is expected given the structural consequence of the low-rank latent bottleneck, such as $r = 8$ vectors in 32 heads can not be mutually orthogonal. Therefore, HyRA reparameterizes pretrained head families into more flexible low-rank editing directions. Combined with the consistent gain over LoRA in Table 3 and over coordinate-wise gating alternatives in Table 4, this suggests that HyRA indeed captures head-specific adaptation signals.

6 Related Work

Parameter-Efficient Fine-Tuning (PEFT) aims to adapt Large language models (LLMs) to downstream tasks by tuning only a small subset of parameters, significantly reducing computational and memory costs. Low-rank adaptation (LoRA) (Hu et al., 2022) fine-tunes a pretrained model by learning a low-rank update $\Delta\mathbf{W} = \mathbf{BA}$ for each frozen weight matrix. Optimization-centric methods focus on improving initialization and training dynamics. For example, DoRA (Liu et al., 2024b) decomposes pretrained weights into a magnitude vector and a direction matrix, employing LoRA for directional updates. PiSSA (Meng et al., 2024) initializes low-rank factors using the principal singular components. LoRA-GA (Wang et al., 2024a) aligns the gradients of low-rank matrix product with those of full fine-tuning at the first step. Linear structural variants attempt to alleviate



(a) Frozen head-wise outputs $\bar{y}_h = W^h x$.

(b) HyRA learned directions w_h.

Figure 4: **Per-head similarity at q-projection of last layer on Llama3.1-8B.** Squared cosine $\cos^2(\cdot, \cdot)$ between heads, computed on ARC-Challenge validation dataset. **(a)** Frozen per-head outputs $\bar{y}_h$ exhibit a 4×4 block-diagonal pattern induced by Grouped Query Attention, where each KV head is shared across four query heads. **(b)** HyRA's learned Householder directions w_h largely break this structure, assigning distinct adaptation directions to heads that share key-value computations.

the rank bottleneck without increasing trainable parameters. For example, AdaLoRA (Zhang et al., 2023) dynamically adjusts the rank of adapter matrices among layers according to a sensitivity based importance score. HiRA (Huang et al., 2025) combines low-rank updates with pretrained weights via a Hadamard product, and KronA (Edalati et al., 2025) replaces LoRA's matrix product with a Kronecker product. TopLoRA (Li et al., 2025) introduces a token-wise diagonal matrix to modulate low-rank updates. LoRA-XS (Bałazy et al., 2024) trains a small intermediate matrix between frozen low-rank factors derived from pretrained weights via SVD.

Beyond linear structural variants, another line of work enhances LoRA capacity through mixture-of-experts (Jacobs et al., 1991; Jordan and Jacobs, 1994) designs. MoELoRA-style methods combine multiple LoRA adapters as experts and learn a router to aggregate them. Several variants have since been propose. LoRAMoE (Dou et al., 2023) introduces MoEstyle plugins to enhance downstream performance while mitigating knowledge forgetting. MoLA (Gao et al., 2024a) analyzes expert allocation patterns across layers. ReMoLE, which adapts the ReLU-based routing from ReMoE (Wang et al., 2024b), and LD-MoLE (Zhuang et al., 2025) replace the non-differentiable TopK selection with a differentiable routing function while regulating the number of activated experts.

In this paper, we apply the polar decomposition to $\mathbf{B}$ and $\mathbf{A}$ and obtain $\Delta\mathbf{W} = \mathbf{U}_B\mathbf{M}\mathbf{V}_A^\top$ with $\mathbf{M} = \mathbf{H}_B\mathbf{H}_A$. This formulation reveals that all attention heads are constrained to share the output of a single latent operator $\mathbf{M}$. These LoRA variants do not explicitly consider the underlying multi-head structure. Although LoRA-XS and TopLoRA introduce intermediate matrices in decompositions of the form $\mathbf{BMA}$ and MoELoRA-style methods use token-level expert routing, none explicitly models the distinct roles of attention heads. In contrast, our analysis identifies the shared latent operator $\mathbf{M}$ as the expressiveness limitation of LoRA under multi-head attention. Based on this insight, HyRA introduces head-wise projections to better align adaptation with the native structure of the model and thereby improve the expressiveness of LoRA. Moreover, experimental results show that their performance does not match that of HyRA.

7 Conclusion

In this work, we identify the shared latent space of LoRA as a structural bottleneck under multi-head attention that limits expressiveness. To solve this limitation, we introduced **HyRA**, a parameter-efficient fine-tuning method that modifies this shared latent space by learning the head-wise projection. Specifically, by adjusting LoRA weights with head-wise, input-dependent Householder transformation, HyRA can have finer-grained adaptation, even with a limited rank and targted module settings. Across diverse model architectures and downstream benchmarks, HyRA consistently achieves the highest accuracy among PEFT baselines, while maintaining strong parameter efficiency.

AI USE STATEMENT

Generative AI was used to polish the manuscript, mainly to improve grammar, wording, and readability. The edits helped make the writing clearer while keeping the original meaning of technical content. It should be noted that generative AI was not extended to the research ideas, methodology and experiment design. These parts were carried out independently by the authors. AI was also not involved in the analysis of the data or interpreting the results. All AI-assisted revisions were checked to ensure that they did not alter the scientific claims. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work follows the ICLR Code of Ethics. No human participants or animal experiments were involved, and no personally identifiable information was used. All datasets and research resources were used with their applicable licenses and usage guidelines. We recognize the importance of considering potential bias, privacy risks, and misuse when applying the proposed methods. We report our methods and findings transparently, acknowledge prior work, and describe the limitations of our study. The authors take full responsibility for the integrity of the research and the claims presented in this paper.

REPRODUCIBILITY STATEMENT

To support reproducibility, we describe the proposed method and experimental procedures in detail in the paper and appendix. We provide the dataset descriptions, evaluation metrics, implementation details, and key hyperparameter settings needed to reproduce the experiments. We also specify the computing environment and hardware used. These details are intended to help readers implement the method and verify the reported results. We will publicly release the code to support reproduction of the experiments.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

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.

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, 2023.

Seungju Back, Dongwoo Lee, Naun Kang, Taehee Lee, SK Hong, Youngjune Gwon, and Sungjin Ahn. Understanding lora as knowledge memory: An empirical analysis. *arXiv preprint arXiv:2603.01097*, 2026.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. Lora-xs: Low-rank adaptation with extremely small number of parameters. *arXiv preprint arXiv:2405.17604*, 2024.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *arXiv preprint arXiv:2312.09979*, 4(7), 2023.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Krona: Parameter-efficient tuning with kronecker adapter. In *Enhancing LLM Performance: Efficacy, Fine-Tuning, and Inference Techniques*, pages 49–65. Springer, 2025.

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*, 2024a.

Chongyang Gao, Kezhen Chen, Jinmeng Rao, Ruibo Liu, Baochen Sun, Yawen Zhang, Daiyi Peng, Xiaoyuan Guo, and VS Subrahmanian. Mola: Moe lora with layer-wise expert allocation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5097–5112, 2025.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform. *arXiv preprint arXiv:2405.03003*, 2024b.

Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU press, 2013.

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.

Riccardo Grazzi, Julien Siems, Arber Zela, Jorg KH Franke, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. In *13th International Conference on Learning Representations Iclr 2025*, pages 1–33. ICLR, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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

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

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.

Samy Jelassi, David Brandfonbrener, Sham M Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. *arXiv preprint arXiv:2402.01032*, 2024.

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

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *2023 IEEE/CVF international conference on computer vision (ICCV)*, pages 3992–4003. IEEE, 2023.

Anirban Laha, Saneem Ahmed Chemmengath, Priyanka Agrawal, Mitesh Khapra, Karthik Sankaranarayanan, and Harish G Ramaswamy. On controllable sparse alternatives to softmax. *Advances in neural information processing systems*, 31, 2018.

Shiwei Li, Xiandi Luo, Haozhao Wang, Xing Tang, Ziqiang Cui, Dugang Liu, Yuhua Li, Xiuqiang He, and Ruixuan Li. Beyond higher rank: Token-wise input-output projections for efficient low-rank adaptation. *arXiv preprint arXiv:2510.23123*, 2025.

Shiwei Li, Xiandi Luo, Haozhao Wang, Xing Tang, Ziqiang Cui, Dugang Liu, Yuhua Li, Yichen Li, Ruixuan Li, et al. Bora: towards more expressive low-rank adaptation with block diversity. In *International Conference on Learning Representations*, volume 2026, pages 143975–143991, 2026.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. Relora: High-rank training through low-rank updates. In *International Conference on Learning Representations*, volume 2024, pages 49405–49421, 2024.

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*, 2024a.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024b.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, Benjamin Bossan, and Marian Tietz. PEFT: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, 2022.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2024.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pages 28085–28128, 2025.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Increasing the expressivity of deltanet through products of householders. *arXiv e-prints*, pages arXiv–2502, 2025.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. *Advances in Neural Information Processing Systems*, 37:54905–54931, 2024a.

Ziteng Wang, Jun Zhu, and Jianfei Chen. Remoe: Fully differentiable mixture-of-experts with relu routing. *arXiv preprint arXiv:2412.14711*, 2024b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Songlin Yang, Yikang Shen, Kaiyue Wen, Shawn Tan, Mayank Mishra, Liliang Ren, Rameswar Panda, and Yoon Kim. Path attention: Position encoding via accumulating householder transformations. *arXiv preprint arXiv:2505.16381*, 2025b.

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.

Ziyu Zhao, Yixiao Zhou, Xin Yu, Zhi Zhang, Didi Zhu, Tao Shen, Zexi Li, Jinluan Yang, Xuwu Wang, Jing Su, et al. Each rank could be an expert: Single-ranked mixture of experts lora for multi-task learning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1998–2009, 2026.

Yuan Zhuang, Yi Shen, Yuexin Bian, Qing Su, Shihao Ji, Yuanyuan Shi, and Fei Miao. Ld-mole: Learnable dynamic routing for mixture of lora experts. *arXiv preprint arXiv:2509.25684*, 2025.

A Theoretical Analysis

In this section, we first analyze the head-dependent effect of latent directions, then discuss why this motivates Householder-like adaptation, and finally clarify its relationship to low-rank adaptation.

A.1 Head-Dependent Directional Sensitivity

We first examine how the effect of a latent direction depends jointly on the shared representation and the head-specific readout.

Proposition 1. For a given head h, a fixed unit direction $\mathbf{w}$, and a fixed scalar β, $G_h(\mathbf{w}) := \mathbb{E}\big[(\mathbf{w}^\top \mathbf{u})^2 \, \|\mathbf{B}_h \mathbf{w}\|_2^2\big]$, where $\mathbf{u} = \mathbf{A}x$. Under the modulation $\mathcal{H}_\beta(\mathbf{w}) = \mathbf{I} - \beta \mathbf{w}\mathbf{w}^\top$, the expected squared output perturbation of head h is given by $\mathbb{E}[\|\delta y_h\|_2^2] = \beta^2 G_h(\mathbf{w})$.

Proof. Given $\mathcal{H}_\beta(\mathbf{w})\mathbf{u} = \mathbf{u} - \beta(\mathbf{w}^\top \mathbf{u})\mathbf{w}$, the output change of head h is $\delta y_h = \mathbf{B}_h \mathcal{H}_\beta(\mathbf{w})\mathbf{u} - \mathbf{B}_h \mathbf{u} = -\beta(\mathbf{w}^\top \mathbf{u})\mathbf{B}_h \mathbf{w}$, which shows that the transformation modifies only the component of $\mathbf{u}$ along the direction $\mathbf{w}$, with the strength controlled by β. Taking the squared Euclidean norm and averaging over inputs, we obtain

$$\mathbb{E}[\|\delta y_h\|_2^2] = \beta^2 \mathbb{E}\big[(\mathbf{w}^\top \mathbf{u})^2 \|\mathbf{B}_h \mathbf{w}\|_2^2\big] = \beta^2 G_h(\mathbf{w}), \tag{7}$$

where $G_h(\mathbf{w})$ measures the effect of directional modulation on the output of head h, and β controls its strength. Consequently, for the fixed nonzero β, directions with larger $G_h(\mathbf{w})$ receive stronger optimization. □

This proposition shows that the same latent direction can have different optimization effects across heads due to their distinct readouts $\mathbf{B}_h$, motivating head-dependent directional adaptation.

A.2 Why is Householder used for head adaptation?

According to Proposition 1, Householder-like modulation offers three advantages. First, $G_h(\mathbf{w})$ depends explicitly on $\mathbf{B}_h$, different heads favor different directions, allowing each head to learn updates aligned with its own functional role. Second, it provides better interpretability: the learned direction $\mathbf{w}$ reflects both data statistics through $(\mathbf{w}^\top \mathbf{u})^2$ and head sensitivity through $\|\mathbf{B}_h \mathbf{w}\|_2^2$, making it a meaningful and interpretable direction of adaptation. Third, it improves parameter efficiency: for each head, the update is concentrated on a small number of directions with large $G_h(\mathbf{w})$, thereby capturing the most significant variations under a limited parameter budget.

A.3 Comparison with Low-Rank Adaptation

HyRA introduces a per-head dynamic operator $\mathcal{H}_h(x)$, extending the shared projection to head-specific transformations. We expand $\mathcal{H}_h(x)$ and analyze how it affects the latent representation. With the Householder-like parameterization $\mathcal{H}_h(x) = \mathbf{I} - \beta_h(x)\, \mathbf{w}_h(x)\mathbf{w}_h(x)^\top$, let $\mathbf{u} = \mathbf{A}x$, the output can be decomposed as

$$\Delta \mathbf{W}^h x = \mathbf{B}_h \big(\mathbf{u} - \beta_h(x) \langle \mathbf{u}, \mathbf{w}_h(x) \rangle \mathbf{w}_h(x)\big). \tag{8}$$

This decomposition shows that HyRA performs a directional transformation in the latent space. Specifically, the projection of $\mathbf{u}$ onto $\mathbf{w}_h(x)$, given by $\langle \mathbf{u}, \mathbf{w}_h(x) \rangle \mathbf{w}_h(x)$, is rescaled by $(1 - \beta_h(x))$, while the orthogonal component is preserved. Importantly, $\mathbf{w}_h(x)$ is dynamically generated from the input, enabling each head to identify token-specific components that should be adjusted for its transformation. Compare to HyRA, standard LoRA uniformly processes the full latent vector u with a single shared operator and lacks direction-wise control. When $\beta_h(x) \equiv 0$, HyRA applies no directional adjustment and reverts to standard LoRA.

It worth noting that MoELoRA-style methods, such as ReMoLE and LD-MoLE, achieve input adaptivity by routing each input x to multiple LoRA experts with independent low-rank updates. However, they rely on additional adapters and routers, introducing extra parameters while not directly addressing the limitation of the shared latent transformation. TopLoRA employs a diagonal matrix Σ_x as a learned token-wise gate over latent dimensions. However, such diagonal modulation scales feature dimensions, lacks interactions between feature directions, and ignores the intrinsic multi-head structure by treating all heads uniformly. Unlike the above methods, HyRA directly modulates

the latent representation and constructs a distinct operator for each head, enabling adaptation beyond independent feature scaling.

B Scalability Analysis

B.1 Rank Ablation

We study how HyRA scales with the underlying LoRA rank r. Table 5 reports HyRA's accuracy on LLaMA-3.2-3B with $r \in \{2, 4, 8, 16, 32, 64\}$. The average accuracy improves monotonically with r. HyRA continues to improve up to $r = 64$, suggesting that the Householder modulation effectively utilizes the additional latent capacity without saturation in the rank range we study.

Table 5: The accuracy of HyRA with varying ranks using Llama3.2-3B model.

	Method	MMLU_P	ARC-C	ARC-E	OBQA	CSQA	SWAG	Hella	CoLA	RTE	Avg
Llama3.2-3B	HyRA($r = 2$)	43.44	73.16	87.06	82.73	80.39	89.14	92.29	84.75	87.50	80.05
	HyRA($r = 4$)	43.83	74.01	87.12	83.21	81.26	90.48	91.45	84.99	87.89	80.47
	HyRA($r = 8$)	47.52	75.54	89.78	84.53	82.72	91.51	93.99	85.31	90.82	82.42
	HyRA($r = 16$)	48.63	76.95	90.23	85.20	82.74	91.34	94.50	87.17	92.24	83.13
	HyRA($r = 32$)	49.84	77.37	90.66	85.73	82.85	91.55	94.71	86.72	92.29	83.52
	HyRA($r = 64$)	50.86	77.93	90.93	85.82	84.31	91.63	94.04	87.98	93.19	84.08

B.2 Target Modules Abalation

We next investigate how HyRA's performance depends on the choice of adapted modules. Table 6 reports the accuracy of HyRA on LLaMA-3.2-3B with $r = 8$ as the target set is progressively expanded from Q alone to QKVOGUD. Here, G, U, and D denote the gate, up, and down projections in the FFN, respectively.

The average accuracy improves monotonically as more modules are adapted, increasing from 79.14 for Q to 79.87 for QK, 81.11 for QKV, 82.07 for QKVO, and 82.42 for QKVOGUD. Each additional attention projection has a meaningful improvement, with the value projection contributing the largest single gain of $+1.24$ from QK to QKV. Adapting all four attention projections in QKVO already achieves 81.84, recovering most of HyRA's full performance of 82.42 without adapting the FFN modules. Adding the three FFN projections G, U, and D further improves the average accuracy by $+0.35$, suggesting that HyRA's head-wise modulation extends beyond attention to neuron-grouped FFN modules, although with smaller gains per module. This trend is consistent with the motivation of HyRA, since multi-head attention provides the most natural structural substrate for head-wise Householder modulation, whereas FFN modules, whose heads correspond only loosely to neuron groups, exhibit weaker architectural alignment and therefore smaller gains.

Table 6: The accuracy of HyRA with varying target modules selection using Llama3.2-3B model.

	Method	MMLU_P	ARC-C	ARC-E	OBQA	CSQA	SWAG	Hella	CoLA	RTE	Avg
Llama3.2-3B	HyRA(Q)	43.87	74.01	87.85	79.39	77.86	90.16	92.14	85.10	81.88	79.14
	HyRA(QK)	44.21	74.43	86.63	81.25	77.69	90.40	92.66	84.19	87.39	79.87
	HyRA(QKV)	45.62	75.00	89.24	82.34	79.49	91.14	93.41	85.29	88.41	81.11
	HyRA(QKVO)	47.98	74.84	89.46	84.01	81.90	91.29	93.57	85.49	89.70	82.07
	HyRA(QKVOGUD)	47.52	75.54	89.78	84.53	82.72	91.51	93.99	85.31	90.82	82.42

C Datasets

In this section, we describe the datasets and experimental setup in more detail. Each dataset consists of training, validation, and test splits. All experiments are conducted by training on the training split and evaluating on the validation split, while the test split is not used.

Table 7: Shared hyperparameter grid swept for all methods in NLP benchmarks.

Hyperparameter	Search range
Cutoff Length	1024
Learning rate	{1e-4, 2e-4, 4e-4, 5e-4}
Weight decay	0.01
Target Modules	All
Optimizer	AdamW
LR scheduler	Cosine decay
Batch size	16
Gradient accumulation steps	1
Warm-up ratio	{0, 0.01}
Precision	bf16

MMLU-Pro: As an extension of MMLU, MMLU-Pro increases the difficulty of evaluation by expanding the number of answer choices from 4 to 10. It features more challenging, reasoning-intensive questions and filters out trivial and noisy items. The stronger performance of chain-of-thought reasoning on this benchmark further highlights its emphasis on deeper reasoning. The dataset contains 12,102 samples.

ARC (AI2 Reasoning Challenge): This benchmark evaluates grade-school science question answering and consists of two subsets: ARC-Easy, which focuses on relatively straightforward questions, and ARC-Challenge, which involves more complex reasoning and deeper domain knowledge. ARC-Easy contains 2,251 examples in the training split, 570 in the validation split, and 2,376 in the test split. ARC-Challenge contains 1,119 examples in the training split, 299 in the validation split, and 1,172 in the test split.

CommonsenseQA: CommonsenseQA is a five-way multiple-choice question answering benchmark designed to evaluate a model's ability to apply diverse forms of commonsense knowledge. It consists of 12,102 questions, each associated with one correct answer and four distractors. The dataset is split into 9,741 training samples, 1,221 validation samples, and 1,140 test samples.

OpenBookQA: OpenBookQA is a four-way multiple-choice question answering benchmark that evaluates both scientific knowledge and language understanding in the setting of complex question answering. Modeled after open-book exams, it provides a collection of scientific facts that must be combined with broader commonsense knowledge to answer each question. In contrast to simple fact-retrieval tasks, OpenBookQA emphasizes multi-step reasoning, external knowledge integration, and deeper text comprehension. It consists of 4,957 training samples, 500 validation samples, and 500 test samples.

SWAG: This benchmark evaluates commonsense reasoning through a task in which the model must select the most plausible continuation of a given scenario. Each example is presented as a four-way multiple-choice question, consisting of one correct answer and three adversarially generated distractors. It includes 73,546 training samples, 20,006 validation samples, and 20,005 test samples.

HellaSWAG: This benchmark is designed to evaluate a model's ability to complete sentences in a coherent and contextually appropriate manner. Similar to SWAG, each example is presented as a four-way multiple-choice question, where the task is to select the most plausible continuation. The challenge goes beyond surface-level pattern matching and requires an understanding of meaning, context, and commonsense reasoning. Although this task is trivial for humans with rich real-world and linguistic experience, it remains challenging for machines. The dataset contains 39,900 training samples, 10,000 validation samples, and 10,000 test samples.

CoLA (Corpus of Linguistic Acceptability): This dataset is part of the General Language Understanding Evaluation (GLUE) benchmark and contains 10,657 sentences collected from 23 linguistics publications, each labeled for grammatical acceptability by the original authors. Its public release provides 9,594 sentences for training and development, with 1,063 test sentences held out.

RTE (Recognizing Textual Entailment): This benchmark is part of the General Language Understanding Evaluation (GLUE) suite and consists of a series of annual entailment challenges. Its

examples are drawn from news and Wikipedia text. All datasets are reformulated as a binary classification task of entailment versus not entailment, yielding 2,490 training samples, 277 validation samples, and 3,000 test samples.

Table 8: Method-specific hyperparameters in NLP benchmarks. Values without a search range are fixed across all experiments.

Method	**Hyperparameter**	**Value**
LoRA / DoRA / TopLoRA / HyRA	LoRA scaling α	$2r$
	LoRA dropout	0.1
TopLoRA	Diagonal Matrix Optimization	Exp+RMSNorm
HyRA	Generator hidden width	64
	Householder magnitude range	$\beta_h \in (0, 2)$
	Generator activation	GELU
ReMoLE	Number of experts per layer	8
	ReLU-based routing	follows(Wang et al., 2024b)
LD-MoLE	Number of experts per layer	8
	Sparsegen routing	follows (Zhuang et al., 2025)
	Sparsity regularization coef	follows (Zhuang et al., 2025)

D Implementation Details for Head-wise Heterogeneity-Based Margin Gain Analysis

For each example, we define margin gain as the change, relative to the frozen pretrained model, in the logit margin between the correct option and the highest-scoring incorrect option. To characterize head-wise heterogeneity, we compute the gradient of the task loss with respect to the pre-RoPE query activations of each query head and map these gradients back to the shared input feature space using the corresponding query projection matrix (W_Q). For each analyzed layer, heterogeneity is measured by the normalized dispersion of the mapped head-wise gradients around their mean. The per-example head attention heterogeneity (HAD) score is obtained by averaging the valid layer-wise heterogeneity scores. For Llama-3.2-3B, we use all 24 query heads and analyze layers 7, 14, 21, and 28.

For Figure 2, we randomly sample 500 examples without replacement from each dataset using seed 2026. Within each dataset, examples are ranked by HAD and divided into 20 equal-sized bins of 25 examples. For each fine-tuning method, we compute the mean margin gain within each bin and then take an equal-weight average of the corresponding bin means across the two datasets, so that each plotted point summarizes 50 examples. Finally, a quadratic polynomial is fitted to the 20 bin-level means to visualize the descriptive trend of margin gain across increasing HAD quantiles. All fine-tuning methods are trained for 10 epochs using AdamW with training seed 42. The fitted curves are intended solely as descriptive summaries and do not constitute statistical significance tests.

E Hyperparameter Grids and Tuning Protocol

This section reports the hyperparameter search space used for HyRA and all baselines, along with the final settings selected for the results in Tables 1 and 3–4. Half-precision computation is used to improve memory efficiency and accelerate the training process. Following (Zhuang et al., 2025), HyRA and the baseline methods were applied to all target modules, including Q, K, V, O, Up, Gate, and Down projections, where we extend HyRA to architectures without an explicit head structure. For FFN modules, which lack an inherent head structure, we partition the output dimension into 32 uniform groups for HyRA's head-wise modulation. The same target modules were adapted for all methods. In all experiments, we set the scaling factor of LoRA to $\alpha = 2r$, where r is the LoRA rank.

E.1 HYPERPARAMETER AND TRAINING SETUP FOR NLP BENCHMARKS

To ensure fair comparison, all LoRA variants: HyRA, LoRA, DoRA, and TopLoRA follow the same training pipeline and hyperparameter selection procedure on each task. For each method, we sweep the same hyperparameter grid (Table 7), select the best configuration based on validation accuracy, and report test accuracy at the end of training. All MoELoRA-style variants: ReMoLE, and LD-MoLE follow the (Zhuang et al., 2025). Method-specific hyperparameters inherited from prior work (e.g., LoRA α, MoE expert count) follow each baseline's published defaults.

Beyond the shared grid, each method has internal hyperparameters that follow each baseline's published recommendations as shown in Table 8.

E.2 HYPERPARAMETER AND TRAINING SETUP FOR COMPUTER VISION BENCHMARKS

Table 9: Hyperparameters and computing resources used in the vision tasks.

Model	Hyperparameter	OxfordPets	StanfordCars	CIFAR10	DTD	EuroSAT	FGVC	RESISC45	CIFAR100
	Optimizer	AdamW							
	Weight Decay	0.01							
	LR Schedule	Linear							
	Epoch	20							
BASE	# GPUs	8							
	Learning Rate (Head)	1E-3	5E-4	5E-4	2E-3	2E-4	1E-3	1E-3	1E-4
	Rank	$r = 16$							
	Batch Size Per GPU	128							
LARGE	# GPUs	8							
	Learning Rate (Head)	2E-3	1E-4	5E-4	2E-3	2E-4	1E-2	2E-3	5E-4
	Rank	4							
	Batch Size Per GPU	32							