FORWARD-ONLY LORA-MOE ALLOCATION VIA MODULE ACTIVATION GEOMETRY

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank adaptation (LoRA) mixtures of experts (LoRA-MoE) increase adaptation capacity through conditional expert activation. However, many conventional LoRA-MoE methods adopted in large language models (LLMs) typically restrict MoE placement to predefined modules, such as the feed-forward networks (FFNs), and use uniform expert configurations across them. This design overlooks that different modules may need different computational capacity for a downstream task. For a more adaptive and more efficient solution, we introduce BLAde (*Budget-constrained LoRA Mixture-of-Experts Adaptive Allocation*), a task-adaptive framework that jointly determines where MoE capacity should be introduced and how the available factor budget should be allocated across LoRA modules. Using lightweight statistics collected from forward activations, BLAde identifies promising modules for conditional expert specialization, adaptively determines their number of experts and top-k routing configurations, and reallocates LoRA ranks across the remaining modules under a factor budget. In this way, BLAde reformulates LoRA-MoE design from a manually specified, architecture-wide configuration into a task-adaptive and budget-aware allocation problem, allocating conditional capacity according to module activation geometry.

1 INTRODUCTION

Low-rank adaptation (LoRA) (Hu et al., 2022) reduces the cost of fine-tuning large language models (LLMs) by learning low-rank updates while keeping pretrained weights frozen. Rank-adaptive methods dynamically allocate this capacity across target modules (Zhang et al., 2023b;a; Valipour et al., 2023; Paischer et al., 2025), while LoRA mixture-of-experts methods introduce conditional specialization by routing different inputs to different low-rank experts. However, expert placement is typically determined by architecture-level heuristics, with routed adapters assigned to predefined module types (Li et al., 2024). Such a placement rule fixes which modules can use conditional specialization before considering the adaptation needs of a downstream task. Although expert counts and ranks can be adjusted (Qing et al., 2024; Chen et al., 2025), adjusting capacity within a prescribed placement does not by itself determine whether each module needs routed experts. This motivates selecting MoE placement from task-dependent evidence, allowing individual modules to receive conditional capacity according to their adaptation needs.

Prior work shows that adaptation needs to be different across modules: AdaLoRA and EVA allocate heterogeneous ranks (Zhang et al., 2023b; Paischer et al., 2025), while MoLA and AlphaLoRA vary expert allocation across layers (Gao et al., 2025; Qing et al., 2024). This motivates jointly considering shared adaptation and conditional specialization under a fixed budget. A rank-r LoRA uses $r(d_i + d_o)$ parameters, whereas E rank-r_e experts use $Er_e(d_i + d_o)$. Thus, allocating more capacity to routed modules reduces the rank available elsewhere, while a fixed parameter budget for a routed module can be divided among more experts, each with a lower rank. MoE placement, expert configuration, and LoRA ranks should therefore be optimized jointly.

We introduce BLAde (*Budget-constrained LoRA Mixture-of-Experts Adaptive Allocation*), which makes these decisions using a short forward-only calibration pass. Our central premise is that deciding whether a module needs token-specific experts and estimating the rank needed by a shared adapter require different statistics. The variance of normalized outputs indicates how much a module may benefit from token-specific experts. We also combine selected input and output coordinates

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.

from each token into a compact feature vector, whose covariance effective rank helps guide rank allocation. BLAde uses response variance to select routed modules and assign expert counts. It uses effective rank to determine MoE eligibility, set the number of experts activated per token, and redistribute LoRA ranks. The budget counts the parameters in the LoRA input and output factors across all modules and experts; router parameters are counted separately. We motivate these statistics theoretically and evaluate their connection to gradient geometry empirically. We evaluate BLAde on reasoning and generation tasks across diverse LLMs. BLAde outperforms LoRA and other baselines without substantial additional overhead.

2 Related Work

We review three lines of related work: adaptive-rank LoRA, focusing on criteria for redistributing low-rank capacity; LoRA-MoE, focusing on MoE architectures and allocation; and data-dependent adaptation, focusing on activation- and gradient-based signals for guiding adaptation.

Adaptive-Rank LoRA. Adaptive-rank LoRA methods adapt rank allocation or support multiple ranks within one model. AdaLoRA (Zhang et al., 2023b) dynamically prunes singular components, DyLoRA (Valipour et al., 2023) trains nested low-rank representations that support multiple ranks, IncreLoRA (Zhang et al., 2023a) progressively expands adapter capacity with rank-one components, and EVA (Paischer et al., 2025) allocates ranks according to the explained variance of input activations. Methods that redistribute ranks use criteria such as singular-component importance or activation variance to estimate module-wise adaptation demand. However, LoRA ultimately parameterizes weight updates, and these scores do not directly indicate which updates reduce the downstream loss. Instead, gradients provide this information locally, making gradient statistics a useful reference for adaptation-capacity allocation.

LoRA-MoE Methods. LoRA-MoE extends low-rank adaptation with conditional expert routing while keeping the backbone frozen. Early methods explore different expert structures: MoLE (Wu et al., 2024) composes separately trained LoRA branches, MixLoRA (Li et al., 2024) introduces LoRA experts into feed-forward blocks, and HydraLoRA (Tian et al., 2024) shares input projections across routed branches. Subsequent methods adapt expert configurations through layer-wise expert counts (Gao et al., 2025; Qing et al., 2024), adaptive routing (Liu & Luo, 2024), rank-aware experts (Zhang et al., 2025a), or joint expert-count and rank allocation (Chen et al., 2025). However, most methods either retain predefined MoE placements or optimize expert configurations only after deciding where mixtures are introduced (Chen et al., 2024; Luo et al., 2024; Zadouri et al., 2024; Tian et al., 2024).

Data-Dependent Adaptation Geometry. Data-dependent adaptation methods use activations or gradients to guide parameter-efficient fine-tuning. FILet (Feng et al., 2026) and LoRA-DA (Zhang et al., 2026) approximate Fisher information with K-FAC (Martens & Grosse, 2015), while LoRA-GA (Wang et al., 2024), GoRA (haonan he et al., 2025), and LoRA-One (Zhang et al., 2025b) use gradients to identify task-relevant update directions. While informative, these backward-dependent signals introduce additional calibration overhead. In contrast, EVA (Paischer et al., 2025) uses forward activation SVD and explained variance to initialize LoRA and redistribute ranks without gradients. Motivated by this, BLAde adopts a forward-only strategy, using activation geometry to guide conditional specialization and capacity allocation across modules.

3 Method

Figure 1 provides an overview of BLAde. Given a pretrained model and a calibration set, we profile every candidate module using only forward activations. BLAde then uses two complementary scores to allocate a fixed factor budget: variance v_l governs conditional specialization, while normalized effective rank q_l guides MoE eligibility, active expert counts, and ordinary LoRA ranks. The resulting heterogeneous architecture assigns each module either an ordinary LoRA adapter or a token-routed LoRA-MoE configuration. Appendix A summarizes the notation, and Appendix B presents the allocation rules and Algorithm 1.

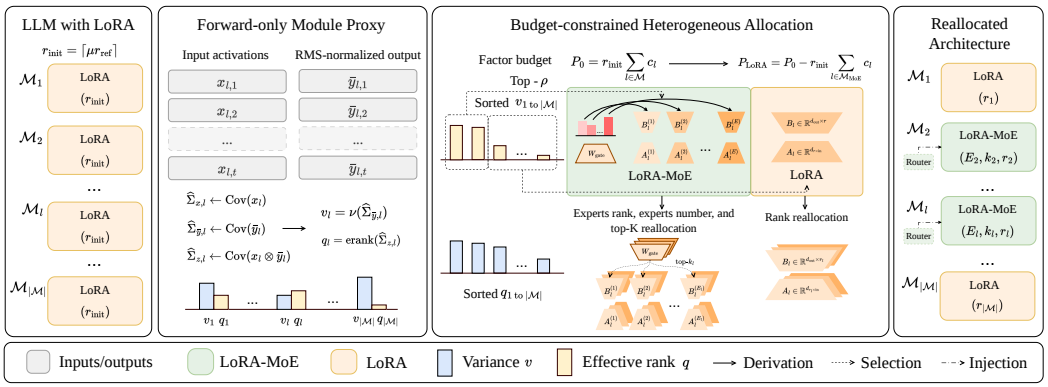


Figure 1: Overview of BLAde. Our framework consists of two stages. (a) *Forward-only Module Proxy* collects activations from a small calibration set and computes two complementary module-wise statistics: variance v_l and marginal-normalized joint effective rank q_l. (b) *Budget-Constrained Heterogeneous Allocation* uses these statistics to jointly determine the adapter architecture and capacity of each module under a fixed factor budget. Here, l and t index modules and tokens; $\boldsymbol{x}_{l,t}$ and $\bar{\boldsymbol{y}}_{l,t}$ denote the input and RMS-normalized output activations; and E_l, k_l, and r_l denote the number of experts, the number of active experts, and the allocated rank, respectively.

3.1 Problem Formulation

Let $\mathcal{M}$ denote the set of target linear modules. For each $l \in \mathcal{M}$, the frozen weight is $\boldsymbol{W}_l \in \mathbb{R}^{d_{o,l} \times d_{i,l}}$, and each unit increase in LoRA rank adds $c_l = d_{i,l} + d_{o,l}$ parameters. A uniform rank-r_{ref} LoRA baseline defines the shared factor budget, while m_l denotes whether the adapter of module l is converted to a routed configuration with E_l rank-$r_{e,l}$ experts:

$$P_0 = r_{\text{ref}} \sum_{l \in \mathcal{M}} c_l, \qquad \sum_{l \in \mathcal{M}} c_l \left[m_l E_l r_{e,l} + (1 - m_l) r_l \right] \le P_0. \tag{1}$$

BLAde uses this budget to make two coupled decisions: which modules should become token-routed MoE, and how much rank the remaining ordinary LoRA modules should receive. Let $a = \{m_l, E_l, k_l, r_{e,l}, r_l\}_{l \in \mathcal{M}}$ denote an architecture, and let $\mathcal{A}(P_0)$ contain the architectures satisfying Equation 1. Ideally, we would train the adapter and router parameters for each candidate architecture and select the one with the lowest task loss:

$$a^* \in \arg\min_{a \in \mathcal{A}(P_0)} \min_{\boldsymbol{\theta}_a} \mathbb{E}_{(\xi, y^\star) \sim \mathcal{D}} \left[\ell(f(\xi; \boldsymbol{\theta}_a, a), y^\star) \right], \tag{2}$$

where $\mathcal{D}$ is the downstream data, ξ an input, $y^\star$ its target, f the model, ℓ the task loss, and $\boldsymbol{\theta}_a$ the adapter and router parameters. Solving Equation 2 requires training many candidate architectures, which is computationally expensive. We therefore use inexpensive forward statistics to guide two decisions: how much rank to assign to a shared adapter, and whether a module may benefit from token-specific experts.

3.2 Preliminaries: Effective Rank and Variance

Our goal is to allocate module-level adaptation capacity (rank and expert configuration) according to the gradient geometry of downstream adaptation. Since exact gradient information requires backward calibration, we instead use lightweight forward statistics to construct two proxy scores.

3.2.1 Effective Rank: Measuring Required Subspace Dimension

Consider a nonzero unobserved covariance $\boldsymbol{C} \succeq 0$ of ideal token-specific adapter corrections after subtracting their mean.

Let $\lambda_1, \ldots, \lambda_d \ge 0$ be the eigenvalues of $\boldsymbol{C}$ and define the normalized spectrum as

$$\pi_i = \frac{\lambda_i}{\sum_j \lambda_j} = \frac{\lambda_i}{\operatorname{Tr}(\boldsymbol{C})}, \qquad \sum_i \pi_i = 1. \tag{3}$$

Imagine choosing an eigenvector with probability π_i, proportional to its eigenvalue. If we repeat this independently, the probability of choosing the same direction twice is $\kappa(\boldsymbol{C}) = \sum_i \pi_i^2$. Intuitively, if all variance concentrates in one direction ($\pi_1 = 1$), then $\kappa(\boldsymbol{C}) = 1$ (always the same). If variance is uniformly spread over r directions ($\pi_i = 1/r$ for $i \leq r$), then $\kappa(\boldsymbol{C}) = r \cdot (1/r)^2 = 1/r$ (lower collision probability). Thus, a low κ indicates that variance is spread across many directions. We define the effective rank as the inverse concentration:

$$\mathrm{erank}(\boldsymbol{C}) := \frac{1}{\kappa(\boldsymbol{C})} = \frac{1}{\sum_i \pi_i^2} = \frac{[\mathrm{Tr}(\boldsymbol{C})]^2}{\sum_i \lambda_i^2} = \frac{\mathrm{Tr}(\boldsymbol{C})^2}{\mathrm{Tr}(\boldsymbol{C}^2)}. \tag{4}$$

Here the last equality holds because $\mathrm{Tr}(\boldsymbol{C}^2) = \sum_i \lambda_i^2$ for any covariance matrix. This participation-ratio definition measures effective dimensionality: a covariance with r equal nonzero eigenvalues has $\mathrm{erank}(\boldsymbol{C}) = r$, whereas a covariance concentrated on one direction has $\mathrm{erank}(\boldsymbol{C}) = 1$. We use effective rank to summarize spectral spread and guide capacity allocation. Appendix C.1 provides the supporting rank bound and discusses its connection to LoRA capacity.

3.2.2 Variance: Bounding the Value of Conditional Specialization

Consider an unobserved ideal token-specific adapter correction $\boldsymbol{U} \in \mathbb{R}^d$. If every token uses the same correction vector $\boldsymbol{a}$, the smallest achievable mean squared error is

$$D_1 = \min_{\boldsymbol{a}} \mathbb{E}\|\boldsymbol{U} - \boldsymbol{a}\|_2^2 \overset{\boldsymbol{a}=\mathbb{E}[\boldsymbol{U}]}{=} \mathrm{Tr}(\mathrm{Cov}(\boldsymbol{U})). \tag{5}$$

Let $Z \in \{1, \ldots, E\}$ denote the group assigned to each token. If each group uses its own correction vector, the optimal vector for group e is $\mathbb{E}[\boldsymbol{U} \mid Z = e]$. The resulting mean squared error is

$$D(Z) = \mathbb{E}\|\boldsymbol{U} - \mathbb{E}[\boldsymbol{U} \mid Z]\|_2^2 = \mathbb{E}[\mathrm{Tr}(\mathrm{Cov}(\boldsymbol{U} \mid Z))]. \tag{6}$$

By the law of total covariance, the improvement from conditioning is bounded by

$$D_1 - D(Z) \leq \mathrm{Tr}(\mathrm{Cov}(\boldsymbol{U})). \tag{7}$$

Thus, group-specific corrections can reduce the mean squared error by at most the total variance (Appendix C.2). Motivated by this result, we define the average coordinate variance as

$$\nu(\boldsymbol{C}) = \frac{\mathrm{Tr}(\mathrm{Cov}(\boldsymbol{U}))}{d} = \frac{\mathrm{Tr}(\boldsymbol{C})}{d}, \tag{8}$$

which is normalized by dimension to enable comparison across modules of different widths. This simplified model motivates using variance to identify modules that may benefit from separate corrections for different token groups, even though it does not uniquely determine the optimal expert count.

3.2.3 Proxy Scores from Forward Activations

The preceding analysis motivates two signals: variance bounds the possible reduction in approximation error from group-specific corrections, while effective rank helps assess how many directions are needed to represent the ideal corrections. For a frozen linear module, let $\boldsymbol{y}_{l,t} = \boldsymbol{W}_l \boldsymbol{x}_{l,t}$ and normalize each token's output vector by its length:

$$\bar{\boldsymbol{y}}_{l,t} = \sqrt{d_{o,l}} \frac{\boldsymbol{y}_{l,t}}{\|\boldsymbol{y}_{l,t}\|_2 + \epsilon}. \tag{9}$$

Here $\epsilon > 0$ stabilizes denominators. This reduces differences in output magnitude across tokens, so the statistics mainly reflect differences in output direction. Let $\boldsymbol{P}_{x,l}$ and $\boldsymbol{P}_{y,l}$ select p evenly spaced input and output coordinates, and define a paired token feature

$$\boldsymbol{z}_{l,t} = (\boldsymbol{P}_{x,l} \boldsymbol{x}_{l,t}) \otimes (\boldsymbol{P}_{y,l} \bar{\boldsymbol{y}}_{l,t}), \qquad \widehat{\boldsymbol{\Sigma}}_{z,l} = \mathrm{Cov}_t(\boldsymbol{z}_{l,t}). \tag{10}$$

Here Cov_t denotes centered sample covariance over retained tokens (Appendix B.1). Unlike a product of marginal covariances, $\widehat{\boldsymbol{\Sigma}}_{z,l}$ retains the dependence between the input and response of the same token. Let $\boldsymbol{Q}_{x,l}$ and $\boldsymbol{Q}_{y,l}$ retain up to s coordinates to compute the input and output covariances separately, and denote $\widehat{\boldsymbol{\Sigma}}_{x,l} = \mathrm{Cov}_t(\boldsymbol{Q}_{x,l} \boldsymbol{x}_{l,t})$ and $\widehat{\boldsymbol{\Sigma}}_{\bar{y},l} = \mathrm{Cov}_t(\boldsymbol{Q}_{y,l} \bar{\boldsymbol{y}}_{l,t})$.

To evaluate the forward scores, we compute reference scores from the empirical covariance of sampled token-level gradient contributions $\boldsymbol{x}_{l,t} \otimes \boldsymbol{\delta}_{l,t}$ on a separate data split, then compare the resulting module rankings. Table 1 reports macro-averages over eight reasoning datasets, where $\rho_{\mathrm{S}}(v)$ and $\rho_{\mathrm{S}}(q)$ are Spearman correlations between forward and gradient-based scores. Output variation better predicts gradient variance, while marginal normalization of the paired joint statistic yields positive $\rho_{\mathrm{S}}(q)$ on every task and the strongest average rank agreement. In contrast, the marginal product reduces variance correlation and both overlap metrics, consistent with $\mathrm{Cov}(\boldsymbol{x} \otimes \boldsymbol{y}) \neq \mathrm{Cov}(\boldsymbol{x}) \otimes \mathrm{Cov}(\boldsymbol{y})$ in general.

Table 1: Preliminary averaged forward–oracle agreement using LLaMA3-8B.

Signal	$\rho_{\mathrm{S}}(v)$	$\rho_{\mathrm{S}}(q)$	MoE overlap	Rank overlap
$\boldsymbol{x}$ only	-0.243	-0.095	0.047	0.152
$\bar{\boldsymbol{y}}$ only	**0.672**	-0.150	0.414	0.176
$\boldsymbol{x} \times \bar{\boldsymbol{y}}$	0.190	-0.126	0.215	0.160
BLAde	**0.672**	**0.229**	**0.441**	**0.352**

These results motivate using normalized outputs for the variance score and paired input-output features for the effective-rank score:

$$v_l = \nu(\widehat{\boldsymbol{\Sigma}}_{\bar{y},l}), \qquad q_l = \frac{\mathrm{erank}(\widehat{\boldsymbol{\Sigma}}_{z,l})}{\max\{\mathrm{erank}(\widehat{\boldsymbol{\Sigma}}_{x,l})\,\mathrm{erank}(\widehat{\boldsymbol{\Sigma}}_{\bar{y},l}), \epsilon\}}. \tag{11}$$

Appendix D.2 states the assumption underlying the output-variance signal. Here, v_l measures how much normalized outputs vary across tokens. To compute q_l we divide the effective rank of the paired-feature covariance by the product of the input and output effective ranks, with ϵ preventing division by zero. This gives a relative score for comparing modules. The allocation rules are monotone in these scores; the scores do not determine the optimal ranks or expert counts. Appendix G.1 reports dataset-level agreement, and Appendix D.3 summarizes the conditions and limitations of relating the joint statistic to gradient geometry.

3.3 Forward-Only Module Proxy

For a shared linear module, each position contributes an outer product to the weight gradient:

$$\boldsymbol{g}_{l,t} = \mathrm{vec}(\boldsymbol{\delta}_{l,t}\boldsymbol{x}_{l,t}^{\top}) = \boldsymbol{x}_{l,t} \otimes \boldsymbol{\delta}_{l,t}, \qquad \nabla_{\boldsymbol{W}_l}\ell = \sum_t \boldsymbol{\delta}_{l,t}\boldsymbol{x}_{l,t}^{\top}, \tag{12}$$

where $\boldsymbol{\delta}_{l,t} = \nabla_{\boldsymbol{y}_{l,t}}\ell$ is the gradient with respect to the module output at token t. Computing it for every candidate module would require backward calibration. We instead use $\bar{\boldsymbol{y}}_{l,t}$ as a forward-only substitute for the backward error. It accumulates the output covariance for v_l, and the marginal and paired covariances defined above for q_l. This construction does not assert that $\bar{\boldsymbol{y}}_l = \boldsymbol{\delta}_l$; it preserves the input–response interaction pattern that a marginal Kronecker product discards.

Appendix B.1 provides streaming computation details, Appendix D.1 explains the marginal normalization in q_l, and Appendices C.3 and D.3 characterize when the forward proxy preserves gradient effective rank and why such agreement is not guaranteed in general.

3.4 Budget-Constrained Heterogeneous Allocation

Let $M = |\mathcal{M}|$. We first apply an effective-rank gate, then select at most $\lceil \rho M \rceil$ modules with the largest v_l among those above the η-quantile of q_l. This filters out modules with relatively low q_l.

For each selected module, E_l is assigned monotonically based on v_l as a budget-aware heuristic, since variation bounds specialization value but does not uniquely determine the optimal expert count. Adapters are initially instantiated at $r_{\mathrm{init}} = \lceil \mu r_{\mathrm{ref}} \rceil$, giving each expert rank $r_{e,l} = r_{\mathrm{init}}/E_l$ and each routed module the same total expert rank $E_l r_{e,l} = r_{\mathrm{init}}$. In contrast, k_l is determined by the relative rank of q_l within the final routed set, with $2 \leq k_l < E_l$. Thus, v_l controls the number of available

experts, while the relative ranking of q_l controls the number activated for each token. Let $\mathcal{M}_{\mathrm{MoE}}$ denote the selected routed modules. The remaining factor budget for ordinary LoRA is

$$P_{\mathrm{LoRA}} = P_0 - r_{\mathrm{init}} \sum_{l \in \mathcal{M}_{\mathrm{MoE}}} c_l. \tag{13}$$

We set ordinary LoRA ranks proportional to q_l, subject to lower and upper limits:

$$\widetilde{r}_l = \min\{r_{\mathrm{init}}, \max\{r_{\min}, \beta q_l\}\}, \qquad \sum_{l \notin \mathcal{M}_{\mathrm{MoE}}} c_l \widetilde{r}_l \le P_{\mathrm{LoRA}}. \tag{14}$$

Here $\widetilde{r}_l$ is the continuous rank before rounding, and $\beta \ge 0$ is a shared scale chosen to satisfy the weighted budget. If the remaining budget cannot give every ordinary LoRA module at least rank $r_{\min}$, we change the routed module with the smallest v_l back to ordinary LoRA and recompute the allocation. Appendix B.2 gives the discrete expert map, relative-rank-based top-k assignment, and integer rounding, together with the complete procedure in Algorithm 1.

For a converted module, let $\boldsymbol{R}_l \in \mathbb{R}^{E_l \times d_{i,l}}$ be its learned router. It defines the routing probabilities and active expert set

$$\boldsymbol{p}_{l,t} = \mathrm{softmax}(\boldsymbol{R}_l \boldsymbol{x}_{l,t}), \qquad \mathcal{T}_{l,t} = \mathrm{TopK}(\boldsymbol{p}_{l,t}, k_l), \qquad \widehat{p}_{l,t,e} = \frac{p_{l,t,e}}{\sum_{e' \in \mathcal{T}_{l,t}} p_{l,t,e'}}. \tag{15}$$

Only selected experts are executed, and the module output is

$$\boldsymbol{h}_{l,t} = \boldsymbol{W}_l \boldsymbol{x}_{l,t} + \frac{\alpha}{k_l r_{e,l}} \sum_{e \in \mathcal{T}_{l,t}} \widehat{p}_{l,t,e} \boldsymbol{B}_{l,e} \boldsymbol{A}_{l,e} \boldsymbol{x}_{l,t}. \tag{16}$$

The scaling $\alpha/(k_l r_{e,l})$ includes an additional $1/k_l$ factor after probability renormalization. Appendix B details forward profiling and allocation, while Appendix E lists the experimental settings.

4 Experiments

We evaluate BLAde on reasoning accuracy, generation accuracy, robustness to allocation settings, and initialization and training costs. Shared settings are given in Appendix E.1, with reasoning and generation schedules in Appendices E.2 and E.3. All experiments are conducted on a single NVIDIA A100 80GB GPU.

4.1 Main Experiments

Table 2 compares BLAde with standard LoRA (Hu et al., 2022), rank-adaptive methods AdaLoRA (Zhang et al., 2023b) and EVA (Paischer et al., 2025), and adaptive LoRA-MoE methods MoRE (Zhang et al., 2025a), AdaMoLE (Liu & Luo, 2024), AlphaLoRA (Qing et al., 2024), and GuiLoMo (Chen et al., 2025), using a reference rank of $r_{\mathrm{ref}} = 32$ on natural language reasoning tasks (Clark et al., 2019; Bisk et al., 2019; Sap et al., 2019; Zellers et al., 2019; Sakaguchi et al., 2019; Clark et al., 2018; Mihaylov et al., 2018). For EVA, we reproduce both $\rho_{\mathrm{EVA}} = 1$ and $\rho_{\mathrm{EVA}} = 2$, where ρ_{EVA} is its rank-redistribution parameter. We use a common reference rank and task-specific training schedules; realized trainable parameter counts differ and are reported in #Params. We additionally report BLAde-G, which replaces our forward-only proxy with backward-gradient statistics while retaining the same allocation procedure, serving as a reference for evaluating the quality of our forward proxy for allocation. Specifically, let $\widehat{\boldsymbol{\Sigma}}_{g,l} = \mathrm{Cov}_t(\boldsymbol{g}_{l,t})$, where $\boldsymbol{g}_{l,t} = \boldsymbol{x}_{l,t} \otimes \boldsymbol{\delta}_{l,t}$ and $\boldsymbol{\delta}_{l,t} = \nabla_{\boldsymbol{y}_{l,t}} \ell$. Its allocation scores are

$$q_l^{\mathrm{G}} = \mathrm{erank}(\widehat{\boldsymbol{\Sigma}}_{g,l}) = \frac{\mathrm{Tr}(\widehat{\boldsymbol{\Sigma}}_{g,l})^2}{\mathrm{Tr}(\widehat{\boldsymbol{\Sigma}}_{g,l}^2)}, \qquad v_l^{\mathrm{G}} = \nu(\widehat{\boldsymbol{\Sigma}}_{g,l}) = \frac{\mathrm{Tr}(\widehat{\boldsymbol{\Sigma}}_{g,l})}{d_{i,l} d_{o,l}}. \tag{17}$$

Here Cov_t denotes centered sample covariance over calibration tokens; q_l^{G} and v_l^{G} replace q_l and v_l in the allocator.

Across all three backbones, BLAde achieves higher average accuracy than all baselines reporting results on all eight tasks. Relative to LoRA, the displayed averages improve with only 1.5M, 1.4M, and 1.1M additional trainable parameters. It also exceeds AdaMoLE and AlphaLoRA while using fewer parameters. The averages remain within 0.2 points of BLAde-G, indicating that the forward proxy can support competitive allocations without backward calibration. Appendix G.2 shows the task-dependent module placements and ranks behind these aggregate results.

Table 2: Experimental results on natural language reasoning tasks. All methods use reference rank $r_{\text{ref}} = 32$; BLAde and BLAde-G use $\rho = 0.2$ and $\mu = 1.25$. #Params includes routers and is reported in millions (M); Avg. is the mean of the eight displayed task scores. For each method, the best result for each dataset within each backbone is **bolded**. We further present standard deviations and statistical significance tests in Appendix F.

Model	Method	#Params	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	Avg.
LLaMA2-7B	LoRA	56.1M	74.4	84.8	82.4	93.5	85.1	86.1	68.1	81.0	81.9
	AdaLoRA	56.1M	67.8	82.0	78.0	86.8	83.4	83.8	66.4	81.2	78.7
	EVA (ρ_{EVA} = 1)	56.1M	73.5	85.1	82.0	94.0	**86.5**	86.0	67.0	82.2	82.0
	EVA (ρ_{EVA} = 2)	49.0M	65.0	84.6	81.3	93.8	82.2	76.1	63.9	79.2	78.3
	MoRE	50.0M	71.6	85.0	81.8	93.6	85.7	85.1	67.9	82.4	81.6
	AdaMoLE	73.5M	74.5	85.1	82.2	94.1	85.5	85.9	70.4	81.8	82.4
	AlphaLoRA	112.4M	74.2	85.3	81.9	93.7	85.4	84.8	67.3	81.2	81.7
	GuiLoMo	51.9M	74.5	85.0	81.7	93.3	85.1	86.5	67.2	82.2	81.9
	BLAde	57.6M	**74.9**	**86.2**	82.6	**94.2**	86.2	**87.0**	**72.5**	83.0	**83.3**
	BLAde-G	57.6M	74.4	85.8	**83.1**	94.1	**86.5**	86.6	71.1	**83.8**	83.2
LLaMA3-8B	LoRA	56.6M	75.4	88.2	82.1	95.4	88.1	92.8	81.0	88.4	86.4
	AdaLoRA	56.6M	69.9	85.6	79.9	92.5	83.0	92.6	80.9	85.0	83.7
	EVA (ρ_{EVA} = 1)	56.6M	74.1	87.8	82.7	96.0	88.0	88.2	77.4	87.0	85.2
	EVA (ρ_{EVA} = 2)	43.2M	71.3	89.0	81.3	94.4	87.5	88.6	73.2	86.6	84.0
	MoRE	52.9M	73.9	88.9	82.7	94.3	86.5	92.9	80.9	86.8	85.9
	AdaMoLE	75.0M	75.5	90.4	82.5	95.3	87.5	91.7	81.8	88.6	86.7
	AlphaLoRA	118.0M	74.4	89.6	81.8	95.6	87.9	92.5	80.9	87.8	86.3
	GuiLoMo	55.0M	76.2	89.3	82.4	95.5	88.8	93.0	80.7	87.0	86.6
	BLAde	58.0M	76.7	90.3	**83.2**	**96.3**	**90.1**	93.2	82.3	89.2	87.7
	BLAde-G	58.0M	**77.0**	**91.0**	82.7	96.2	89.8	**93.4**	**82.6**	**90.2**	**87.9**
LLaMA3.2-3B	LoRA	33.0M	72.8	84.7	80.6	93.5	84.5	87.4	74.7	78.8	82.1
	AdaLoRA	33.0M	70.6	83.3	75.7	88.6	84.3	86.5	74.0	81.0	80.5
	EVA (ρ_{EVA} = 1)	33.0M	73.1	84.8	81.0	94.2	86.0	85.6	69.7	**81.4**	82.0
	EVA (ρ_{EVA} = 2)	26.5M	65.9	83.4	78.4	91.4	82.2	85.1	59.6	76.8	77.9
	MoRE	30.9M	71.2	83.5	80.8	92.7	84.0	86.7	75.2	80.6	81.8
	AdaMoLE	44.5M	73.0	85.1	80.3	93.2	85.7	87.4	74.4	79.8	82.4
	AlphaLoRA	78.2M	72.8	84.6	81.0	93.8	85.1	88.1	73.2	80.6	82.4
	GuiLoMo	21.7M	72.8	85.4	80.4	93.7	85.2	87.4	71.8	79.4	82.0
	BLAde	34.1M	73.3	**86.4**	81.2	**94.5**	86.3	**88.5**	75.2	81.2	83.3
	BLAde-G	34.1M	**73.7**	85.6	**81.5**	94.3	**86.6**	88.3	**76.1**	**81.4**	**83.4**

4.2 Natural Language Generation Experiments

We further evaluate GSM8K (Cobbe et al., 2021), MBPP (Austin et al., 2021), HumanEval (Chen et al., 2021), and Natural Questions (NQ) (Kwiatkowski et al., 2019) on LLaMA3-8B (Grattafiori et al., 2024), Gemma-7B (Mesnard et al., 2024), and Mistral-7B (Jiang et al., 2023). As shown in Table 3, BLAde consistently outperforms LoRA across all three backbones.

We compare BLAde with representative rank-allocation method EVA, and adaptive LoRA-MoE methods, AdaMoLE and AlphaLoRA. BLAde achieves the best or second-best performance across all datasets. AdaMoLE attains slightly higher GSM8K accuracy on LLaMA3-8B and Gemma-7B, but uses additional trainable parameters and incurs higher computational overhead. Overall, these results demonstrate that BLAde effectively allocates rank and expert capacity to improve performance across natural language generation tasks.

Table 3: Results on natural language generation tasks. GSM8K and NQ are evaluated by answer accuracy, while MBPP and HumanEval are evaluated by pass@1. We use reference rank $r_{\text{ref}} = 16$; BLAde uses $\rho = 0.2$ and $\mu = 1.25$. For HumanEval, models are trained on the MBPP training set. Results are grouped by backbone model. For each method, we report the median over three runs, and the best result for each dataset within each backbone is **bolded**.

	LLaMA3-8B				Gemma-7B				Mistral-7B			
Method	GSM8K	MBPP	HumanEval	NQ	GSM8K	MBPP	HumanEval	NQ	GSM8K	MBPP	HumanEval	NQ
LoRA	61.6	47.4	41.5	39.8	60.5	45.0	34.8	36.2	58.9	40.6	30.4	43.0
EVA (ρ_{EVA} = 1)	61.3	43.2	38.4	39.5	58.6	40.2	29.8	33.1	54.7	30.8	22.6	40.6
EVA (ρ_{EVA} = 2)	57.9	37.0	31.1	39.2	56.8	38.0	22.0	33.4	54.1	31.6	25.0	40.9
AdaMoLE	**63.9**	**51.0**	41.5	40.4	**62.5**	47.8	35.4	35.2	60.7	41.8	31.1	44.2
AlphaLoRA	62.2	48.8	40.2	41.0	61.6	43.0	31.7	34.3	59.5	39.8	29.9	43.5
BLAde	63.5	50.2	**44.5**	**41.8**	62.2	**48.4**	**36.0**	**36.9**	**61.0**	**42.6**	**32.3**	**44.3**

4.3 HYPERPARAMETER SENSITIVITIES

We evaluate sensitivity to the target rank r_{ref}, routed-module fraction ρ, initial-rank multiplier μ, and calibration-set size N_{cal} on LLaMA3-8B. Figure 2 reports the equal-weight mean accuracy across the eight reasoning datasets; the corresponding per-task results and experimental details are provided in Appendix G.3 (Tables 11–14). Each mean is computed directly from the eight per-task scores rather than from the rounded averages displayed in the tables.

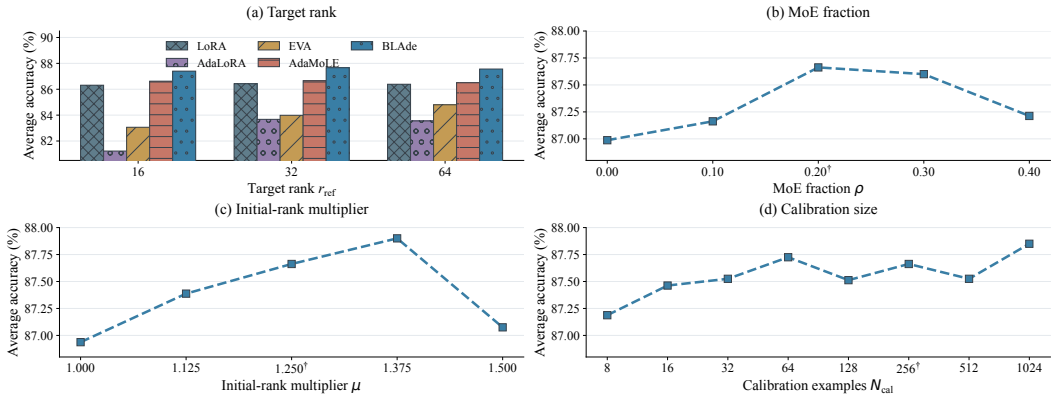


Figure 2: Hyperparameter sensitivity on LLaMA3-8B, measured by mean accuracy across eight reasoning tasks. (a) Target-rank comparison of five adaptation methods. (b–d) Sensitivity of BLAde to the routed-module fraction ρ, initial-rank multiplier μ, and calibration-set size N_{cal}, respectively, with all unspecified hyperparameters fixed to their defaults. Daggers denote the default settings.

Across all rank settings, BLAde achieves the highest mean accuracy, consistently outperforming all baselines. Interestingly, rank-adaptive methods such as AdaLoRA and EVA underperform fixed-rank LoRA, indicating that adaptive rank allocation alone does not necessarily improve downstream performance. In contrast, BLAde consistently maintains strong performance across different rank budgets, suggesting that jointly allocating rank and conditional expert capacity provides a more effective use of the available adaptation budget. These results further support the robustness of BLAde's allocation strategy across the tested ranks.

The MoE fraction ρ and initial-rank multiplier μ control how aggressively capacity is reallocated toward routed modules, subject to divisibility and budget constraints. Increasing ρ from 0.20 to 0.30–0.40 reduces accuracy, while increasing μ beyond 1.25 yields the highest mean at 1.375. These results suggest that moderate reallocation is preferable to aggressively concentrating capacity in routed modules. One possible explanation is that excessive reallocation leaves insufficient capacity for standard LoRA modules, although our sweeps do not establish this causal relationship.

We further vary the calibration-set size from 8 to 1,024 examples. Accuracy remains broadly stable across this range, with the highest mean obtained at 1,024 examples but no monotonic improvement as the calibration set grows. This robustness indicates that BLAde can derive informative allocation signals from relatively small calibration sets, requiring only limited data for effective allocation.

4.4 INITIALIZATION OVERHEAD

Figure 3 compares BLAde with EVA, LoRA-GA, AlphaLoRA, GuiLoMo, PiSSA (Meng et al., 2024), and BLAde-G. Additional peak GPU memory is measured above the memory already occupied by the frozen backbone and any adapters present before initialization. BLAde avoids backward calibration and SVD of weights or gradients. GuiLoMo requires performing alternating gradient updates of model parameters and GuidedSelection vectors during its bilevel allocation search, resulting in substantially higher initialization overhead. EVA has the closest measured overhead; BLAde has lower initialization time than PiSSA under these settings. Compared with BLAde-G and LoRA-GA, it reduces memory, computation, and time. BLAde-G achieves slightly higher average accuracy on two of the three backbones, at substantially higher calibration cost.

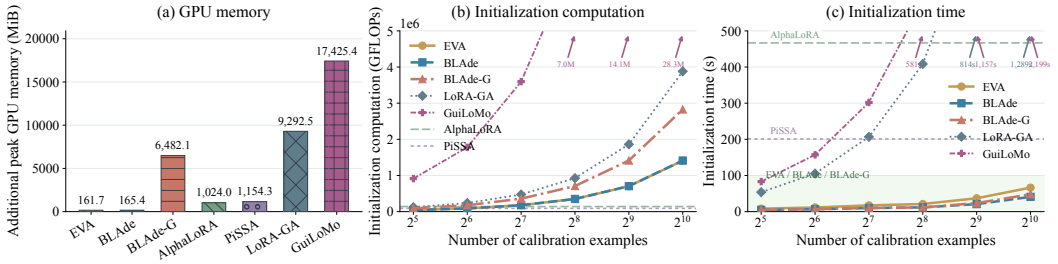


Figure 3: Initialization overhead comparison. We compare the (a) additional peak GPU memory over standard LoRA, (b) floating-point operations (FLOPs), and (c) wall-clock time. Results are averaged across eight reasoning tasks on LLaMA3-8B. Dashed lines in (b–c) denote calibration-size-independent baselines.

4.5 Training Overhead

Figure 4 shows that BLAde achieves lower training time, forward latency, and forward memory than AdaMoLE, AlphaLoRA, and GuiLoMo. In particular, BLAde incurs only slightly higher overhead than standard LoRA. GuiLoMo adds alternating model and selection-vector updates during initialization; its broad expert placement and active expert ranks can also increase training work, although these measurements do not isolate their effects. BLAde still requires more training time and forward memory than AdaLoRA and MoRE.

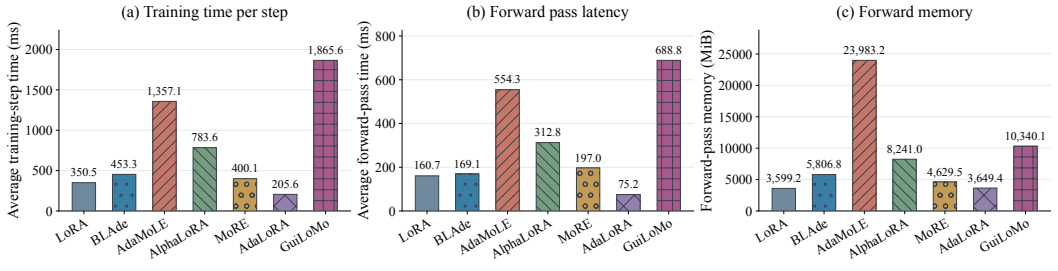


Figure 4: Training overhead comparison. We compare (a) training time per step, (b) forward-pass latency, and (c) forward-pass memory. Results are averaged across eight reasoning tasks using LLaMA3-8B as the backbone.

5 Conclusion

We presented BLAde, a forward-only framework that jointly determines where to introduce LoRA-MoE capacity and how to distribute it under a reference factor budget. Response variance guides module selection and expert counts, while marginal-normalized paired effective rank guides active expert counts and LoRA ranks. Together, these signals yield task-dependent adapter configurations without backward calibration. Experiments show improved reasoning and generation performance over LoRA, with accuracy close to the gradient-based variant and substantially lower initialization memory. Although forward statistics remain proxies and routers add overhead, the framework provides a practical approach to allocating conditional capacity according to activation geometry.

5.1 Limitations

BLAde inherits the deployment limitations of LoRA-MoE: token-dependent weight updates cannot be merged into the pretrained weights, introducing additional inference latency. Moreover, allocation statistics are approximated for computational efficiency, making our allocation strategy a practical trade-off between adaptation quality and computational cost rather than a theoretically optimal solution. Finally, initialization still requires a small amount of data prepared; these examples may come from a related domain rather than the exact target task, but the method is not data-free.

GENERATIVE AI USAGE STATEMENT

We used GPT-5.6 sol to assist with manuscript revision, related-work discovery, method development, and mathematical proof drafting. For manuscript revision, we first prepared the drafts ourselves, used the model to suggest corrections and improvements to clarity and language, and then reviewed and revised the suggested changes. For related-work discovery, we asked the model to identify a broad set of potentially relevant methods and provide URLs to the corresponding sources. We assessed these candidates and made the final decisions about which works to discuss and cite.

For method development, we formulated the initial ideas and designed the overall framework, then used the model to brainstorm technically feasible implementation details and refine the initial design. We evaluated these suggestions through iterative discussions and retained responsibility for all resulting design decisions. For the mathematical analysis, we provided the propositions and intended proof directions and used the model to generate initial proof drafts. We then examined the arguments, corrected errors and gaps, and iteratively revised the proofs with AI-assisted editing before accepting the final versions. Thus, the model's assistance extended beyond language editing to substantive suggestions on implementation details and proof arguments.

All AI-assisted material included in this paper was reviewed and revised by the authors. We take full responsibility for the final manuscript, including the accuracy of its references, methodological claims, mathematical arguments, and reported results.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Shaoxiang Chen, Zequn Jie, and Lin Ma. Llava-mole: Sparse mixture of lora experts for mitigating data conflicts in instruction finetuning mllms, 2024. URL https://arxiv.org/abs/2401.16160.

Xinrong Chen, Hengyuan Zhang, Yingmin Qiu, Xiao Liang, Ziyue Li, Guanyu Wang, Weiping Li, Tong Mo, Hayden Kwok-Hay So, and Ngai Wong. GuiLoMo: Allocating experts and ranks for LoRA-MoE via bilevel optimization with GuidedSelection vectors. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 7552–7567, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.399. URL https://aclanthology.org/2025.findings-emnlp.399/.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Zhi-Quan Feng, Ying-Jia Lin, and Hung-Yu Kao. Learning in the fisher subspace: A guided initialization for loRA fine-tuning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=P76Q8Pi1E9.

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 Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5112–5127, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.284. URL https://aclanthology.org/2025.findings-naacl.284/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor

Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

haonan he, Peng Ye, Yuchen Ren, yuan yuan, LuyangZhou, ShucunJu, and lei chen. GoRA: Gradient-driven adaptive low rank adaptation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=d1dL1ymD6N.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

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

Zefang Liu and Jiahua Luo. Adamole: Fine-tuning large language models with adaptive mixture of low-rank adaptation experts. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=ndY9qFf9Sa.

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

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/martens15.html.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=6ZBHIEtdP4.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec,

Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Fabian Paischer, Lukas Hauzenberger, Thomas Schmied, Benedikt Alkin, Marc Peter Deisenroth, and Sepp Hochreiter. One initialization to rule them all: Fine-tuning via explained variance adaptation, 2025. URL https://openreview.net/forum?id=DM6Q45HWSk.

Peijun Qing, Chongyang Gao, Yefan Zhou, Xingjian Diao, Yaoqing Yang, and Soroush Vosoughi. AlphaLoRA: Assigning LoRA experts based on layer training quality. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 20511–20523, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1141. URL https://aclanthology.org/2024.emnlp-main.1141/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

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

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Cheng zhong Xu. HydraloRA: An asymmetric loRA architecture for efficient fine-tuning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=qEpi8uWX3N.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3274–3287, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.239. URL https://aclanthology.org/2023.eacl-main.239/.

Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-rank adaptation with gradient approximation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=VaLAWrLHJv.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of loRA experts. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uWvKBCYh4S.

Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermis, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=EvDeiLv7qc.

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

Dacao Zhang, Kun Zhang, Shimao Chu, Le Wu, Xin Li, and Si Wei. More: A mixture of low-rank experts for adaptive multi-task learning. *arXiv preprint arXiv:2505.22694*, 2025a. URL https://arxiv.org/abs/2505.22694.

Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. Increlora: Incremental parameter allocation method for parameter-efficient fine-tuning, 2023a. URL https://arxiv.org/abs/2308.12043.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=lq62uWRJjiY.

Qingyue Zhang, Chang Chu, Tianren Peng, Qi Li, Xiangyang Luo, Zhihao Jiang, and Shao-Lun Huang. LoRA-DA: Data-aware initialization for low-rank adaptation via asymptotic analysis, 2026. URL https://openreview.net/forum?id=4nNqVpC3td.

Yuanhe Zhang, Fanghui Liu, and Yudong Chen. LoRA-one: One-step full gradient could suffice for fine-tuning large language models, provably and efficiently. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=KwIlvmLDLm.

A Symbols and Notation

Table 4 summarizes the notation used in the method and proofs, grouped into allocation settings, vectors and matrices, and statistical operations. The index l identifies a target linear module, while t and e index tokens and experts, respectively. Module-specific dimensions and configurations carry the subscript l. All rank variables use lowercase r: r_{ref} is the global reference rank, r_{init} the temporary initialization rank, r_l the ordinary rank of module l, and $r_{e,l}$ the rank of each expert in that module. The unindexed r denotes a generic rank in the proofs. We distinguish unobserved oracle corrections and their covariance from the empirical activation covariances collected during forward-only calibration; hats on covariance matrices denote sample estimates. The resulting scores v_l and q_l guide the allocation procedure in Appendix B. We reserve ρ for the routed-module fraction, ρ_{EVA} for EVA's parameter, and ρ_{S} for Spearman correlation. In the proofs, $\boldsymbol{f}_l$ denotes the unsketched paired feature and γ_l the common scale induced by the covariance transformation; $\boldsymbol{h}_{l,t}$ denotes the adapted module output. The symbol χ indexes the streamed feature type, and π_i denotes a normalized covariance eigenvalue.

B Allocation Details and Algorithm

We first describe how forward calibration produces the module scores, then specify the budget-constrained allocation rules summarized in Algorithm 1. The allocation assumes integer ranks $r_{\text{ref}} \geq r_{\min} \geq 1$, $\mu \geq 1$, and $\rho, \eta \in [0, 1]$, ensuring that changing all routed modules back to ordinary LoRA makes the minimum ranks affordable within the budget. Numerical settings are listed in Appendix E.

B.1 Forward Statistics and Proxy Scores

For each module and calibration batch, we collect token activations, discard padding tokens, and randomly retain at most T valid tokens. We use the same retained tokens for the input and output statistics. We accumulate statistics for three feature types: selected input coordinates, selected normalized-output coordinates, and their paired features. We retain at most s coordinates for the

Table 4: Summary of symbols and notation.

Symbol	Shape	Meaning
Constants and discrete variables		
l	scalar	index of an eligible target module
$\mathcal{M}, \mathcal{M}_{\mathrm{MoE}}$	sets	profiled modules and selected routed modules
$d_{i,l}, d_{o,l}$	scalar	input and output widths of module l
$r_{\mathrm{ref}}, r_{\mathrm{init}}$	scalar	reference rank and temporary rank, $r_{\mathrm{init}} = \lceil \mu r_{\mathrm{ref}} \rceil$
μ, α	scalar	temporary-rank inflation and LoRA scaling coefficient
m_l	$\{0, 1\}$	indicator that module l is converted to MoE
$E_l, k_l, r_{e,l}, r_l$	scalar	experts, top-k, expert rank, and ordinary LoRA rank
c_l	scalar	rank-one cost, $d_{i,l} + d_{o,l}$
P_0, P_{LoRA}	scalar	reference and remaining factor budgets
ρ	scalar	requested maximum fraction of modules considered for MoE
η	scalar	effective-rank gate quantile
$r_{\min}$	scalar	minimum rank reserved for each ordinary LoRA module
T, s	scalar	token cap per batch and marginal coordinate-sketch width
$\mathcal{E}, \mathcal{E}_{\mathrm{adm}}$	sets	candidate and admissible expert counts
Vectors and matrices		
$\boldsymbol{U}$	$\mathbb{R}^d$	ideal token-specific adapter correction (d coordinates)
$\boldsymbol{C}$	$\mathbb{R}^{d \times d}$	PSD covariance; $\boldsymbol{C} = \mathrm{Cov}(\boldsymbol{U})$ for oracle corrections
$\widehat{\boldsymbol{C}}$	$\mathbb{R}^{d \times d}$	Sample estimate from $n > 1$ vectors $\boldsymbol{U}_t$: $\widehat{\boldsymbol{C}} = \frac{1}{n-1} \sum_{t=1}^{n} (\boldsymbol{U}_t - \bar{\boldsymbol{U}})(\boldsymbol{U}_t - \bar{\boldsymbol{U}})^\top$, where $\bar{\boldsymbol{U}}$ is the sample mean; oracle corrections are unobserved in forward-only profiling
$\boldsymbol{x}_l, \boldsymbol{y}_l, \boldsymbol{\delta}_l, \bar{\boldsymbol{y}}_l$	$\mathbb{R}^{d_{i,l}}, \mathbb{R}^{d_{o,l}}, \mathbb{R}^{d_{o,l}}, \mathbb{R}^{d_{o,l}}$	input, output, error, and RMS-normalized output
$\boldsymbol{W}_l$	$\mathbb{R}^{d_{o,l} \times d_{i,l}}$	frozen projection weight
$\boldsymbol{A}_l, \boldsymbol{B}_l$	$\mathbb{R}^{r_l \times d_{i,l}}, \mathbb{R}^{d_{o,l} \times r_l}$	LoRA down- and up-projection factors
$\boldsymbol{A}_{l,e}, \boldsymbol{B}_{l,e}$	$\mathbb{R}^{r_{e,l} \times d_{i,l}}, \mathbb{R}^{d_{o,l} \times r_{e,l}}$	factors of routed expert e
$\boldsymbol{R}_l$	$\mathbb{R}^{E_l \times d_{i,l}}$	token router of a converted module
$\boldsymbol{u}^{\chi}_{l,t}, \boldsymbol{s}^{\chi}_l$	vectors	streamed feature and its sum, $\chi \in \{x, \bar{y}, z\}$; see Eq. 18
$\boldsymbol{P}_{x,l}, \boldsymbol{P}_{y,l}$	linear maps	deterministic coordinate projections for the paired sketch
$\boldsymbol{z}_{l,t}$	$\mathbb{R}^{p^2}$	paired feature $(\boldsymbol{P}_{x,l}\boldsymbol{x}_{l,t}) \otimes (\boldsymbol{P}_{y,l}\bar{\boldsymbol{y}}_{l,t})$
$\boldsymbol{G}^{\chi}_l, \widehat{\boldsymbol{\Sigma}}_{\bar{y},l}, \widehat{\boldsymbol{\Sigma}}_{z,l}$	matrices	streaming Gram matrix and forward covariance estimates
$\boldsymbol{\Sigma}_{x,l}, \boldsymbol{\Sigma}_{\delta,l}, \boldsymbol{\Sigma}_{\bar{y},l}$	covariance matrices	population input, error, and normalized-response covariances
Operations and derived scores		
$\mathrm{vec}(\cdot)$	matrix $\to$ vector	column-wise vectorization
$\otimes$	matrix $\times$ matrix	Kronecker product
$\mathrm{Tr}(\cdot)$	square matrix $\to \mathbb{R}$	matrix trace
$\mathrm{Cov}(\boldsymbol{z})$	random vector $\to$ matrix	$\mathbb{E}[(\boldsymbol{z} - \mathbb{E}[\boldsymbol{z}])(\boldsymbol{z} - \mathbb{E}[\boldsymbol{z}])^\top]$
$\mathrm{erank}(\cdot)$	PSD matrix $\to \mathbb{R}_+$	effective rank for nonzero PSD matrices; $\mathrm{erank}(0) = 1$
$\nu(\cdot)$	covariance $\to \mathbb{R}_+$	mean coordinate variance
v_l, q_l	$\mathbb{R}_+, \mathbb{R}_+$	variance and normalized paired effective-rank scores

separate input and output covariances, and p coordinates from each to construct the paired feature $\boldsymbol{z}_{l,t}$ in Eq. 10. We denote these three feature vectors by $\boldsymbol{u}^{\chi}_{l,t}$, with $\chi \in \{x, \bar{y}, z\}$: $\boldsymbol{u}^{x}_{l,t} = \boldsymbol{Q}_{x,l}\boldsymbol{x}_{l,t}$, $\boldsymbol{u}^{\bar{y}}_{l,t} = \boldsymbol{Q}_{y,l}\bar{\boldsymbol{y}}_{l,t}$, and $\boldsymbol{u}^{z}_{l,t} = \boldsymbol{z}_{l,t}$. The stored statistics are the retained-token count n^{χ}_l, vector sum, and Gram matrix:

$$n^{\chi}_l, \qquad \boldsymbol{s}^{\chi}_l = \sum_t \boldsymbol{u}^{\chi}_{l,t}, \qquad \boldsymbol{G}^{\chi}_l = \sum_t \boldsymbol{u}^{\chi}_{l,t} \boldsymbol{u}^{\chi\top}_{l,t}, \tag{18}$$

For each feature type with at least one retained token ($n^{\chi}_l \geq 1$), the centered sample covariance is

$$\widehat{\boldsymbol{\Sigma}}_{\chi,l} = \frac{\boldsymbol{G}^{\chi}_l - (n^{\chi}_l)^{-1} \boldsymbol{s}^{\chi}_l \boldsymbol{s}^{\chi\top}_l}{\max(n^{\chi}_l - 1, 1)}. \tag{19}$$

For $n^{\chi}_l = 1$, this expression returns zero; the usual sample-covariance formula requires $n^{\chi}_l > 1$. Profiling requires at least one retained token per module. All statistics are detached, accumulated on CPU, and discarded after allocation. The joint covariance has dimension at most $p^2 \times p^2$. Selecting a fixed subset of coordinates limits memory use. However, the resulting covariance may not represent

the covariance of the full feature vector without bias, and the result can depend on the coordinate system. The resulting covariances define v_l and q_l through Eq. 11. Only these scores are needed for allocation; token activations and accumulated moments can then be released.

Algorithm 1 BLAde: forward-only capacity allocation

Require: Modules $\mathcal{M}$, calibration data $\mathcal{D}_{\text{cal}}$, reference rank r_{ref}
Require: Allocation settings $\mu, \rho, \eta, r_{\min}$ and candidate expert counts $\mathcal{E}$
Ensure: Routed configurations $\{E_l, k_l, r_{e,l}\}_{l\in\mathcal{M}_{\text{MoE}}}$ and ordinary ranks $\{r_l\}_{l\notin\mathcal{M}_{\text{MoE}}}$
$r_{\text{init}} \leftarrow \lceil \mu r_{\text{ref}} \rceil$; $P_0 \leftarrow r_{\text{ref}} \sum_{l\in\mathcal{M}} c_l$
1. Profile module geometry
Collect centered activation covariances on $\mathcal{D}_{\text{cal}}$ using forward passes only (Appendix B.1)
Compute $\{v_l, q_l\}_{l\in\mathcal{M}}$ using Eq. 11
2. Select routed modules under the budget
$\mathcal{M}_{\text{cand}} \leftarrow \{l \in \mathcal{M} : q_l \geq \text{Quantile}(\{q_i : i \in \mathcal{M}\}, \eta)\}$
$\mathcal{M}_{\text{MoE}} \leftarrow$ the highest-v_l candidates, up to $\lceil \rho|\mathcal{M}| \rceil$ modules
Compute admissible expert counts $\mathcal{E}_{\text{adm}}$ using Eq. 20
if $\mathcal{E}_{\text{adm}} = \varnothing$ **then**
    $\mathcal{M}_{\text{MoE}} \leftarrow \varnothing$
end if
Compute P_{LoRA} using Eq. 13
while $r_{\min} \sum_{l\notin\mathcal{M}_{\text{MoE}}} c_l > P_{\text{LoRA}}$ **do**
    Remove the lowest-v_l module from $\mathcal{M}_{\text{MoE}}$ and update P_{LoRA}
end while
3. Allocate expert and ordinary capacity
for $l \in \mathcal{M}_{\text{MoE}}$ **do**
    Assign $(E_l, r_{e,l})$ by Eq. 21 and k_l by Eq. 23
end for
Solve Eq. 14 for $\{\widetilde{r}_l\}_{l\notin\mathcal{M}_{\text{MoE}}}$ by bisection on β
Set $r_l \leftarrow \lfloor \widetilde{r}_l \rfloor$ and distribute the residual budget using Appendix B.2
return $\mathcal{M}_{\text{MoE}}$, $\{E_l, k_l, r_{e,l}\}_{l\in\mathcal{M}_{\text{MoE}}}$, $\{r_l\}_{l\notin\mathcal{M}_{\text{MoE}}}$

B.2 Budget-Constrained Allocation

Given the scores, retain modules at or above the η-quantile of q_l and select up to $\lceil \rho|\mathcal{M}| \rceil$ candidates with the largest v_l. Denote this routed set by $\mathcal{M}_{\text{MoE}}$.

For $r_{\text{init}} = \lceil \mu r_{\text{ref}} \rceil$, let $\mathcal{E}_{\text{adm}}$ be the set of admissible expert counts drawn from the candidate set $\mathcal{E}$. A count is admissible only if it is at least three, does not exceed r_{init}, and divides r_{init} exactly. The divisibility condition ensures that assigning $r_{e,l} = r_{\text{init}}/E_l$ gives every expert an integer rank. Thus,

$$\mathcal{E}_{\text{adm}} = \{E \in \mathcal{E} : 3 \leq E \leq r_{\text{init}},\ r_{\text{init}} \bmod E = 0\} = \{E^{(1)} < \cdots < E^{(|\mathcal{E}_{\text{adm}}|)}\}. \tag{20}$$

If $\mathcal{E}_{\text{adm}}$ is nonempty, its elements are ordered increasingly as shown above, and the allocator maps each candidate module's variance score to one of these admissible counts. If the set is empty, no candidate count satisfies the constraints, so no module is converted to MoE and all modules remain ordinary LoRA. For candidate l, define $u_l = (v_l - v_{\min})/\max(v_{\max} - v_{\min}, \epsilon)$ using extrema over all profiled modules. If $\mathcal{E}_{\text{adm}}$ is nonempty, the implementation assigns

$$E_l = E^{(1+\min\{\lfloor |\mathcal{E}_{\text{adm}}| u_l \rfloor, |\mathcal{E}_{\text{adm}}|-1\})}, \qquad r_{e,l} = r_{\text{init}}/E_l. \tag{21}$$

The selected E_l is therefore always admissible, and the corresponding expert rank is an integer.

After selecting routed modules, we reserve their factor costs using Eq. 13. If the remaining budget cannot support rank $r_{\min}$ for every ordinary module, we change the routed module with the smallest v_l back to ordinary LoRA and repeat until every ordinary module can receive at least rank $r_{\min}$.

After budget-feasibility adjustments, compute each routed module's relative rank score ζ_l using only the final routed set $\mathcal{M}_{\text{MoE}}$:

$$\zeta_l = \frac{|\{i \in \mathcal{M}_{\text{MoE}} : q_i < q_l\}| + \frac{1}{2}|\{i \in \mathcal{M}_{\text{MoE}} : q_i = q_l\}|}{|\mathcal{M}_{\text{MoE}}|}. \tag{22}$$

The numerator counts routed modules with a smaller q score and gives half weight to modules tied with l; division by the routed-set size produces a score in $[0, 1]$. Thus, modules with larger q_l receive larger ζ_l, while tied modules receive the same score. For example, a unique lowest (highest) score has relative rank score $1/(2M_{\mathrm{MoE}})$ $(1 - 1/(2M_{\mathrm{MoE}}))$, where $M_{\mathrm{MoE}} = |\mathcal{M}_{\mathrm{MoE}}|$. We use this relative position, rather than the raw scale of q_l, to interpolate the active expert count between two and $E_l - 1$:

$$k_l = \begin{cases} 2, & E_l \le 3, \\ 2 + \mathrm{round}(\zeta_l(E_l - 3)), & E_l > 3, \end{cases} \tag{23}$$

which ensures $2 \le k_l < E_l$.

For ordinary modules, the continuous cost in Equation 14 is monotone in β, so bisection finds a feasible scale. We first round each continuous rank down to an integer. We then use the remaining budget for additional rank increments, prioritizing modules by $(\widetilde{r}_l - \lfloor \widetilde{r}_l \rfloor)/c_l$, from largest to smallest. This preserves the factor budget but is not claimed to solve the integer problem optimally.

The selected module set determines $m_l = 1$ for routed modules and $m_l = 0$ otherwise.

C Proofs

The following bounds motivate the allocation signals in Sections 3.2 and 3.4; they do not identify optimal ranks or expert counts.

C.1 Proof of Theorem C.1: Necessary Rank Bound

Theorem C.1 (Necessary Rank Bound). *Let $\boldsymbol{C} \in \mathbb{R}^{d\times d}$ be PSD with $\mathrm{Tr}(\boldsymbol{C}) > 0$, let $r \in \{1, \dots, d\}$, and let $\boldsymbol{P}$ be an orthogonal projector of rank at most r. Then $\mathrm{Tr}(\boldsymbol{PC})/\mathrm{Tr}(\boldsymbol{C}) \le \sqrt{r/\mathrm{erank}(\boldsymbol{C})}$. Retaining a fraction $\tau \in (0, 1]$ of the variance therefore requires $r \ge \tau^2 \mathrm{erank}(\boldsymbol{C})$. For a centered oracle correction, the same bound is necessary for a rank-r LoRA correction to retain that fraction of energy.*

Proof. Let $\boldsymbol{C} \succeq 0$ have eigenvalues $\lambda_1 \ge \cdots \ge \lambda_d \ge 0$, and let $\boldsymbol{P}$ be any orthogonal projector with $\mathrm{rank}(\boldsymbol{P}) \le r$. By the Ky Fan maximum principle, followed by Cauchy–Schwarz,

$$\mathrm{Tr}(\boldsymbol{PC}) \le \sum_{i=1}^{r} \lambda_i \le \sqrt{r \sum_{i=1}^{r} \lambda_i^2} \le \sqrt{r\,\mathrm{Tr}(\boldsymbol{C}^2)}. \tag{24}$$

Dividing by $\mathrm{Tr}(\boldsymbol{C}) > 0$ and using $\mathrm{erank}(\boldsymbol{C}) = \mathrm{Tr}(\boldsymbol{C})^2/\mathrm{Tr}(\boldsymbol{C}^2)$ gives

$$\frac{\mathrm{Tr}(\boldsymbol{PC})}{\mathrm{Tr}(\boldsymbol{C})} \le \sqrt{\frac{r}{\mathrm{erank}(\boldsymbol{C})}}. \tag{25}$$

Therefore, if some rank-r representation retains at least a fraction $\tau \in (0, 1]$ of the total variance, then

$$r \ge \tau^2 \mathrm{erank}(\boldsymbol{C}). \tag{26}$$

The same statement has a direct approximation interpretation. For a centered random vector $\boldsymbol{z}$ with covariance $\boldsymbol{C}$,

$$\min_{\boldsymbol{P}=\boldsymbol{P}^\top=\boldsymbol{P}^2,\,\mathrm{rank}(\boldsymbol{P})\le r} \mathbb{E}\|\boldsymbol{z} - \boldsymbol{Pz}\|_2^2 = \mathrm{Tr}(\boldsymbol{C}) - \sum_{i=1}^{r} \lambda_i(\boldsymbol{C}). \tag{27}$$

When variance is spread across many directions, retaining a fixed fraction of it requires a sufficiently large rank. Equality is attained when $\boldsymbol{C}$ has exactly r equal positive eigenvalues and the projector retains their full support ($\tau = 1$). Equal eigenvalues alone do not ensure equality for a smaller retained subspace.

For a direct LoRA interpretation, a rank-r update $\Delta\boldsymbol{W} = \boldsymbol{BA}$ produces corrections in $\mathrm{col}(\boldsymbol{B})$, whose dimension is at most r. If $\boldsymbol{U}$ is the centered oracle correction and $\boldsymbol{P_B}$ projects onto this

column space, then the orthogonal projection is the closest vector to $\boldsymbol{U}$ in this subspace. Any LoRA correction in the same subspace therefore has at least as much squared error: $\mathbb{E}\|\boldsymbol{U} - \Delta\boldsymbol{W}\boldsymbol{x}\|_2^2 \geq \mathbb{E}\|\boldsymbol{U} - \boldsymbol{P_B}\boldsymbol{U}\|_2^2$. Thus Equation 25 is a necessary LoRA-rank bound when $\boldsymbol{C} = \mathrm{Cov}(\boldsymbol{U})$. We do not observe this oracle covariance directly. Instead, we use the effective rank of the forward activation geometry as a spectral-spread proxy for capacity demand. The bound motivates this choice, but substituting the forward proxy covariance for the oracle covariance does not yield a lower bound on the oracle-optimal LoRA rank.

This bound gives a necessary minimum rank, but does not determine the smallest rank that achieves a given approximation error. For example, the normalized spectra $(1/2, 1/2, 0)$ and $(2/3, 1/6, 1/6)$ both have $\sum_i \pi_i^2 = 1/2$ and thus effective rank 2, but their best rank-one subspaces retain respectively $1/2$ and $2/3$ of the energy; their algebraic ranks also differ. Effective rank therefore supplies a spectral-spread summary and the necessary bound above, while a target error and the full spectral tail would be required to determine an exact rank. □

C.2 Proof of Theorem C.2: Conditional Prototype Gain

Theorem C.2 (Conditional Prototype Gain). *Let $\boldsymbol{U} \in \mathbb{R}^d$ have a finite second moment, and let $Z \in \{1, \ldots, E\}$ be a routing variable. The minimum mean squared errors obtained using one shared correction vector and separate correction vectors for the routed groups satisfy $D_1 = \mathrm{Tr}(\mathrm{Cov}(\boldsymbol{U}))$ and $D(Z) = \mathbb{E}[\mathrm{Tr}(\mathrm{Cov}(\boldsymbol{U} \mid Z))]$. Consequently, $0 \leq D_1 - D(Z) = \mathrm{Tr}(\mathrm{Cov}(\mathbb{E}[\boldsymbol{U} \mid Z])) \leq \mathrm{Tr}(\mathrm{Cov}(\boldsymbol{U}))$.*

Proof. Let $\boldsymbol{U} \in \mathbb{R}^d$ be an ideal token-specific adapter correction with a finite second moment. Using one correction vector for all tokens gives the minimum mean squared error

$$D_1 = \min_{\boldsymbol{a}} \mathbb{E}\|\boldsymbol{U} - \boldsymbol{a}\|_2^2 \overset{\boldsymbol{a}=\mathbb{E}[\boldsymbol{U}]}{=} \mathrm{Tr}(\mathrm{Cov}(\boldsymbol{U})), \tag{28}$$

because the minimizer is $\boldsymbol{a} = \mathbb{E}\boldsymbol{U}$. For any routing variable $Z \in \{1, \ldots, E\}$, the optimal correction vectors for the routed groups are $\boldsymbol{a}_e = \mathbb{E}[\boldsymbol{U} \mid Z = e]$, giving

$$D(Z) = \mathbb{E}\|\boldsymbol{U} - \mathbb{E}[\boldsymbol{U} \mid Z]\|_2^2 = \mathbb{E}[\mathrm{Tr}(\mathrm{Cov}(\boldsymbol{U} \mid Z))]. \tag{29}$$

The law of total covariance states

$$\mathrm{Cov}(\boldsymbol{U}) = \mathbb{E}[\mathrm{Cov}(\boldsymbol{U} \mid Z)] + \mathrm{Cov}(\mathbb{E}[\boldsymbol{U} \mid Z]). \tag{30}$$

Taking traces yields

$$D_1 - D(Z) = \mathrm{Tr}(\mathrm{Cov}(\mathbb{E}[\boldsymbol{U} \mid Z])) \leq \mathrm{Tr}(\mathrm{Cov}(\boldsymbol{U})), \tag{31}$$

which proves Equation 7. Dividing by d gives the per-coordinate bound in terms of ν. Zero variance precludes any gain; positive variance permits, but does not ensure, a gain. For example, a routing variable Z independent of $\boldsymbol{U}$ yields no improvement. Variance therefore bounds the benefit of specialization without determining the optimal expert count. □

C.3 Proof of Theorem C.3: Conditional Effective-Rank Preservation

Theorem C.3 (Conditional Effective-Rank Preservation). *Let $\boldsymbol{f}_l = \boldsymbol{x}_l \otimes \bar{\boldsymbol{y}}_l$ and $\boldsymbol{g}_l = \boldsymbol{x}_l \otimes \boldsymbol{\delta}_l$ have finite covariance, where $\boldsymbol{x}_l \in \mathbb{R}^{d_{i,l}}$ and $\bar{\boldsymbol{y}}_l, \boldsymbol{\delta}_l \in \mathbb{R}^{d_{o,l}}$. Suppose a fixed matrix $\boldsymbol{T}_l \in \mathbb{R}^{d_{o,l} \times d_{o,l}}$ and a fixed scalar $\gamma_l > 0$ satisfy $\boldsymbol{\delta}_l = \boldsymbol{T}_l \bar{\boldsymbol{y}}_l$ almost surely on the calibration distribution and $\boldsymbol{T}_l^\top \boldsymbol{T}_l = \gamma_l \boldsymbol{I}_{d_{o,l}}$. Then, with $\boldsymbol{S}_l = \boldsymbol{I}_{d_{i,l}} \otimes \boldsymbol{T}_l$, $\mathrm{Cov}(\boldsymbol{g}_l) = \boldsymbol{S}_l \mathrm{Cov}(\boldsymbol{f}_l) \boldsymbol{S}_l^\top$. If $\mathrm{Cov}(\boldsymbol{f}_l) \neq 0$, then $\mathrm{erank}(\mathrm{Cov}(\boldsymbol{g}_l)) = \mathrm{erank}(\mathrm{Cov}(\boldsymbol{f}_l))$.*

The theorem establishes equality of covariance effective ranks under the stated assumptions; it does not reconstruct the gradients. For a linear module, the backward error is $\boldsymbol{\delta}_l = \nabla_{\boldsymbol{y}_l} \ell$, and the per-position gradient contribution is (suppressing t)

$$\boldsymbol{g}_l = \mathrm{vec}(\boldsymbol{\delta}_l \boldsymbol{x}_l^\top) = \boldsymbol{x}_l \otimes \boldsymbol{\delta}_l. \tag{32}$$

The full shared-weight gradient sums these contributions over token positions, as in Eq. 12. The forward feature $\boldsymbol{f}_l = \boldsymbol{x}_l \otimes \bar{\boldsymbol{y}}_l$ retains the same input factor and replaces the unavailable backward-error factor with the normalized response. Here $\boldsymbol{f}_l$ denotes the unsketched paired feature used in this theorem. Its covariance retains input–response dependence; independence between the two factors is not required.

In general, the backward error is produced by the chain rule. Let $\boldsymbol{o} = F_l(\boldsymbol{y}_l; \boldsymbol{c})$ denote the final prediction obtained from the downstream computation, where $\boldsymbol{c}$ collects the other incoming activations and context held fixed when differentiating with respect to $\boldsymbol{y}_l$. For target $y^\star$ and loss $\ell(\boldsymbol{o}, y^\star)$, define the downstream Jacobian $\boldsymbol{J}_{F_l}(\boldsymbol{y}_l; \boldsymbol{c}) = \partial F_l(\boldsymbol{y}_l; \boldsymbol{c})/\partial \boldsymbol{y}_l$. Then

$$\boldsymbol{\delta}_l = \nabla_{\boldsymbol{y}_l} \ell(F_l(\boldsymbol{y}_l; \boldsymbol{c}), y^\star) = \boldsymbol{J}_{F_l}(\boldsymbol{y}_l; \boldsymbol{c})^\top \, \nabla_{\boldsymbol{o}} \ell(\boldsymbol{o}, y^\star)|_{\boldsymbol{o}=F_l(\boldsymbol{y}_l;\boldsymbol{c})} \,. \tag{33}$$

The loss gradient $\nabla_{\boldsymbol{o}} \ell(\boldsymbol{o}, y^\star)$ measures the sensitivity of the loss to the prediction, and the transposed Jacobian propagates this signal back to the module output. Thus, $\boldsymbol{\delta}_l$ depends on both the prediction–target discrepancy and the downstream network's local response to changes in $\boldsymbol{y}_l$. The Jacobian can vary with the input and activation pattern, while the loss gradient also depends on the target. In a Transformer, the downstream computation can couple token positions, so a token's backward error can reflect losses at other positions as well.

Equation 33 does not generally reduce to a fixed matrix multiplying $\bar{\boldsymbol{y}}_l$. In particular, the downstream Jacobian alone is not the matrix $\boldsymbol{T}_l$ in this theorem: it acts on a loss-gradient signal, not directly on the normalized module response. The assumed relation must account for their combined effect across the calibration distribution. The matrix $\boldsymbol{T}_l$ therefore expresses an additional, hypothetical relation between the normalized response and the backward error:

$$\boldsymbol{\delta}_l = \boldsymbol{T}_l \bar{\boldsymbol{y}}_l, \qquad \boldsymbol{T}_l^\top \boldsymbol{T}_l = \gamma_l \boldsymbol{I}_{d_{o,l}}, \quad \gamma_l > 0. \tag{34}$$

It is not a parameter estimated or learned by the allocation algorithm. For each module, the same $\boldsymbol{T}_l$ must apply across calibration examples and tokens. The second condition says that $\boldsymbol{T}_l/\sqrt{\gamma_l}$ is orthogonal: it permits rotation or reflection and a common scaling of all directions, since $\|\boldsymbol{T}_l \boldsymbol{u}\|_2^2 = \gamma_l \|\boldsymbol{u}\|_2^2$ for every $\boldsymbol{u} \in \mathbb{R}^{d_{o,l}}$. An arbitrary linear map can stretch different directions by different amounts and need not preserve effective rank.

These assumptions do not follow from the forward computation. For example, a squared loss $\ell = \frac{1}{2}\|\boldsymbol{y}_l - \boldsymbol{y}^\star\|_2^2$ gives $\boldsymbol{\delta}_l = \boldsymbol{y}_l - \boldsymbol{y}^\star$, so the same response with different targets need not produce the same backward error. Even if $\boldsymbol{y}^\star = 0$, expressing $\boldsymbol{y}_l$ in terms of its normalized response introduces a token-dependent scale unless the output norm is constant. Thus, a fixed scaled isometry is not guaranteed in a Transformer, and allowing a separate $\boldsymbol{T}_l$ for every token would not satisfy the theorem's assumption.

Proof. The Kronecker-product identity $(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{u} \otimes \boldsymbol{v}) = (\boldsymbol{A}\boldsymbol{u}) \otimes (\boldsymbol{B}\boldsymbol{v})$ gives

$$\begin{aligned} \boldsymbol{g}_l &= \boldsymbol{x}_l \otimes (\boldsymbol{T}_l \bar{\boldsymbol{y}}_l) \\ &= (\boldsymbol{I}_{d_{i,l}} \otimes \boldsymbol{T}_l)(\boldsymbol{x}_l \otimes \bar{\boldsymbol{y}}_l) = \boldsymbol{S}_l \boldsymbol{f}_l. \end{aligned} \tag{35}$$

The identity $\boldsymbol{I}_{d_{i,l}}$ leaves the input factor unchanged, while $\boldsymbol{T}_l$ transforms only the response factor. This input-space identity is distinct from the output-space identity $\boldsymbol{I}_{d_{o,l}}$ in Eq. 34. Both joint features lie in $\mathbb{R}^{d_{i,l} d_{o,l}}$, and

$$\boldsymbol{S}_l^\top \boldsymbol{S}_l = \boldsymbol{I}_{d_{i,l}} \otimes (\boldsymbol{T}_l^\top \boldsymbol{T}_l) = \gamma_l \boldsymbol{I}_{d_{i,l} d_{o,l}}. \tag{36}$$

Write $\boldsymbol{C}_f = \mathrm{Cov}(\boldsymbol{f}_l)$ and $\boldsymbol{C}_g = \mathrm{Cov}(\boldsymbol{g}_l)$. Since $\boldsymbol{S}_l$ is fixed, subtracting the mean before applying $\boldsymbol{S}_l$ gives the same result as applying $\boldsymbol{S}_l$ first and then subtracting the transformed mean: $\boldsymbol{g}_l - \mathbb{E}\boldsymbol{g}_l = \boldsymbol{S}_l(\boldsymbol{f}_l - \mathbb{E}\boldsymbol{f}_l)$. Hence $\boldsymbol{C}_g = \boldsymbol{S}_l \boldsymbol{C}_f \boldsymbol{S}_l^\top$. Cyclicity of the trace then yields

$$\begin{aligned} \mathrm{Tr}(\boldsymbol{C}_g) &= \mathrm{Tr}(\boldsymbol{C}_f \boldsymbol{S}_l^\top \boldsymbol{S}_l) = \gamma_l \,\mathrm{Tr}(\boldsymbol{C}_f), \\ \mathrm{Tr}(\boldsymbol{C}_g^2) &= \mathrm{Tr}(\boldsymbol{C}_f \boldsymbol{S}_l^\top \boldsymbol{S}_l \boldsymbol{C}_f \boldsymbol{S}_l^\top \boldsymbol{S}_l) = \gamma_l^2 \,\mathrm{Tr}(\boldsymbol{C}_f^2). \end{aligned} \tag{37}$$

For nonzero positive-semidefinite $\boldsymbol{C}_f$, the denominator below is positive, and the common scale cancels:

$$\mathrm{erank}(\boldsymbol{C}_g) = \frac{\gamma_l^2 \,\mathrm{Tr}(\boldsymbol{C}_f)^2}{\gamma_l^2 \,\mathrm{Tr}(\boldsymbol{C}_f^2)} = \mathrm{erank}(\boldsymbol{C}_f). \tag{38}$$

Equivalently, all nonzero covariance eigenvalues are multiplied by γ_l, so their normalized spectrum is unchanged. □

However, the relations $\boldsymbol{\delta}_l = \boldsymbol{T}_l \bar{\boldsymbol{y}}_l$ and $\boldsymbol{T}_l^\top \boldsymbol{T}_l = \gamma_l \boldsymbol{I}$ are additional assumptions. Without restrictions on labels or objectives, forward information alone cannot recover the true gradient covariance in general. To see this, consider the squared loss with $x \in \{-1, +1\}$ uniformly and $y = wx$ at $w = 1$. The target $y^\star = x$ gives zero gradient, whereas $y^\star = 1$ gives $\nabla_w \ell = 1 - x$ with nonzero variance, despite identical forward (x, y) distributions. Conversely, at $w = 0$ and $y^\star = 1$, the output proxy is zero while the gradient $-x$ has nonzero variance. Thus, forward statistics cannot unconditionally determine gradient covariance, and the scores used by BLAde serve as proxies rather than exact estimators of gradient statistics.

Nevertheless, the proof shows that paired forward features preserve the effective rank of gradient features under the stated structural assumptions. This provides a theoretical basis for using paired forward geometry as a proxy for gradient spectral spread when allocating adaptation capacity. The theorem concerns unsketched paired features; it does not establish preservation after coordinate sketching or marginal normalization in q_l. The practical usefulness of the implemented proxy is evaluated through the proxy–oracle agreement experiment in Appendix G.1.

D Justifications

D.1 Choice of Allocation Statistics

For nonzero positive-semidefinite matrices, participation-ratio effective rank is multiplicative under a Kronecker product:

$$\operatorname{erank}(\boldsymbol{C}_x \otimes \boldsymbol{C}_y) = \operatorname{erank}(\boldsymbol{C}_x) \operatorname{erank}(\boldsymbol{C}_y). \tag{39}$$

Thus, the denominator of q_l in Eq. 11 provides the effective rank of the corresponding marginal Kronecker-product baseline. This does not imply that $\operatorname{Cov}(\boldsymbol{x} \otimes \bar{\boldsymbol{y}}) = \operatorname{Cov}(\boldsymbol{x}) \otimes \operatorname{Cov}(\bar{\boldsymbol{y}})$; the numerator is instead computed from paired input–output samples so that it retains their dependence. The resulting q_l is a relative allocation score rather than an exact estimator of gradient effective rank or optimal rank.

D.2 Output Variance for Specialization

The conditional-gain bound in Eq. 31 applies to the unavailable ideal correction, not directly to $\bar{\boldsymbol{y}}_l$. Using $v_l = \nu(\widehat{\boldsymbol{\Sigma}}_{\bar{y},l})$ therefore assumes that modules with larger normalized-output variance tend to benefit more from separate corrections for different token groups. This assumption concerns the ordering of modules rather than the exact gain or optimal expert count. Table 1 evaluates it indirectly through agreement with gradient-based module rankings.

D.3 Validity and Limits of Forward-Only Proxies

The forward statistics depend only on module inputs and normalized outputs, whereas gradient statistics also depend on targets, the loss, and the downstream Jacobian. Theorem C.3 gives a sufficient condition under which the unsketched paired forward feature preserves gradient effective rank, while Appendix C.3 shows why this condition is not guaranteed in general. The implemented scores additionally use coordinate sketches and marginal normalization, so their agreement with gradient geometry is evaluated empirically in Appendix G.1 rather than assumed.

E Hyperparameters

This section lists the shared allocation settings and task-specific training schedules. The calibration-set size counts examples, whereas T limits the valid tokens retained per module and batch.

E.1 Shared Allocation and Optimization Settings

Table 5 gives the default settings; task-specific values and sensitivity sweeps override these where stated.

For routed modules, expert input factors and routers are initialized from zero-mean Gaussian distributions, while all expert output factors are initialized to zero. Standard LoRA adapters retain or

Table 5: Shared BLAde allocation and optimization hyperparameters.

Hyperparameter	Value
Calibration examples	256
Target modules	`q_proj`, `k_proj`, `v_proj`, `up_proj`, `down_proj`
Marginal sketch width s	64
Joint coordinates per factor p	8
Joint sketch width p^2	64
Tokens per module per batch T	64
MoE fraction ρ	0.20
Effective-rank gate quantile η	0.50
Candidate expert counts $\mathcal{E}$	$\{2, 4, 8, 16\}$
Initial-rank multiplier μ	1.25
Minimum ordinary rank $r_{\min}$	$\max(2, \lfloor r_{\text{ref}}/8 \rfloor)$
LoRA dropout	0
Expert-similarity coefficient λ_{sim}	10^{-3}
Router-balance coefficient λ_{bal}	0
Router learning rate	5×10^{-6}
Optimizer / scheduler	AdamW / linear
Weight decay	0.1
Random seed in current commands	25252

truncate the corresponding rows of the temporary rank-r_{init} initialization, with their output factors also initialized to zero. All experiments are conducted in BF16 precision.

E.2 Reasoning-Task Settings

Table 6 lists sequence lengths, batch sizes, training durations, and backbone-specific learning rates for the eight reasoning tasks.

Table 6: Reasoning-task hyperparameters for BLAde. Batch size is per device; gradient accumulation is one. The LoRA target rank is $r_{\text{ref}} = 32$ and $\alpha = 64$ in the main comparison.

	ARC-C	ARC-E	BoolQ	HellaS.	OBQA	PIQA	SIQA	WinoG.
Max. length	200	200	900	360	128	240	128	240
Batch size	8	8	8	4	8	4	8	8
Epochs	14	5	4	3	12	5	3	3
Warmup steps	200	200	800	2000	500	400	500	500
LR, LLaMA2-7B	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	8×10^{-5}	8×10^{-5}	7×10^{-5}
LR, LLaMA3-8B	8×10^{-5}	8×10^{-5}	8×10^{-5}	6×10^{-5}	8×10^{-5}	6×10^{-5}	9×10^{-5}	8×10^{-5}
LR, LLaMA3.2-3B	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}	10^{-4}

E.3 Generation-Task Settings

Table 7 gives the generation and training settings. HumanEval is evaluated after training on MBPP, as in the main experiments.

Table 7: Generation-task hyperparameters for BLAde on LLaMA3-8B, Gemma-7B, and Mistral-7B. The LoRA target rank is $r_{\text{ref}} = 16$, $\alpha = 32$, and 256 calibration examples are used.

	GSM8K	MBPP	HumanEval	NQ
Max. input length	768	1024	1024	256
Max. new tokens	384	512	512	32
Batch size	4	2	4	8
Gradient accumulation	2	2	2	2
Effective batch size	8	4	8	16
Epochs	3	4	4	3
Warmup steps	100	20	20	500
Learning rate	8×10^{-5}	8×10^{-5}	8×10^{-5}	8×10^{-5}

F STATISTICAL SIGNIFICANCE

We select representative baselines, including LoRA Hu et al. (2022), EVA ($\rho = 2$) Paischer et al. (2025), AdaMoLE Liu & Luo (2024), and AlphaLoRA Qing et al. (2024), and report standard deviations and statistical significance versus BLAde on reasoning datasets in Tables 8 and 9, respectively. Although not every improvement is statistically significant, the gains in average performance across the three runs are significant in most comparisons.

Table 8: Standard deviations computed from the accuracies across three runs.

Backbone	Method	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	Avg.
LLaMA2-7B	LoRA	0.839	0.737	0.557	0.656	0.473	0.600	1.168	1.604	0.808
	EVA ($\rho = 2$)	0.700	0.900	0.757	0.945	0.700	0.577	0.436	1.701	0.807
	AdaMoLE	0.833	0.833	0.600	0.462	0.721	1.100	0.321	0.611	0.590
	AlphaLoRA	0.681	0.265	0.300	0.379	0.416	0.666	1.050	0.643	0.031
	BLAde	0.379	0.379	0.493	0.153	0.874	0.252	0.819	0.702	0.313
LLaMA3-8B	LoRA	0.416	0.252	0.231	0.321	0.624	0.981	0.702	1.200	0.513
	EVA ($\rho = 2$)	0.929	0.379	0.231	0.850	0.306	0.200	0.666	0.902	0.258
	AdaMoLE	0.513	0.666	0.611	0.721	0.153	0.416	0.436	0.702	0.153
	AlphaLoRA	0.493	0.755	0.252	0.643	0.551	0.529	0.757	1.114	0.356
	BLAde	0.058	0.681	0.473	0.200	0.709	0.557	0.693	1.026	0.355
LLaMA3.2-3B	LoRA	0.819	1.150	0.551	0.473	0.351	0.208	1.185	1.665	0.741
	EVA ($\rho = 2$)	0.737	0.361	0.321	0.764	0.265	0.289	0.462	0.924	0.109
	AdaMoLE	0.666	0.300	0.458	0.635	0.058	0.929	0.794	0.611	0.076
	AlphaLoRA	0.306	0.252	0.458	0.681	0.721	0.721	0.802	0.924	0.394
	BLAde	0.265	0.208	0.416	0.265	0.200	0.603	0.737	0.693	0.056

Table 9: Illustrative two-sided paired t-test p-values versus BLAde.

Backbone	Baseline	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	Avg.
LLaMA2-7B	LoRA	0.1027	0.0825	0.0942	0.1186	0.0474	0.0441	0.0021	0.2403	0.0370
	EVA ($\rho = 2$)	0.0005	0.0487	0.0158	0.2662	0.0070	0.0004	0.0012	0.1051	0.0033
	AdaMoLE	0.1994	0.1876	0.0488	0.4380	0.3715	0.1473	0.0657	0.2705	0.0369
	AlphaLoRA	0.6720	0.0488	0.0234	0.1660	0.3061	0.0133	0.0376	0.1090	0.0117
LLaMA3-8B	LoRA	0.1181	0.0691	0.2705	0.0533	0.0063	0.0778	0.0102	0.0043	0.0066
	EVA ($\rho = 2$)	0.0164	0.3206	0.0679	0.0373	0.0153	0.0032	0.0011	0.0508	0.0004
	AdaMoLE	0.1185	0.0377	0.0102	0.1515	0.0412	0.0822	0.0249	0.1916	0.0453
	AlphaLoRA	0.0191	0.9715	0.0153	0.0831	0.1189	0.2123	0.0028	0.0010	0.0006
LLaMA3.2-3B	LoRA	0.2029	0.1477	0.0142	0.0125	0.0023	0.1183	0.3331	0.1846	0.0749
	EVA ($\rho = 2$)	0.0013	0.0108	0.0271	0.0083	0.0002	0.0040	0.0001	0.0009	0.0001
	AdaMoLE	0.9561	0.0513	0.3172	0.0827	0.0188	0.0460	0.0019	0.2494	0.0069
	AlphaLoRA	0.0051	0.0210	0.0377	0.3268	0.0820	0.0942	0.0057	0.7551	0.0772

G FULL RESULTS

G.1 FULL RESULTS OF TABLE 1: PROXY–ORACLE AGREEMENT

For each task, we shuffle the training split with a fixed seed and use disjoint sets of 256 examples for forward calibration and the backward oracle. For a linear module, each sampled per-position gradient contribution is $\boldsymbol{g}_t = \boldsymbol{x}_t \otimes \boldsymbol{\delta}_t$. We compute its covariance spectrum without materializing this high-dimensional vector. For n sampled tokens, let $\boldsymbol{X}$ and $\boldsymbol{\Delta}$ have rows $\boldsymbol{x}_t^\top$ and $\boldsymbol{\delta}_t^\top$, respectively. The uncentered sample Gram matrix is

$$\boldsymbol{K} = (\boldsymbol{X}\boldsymbol{X}^\top) \odot (\boldsymbol{\Delta}\boldsymbol{\Delta}^\top), \tag{40}$$

where $\odot$ denotes elementwise multiplication. After subtracting each row mean and column mean, adding back the overall mean, and dividing by $n-1$, its nonzero eigenvalues equal those of the empirical gradient covariance. We sample up to 256 valid tokens per module across oracle batches and use $p = 16$ for the diagnostic joint sketch, compared with $p = 8$ in allocation. Thus this diagnostic evaluates a larger sketch and does not directly validate agreement at $p = 8$. Correlations are computed across the 160 profiled modules. MoE overlap is recall against the oracle top-ρ variance set after the effective-rank gate; rank overlap is recall of the oracle top-ρ effective-rank set, with $\rho = 0.2$. Each overlap score is the fraction of modules in the corresponding gradient-based reference set that are also selected by the forward scores.

Table 10 reports all eight tasks. "Input $\times$ output" multiplies marginal scores, whereas BLAde uses output-only v_l and divides paired-joint effective rank by the product of the marginal effective ranks. Normalized response variance is consistently predictive. The normalized joint score yields positive $\rho_S(q)$ on all eight tasks and improves the mean over the marginal-product baseline from -0.126 to 0.229 in $\rho_S(q)$ and from 0.160 to 0.352 in rank overlap.

Table 10: Full forward–oracle agreement on eight reasoning tasks using LLaMA3-8B. Bold marks the best value within each task and metric; ties are both marked.

Dataset	Signal	$\rho_S(v)$	$\rho_S(q)$	MoE overlap	Rank overlap
ARC-C	$\boldsymbol{x}$ only	-0.273	-0.107	0.000	0.094
	$\bar{\boldsymbol{y}}$ only	**0.658**	-0.119	0.344	0.094
	$\boldsymbol{x} \times \bar{\boldsymbol{y}}$	0.182	-0.105	0.250	0.094
	BLAde	**0.658**	**0.251**	**0.438**	**0.344**
ARC-E	$\boldsymbol{x}$ only	-0.231	-0.174	0.063	0.063
	$\bar{\boldsymbol{y}}$ only	**0.693**	-0.184	**0.469**	0.125
	$\boldsymbol{x} \times \bar{\boldsymbol{y}}$	0.214	-0.180	0.281	0.125
	BLAde	**0.693**	**0.241**	0.438	**0.344**
BoolQ	$\boldsymbol{x}$ only	-0.362	-0.032	0.031	0.188
	$\bar{\boldsymbol{y}}$ only	**0.656**	-0.091	0.375	0.188
	$\boldsymbol{x} \times \bar{\boldsymbol{y}}$	0.067	-0.085	0.188	0.156
	BLAde	**0.656**	**0.311**	**0.406**	**0.438**
HellaS.	$\boldsymbol{x}$ only	-0.176	0.029	0.063	**0.281**
	$\bar{\boldsymbol{y}}$ only	**0.575**	-0.123	0.406	0.188
	$\boldsymbol{x} \times \bar{\boldsymbol{y}}$	0.251	-0.061	0.250	**0.281**
	BLAde	**0.575**	**0.179**	**0.469**	**0.281**
OBQA	$\boldsymbol{x}$ only	-0.272	-0.106	0.063	0.094
	$\bar{\boldsymbol{y}}$ only	**0.683**	0.013	**0.531**	0.219
	$\boldsymbol{x} \times \bar{\boldsymbol{y}}$	0.142	0.017	0.219	0.156
	BLAde	**0.683**	**0.093**	0.438	**0.281**
PIQA	$\boldsymbol{x}$ only	-0.214	-0.013	0.063	0.188
	$\bar{\boldsymbol{y}}$ only	**0.681**	-0.272	0.344	0.188
	$\boldsymbol{x} \times \bar{\boldsymbol{y}}$	0.189	-0.176	0.188	0.188
	BLAde	**0.681**	**0.250**	**0.438**	**0.406**
SIQA	$\boldsymbol{x}$ only	-0.176	-0.057	0.063	0.219
	$\bar{\boldsymbol{y}}$ only	**0.689**	-0.194	0.406	0.188
	$\boldsymbol{x} \times \bar{\boldsymbol{y}}$	0.243	-0.148	0.188	0.156
	BLAde	**0.689**	**0.222**	**0.438**	**0.313**
WinoG.	$\boldsymbol{x}$ only	-0.243	-0.297	0.031	0.094
	$\bar{\boldsymbol{y}}$ only	**0.737**	-0.231	0.438	0.219
	$\boldsymbol{x} \times \bar{\boldsymbol{y}}$	0.231	-0.268	0.156	0.125
	BLAde	**0.737**	**0.283**	**0.469**	**0.406**
Avg.	$\boldsymbol{x}$ only	-0.243	-0.095	0.047	0.152
	$\bar{\boldsymbol{y}}$ only	**0.672**	-0.150	0.414	0.176
	$\boldsymbol{x} \times \bar{\boldsymbol{y}}$	0.190	-0.126	0.215	0.160
	BLAde	**0.672**	**0.229**	**0.441**	**0.352**

G.2 Full Results of Section 4.1: Task-Wise Module Allocations

Figures 5–12 show the complete LLaMA3-8B allocations with $(r_{\text{ref}}, \rho, \mu) = (32, 0.20, 1.25)$, produced from the same 256-example calibration sets used in the reasoning experiments. Columns follow Transformer depth, and rows separate the five eligible projection families. The panels report total stored rank, the LoRA/MoE decision, expert count, rank per expert, and active top-k; dashes in the last three panels denote ordinary LoRA modules, for which expert-specific quantities do not apply. All figures share the same color limits so that allocations can be compared across tasks.

At the fixed MoE fraction $\rho = 0.2$, each task converts 32 of the 160 profiled modules. The maps reveal a recurring preference for `v_proj` and MLP projections, but not a fixed FFN-only rule: BoolQ also selects `q_proj` and `k_proj` projections, and both placement across depth and ordinary-LoRA ranks vary by task. These plots are descriptive outcomes for the reported model, calibration sample, and seed; they are not intended to establish a universal specialization pattern across architectures.

G.3 Full Results of Section 4.3: Allocation and Calibration Sensitivity

All experiments in this section use LLaMA3-8B on the eight reasoning tasks. We examine the target rank r_{ref}, the MoE fraction ρ, and the initial-rank multiplier μ, as well as the calibration-set size. Tables 11–14 report the per-task results.

The rank experiment uses $r_{\text{ref}} \in \{16, 32, 64\}$ and compares LoRA, AdaLoRA, EVA, AdaMoLE, and BLAde. For BLAde, the rank experiment fixes $(\rho, \mu) = (0.20, 1.25)$. The separate ρ and μ sweeps use $r_{\text{ref}} = 32$ and change one hyperparameter at a time around this default. For the rank, ρ, and μ sweeps, the calibration set contains 256 examples per task; calibration examples, token-sampling seeds, and coordinate selections are held fixed across BLAde settings. Allocation is recomputed for each setting, and adapters are trained from initialization with the same task-specific schedule. The remaining settings follow Appendix E.

Task scores are the median accuracies over three runs, in percent. Avg. is the unweighted mean of the eight per-task medians, computed only when all measurements are available. Since allocation can depend on the task, #Params is the task-mean total number of trainable parameters in millions, including routers where present, as logged after initialization. Parameter means are reported only when counts for all eight tasks are available.

Table 11 groups the five methods by the reference rank r_{ref}. For BLAde, changing r_{ref} changes the reference factor budget $P_0 = r_{\text{ref}} \sum_{l \in \mathcal{M}} c_l$; the temporary rank is $r_{\text{init}} = \lceil 1.25 r_{\text{ref}} \rceil$, giving 20, 40, and 80 for the three settings. Thus, comparisons within a rank group assess the methods at the same reference rank, while comparisons across groups assess their response to a larger budget. The realized parameter count must also be considered because adaptive ranks and router overhead can yield different totals at the same reference rank.

Table 11: Rank sensitivity on LLaMA3-8B. Each rank group compares LoRA, AdaLoRA, EVA, AdaMoLE, and BLAde. For BLAde, $\rho = 0.20$ and $\mu = 1.25$. Task scores and Avg. are accuracies (%); #Params is the task-mean trainable parameter count (M), including routers.

r_{ref}	Method	#Params	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	Avg.
16	LoRA	28.3M	75.1	89.3	82.6	95.6	88.2	92.5	79.4	87.8	86.3
	AdaLoRA	28.3M	69.0	83.9	76.8	89.8	73.3	92.3	80.5	84.2	81.2
	EVA ($\rho_{\text{EVA}} = 2$)	21.4M	70.9	88.1	82.1	94.9	82.2	87.9	73.2	85.2	83.1
	AdaMoLE	42.5M	75.8	90.1	82.4	95.4	87.1	92.1	82.1	88.0	86.6
	BLAde	29.6M	76.1	90.3	83.0	96.0	89.5	93.9	81.6	88.8	87.4
32	LoRA	56.6M	75.4	88.2	82.1	95.4	88.1	92.8	81.0	88.4	86.4
	AdaLoRA	56.6M	69.9	85.6	79.9	92.5	83.0	92.6	80.9	85.0	83.7
	EVA ($\rho_{\text{EVA}} = 2$)	43.2M	71.3	89.0	81.3	94.4	87.5	88.6	73.2	86.6	84.0
	AdaMoLE	75.0M	75.5	90.4	82.5	95.3	87.5	91.7	81.8	88.6	86.7
	BLAde	58.0M	76.7	90.3	83.2	96.3	90.1	93.2	82.3	89.2	87.7
64	LoRA	113.2M	75.6	87.7	82.4	95.7	88.5	93.1	80.7	87.4	86.4
	AdaLoRA	113.2M	71.0	86.3	80.0	91.9	79.6	92.5	81.1	86.0	83.6
	EVA ($\rho_{\text{EVA}} = 2$)	88.7M	71.7	88.7	81.6	94.6	87.6	90.1	78.7	85.4	84.8
	AdaMoLE	140.1M	75.7	90.0	82.1	95.2	88.0	92.1	80.7	88.2	86.5
	BLAde	115.6M	76.4	89.9	83.0	96.2	89.4	93.7	82.1	89.8	87.6

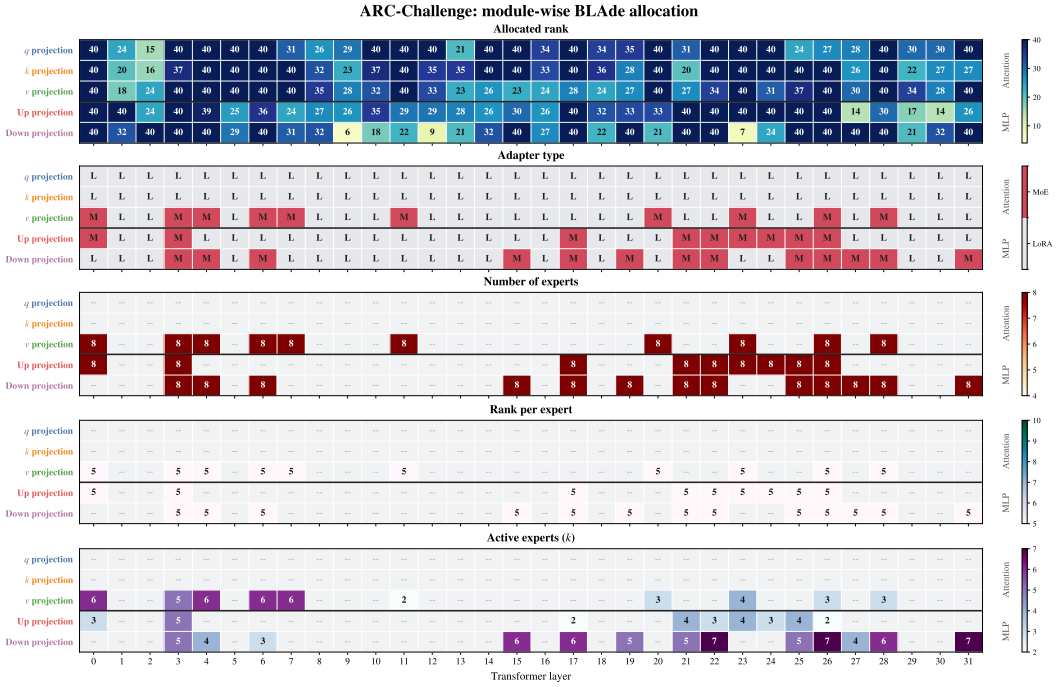


Figure 5: Module-wise BLAde allocation for ARC-Challenge. Colors and cell values show the allocation assigned to every projection at every layer.

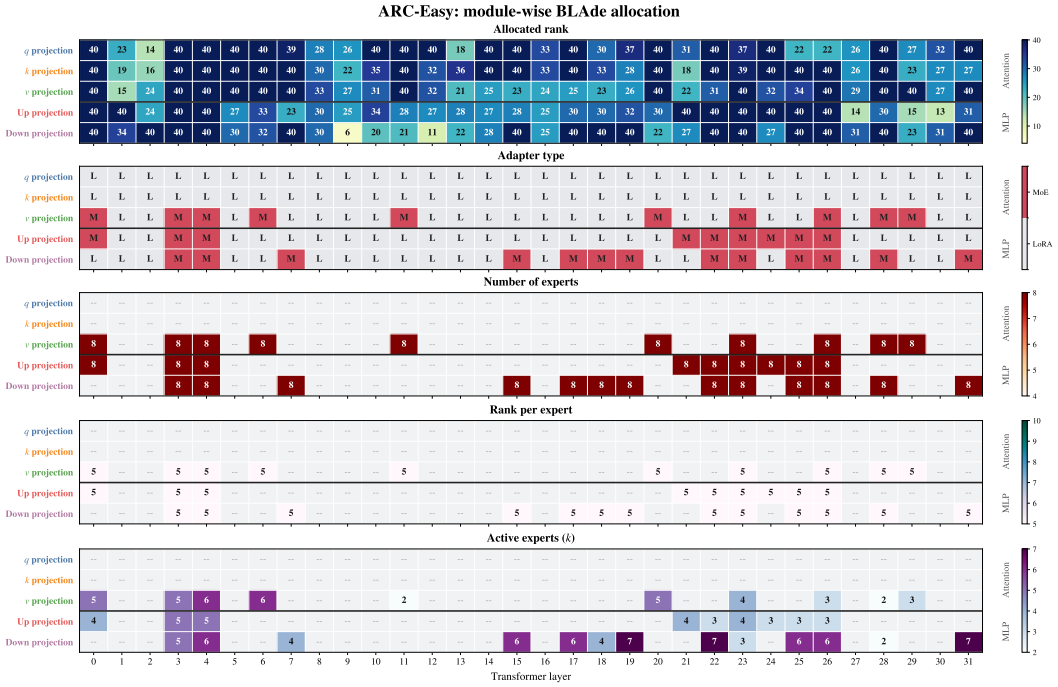


Figure 6: Module-wise BLAde allocation for ARC-Easy.

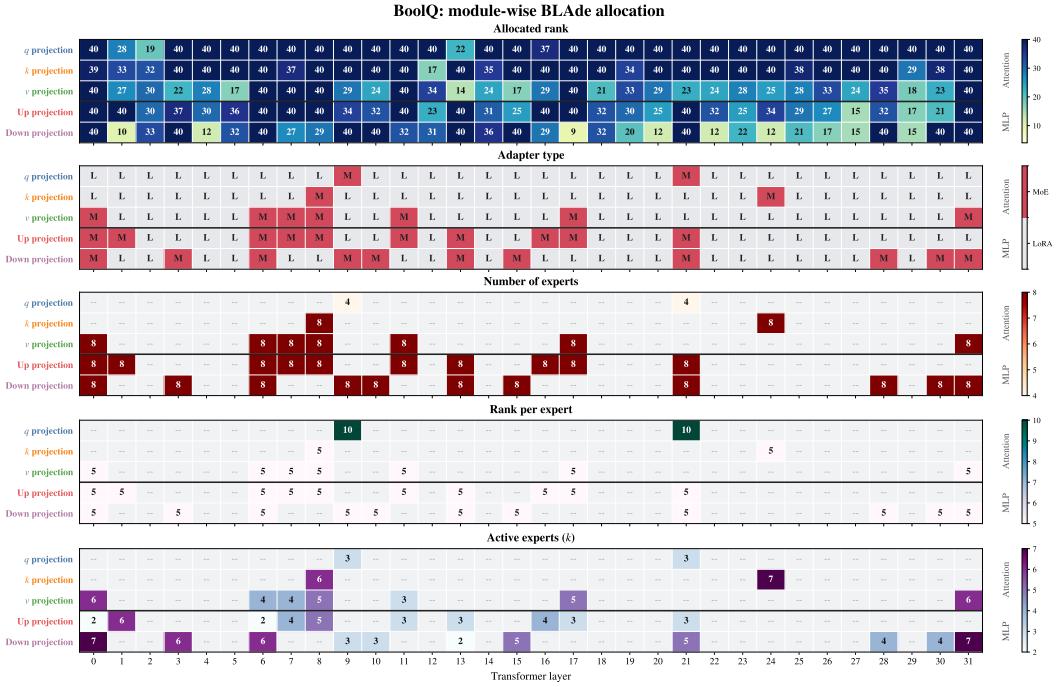


Figure 7: Module-wise BLAde allocation for BoolQ. In contrast to most other tasks, the selected MoE set includes several `q_proj` and `k_proj` projections.

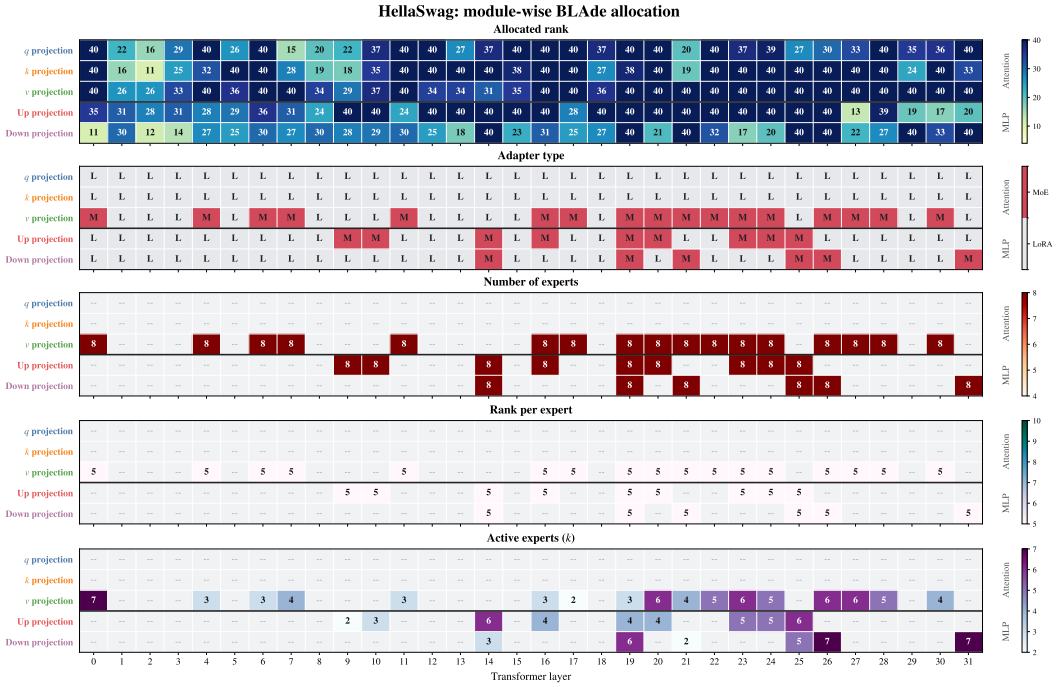


Figure 8: Module-wise BLAde allocation for HellaSwag.

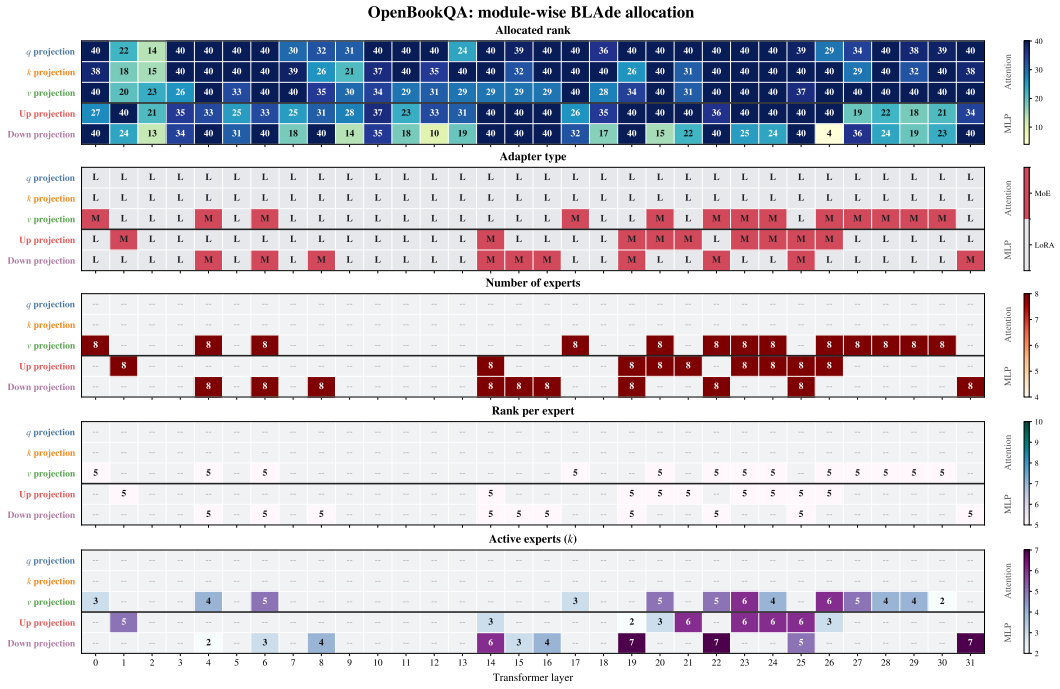


Figure 9: Module-wise BLAde allocation for OpenBookQA.

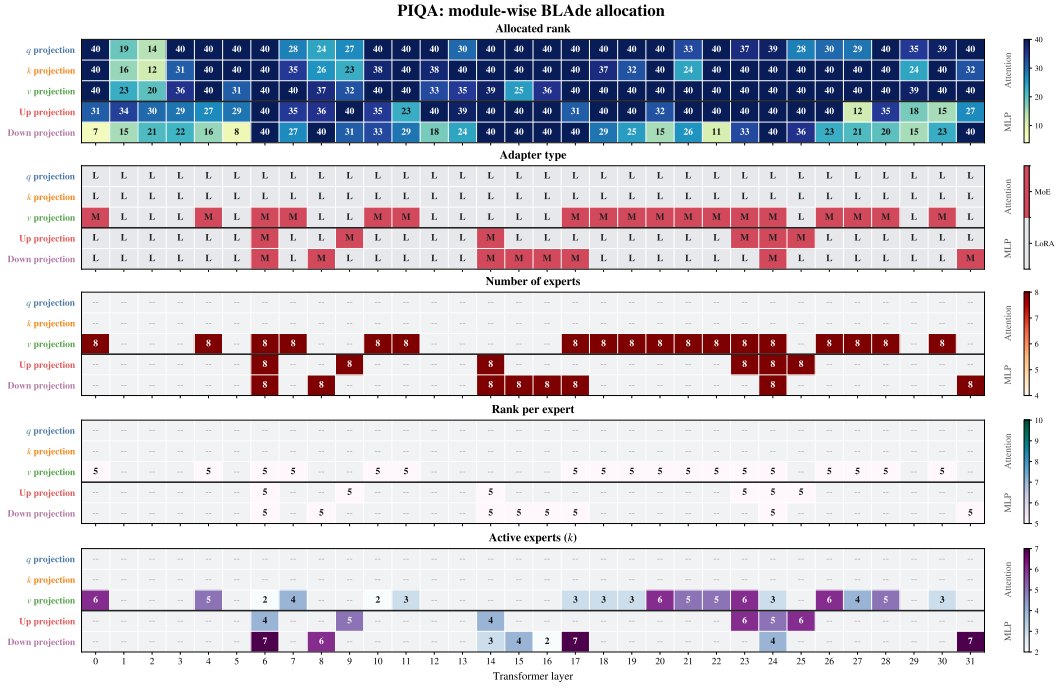


Figure 10: Module-wise BLAde allocation for PIQA.

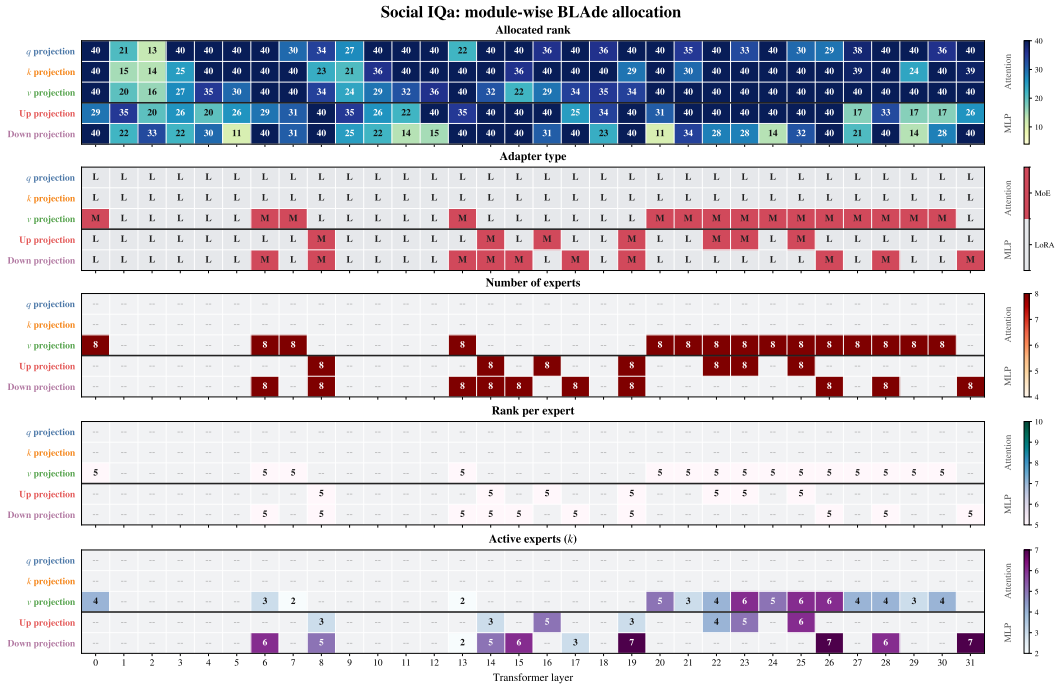


Figure 11: Module-wise BLAde allocation for SIQA.

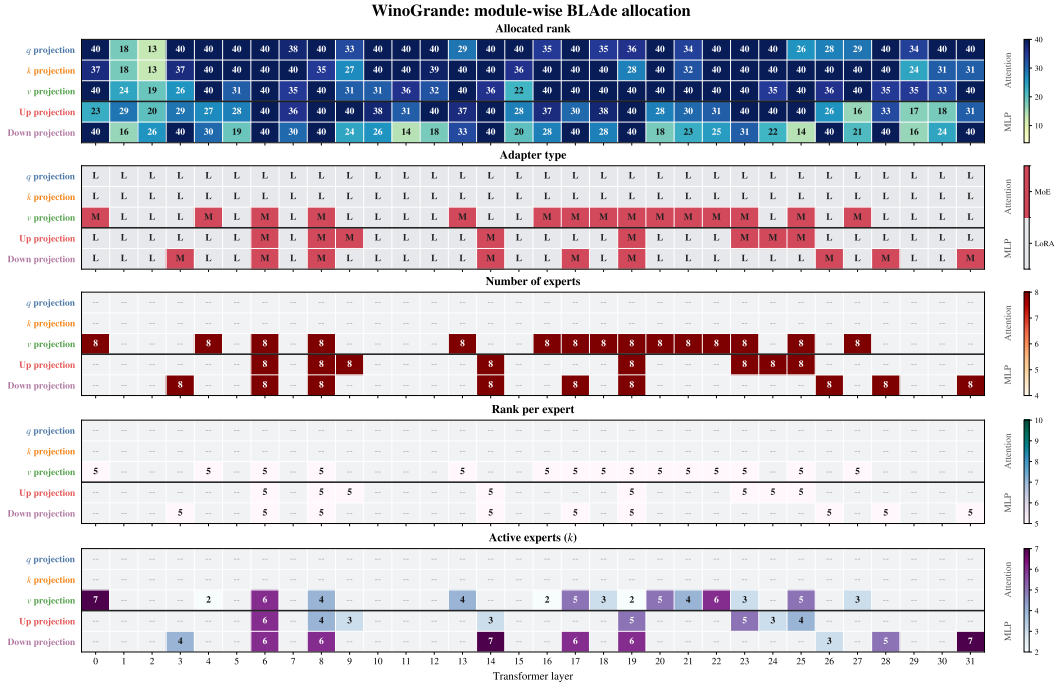


Figure 12: Module-wise BLAde allocation for WinoGrande.

The ρ sweep uses $\rho \in \{0, 0.10, 0.20, 0.30, 0.40\}$ with $r_{\text{ref}} = 32$ and $\mu = 1.25$. At $\rho = 0$, no module is routed; the remaining procedure still allocates ordinary LoRA ranks according to the proxy scores, making this an adaptive ordinary-LoRA control. Increasing ρ allows more modules to receive routed capacity, subject to the effective-rank gate and budget feasibility. The realized MoE fraction can therefore be smaller than the requested fraction. Table 12 isolates this change in placement under a fixed reference factor budget.

Table 12: MoE-fraction sensitivity of BLAde on LLaMA3-8B with $r_{\text{ref}} = 32$ and $\mu = 1.25$. Task scores and Avg. are accuracies (%). The dagger marks the default.

ρ	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	Avg.
0.00	75.7	89.7	82.8	95.7	88.6	92.9	81.5	89.0	87.0
0.10	75.5	90.1	83.4	96.1	89.6	92.6	81.2	88.8	87.2
0.20†	76.7	90.3	83.2	96.3	90.1	93.2	82.3	89.2	87.7
0.30	76.5	90.9	83.2	95.9	89.7	93.1	82.1	89.4	87.6
0.40	75.8	89.9	82.7	96.1	89.5	92.9	82.2	88.6	87.2

The μ sweep uses $\mu \in \{1.0, 1.125, 1.25, 1.375, 1.5\}$ with $r_{\text{ref}} = 32$ and $\rho = 0.20$. These values give temporary ranks $r_{\text{init}} \in \{32, 36, 40, 44, 48\}$, respectively, as shown in Table 13. The reference budget continues to use r_{ref}, rather than r_{init}. A larger temporary rank reserves more factor capacity per routed module and leaves less capacity for ordinary adapters if the routed set is unchanged. When minimum ordinary ranks become infeasible, the allocator changes routed candidates back to ordinary LoRA as specified in Appendix B.

The divisibility constraint on r_{init} also changes the admissible expert counts. Differences between settings can therefore reflect changes in expert count, rank per expert, ordinary rank, or the realized routed set.

Table 13: Rank-inflation sensitivity of BLAde on LLaMA3-8B with $r_{\text{ref}} = 32$ and $\rho = 0.20$. Task scores and Avg. are accuracies (%). The dagger marks the default.

μ	r_{init}	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	Avg.
1.000	32	75.6	89.3	82.4	95.9	89.3	92.7	80.9	89.4	86.9
1.125	36	76.3	90.4	82.4	96.3	90.0	93.4	81.3	89.0	87.4
1.250†	40	76.7	90.3	83.2	96.3	90.1	93.2	82.3	89.2	87.7
1.375	44	76.3	90.4	83.5	96.4	89.9	93.6	83.1	90.0	87.9
1.500	48	76.0	90.1	82.9	96.0	89.2	93.1	80.5	88.8	87.1

The rank sweep varies the reference budget, whereas the ρ and μ sweeps vary its allocation. These separate sweeps measure local sensitivity around the default, not the optimal joint $(r_{\text{ref}}, \rho, \mu)$ configuration.

Table 14 reports the effect of the number of calibration examples on downstream fine-tuning accuracy. We vary the calibration-set size $N_{\text{cal}} \in \{8, 16, 32, 64, 128, 256, 512, 1024\}$ while fixing $(r_{\text{ref}}, \rho, \mu) = (32, 0.20, 1.25)$ on LLaMA3-8B. This changes the data used to collect initialization statistics and allocate adapters; the task-specific fine-tuning schedule is kept fixed. The 256-example row reproduces the default setting.

Table 14: Calibration-set-size sensitivity of BLAde on LLaMA3-8B with $r_{\mathrm{ref}} = 32$, $\rho = 0.20$, and $\mu = 1.25$. N_{cal} is the number of examples used for calibration. Task scores and Avg. are fine-tuning accuracies (%). The dagger marks the default.

N_{cal}	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	Avg.
8	75.7	90.1	82.9	95.7	89.3	93.3	81.9	88.6	87.2
16	75.9	90.3	82.4	96.0	89.6	93.9	82.6	89.0	87.5
32	75.3	91.0	82.3	96.2	88.9	93.9	82.6	90.0	87.5
64	76.6	90.0	82.8	96.3	89.7	93.4	83.4	89.6	87.7
128	75.6	89.7	83.5	96.0	90.3	93.5	82.1	89.4	87.5
$256^{\dagger}$	76.7	90.3	83.2	96.3	90.1	93.2	82.3	89.2	87.7
512	76.6	90.6	83.1	96.1	89.5	93.5	82.0	88.8	87.5
1024	76.7	90.5	83.3	96.4	89.8	93.6	82.3	90.2	87.9