QUERYABLE LORA: INSTRUCTION-REGULARIZED ROUTING OVER SHARED LOW-RANK UPDATE ATOMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank adaptation (LoRA) applies the same learned update to every input, even when an adapter must perform multiple tasks. Queryable LoRA is a parameter-efficient fine-tuning (PEFT) approach that adapts its low-rank update to each input by retrieving a sparse mixture of rank-space atoms from a shared memory. A block uses its current representation and a summary of earlier blocks to select the atoms. In the Instruction-Queryable LoRA (IQ) approach, an instruction embedding also guides the query and defines a prior over atoms. We show that the router solves a KL-regularized retrieval problem. We also demonstrate that its updates stay norm-bounded across routing switches and that the blockwise gradients factorize exactly. To test transfer, we trained one adapter on a mixture of source tasks and chose its checkpoint on validation data. We then evaluated it on held-out tasks with unseen instructions. Across model sizes, Queryable LoRA matched or exceeded LoRA in most paired runs. In the main comparison, both variants exceeded every baseline in average accuracy at a training cost close to LoRA's.

1 INTRODUCTION

Low-rank adaptation (LoRA) learns a compact correction to a frozen weight matrix (Hu et al., 2022), and the same correction is applied to every input. When one adapter has to serve several tasks, different inputs may need different corrections, and the right correction for a layer can also depend on what earlier layers have computed. Dynamic adapters provide this flexibility through input-dependent experts or generated parameters (Luo et al., 2024; Charakorn et al., 2025; Abdalla et al., 2025), often at the price of many more trainable parameters or much longer training. We ask whether a small shared memory can reach the accuracy of strong dynamic adapters while keeping runtime close to standard LoRA.

We answer it with a new adapter design. Queryable LoRA keeps the usual low-rank factors at every adapted projection and adds a bank of small matrices, called atoms, shared across layers and projections. At the start of each block of layers, a router uses the current rank-space state and summaries of earlier blocks to select and mix atoms from the bank. Each adapted projection applies this mixture to its compressed activation and uses a learned gate to scale the resulting correction inside the LoRA bottleneck. The shared bank lets different blocks reuse atoms, while the summaries let later routers condition their choices on what earlier blocks have computed. Instruction-Queryable (IQ) LoRA also uses an instruction embedding to guide the query and place a prior on the learned atoms. Our theory explains the router through KL-regularized retrieval and gives an update-norm bound and exact blockwise gradient identities.

To test whether this design helps when the task changes, we trained a single adapter on a mixture of source tasks with task-specific instructions and chose each method's checkpoint on the same validation set. We then tested transfer to held-out benchmarks with unseen instructions. Against static and dynamic baselines, IQ had the best mean rank, and both variants scored above LoRA and DoRA on every benchmark (Table 1). They also exceeded every baseline in average accuracy at close to LoRA's training time (Figure 2). The advantage over LoRA held up in seed-by-seed comparisons across model sizes and in a larger evaluation on the full test sets.

The ablation study shows where this benefit comes from, since removing the summary of earlier blocks brought accuracy down to about the level of LoRA (Table 5). At inference, the added prefill

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.

time was a few milliseconds for long prompts, and its share of the total shrank as prompts and batches grew (Section 5.4).

2 Related Work

Several methods have attempted to modify LoRA to address the limits of a single fixed update. For example, magnitude-direction variants restructure the low-rank matrices to improve optimization dynamics (Liu et al., 2024), and RepLoRA reparameterizes the LoRA matrices to learn them from less data (Truong et al., 2025). MoSLoRA adds a trainable mixing matrix between the two low-rank factors (Wu et al., 2024). Sharing-based variants reduce redundant local structure across layers (Song et al., 2024; Li et al., 2024). While these methods improve training efficiency, their final adapters remain static. They do not allow the low-rank transformation to adapt jointly to the current hidden state and the preceding depth-wise computation inside the forward pass. Queryable LoRA shares small matrices in rank space and retrieves their mixture using the current input and computations from earlier blocks.

Alternatively, mixture-of-experts approaches like MixLoRA dynamically construct input-tailored low-rank matrices to mitigate task interference (Shen et al., 2024). MoE-LoRA and Lily use expert or interconnected adapter structures to make adaptation input-dependent (Luo et al., 2024; Zhong et al., 2025). Methods such as Text-to-LoRA and ZHyper generate adapter parameters from conditioning information (Charakorn et al., 2025; Abdalla et al., 2025; Liu et al., 2026), and conditioning can also guide the fusion of trained LoRAs (Shukla et al., 2026). Other work adapts the low-rank capacity itself by allocating rank during training (Su et al., 2026) or by learning when a query needs extra reasoning (Zhang et al., 2026). In contrast to static LoRA, parameter-heavy hypernetworks, and methods that synthesize weights directly from text, our approach dynamically assembles updates from a globally shared memory of rank-space atoms.

3 Queryable Low-Rank Adaptation

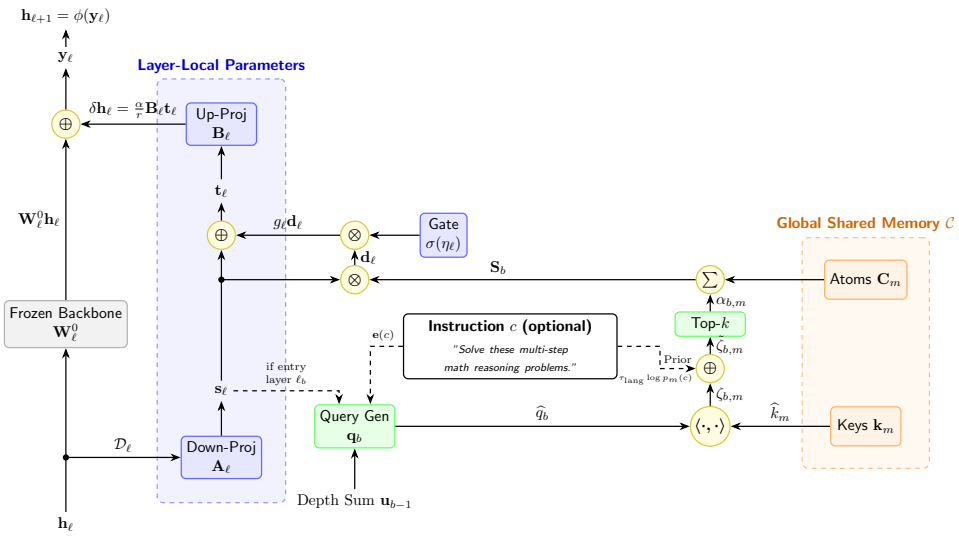


Figure 1: Overview of Queryable LoRA. A block builds its query from the current rank-space state and a summary of earlier blocks, then retrieves a mixture of shared atoms. The resulting operator acts inside the LoRA bottleneck through a learned gate. In IQ, the instruction embedding enters the query and defines a prior over atoms.

Shared rank-space updates. For a frozen projection $\mathbf{W}_\ell \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, let $\mathbf{A}_\ell \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\mathbf{B}_\ell \in \mathbb{R}^{d_{\text{out}} \times r}$ be trainable low-rank factors. To share the cost of routing and encourage consistent

structural adaptations, we group consecutive layers into routing blocks so that the router selects one atom mixture at block entry and all adapted projections in the block reuse it. Layer ℓ belongs to routing block B_b, and all adapted projections and layers share a bank of M trainable atoms $\mathbf{C}_m \in \mathbb{R}^{r\times r}$ (Figure 1). Given input state $\mathbf{h}_\ell$ and instruction c, the update is

$$\begin{aligned}\Delta\mathbf{W}^\ell(\mathbf{h}_\ell; b, c) &= \frac{\alpha_{\mathrm{L}}}{r}\mathbf{B}_\ell[\mathbf{I}_r + g_\ell \mathbf{S}_b(c)]\mathbf{A}_\ell \\ \mathbf{S}_b(c) &= \sum_{m\in I_b} \alpha_{b,m}(c)\mathbf{C}_m \\ g_\ell &= \sigma(\eta_\ell)\end{aligned} \tag{1}$$

Here α_{L} is the LoRA scaling and η_ℓ is a learned gate parameter. The set I_b contains the top-k atoms, and their routing weights are nonnegative and sum to one. The layer output is $\mathbf{W}_\ell\mathbf{h}_\ell + \Delta\mathbf{W}^\ell\mathbf{h}_\ell$, and each block retrieves its operator once and reuses it across its projections.

State and depth query. The operator $\mathbf{S}_b(c)$ is computed once per block using a router conditioned on the current block-entry state, the depth summary of earlier blocks, and, optionally, the external language instruction c. Let $\mathbf{s}_{\mathrm{entry}} \in \mathbb{R}^r$ be the rank-space state pooled over the sequence at block entry, and let $\bar{\mathbf{s}}_i \in \mathbb{R}^r$ summarize preceding block i. A frozen encoder maps the task instruction to $\mathbf{e}(c) \in \mathbb{R}^{d_e}$, which enters the d_k-dimensional query space through trainable projections,

$$\begin{aligned}\mathbf{q}_b^{(0)} &= \mathbf{w}_{\ell_b} + \mathbf{Q}_{\mathrm{cur}}\mathbf{s}_{\mathrm{entry}} + \lambda_{\mathrm{ctx}}\mathbf{Q}_{\mathrm{ctx}}\mathbf{e}(c) \\ \mathbf{q}_b &= \mathbf{q}_b^{(0)} + \mathbf{Q}_{\mathrm{dep}}\mathbf{u}_{b-1}^{\mathrm{att}} \\ \mathbf{u}_{b-1}^{\mathrm{att}} &= \sum_{i<b} \beta_{i\to b}\bar{\mathbf{s}}_i\end{aligned} \tag{2}$$

To form the depth summary, Queryable averages the summaries of earlier blocks in the same forward pass. IQ instead compares learned projections of the preliminary query $\mathbf{q}_b^{(0)}$ and each earlier block's summary, then applies softmax to compute the weights $\beta_{i\to b}$ for the current input. The router adds the projected summary to $\mathbf{q}_b^{(0)}$ to form $\mathbf{q}_b$; both variants use a zero summary for the first block (Appendix A gives the projections).

Instruction-regularized retrieval. The block then compares its query with a learned key $\mathbf{k}_m \in \mathbb{R}^{d_k}$ for each atom. Write $\widehat{\mathbf{v}} = \mathrm{RMSNorm}(\mathbf{v})$. Positive temperatures $T_{\mathrm{attn}}, T_{\mathrm{lang}}$ control retrieval. The state logits and instruction prior are

$$\begin{aligned}\zeta_{b,m} &= \frac{\langle \widehat{\mathbf{q}}_b, \widehat{\mathbf{k}}_m\rangle}{\sqrt{d_k}T_{\mathrm{attn}}} \\ p_m(c) &= \mathrm{softmax}_m\left(\frac{\langle \widehat{\mathbf{R}_{\mathrm{ctx}}\mathbf{e}(c)}, \widehat{\mathbf{k}}_m\rangle}{\sqrt{d_k}T_{\mathrm{lang}}}\right) \\ \widetilde{\zeta}_{b,m} &= \zeta_{b,m} + \tau_{\mathrm{lang}}\log p_m(c)\end{aligned} \tag{3}$$

In IQ, the router adds the projected instruction embedding $\lambda_{\mathrm{ctx}}\mathbf{Q}_{\mathrm{ctx}}\mathbf{e}(c)$ to the query and uses a separate projection of the embedding to compute the prior $p_m(c)$ over the same atom bank. It adds $\tau_{\mathrm{lang}}\log p_m(c)$ to each atom's query–key score, so a higher prior probability favors that atom when the query–key scores are equal. The router then selects the k largest combined scores, $I_b = \mathrm{TopK}_m(\widetilde{\zeta}_{b,m})$, and applies softmax over I_b to obtain the mixture weights.

This routing distribution solves the KL-regularized objective in Theorem 4.1.

Limiting cases and usage. Setting $g_\ell = 0$ in Equation (1) recovers LoRA, and setting $\tau_{\mathrm{lang}} = \lambda_{\mathrm{ctx}} = 0$ gives state-and-depth routing with the same depth-pooling rule (Proposition B.7). We use instruction regularization for zero-shot or cross-task transfer, where evaluation instructions differ from training instructions. With a single fixed instruction, the prior reduces to a constant task-level bias. The main Queryable and IQ comparisons use the pooling choices above (Appendix F tests the instruction input with the other model components fixed).

4 THEORETICAL RESULTS

Queryable's router uses query–key scores to favor atoms that match the block's query and the instruction prior to favor atoms associated with the task description. For a fixed active set, its softmax directly computes the unique weights that maximize the average query–key score minus KL divergence from the tempered instruction prior (Theorem 4.1).

We use the update, routed operator, block query, language prior, and joint routing logits from §3. For a block b, let $I = I_b \subseteq [M]$ denote the active atom set used by the router, with $I = [M]$ for dense softmax. We write $\boldsymbol{\alpha}_{b,I}(c) \in \Delta(I)$ for the routing distribution restricted to I, where $\Delta(I)$ is the probability simplex over I. Since the router uses logits $\widetilde{\zeta}_{b,m}(c) = \zeta_{b,m} + \tau_{\text{lang}} \log p_m(c)$, define the corresponding tempered instruction prior on I by $\boldsymbol{\pi}^{(\tau)}_{I,m}(c) = \frac{p_m(c)^{\tau_{\text{lang}}}}{\sum_{j\in I} p_j(c)^{\tau_{\text{lang}}}}$ for $m \in I$. Here $\boldsymbol{\zeta}_{b,I}$ collects the state logits on I, $\mathrm{KL}(\cdot\|\cdot)$ is KL divergence on $\Delta(I)$, and $H(\boldsymbol{a}) = -\sum_{m\in I} a_m \log a_m$. All norms stated are operator norms unless explicitly marked by $\|\cdot\|_F$. The norm statement assumes bounded atoms and factors, $\|\mathbf{C}_m\| \le R_C$, $\|\mathbf{A}_\ell\| \le R_A$, $\|\mathbf{B}_\ell\| \le R_B$, and bounded block summaries $\|\bar{\mathbf{s}}_i\| \le R_s$. For the gradient identities, during a fixed forward pass we write $\mathbf{d}_\ell = \mathbf{S}_b(c)\mathbf{s}_\ell$, $\mathbf{t}_\ell = \mathbf{s}_\ell + g_\ell \mathbf{d}_\ell$, and $\mathbf{r}_\ell = \nabla_{\mathbf{t}_\ell}\mathcal{L}$.

Theorem 4.1 (Variational characterization of instruction-regularized retrieval)**.** *Fix a block b, instruction c, and active set I. The routing distribution $\boldsymbol{\alpha}_{b,I}(c)$ defined in* (29) *is the unique solution of*

$$\boldsymbol{\alpha}_{b,I}(c) = \arg\max_{\boldsymbol{a}\in\Delta(I)} \Big\{ \langle \boldsymbol{a}, \boldsymbol{\zeta}_{b,I}\rangle - \mathrm{KL}\Big(\boldsymbol{a} \,\Big\|\, \boldsymbol{\pi}^{(\tau)}_I(c)\Big)\Big\} \tag{4}$$

where $\Delta(I)$ is the probability simplex on I. Equivalently,

$$\boldsymbol{\alpha}_{b,I}(c) = \arg\max_{\boldsymbol{a}\in\Delta(I)} \big\{ \langle \boldsymbol{a}, \boldsymbol{\zeta}_{b,I} + \tau_{\text{lang}} \log \mathbf{p}_I(c)\rangle + H(\boldsymbol{a})\big\} \tag{5}$$

where $H(\boldsymbol{a}) = -\sum_{m\in I} a_m \log a_m$.

When the state query and instruction prior favor different atoms, the prior probability of a state-preferred atom bounds the resulting gap in state utility (Corollary 4.1.1).

Corollary 4.1.1 (State-prior tradeoff bound)**.** *Let $m_b^\star \in \arg\max_{m\in I} \zeta_{b,m}$. Under the hypotheses of Theorem 4.1,*

$$0 \le \max_{m\in I} \zeta_{b,m} - \langle \boldsymbol{\alpha}_{b,I}(c), \boldsymbol{\zeta}_{b,I}\rangle \le \log \frac{1}{\pi^{(\tau)}_{I,m_b^\star}(c)} \tag{6}$$

The router averages the selected atoms with nonnegative weights that sum to one, so the mixture's norm cannot exceed the largest atom norm. Each adapted projection combines this mixture with its low-rank factors and a gate between zero and one; the assumed bounds on the atoms and factors therefore bound the resulting update even when the router selects different atoms (Theorem 4.2). For depth conditioning, the router also averages bounded block summaries, so the resulting summary cannot have a larger norm than the largest block summary.

Theorem 4.2 (Norm-controlled dynamic updates and bounded depth summaries)**.** *Under Assumptions B.1 and B.2, the following statements hold for every instruction c and block b.*

1. *The routed operator belongs to the convex hull of the atom bank, $\mathbf{S}_b(c) \in \mathrm{conv}\{\mathbf{C}_1, \dots, \mathbf{C}_M\}$, and its operator norm is uniformly bounded, $\|\mathbf{S}_b(c)\| \le R_C$.*

2. *For every adapted layer $\ell \in B_b$,*

$$\|\Delta\mathbf{W}^\ell(\mathbf{h}_\ell; b, c)\| \le \frac{\alpha_{\mathrm{L}}}{r} \|\mathbf{B}_\ell\| \,(1 + g_\ell R_C)\, \|\mathbf{A}_\ell\| \le \frac{\alpha_{\mathrm{L}}}{r} R_B(1 + R_C) R_A \tag{7}$$

   Consequently, for every vector $\mathbf{x}$,

$$\|\Delta\mathbf{W}^\ell(\mathbf{h}_\ell; b, c)\mathbf{x}\| \le \frac{\alpha_{\mathrm{L}}}{r} R_B(1 + R_C) R_A \,\|\mathbf{x}\| \tag{8}$$

3. *If the attention-style depth summary is given by $\mathbf{u}^{\mathrm{att}}_{b-1} = \sum_{i=1}^{b-1} \beta_{i\to b} \bar{\mathbf{s}}_i$, where, $\beta_{i\to b} \ge 0$, and $\sum_{i=1}^{b-1} \beta_{i\to b} = 1$. Then*

$$\|\mathbf{u}^{\mathrm{att}}_{b-1}\| \le \max_{1\le i\le b-1} \|\bar{\mathbf{s}}_i\| \le R_s \tag{9}$$

Backpropagation sums the gradient contributions from the layers that use a block's routed operator. It multiplies this sum by each retrieved atom's routing weight to obtain that atom's direct value-path gradient from the block (Theorem 4.3). Backpropagation also accumulates each atom's contributions across blocks, and the optimizer uses the resulting gradients to update the shared bank. Thus, multiple layers and blocks can train the same update component.

Theorem 4.3 (Exact blockwise gradient factorization)**.** *Consider $\mathbf{S}_b(c)$ as the blockwise routed operator used by all layers in B_b. Then*

$$\nabla_{\mathbf{S}_b(c)}\mathcal{L} = \sum_{\ell \in B_b} g_\ell \mathbf{r}_\ell \mathbf{s}_\ell^\top \tag{10}$$

Consequently, if the atom values $\{\mathbf{C}_m\}_{m=1}^M$ are independent of the routing keys, then the direct value-path gradients through block b are

$$\nabla_{\mathbf{C}_m}\mathcal{L} = \alpha_{b,m}(c)\nabla_{\mathbf{S}_b(c)}\mathcal{L} \text{ and } \frac{\partial \mathcal{L}}{\partial \alpha_{b,m}(c)} = \left\langle \nabla_{\mathbf{S}_b(c)}\mathcal{L}, \mathbf{C}_m \right\rangle_F \tag{11}$$

$\forall\, m = 1, 2, 3 \ldots, M$. The gate parameter satisfies

$$\frac{\partial \mathcal{L}}{\partial \eta_\ell} = \sigma(\eta_\ell)(1 - \sigma(\eta_\ell)) \left\langle \mathbf{r}_\ell, \mathbf{d}_\ell \right\rangle \tag{12}$$

for $\ell \in B_b$.

For gradient descent taken directly in the active logits, an atom's logit increases when its inner product with the loss gradient is below the routing-weighted average (Corollary 4.3.1). This means stronger alignment with the negative loss gradient relative to that average. Its routing probability also depends on changes in the other active logits.

Corollary 4.3.1 (Exact logit gradients on a fixed active set)**.** *Fix an active set I and define $\psi_{b,m} := \left\langle \nabla_{\mathbf{S}_b(c)}\mathcal{L}, \mathbf{C}_m \right\rangle_F$ and $\bar{\psi}_b := \sum_{j \in I} \alpha_{b,j}(c)\psi_{b,j}$. Let $z_{b,m}(c) := \zeta_{b,m} + \tau_{\text{lang}} \log p_m(c),\ \forall\, m \in I$. Then,*

$$\frac{\partial \mathcal{L}}{\partial z_{b,m}(c)} = \alpha_{b,m}(c)\big(\psi_{b,m} - \bar{\psi}_b\big) \tag{13}$$

Equivalently, writing $p_m(c) = \mathrm{softmax}(\boldsymbol{\rho}(c))_m$, the gradients with respect to the state and raw language logits satisfy:

$$\frac{\partial \mathcal{L}}{\partial \zeta_{b,m}} = \alpha_{b,m}(c)\big(\psi_{b,m} - \bar{\psi}_b\big) \text{ and } \frac{\partial \mathcal{L}}{\partial \rho_m(c)} = \tau_{\text{lang}}\alpha_{b,m}(c)\big(\psi_{b,m} - \bar{\psi}_b\big) \tag{14}$$

Theorem 4.3 and Corollary 4.3.1 give the contribution of block b. The atoms $\mathbf{C}_m$ and the raw language logits $\rho_m(c)$ are shared by all blocks, so their full gradients add these contributions over blocks, for example $\nabla_{\mathbf{C}_m}\mathcal{L} = \sum_b \alpha_{b,m}(c)\nabla_{\mathbf{S}_b(c)}\mathcal{L}$.

Behavior across routing switches. Across top-k switches, the norm bound continues to hold. The Lipschitz and gradient results hold wherever the active set stays fixed (Assumption B.4). Concentrated atom use and low route drift were also observed in the routing experiments (Appendix H).

5 Experiments

5.1 Experimental setup

One adapter, unseen instructions. Our experiments ask whether one adapter trained on a few source tasks can transfer to new tasks and new instructions. We trained each adapter on a mixture of five multiple-choice source tasks, with a task-specific instruction for each task (Table 7 names the tasks). We then tested transfer on held-out subjects from MMLU (Hendrycks et al., 2021a) and held-out categories from MMLU-Pro (Wang et al., 2024), as well as GPQA-Diamond (Rein et al., 2024). We also held out the evaluation instructions during adapter training. Appendix C gives the size of every evaluation set.

Checkpoint selection. Because the target tasks are used only for testing, checkpoints were chosen on the source tasks alone. At the end of each epoch, we evaluated every method on the same 80-example validation set drawn from four source tasks, disjoint from adapter training and target evaluation. We selected the checkpoint with the highest validation accuracy, resolving ties in favor of the later epoch, and used it for both the initial and full-size target evaluations. All main LLM comparisons followed this rule, with three seeds for MMLU and MMLU-Pro and five seeds for GPQA at 7B and 14B (Appendix C).

Implementation. We used Qwen2.5-Instruct models from 1.5B to 14B parameters (Qwen Team, 2024), and Appendix F covers the 0.5B experiments and a separate small-model study. We applied adapters to every attention and MLP projection at rank 8 and scaling 16. Queryable variants used 16 atoms with top-4 routing over blocks of two layers. Table 9 lists the remaining training settings together with the hardware and software, and Appendix C describes the baselines and the instruction encoder.

5.2 Main results

Table 1: Zero-shot transfer at 7B using checkpoints selected on source-task validation data, with evaluation instructions held out during adapter training. Values are mean ± standard deviation over three seeds for MMLU-12 and MMLU-Pro and five for GPQA. Avg. and rank aggregate the three benchmarks, with tied rounded means assigned average ranks. Time covers training and held-out evaluation over 11 runs (Appendix C gives the seeds and the standard-deviation conventions).

Method	MMLU-12	MMLU-Pro	GPQA	Avg.	Rank	Params	Time
LoRA	0.759±.014	0.462±.022	0.333±.034	0.518	6.0	20.19M	1.0×
DoRA	0.741±.019	0.462±.008	0.319±.021	0.507	7.5	21.3M	2.8×
RepLoRA	0.774±.015	**0.487**±.015	0.317±.024	0.526	4.5	667M	1.5×
MoE-LoRA	0.778±.005	0.473±.015	0.325±.015	0.525	4.5	85.3M	4.3×
Lily	0.766±.002	0.480±.010	0.331±.008	0.526	5.0	9.8M	4.3×
ZHyper	**0.779**±.027	0.463±.019	0.333±.020	0.525	3.5	23.7M	1.3×
Queryable LoRA	0.774±.007	0.486±.008	0.341±.033	0.534	2.8	20.24M	1.2×
IQ	0.778±.007	0.483±.017	**0.343**±.016	**0.535**	**2.2**	20.24M	1.2×

Comparison across eight methods. Table 1 compares the two variants with six static and dynamic baselines at 7B. IQ is the only method in the top three on all three benchmarks and has the best mean rank, with Queryable second. Both proposed variants scored above LoRA and DoRA on every benchmark. The strongest baseline changes from one benchmark to the next, which is why the mean rank is a useful summary of consistency. Figure 2 adds training cost to this picture. The two Queryable variants exceed every baseline in average accuracy at close to LoRA's runtime, while the expert-based MoE-LoRA and Lily take several times longer.

Consistency across seeds. Mean accuracies over a few seeds can hide how consistent a difference is, so we also compared Queryable and LoRA run by run. Across eight settings from 1.5B to 14B, Queryable matched or exceeded LoRA in 23 of 27 seed-paired runs (one-sided sign test, $p = 0.0013$). These pairs include two additional sets of MMLU subjects at 7B, and Appendix D gives the counts for each setting, along with the assumptions behind the test.

Full-size evaluation. Small test sets raise a similar question, so we re-evaluated the same adapters on the full test splits of the held-out MMLU subjects and MMLU-Pro categories. On all 9,091 held-out MMLU-Pro questions, Queryable scored 2.7 points above LoRA, with a question-clustered 95% bootstrap interval of $[2.3, 3.1]$. The ordering of the eight methods on MMLU-Pro stayed the same at this size, and IQ and Queryable again have the two best mean ranks (Table 2).

5.3 Scaling with model size

The results so far use a 7B backbone, so we also varied the model size on GPQA. Queryable had a higher mean accuracy than static LoRA on GPQA at every tested scale from 1.5B to 14B and

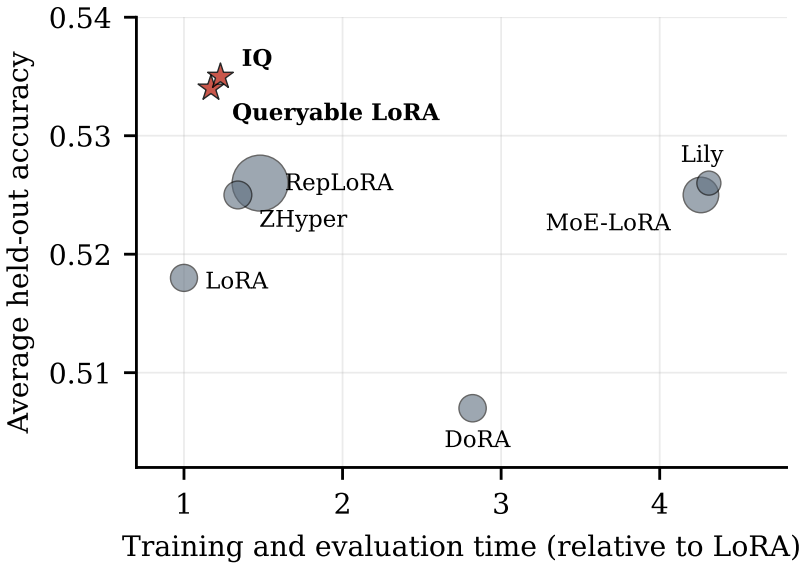


Figure 2: Average held-out accuracy over the three benchmarks against training and evaluation time relative to LoRA at 7B (Table 1). Marker size grows with trainable parameter count. The two Queryable variants exceed every baseline in average accuracy while running close to LoRA's runtime. Seed standard deviations for each benchmark are reported in Table 1.

Table 2: Full-size evaluation of the same 7B adapters selected by validation accuracy, using 3,766 official MMLU-12 test questions and all 9,091 MMLU-Pro questions in the ten held-out categories. GPQA uses the evaluation from Table 1. Values are mean ± standard deviation across three seeds for MMLU-12 and MMLU-Pro and five for GPQA.

Method	MMLU-12 (3,766)	MMLU-Pro (9,091)	GPQA (99)	Mean rank
LoRA	0.671±.007	0.421±.026	0.333±.034	6.2
DoRA	0.668±.009	0.422±.001	0.319±.021	7.3
RepLoRA	0.673±.004	**0.448**±.008	0.317±.024	5.2
MoE-LoRA	0.677±.010	0.437±.010	0.325±.015	4.8
Lily	**0.687**±.005	0.441±.007	0.331±.008	3.3
ZHyper	0.675±.006	0.427±.020	0.333±.020	4.8
Queryable LoRA	0.677±.003	**0.448**±.005	0.341±.033	2.3
IQ	0.682±.002	0.446±.008	**0.343**±.016	**2.0**

matched or exceeded it on all five seeds at 14B (Table 3). The shared bank and routing components add about 58K parameters at every size, so their share of the adapter shrinks as the backbone grows. By comparison, MoE-LoRA and RepLoRA train many more parameters than LoRA (Table 6), and both have lower average accuracy than the two Queryable variants (Table 1).

5.4 Training and inference cost

Training and evaluation time. Parameter counts are only one part of the cost, so we also timed complete runs on the same hardware. The two Queryable variants were the fastest methods after LoRA, and MoE-LoRA and Lily took the longest (Table 4).

Prefill time. We measured Qwen2.5-7B-Instruct on H100 in bf16 with two repetitions of each configuration and applied the low-rank factors at run-time for LoRA and both proposed variants (Table 11). In these measurements, Queryable and IQ added only a few milliseconds of prefill time for long prompts. The backbone performs more computation as prompts and batches grow, while the router launches the same number of kernels per block. This fixed launch count helps explain why the

Table 3: GPQA-Diamond transfer across model sizes using checkpoints selected by validation accuracy, reported as mean ± standard deviation over the stated number of seeds. The shared bank and routing components add about 58K trainable parameters, and the last row gives the overhead relative to LoRA.

	1.5B (n=3)	3B (n=3)	7B (n=5)	14B (n=5)
LoRA	0.290±.041	0.286±.021	0.333±.034	0.404±.007
Queryable LoRA	**0.327**±.006	**0.300**±.025	**0.341**±.033	**0.416**±.013
Param. overhead	0.63%	0.39%	0.29%	0.17%

Table 4: Wall-clock minutes per complete 7B run (from model loading through held-out evaluation), mean ± sample standard deviation over 11 runs, with three each on MMLU-12 and MMLU-Pro and five on GPQA. Ratios use LoRA as reference.

Method	Minutes	Ratio	Method	Minutes	Ratio
LoRA	9.5±.4	1.0×	DoRA	26.8±1.4	2.8×
Queryable	11.1±.7	1.2×	RepLoRA	14.1±.5	1.5×
IQ	11.7±.4	1.2×	MoE-LoRA	40.4±.6	4.3×
ZHyper	12.8±.5	1.3×	Lily	40.9±1.8	4.3×

added prefill time became a smaller percentage of the total for longer prompts and larger batches (Table 11). In a separate comparison with other dynamic adapters, both variants had lower latency than the generated-adapter baselines (Table 12).

Prompt routing and generation. Generation needs one more design choice. During prefill, each block's router selects atoms and mixture weights from the prompt, and we cache that route. At the first decoding step, every block except the last recomputes its route from the new token and the prompt's block summaries. The implementation then reuses these routes for the remaining tokens and resets them for each new request, avoiding further atom selection while still applying the low-rank update at every adapted projection. We timed generation with a 256-token prompt and 128 new tokens at batch one. In this setting, Queryable's throughput was about 29% lower than LoRA's (Appendix E also reports throughput with a route recomputed at each token).

5.5 Ablation study and additional results

Component ablation. We trained separate IQ variants that each remove or change one component, then evaluated them on the same held-out split (Table 5). When we removed the depth summary, the router could no longer consult the explicit summaries of earlier blocks, and mean accuracy fell from 77.0% to 75.2%, close to LoRA's 75.9%. Every variant that retained the summary scored above LoRA and DoRA, supporting the value of this information in this experiment.

Routing at every layer reached the highest accuracy in the ablation and needs twice as many routing calls, so blocks of two layers are the cheaper default. A global bank achieved similar accuracy to block-local banks with slightly fewer parameters (20.24M vs. 20.27M). Every variant used the same training data and checkpoint rule as IQ. The accuracy differences in Table 5 therefore reflect the routing design at a nearly fixed parameter count.

Gradient analysis and small models. To unpack some of the dynamics found in these overall accuracy scores, Appendix G examines the per-layer adapter gradient norms, where the instruction-queryable adapter receives stronger gradient signals across a wider range of layers (Figures 4 and 5). For additional results on a broader battery of $\leq$ 1B models, including Qwen3, LiquidAI LFM2 variants, AMD ReasonLite variants, IBM Granite-4.0-350M, and HuggingFaceTB SmolLM2-360M-Instruct, see Table 16. Here, we select checkpoints with the lowest validation fine-tuning loss, and IQ equals or exceeds LoRA in 29 of 39 cases. The battery also includes three math datasets with open-ended answers, and IQ equals or exceeds LoRA in 18 of their 22 cases. Appendix I tests the same idea on synthetic regression tasks.

Table 5: 7B MMLU-12 ablations using checkpoints selected by validation accuracy (three seeds, mean ± standard deviation). Each variant changes one component of IQ under shared training settings, with atom initialization differing from Table 1 as specified in Appendix C. LoRA and DoRA score $0.759_{\pm.014}$ and $0.741_{\pm.019}$ on the same split.

Variant	Held-out	What it changes
Instruction-Queryable	$0.770_{\pm.011}$	global bank, block 2, depth attention
Per-layer routing (block size 1)	$\mathbf{0.785}_{\pm.005}$	2× routing calls
Block-local atom banks	$0.769_{\pm.004}$	20.27M vs 20.24M parameters
Current-state-only routing	$0.752_{\pm.014}$	removes depth conditioning

6 Conclusion

In conclusion, we introduced a queryable memory of rank-space atoms that lets low-rank adapters retrieve input- and depth-conditioned updates during the forward pass. IQ adds an instruction-based query and prior, so a description of the task can guide which atoms are retrieved. When one adapter was trained on a few source tasks and tested on new ones, both variants reached the accuracy of the strongest dynamic adapters at close to LoRA's training cost. Across model sizes, Queryable also matched or exceeded LoRA in most paired runs, and its added inference time was small for long prompts and larger batches. Open-ended generation and larger models are the next steps for evaluation.

AI USE STATEMENT

We used AI assistants to help draft and edit the text. During the experiments, they helped with experiment design and with writing and debugging the training and evaluation code. They helped find related work and helped write and check parts of the proofs. They also helped launch and monitor runs and write the scripts for aggregation and statistical tests. We used them to interpret results and to check the mathematical notation and the text explaining the theorems. For numerical checks, we recomputed the reported LLM summaries from our run records and compared the synthetic and additional 0.5B results with their source tables. We checked the mathematical notation and gradient-sign explanation against the theorem statements. We reviewed all AI-assisted work and take full responsibility for this paper's content.

ETHICS STATEMENT

Like other fine-tuning methods, Queryable LoRA can support useful customization and harmful fine-tuning. We used existing models and public benchmarks. Deployments should keep standard safeguards and test the adapted model for the intended task, following the access and licensing terms of the underlying assets.

REPRODUCIBILITY STATEMENT

Our code is included in the supplementary material. Appendices A and C give the hyperparameters and training details, including the number of seeds per setting and the checkpoint selection (selection rules for the additional experiments are in Appendix F). All proofs are included in Appendix B.

REFERENCES

M. H. I. Abdalla, Zhipin Wang, Christian Frey, Steffen Eger, and Josif Grabocka. Zhyper: Factorized hypernetworks for conditioned LLM fine-tuning. *arXiv preprint arXiv:2510.19733*, 2025.

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. *arXiv preprint arXiv:2108.07732*, 2021.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-LoRA: Instant transformer adaption. In *International Conference on Machine Learning*, pp. 7485–7514. PMLR, 2025.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019.

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. *arXiv preprint arXiv:1803.05457*, 2018.

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

Nghiem T. Diep, Hien Dang, Tuan Truong, Tan Dinh, Huy Nguyen, and Nhat Ho. DoRAN: Stabilizing weight-decomposed low-rank adaptation via noise injection and auxiliary networks. *arXiv preprint arXiv:2510.04331*, 2025a.

Nghiem T. Diep, Dung Le, Tuan Truong, Tan Dinh, Huy Nguyen, and Nhat Ho. Hypernetwork-driven low-rank adaptation across attention heads. *arXiv preprint arXiv:2510.04295*, 2025b.

Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021b.

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.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1082. URL `https://aclanthology.org/D17-1082`.

Yang Li, Shaobo Han, and Shihao Ji. VB-LoRA: Extreme parameter efficient fine-tuning with vector banks. *Advances in Neural Information Processing Systems*, 37:16724–16751, 2024.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, pp. 32100–32121. PMLR, 2024.

Yewei Liu, Xiyuan Wang, Yansheng Mao, Yoav Gelberg, Haggai Maron, and Muhan Zhang. SHINE: A scalable in-context hypernetwork for mapping context to LoRA in a single pass. In *International Conference on Machine Learning*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

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

Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pp. 1614–1623. PMLR, 2016.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-Math: Unlocking the potential of SLMs in grade school math. *arXiv preprint arXiv:2402.14830*, 2024.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024.

Ying Shen, Zhiyang Xu, Qifan Wang, Yu Cheng, Wenpeng Yin, and Lifu Huang. Multimodal instruction tuning with conditional mixture of LoRA. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 637–648, 2024.

Shreya Shukla, Aditya Sriram, Milinda Kuppur Narayanaswamy, and Hiteshi Jain. qa-FLoRA: Data-free query-adaptive fusion of LoRAs for LLMs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32992–33000, 2026.

Yurun Song, Junchen Zhao, Ian G. Harris, and Sangeetha Abdu Jyothi. ShareLoRA: Parameter efficient and robust large language model fine-tuning via shared low-rank adaptation. *arXiv preprint arXiv:2406.10785*, 2024.

Felix Stollenwerk. On the mathematical relationship between layer normalization and dynamic activation functions. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 674–681, 2026.

Barbara Su, Fangshuo Liao, and Anastasios Kyrillidis. AdaPaD: Adaptive parallel deflation for PEFT with self-correcting rank discovery. *arXiv preprint arXiv:2605.10741*, 2026.

S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved April 8, 2026, from `http://www.sfu.ca/~ssurjano`, 2013.

Tuan Truong, Chau Nguyen, Huy Nguyen, Minh Le, Trung Le, and Nhat Ho. RepLoRA: Reparameterizing low-rank adaptation via the perspective of mixture of experts. In *International Conference on Machine Learning*, pp. 60183–60217. PMLR, 2025.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. SOAP: Improving and stabilizing Shampoo using Adam for language modeling. In *International Conference on Learning Representations*, 2025.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 37, pp. 95266–95290, 2024.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. Mixture-of-subspaces in low-rank adaptation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7880–7899, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Longxiang Zhang, Weilong Dai, Guanghao Zhang, Hao Jiang, and Pipei Huang. Think when needed: Adaptive reasoning-driven multimodal embeddings with a dual-LoRA architecture. *arXiv preprint arXiv:2605.14448*, 2026.

Yibo Zhong, Jinman Zhao, and Yao Zhou. Low-rank interconnected adaptation across layers. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 17005–17029, 2025.

A Implementation Details

A.1 Projection-wise update and dimensions

For projection p in transformer layer ℓ, the input is $\mathbf{H}_\ell^p \in \mathbb{R}^{T\times d_{\text{in}}^p}$ and the low-rank activation is $\mathbf{R}_\ell^p = \mathcal{D}(\mathbf{H}_\ell^p)(\mathbf{A}_\ell^p)^\top \in \mathbb{R}^{T\times r}$, where $\mathcal{D}$ is adapter dropout during training. Equation (1) applies to every attention and MLP projection, with projection-specific factors and gates. An operator $\mathbf{S}_b \in \mathbb{R}^{r\times r}$ is shared within a routing block. The returned low-rank contribution is

$$\frac{\alpha_{\text{L}}}{r}\left(\mathbf{R}_\ell^p + g_\ell^p \mathbf{R}_\ell^p \mathbf{S}_b^\top\right)(\mathbf{B}_\ell^p)^\top \in \mathbb{R}^{T\times d_{\text{out}}^p}.$$

Token pooling produces an r-dimensional routing state, and the operator is cached within its block. Each layer contributes the pooled state of its first eligible projection to the summary.

A.2 Depth attention and instruction channels

The dimensions in Equation (2) are $\mathbf{w}_{\ell_b} \in \mathbb{R}^{d_k}$, $\mathbf{Q}_{\text{cur}}, \mathbf{Q}_{\text{dep}} \in \mathbb{R}^{d_k\times r}$, and $\mathbf{Q}_{\text{ctx}}, \mathbf{R}_{\text{ctx}} \in \mathbb{R}^{d_k\times d_e}$. For learned depth attention, $\mathbf{Q}_{\text{dep},q} \in \mathbb{R}^{d_k\times d_k}$ and $\mathbf{Q}_{\text{dep},k} \in \mathbb{R}^{d_k\times r}$ give

$$\beta_{i\to b} = \text{softmax}_{i<b}\left(\frac{\langle \text{RMSNorm}(\mathbf{Q}_{\text{dep},q}\mathbf{q}_b^{(0)}), \text{RMSNorm}(\mathbf{Q}_{\text{dep},k}\bar{\mathbf{s}}_i)\rangle}{\sqrt{d_k}T_{\text{dep}}}\right).$$

Mean pooling sets $\beta_{i\to b} = 1/(b-1)$, also giving a convex combination to which the depth-summary norm bound applies. The first block receives the zero vector, and RMS normalization follows Zhang & Sennrich (2019).

Queryable uses uniform pooling, and IQ uses learned attention. Removing the depth summary sets its contribution to zero, and turning off depth attention leaves the mean summary. In the component ablation, the baseline explicitly enables depth attention, and parameter counts include every instantiated routing projection, including inactive projections.

The instruction reaches the router through a separate path. A frozen `sentence-transformers/all-MiniLM-L6-v2` encoder followed by trainable projections produces the instruction prior, with $\tau_{\text{lang}} = 0.5$ and $\lambda_{\text{ctx}} = 0.25$ for IQ and both coefficients set to zero for Queryable. Disabling these channels keeps the bank and local factors fixed. The main comparisons use the depth-pooling choices above (for the instruction experiment in Appendix F, we kept the other model components fixed).

Algorithm 1 Blockwise queryable adaptation (one prompt forward pass)

Require: Frozen backbone, low-rank factors and gates, atom bank and keys, router parameters, instruction embedding $\mathbf{e}(c)$, routing blocks $B_1, \ldots, B_B$.

Initialize completed summaries $\mathcal{U} \leftarrow [\,]$.
for $b = 1, \ldots, B$ **do**
  Pool the first eligible rank-space activation to obtain $\mathbf{s}_{\text{entry}}$.
  Form $\mathbf{q}_b^{(0)}$ using Equation (2).
  Retrieve $\mathbf{u}_{b-1}^{\text{att}}$ from $\mathcal{U}$ using depth attention or uniform pooling. Use zero if $b = 1$.
  Form $\mathbf{q}_b$ and the joint logits $\widetilde{\zeta}_{b,m}$ by Equations (2) and (3).
  Set $I_b \leftarrow \text{TopK}_m(\widetilde{\zeta}_{b,m})$ and normalize the active weights.
  Retrieve $\mathbf{S}_b \leftarrow \sum_{m\in I_b} \alpha_{b,m}\mathbf{C}_m$.
  for each adapted projection in B_b **do**
    Apply the frozen projection plus the gated low-rank update of Equation (1).
    Accumulate the eligible pooled rank-state summaries.
  end for
  Append the completed block average $\bar{\mathbf{s}}_b$ to $\mathcal{U}$.
end for
return backbone output and cached per-block prompt routes.

A.3 CACHING AND TRAINABLE-PARAMETER ACCOUNTING

During autoregressive evaluation, the routes are fixed after the first decoding step. At that step, every block except the last recomputes its route once from the new token and the prompt's block summaries. A new request resets the routes, and each update stays constant for all later tokens. During prefill, each block query uses a summary pooled over the sequence. We apply the routed operators and low-rank factors directly, with a separate low-rank multiplication at every adapted projection.

In the implementation, each atom is parameterized as $\mathbf{C}_m = \mathbf{C}_0 + \mathbf{D}_m$, where $\mathbf{C}_0$ is one shared trainable $r \times r$ matrix initialized at zero. The active weights sum to one, so the routed operator equals $\mathbf{C}_0 + \sum_{m \in I_b} \alpha_{b,m} \mathbf{D}_m$. The results of Section 4 hold for these atoms $\mathbf{C}_m$. For $M = 16$ and $r = 8$, the matrices $\mathbf{D}_m$ contain $Mr^2 = 1{,}024$ parameters, and $\mathbf{C}_0$ adds $r^2 = 64$. The approximately 58K added parameters belong to the shared bank and the routing components (Table 6 gives exact trainable counts, including local low-rank factors).

Table 6: Exact trainable parameters across backbone sizes. Queryable and IQ have equal instantiated counts, and the overhead column gives their increase over LoRA.

Backbone	LoRA	Queryable / IQ	Overhead	MoE-LoRA	RepLoRA
0.5B	4,399,104	4,456,936	1.31%	18,579,456	145,872,000
1.5B	9,232,384	9,290,372	0.63%	38,965,248	305,487,168
3B	14,966,784	15,025,084	0.39%	63,221,760	494,956,224
7B	20,185,088	20,243,076	0.29%	85,270,528	666,926,400
14B	34,406,400	34,465,168	0.17%	146,178,048	1,136,814,336

At 7B, DoRA uses 21,317,632 trainable parameters and ZHyper uses 23,686,432. Lily uses 9,803,904. These counts describe parameters during training and are rounded to millions in Table 1.

B Additional Theoretical Analysis

B.1 Setup

We use the notations as stated in Sec. 3. For every block b, the router produces a probability vector

$$\boldsymbol{\alpha}_b(c) = (\alpha_{b,1}(c), \dots, \alpha_{b,M}(c)) \in \Delta_M$$

over a shared memory of rank-space atoms $\{\mathbf{C}_m\}_{m=1}^{M} \subset \mathbb{R}^{r\times r}$. The routed rank-space operator is

$$\mathbf{S}_b(c) := \sum_{m=1}^{M} \alpha_{b,m}(c)\mathbf{C}_m \tag{15}$$

For a layer $\ell \in \mathcal{B}_b$, the realized update is

$$\Delta\mathbf{W}^{\ell}(\mathbf{h}_\ell; b, c) = \frac{\alpha_{\mathrm{L}}}{r}\,\mathbf{B}_\ell\big(\mathbf{I}_r + g_\ell \mathbf{S}_b(c)\big)\mathbf{A}_\ell \tag{16}$$

$$g_\ell = \sigma(\eta_\ell) \in (0,1) \tag{17}$$

where α_{L} is the LoRA scaling factor and r is the rank-space dimension. The symbol α_{L} is used to avoid confusion with the routing weights $\alpha_{b,m}(c)$. The instruction encoder maps a natural-language instruction c to a fixed embedding $\mathbf{e}(c) \in \mathbb{R}^{d_c}$. The state-dependent block query is

$$\mathbf{q}_b^{(0)} = \mathbf{w}_{\ell_b} + \mathbf{Q}_{\mathrm{cur}}\mathbf{s}_{\ell_b}^{\mathrm{entry}} + \lambda_{\mathrm{ctx}}\mathbf{Q}_{\mathrm{ctx}}\mathbf{e}(c) \tag{18}$$

where ℓ_b is the first adapted layer in block b. The attention-style depth summary is

$$\mathbf{u}_{b-1}^{\mathrm{att}} = \sum_{i=1}^{b-1} \beta_{i\to b}\bar{\mathbf{s}}_i \tag{19}$$

$$\beta_{i\to b} = \frac{\exp(\xi_{i\to b})}{\sum_{j=1}^{b-1}\exp(\xi_{j\to b})} \tag{20}$$

with the convention $\mathbf{u}_0^{\mathrm{att}} = \mathbf{0}$. The final query is

$$\mathbf{q}_b = \mathbf{q}_b^{(0)} + \mathbf{Q}_{\mathrm{dep}}\mathbf{u}_{b-1}^{\mathrm{att}} \tag{21}$$

The depth-summary logits are

$$\xi_{i\to b} = \frac{\langle \widehat{\mathbf{q}}_b^{(0)}, \widehat{\boldsymbol{\kappa}}_i\rangle}{\sqrt{d_k}T_{\mathrm{dep}}} \tag{22}$$

$$\widehat{\mathbf{q}}_b^{(0)} = \mathrm{RMSNorm}(\mathbf{Q}_{\mathrm{dep},q}\mathbf{q}_b^{(0)}) \tag{23}$$

$$\widehat{\boldsymbol{\kappa}}_i = \mathrm{RMSNorm}(\mathbf{Q}_{\mathrm{dep},k}\bar{\mathbf{s}}_i). \tag{24}$$

The language-prior logits and state logits are

$$\rho_m(c) = \frac{\langle \mathrm{RMSNorm}(\mathbf{R}_{\mathrm{ctx}}\mathbf{e}(c)), \widehat{\mathbf{k}}_m\rangle}{\sqrt{d_k}T_{\mathrm{lang}}} \tag{25}$$

$$\zeta_{b,m} = \frac{\langle \mathrm{RMSNorm}(\mathbf{q}_b), \widehat{\mathbf{k}}_m\rangle}{\sqrt{d_k}T_{\mathrm{attn}}} \tag{26}$$

where $\widehat{\mathbf{k}}_m := \mathrm{RMSNorm}(\mathbf{k}_m)$. The language prior distribution is

$$p_m(c) = \frac{\exp(\rho_m(c))}{\sum_{j=1}^{M}\exp(\rho_j(c))} \tag{27}$$

The router in the main paper uses fused logits

$$\widetilde{\zeta}_{b,m}(c) = \zeta_{b,m} + \tau_{\mathrm{lang}}\log p_m(c) \tag{28}$$

For full routing, the active set is $I = \{1, \dots, M\}$. For top-k routing, $I = I_b(c)$ denotes the selected active set. On a fixed active set I, the routed weights are

$$\alpha_{b,m}(c) = \frac{\exp(\widetilde{\zeta}_{b,m}(c))}{\sum_{j \in I} \exp(\widetilde{\zeta}_{b,j}(c))} \tag{29}$$

for $m \in I$, and $\alpha_{b,m}(c) = 0$ for $m \notin I$ in the sparse case. The active set I_b is treated as fixed inside differentiability statements. This is automatic away from top-k switching boundaries and is the only local smoothness convention needed for logit-gradient calculations. Matrix inner products are Frobenius inner products, $\langle \mathbf{U}, \mathbf{V} \rangle_F = \mathrm{tr}(\mathbf{U}^\top \mathbf{V})$. We consider the following assumptions:

Assumption B.1. *$\exists$ finite constants $R_C, R_A, R_B > 0$ such that $\|\mathbf{C}_m\| \le R_C$, $\|\mathbf{A}_\ell\| \le R_A$, $\|\mathbf{B}_\ell\| \le R_B$; $\forall$ atom m and every adapted layer ℓ. Matrix norms are operator norms unless explicitly stated otherwise.*

Assumption B.2. *$\exists$ finite constants $R_s, R_e > 0$ such that $\|\bar{\mathbf{s}}_i\| \le R_s$, $\|\mathbf{s}_{\ell_b}^{\mathrm{entry}}\| \le R_s$, $\|\mathbf{e}(c)\| \le R_e$; $\forall$ completed block i, block entry layer ℓ_b, and instruction c in the region of interest.*

Assumption B.3. *Let $\mathrm{RMS}_\varepsilon(\mathbf{x}) := \sqrt{\frac{1}{d}\|\mathbf{x}\|_2^2 + \varepsilon}$ and $\mathrm{RMSNorm}(\mathbf{x}) := \frac{\mathbf{x}}{\mathrm{RMS}_\varepsilon(\mathbf{x})}$. $\exists$ $\rho_{\min} > 0$ such that every vector to which RMS normalization is applied in the router satisfies $\mathrm{RMS}_\varepsilon(\mathbf{x}) \ge \rho_{\min}$.*

Assumption B.4. *For the top-k sparse router, all local differentiability and Lipschitz statements are restricted to a region where the active set $I_b(c)$ is constant. Equivalently, the kth and $(k+1)$st largest fused logits are separated by a positive margin throughout the region.*

B.2 Results

Section 3 defines the language prior as

$$p_m(\boldsymbol{c}) = \frac{\exp(\rho_m(\boldsymbol{c}))}{\sum_{j=1}^{M} \exp(\rho_j(\boldsymbol{c}))} \tag{30}$$

$$\tilde{\zeta}_{b,m}(\boldsymbol{c}) = \zeta_{b,m} + \tau_{\mathrm{lang}} \log p_m(\boldsymbol{c}) \tag{31}$$

where $p_m(c)$ is the language-prior distribution and $\tilde{\zeta}_{b,m}(c)$ are the fused routing logits. Lemma B.5 represents this formulation in an equivalent and simplified sense.

Lemma B.5. *Define $z_{b,m}$ as $z_{b,m}(c) = \zeta_{b,m} + \tau_{\mathrm{lang}} \rho_m(c)$, where, $m = 1, 2, 3, ..., M$. Then, both the usual softmax router and the top-k softmax router obtained from the logits $\{\tilde{\zeta}_{b,m}\}_{m=1}^M$ are equivalent to those obtained from $\{z_{b,m}\}_{m=1}^M$.*

Proof. We observe that:

$$\log p_m(\boldsymbol{c}) = \rho_m(\boldsymbol{c}) - \log\Big(\sum_{j=1}^{M} e^{\rho_j(\boldsymbol{c})}\Big) \tag{32}$$

$$\implies \tilde{\zeta}_{b,m}(\boldsymbol{c}) = \zeta_{b,m} + \tau_{\mathrm{lang}} \rho_m(\boldsymbol{c}) - \tau_{\mathrm{lang}} \log\Big(\sum_{j=1}^{M} e^{\rho_j(\boldsymbol{c})}\Big) \tag{33}$$

Hence, $\{\tilde{\zeta}_{b,m}\}_{m=1}^M$ and $\{z_{b,m}\}_{m=1}^M$ differ by only a scalar quantity, and thus, the softmax probabilities obtained from either logits are the same. □

The router can be understood as solving a precise retrieval problem over the shared atom memory. We state Theorem B.6 below.

Theorem B.6 (Variational characterization of instruction-regularized retrieval)**.** *Fix a block b, instruction c, and active set I. The routing distribution $\boldsymbol{\alpha}_{b,I}(c)$ defined in* (29) *is the unique solution of*

$$\boldsymbol{\alpha}_{b,I}(c) = \arg\max_{\boldsymbol{a} \in \Delta(I)} \Big\{ \langle \boldsymbol{a}, \boldsymbol{\zeta}_{b,I} \rangle - \mathrm{KL}\Big(\boldsymbol{a} \,\Big\|\, \boldsymbol{\pi}_I^{(\tau)}(c)\Big) \Big\} \tag{34}$$

where $\Delta(I)$ is the probability simplex on I. Equivalently,

$$\boldsymbol{\alpha}_{b,I}(c) = \arg\max_{\boldsymbol{a}\in\Delta(I)} \left\{ \langle \boldsymbol{a}, \boldsymbol{\zeta}_{b,I} + \tau_{\text{lang}} \log \mathbf{p}_I(c) \rangle + H(\boldsymbol{a}) \right\} \tag{35}$$

where $H(\boldsymbol{a}) = -\sum_{m\in I} a_m \log a_m$.

Proof. Let

$$\Phi(\boldsymbol{a}) := \langle \boldsymbol{a}, \boldsymbol{\zeta}_{b,I} \rangle - \mathrm{KL}\Big(\boldsymbol{a} \,\Big\|\, \boldsymbol{\pi}_I^{(\tau)}(c) \Big)$$

for $\boldsymbol{a} \in \Delta(I)$. Expanding the KL divergence gives

$$\begin{aligned}
\Phi(\boldsymbol{a}) &= \sum_{m\in I} a_m \zeta_{b,m} - \sum_{m\in I} a_m \log \frac{a_m}{\pi_{I,m}^{(\tau)}(c)} \\
&= \sum_{m\in I} a_m \zeta_{b,m} - \sum_{m\in I} a_m \log a_m + \sum_{m\in I} a_m \log \pi_{I,m}^{(\tau)}(c)
\end{aligned}$$

The map $\boldsymbol{a} \mapsto -\sum_{m\in I} a_m \log a_m$ is strictly concave on the relative interior of the simplex. The remaining terms are linear in $\boldsymbol{a}$. Hence, Φ is strictly concave on $\Delta(I)$ and has at most one maximizer. To compute the maximizer, introduce a Lagrange multiplier $\lambda \in \mathbb{R}$ for the constraint $\sum_{m\in I} a_m = 1$. Hence, the Lagrangian is:

$$\mathcal{J}(\boldsymbol{a}, \lambda) = \sum_{m\in I} a_m \zeta_{b,m} - \sum_{m\in I} a_m \log \frac{a_m}{\pi_{I,m}^{(\tau)}(c)} + \lambda \left(\sum_{m\in I} a_m - 1 \right)$$

$\forall\, m \in I$,

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial a_m} &= \zeta_{b,m} - \frac{\partial}{\partial a_m} \left(a_m \log \frac{a_m}{\pi_{I,m}^{(\tau)}(c)} \right) + \lambda \\
&= \zeta_{b,m} - \log a_m + \log \pi_{I,m}^{(\tau)}(c) - 1 + \lambda
\end{aligned}$$

From the first-order condition, $\frac{\partial \mathcal{J}}{\partial a_m} = 0$. Hence,

$$0 = \zeta_{b,m} - \log a_m + \log \pi_{I,m}^{(\tau)}(c) - 1 + \lambda \tag{36}$$

$$\implies a_m = \exp(\lambda - 1)\, \pi_{I,m}^{(\tau)}(c) \exp(\zeta_{b,m}) \tag{37}$$

$\exp(\lambda - 1)$ is independent of m and is determined by the simplex constraint. Hence,

$$a_m = \frac{\pi_{I,m}^{(\tau)}(c) \exp(\zeta_{b,m})}{\sum_{j\in I} \pi_{I,j}^{(\tau)}(c) \exp(\zeta_{b,j})} \tag{38}$$

Since, $\pi_{I,m}^{(\tau)}(c) = \frac{p_m(c)^{\tau_{\text{lang}}}}{\sum_{q\in I} p_q(c)^{\tau_{\text{lang}}}}$,

$$a_m = \frac{\exp(\zeta_{b,m}) p_m(c)^{\tau_{\text{lang}}}}{\sum_{j\in I} \exp(\zeta_{b,j}) p_j(c)^{\tau_{\text{lang}}}} = \frac{\exp(\zeta_{b,m} + \tau_{\text{lang}} \log p_m(c))}{\sum_{j\in I} \exp(\zeta_{b,j} + \tau_{\text{lang}} \log p_j(c))}$$

This is the required active-set router in (29). Since the objective is strictly concave, this maximizer is unique. To show the entropy form, the definition of $\boldsymbol{\pi}_I^{(\tau)}(c)$ gives

$$\log \pi_{I,m}^{(\tau)}(c) = \tau_{\text{lang}} \log p_m(c) - \log \left(\sum_{j\in I} p_j(c)^{\tau_{\text{lang}}} \right)$$

Using this identity in the expanded expression for Φ gives

$$\Phi(\boldsymbol{a}) = \langle \boldsymbol{a}, \boldsymbol{\zeta}_{b,I} \rangle + H(\boldsymbol{a}) + \tau_{\text{lang}} \langle \boldsymbol{a}, \log \mathbf{p}_I(c) \rangle - \log \left(\sum_{j\in I} p_j(c)^{\tau_{\text{lang}}} \right) \sum_{m\in I} a_m$$

Now, since, $\boldsymbol{a} \in \Delta(I)$, $\sum_{m\in I} a_m = 1$. Hence, the final term is constant in $\boldsymbol{a}$ and does not change the optimizer. Removing this constant gives (35). □

The state-prior tradeoff, in B.6.1, clarifies what is lost when the instruction is allowed to influence routing. The hidden state may prefer one update atom, while the instruction prior may prefer another. B.6.1 demonstrates that the state-matching score lost by using instruction-guided retrieval, is small, whenever the tempered instruction prior assigns sufficient weight to the atom that the hidden state would choose on its own.

Corollary B.6.1 (State-prior tradeoff bound)**.** *Let $m_b^\star \in \arg\max_{m\in I} \zeta_{b,m}$. Under the hypotheses of Theorem B.6,*

$$0 \le \max_{m\in I} \zeta_{b,m} - \langle \boldsymbol{\alpha}_{b,I}(c), \boldsymbol{\zeta}_{b,I}\rangle \le \log \frac{1}{\pi^{(\tau)}_{I,m_b^\star}(c)} \tag{39}$$

Proof. The left inequality follows because $\boldsymbol{\alpha}_{b,I}(c)$ is a probability vector. Indeed,

$$\langle \boldsymbol{\alpha}_{b,I}(c), \boldsymbol{\zeta}_{b,I}\rangle = \sum_{m\in I} \alpha_{b,m}(c)\zeta_{b,m} \le \sum_{m\in I} \alpha_{b,m}(c) \max_{j\in I} \zeta_{b,j} = \max_{j\in I} \zeta_{b,j}$$
$$\implies \max_{m\in I} \zeta_{b,m} - \langle \boldsymbol{\alpha}_{b,I}(c), \boldsymbol{\zeta}_{b,I}\rangle \ge 0$$

For the upper bound, define:

$$\Phi(\boldsymbol{a}) = \langle \boldsymbol{a}, \boldsymbol{\zeta}_{b,I}\rangle - \mathrm{KL}\Big(\boldsymbol{a} \,\Big\|\, \boldsymbol{\pi}^{(\tau)}_I(c)\Big)$$

By Theorem B.6, $\boldsymbol{\alpha}_{b,I}(c)$ maximizes Φ over $\Delta(I)$. Let $\mathbf{e}_{m_b^\star}$ denote the simplex vertex that places all mass on $m_b^\star$. Then,

$$\Phi(\boldsymbol{\alpha}_{b,I}(c)) \ge \Phi(\mathbf{e}_{m_b^\star})$$

Expanding both sides gives

$$\langle \boldsymbol{\alpha}_{b,I}(c), \boldsymbol{\zeta}_{b,I}\rangle - \mathrm{KL}\Big(\boldsymbol{\alpha}_{b,I}(c) \,\Big\|\, \boldsymbol{\pi}^{(\tau)}_I(c)\Big) \ge \zeta_{b,m_b^\star} - \mathrm{KL}\Big(\mathbf{e}_{m_b^\star} \,\Big\|\, \boldsymbol{\pi}^{(\tau)}_I(c)\Big)$$

The KL divergence of a vertex against $\boldsymbol{\pi}^{(\tau)}_I(c)$ is

$$\mathrm{KL}\Big(\mathbf{e}_{m_b^\star} \,\Big\|\, \boldsymbol{\pi}^{(\tau)}_I(c)\Big) = \log \frac{1}{\pi^{(\tau)}_{I,m_b^\star}(c)} \tag{40}$$

$$\implies \langle \boldsymbol{\alpha}_{b,I}(c), \boldsymbol{\zeta}_{b,I}\rangle - \mathrm{KL}\Big(\boldsymbol{\alpha}_{b,I}(c) \,\Big\|\, \boldsymbol{\pi}^{(\tau)}_I(c)\Big) \ge \zeta_{b,m_b^\star} - \log \frac{1}{\pi^{(\tau)}_{I,m_b^\star}(c)} \tag{41}$$

As $\mathrm{KL}(\cdot\|\cdot) \ge 0$,

$$\langle \boldsymbol{\alpha}_{b,I}(c), \boldsymbol{\zeta}_{b,I}\rangle \ge \langle \boldsymbol{\alpha}_{b,I}(c), \boldsymbol{\zeta}_{b,I}\rangle - \mathrm{KL}\Big(\boldsymbol{\alpha}_{b,I}(c) \,\Big\|\, \boldsymbol{\pi}^{(\tau)}_I(c)\Big) \tag{42}$$

From (41) and (42),

$$\langle \boldsymbol{\alpha}_{b,I}(c), \boldsymbol{\zeta}_{b,I}\rangle \ge \zeta_{b,m_b^\star} - \log \frac{1}{\pi^{(\tau)}_{I,m_b^\star}(c)} \tag{43}$$

Rearranging gives us the upper bound in (39). □

Next, Proposition B.7 clarifies the importance of the instruction strength parameter τ_{lang}.

Proposition B.7 (Limiting language regimes)**.** *The instruction-regularized queryable update has the following limiting cases.*

(a) If $\lambda_{\mathrm{ctx}} = 0$ and $\tau_{\mathrm{lang}} = 0$, then the router reduces to the purely state-dependent queryable router.

(b) If $g_\ell = 0$ for every adapted layer, then the realized update reduces to the static LoRA update $\Delta\mathbf{W}^\ell = (\alpha_{\mathrm{L}}/r)\mathbf{B}_\ell\mathbf{A}_\ell$.

(c) Fix an active set I. Suppose that $\rho_m(c)$ has a unique maximizer $m_I^\dagger$ on I. If the state logits $\{\zeta_{b,m}\}_{m\in I}$ are finite and $\tau_{\mathrm{lang}} \to \infty$, then

$$\alpha_{b,m_I^\dagger}(c) \to 1 \tag{44}$$

$$\alpha_{b,m}(c) \to 0 \quad \forall\, m \ne m_I^\dagger \tag{45}$$

$$\mathbf{S}_b(c) \to \mathbf{C}_{m_I^\dagger} \tag{46}$$

(d) If the state logits are constant on I, then the router is the tempered instruction prior $\boldsymbol{\pi}_I^{(\tau)}(c)$. Specifically, if $\tau_{\text{lang}} = 1$, the router is exactly the restricted and renormalized language prior on I.

Proof. For part (a), if $\lambda_{\text{ctx}} = 0$, then the pre-query in (18) becomes

$$\mathbf{q}_b^{(0)} = \mathbf{w}_{\ell_b} + \mathbf{Q}_{\text{cur}} \mathbf{s}_{\ell_b}^{\text{entry}}$$

Hence, the instruction embedding no longer enters the query. If $\tau_{\text{lang}} = 0$, then the fused logit in (28) becomes

$$\widetilde{\zeta}_{b,m}(c) = \zeta_{b,m}$$

Thus, both the active-set selection and the softmax weights depend only on the state logits, which is the state-dependent queryable router. For part (b), substitute $g_\ell = 0$ into (16):

$$\Delta \mathbf{W}^\ell(\mathbf{h}_\ell; b, c) = \frac{\alpha_{\text{L}}}{r} \mathbf{B}_\ell (\mathbf{I}_r + 0 \cdot \mathbf{S}_b(c)) \mathbf{A}_\ell = \frac{\alpha_{\text{L}}}{r} \mathbf{B}_\ell \mathbf{A}_\ell$$

This is the standard static LoRA update. For part (c), by Lemma B.5, on I the router can be written as

$$\alpha_{b,m}(c) = \frac{\exp(\zeta_{b,m} + \tau_{\text{lang}} \rho_m(c))}{\sum_{j \in I} \exp(\zeta_{b,j} + \tau_{\text{lang}} \rho_j(c))}$$

Let $m_I^\dagger$ be the unique maximizer of $\rho_m(c)$ on I. $\forall\, m \neq m_I^\dagger$,

$$\frac{\alpha_{b,m}(c)}{\alpha_{b,m_I^\dagger}(c)} = \exp\left(\zeta_{b,m} - \zeta_{b,m_I^\dagger} + \tau_{\text{lang}}(\rho_m(c) - \rho_{m_I^\dagger}(c))\right)$$

Since $m_I^\dagger$ is the unique maximizer, $\rho_m(c) - \rho_{m_I^\dagger}(c) < 0 \;\; \forall\, m \neq m_I^\dagger$. The state-logit difference is finite by assumption. Hence, $\left(\zeta_{b,m} - \zeta_{b,m_I^\dagger} + \tau_{\text{lang}}(\rho_m(c) - \rho_{m_I^\dagger}(c))\right) \to -\infty$ as $\tau_{\text{lang}} \to \infty$, hence, the ratio tends to zero. Since the routing weights sum to one, $\implies \alpha_{b,m_I^\dagger}(c) \to 1$ and $\alpha_{b,m}(c) \to 0$ for all $m \neq m_I^\dagger$. Substituting into (15) gives $\mathbf{S}_b(c) \to \mathbf{C}_{m_I^\dagger}$. For part (d), suppose $\zeta_{b,m} = \gamma \;\; \forall\, m \in I$. Then,

$$\alpha_{b,m}(c) = \frac{\exp(\gamma) p_m(c)^{\tau_{\text{lang}}}}{\sum_{j \in I} \exp(\gamma) p_j(c)^{\tau_{\text{lang}}}} = \frac{p_m(c)^{\tau_{\text{lang}}}}{\sum_{j \in I} p_j(c)^{\tau_{\text{lang}}}} = \pi_{I,m}^{(\tau)}(c)$$

If $\tau_{\text{lang}} = 1$, this is the required restricted language prior renormalized on I. □

Propositions B.8 and B.9 bound the routed operator and the depth summary.

Proposition B.8. *For every block b and instruction $\boldsymbol{c}$, we have,*

$$S_b(\boldsymbol{c}) = \sum_{m=1}^{M} \alpha_{b,m}(\boldsymbol{c}) C_m \;\; \in \text{conv}\{C_1, C_2, C_3, \ldots, C_M\} \tag{47}$$

$$\|S_b(\boldsymbol{c})\| \leq \sum_{m=1}^{M} \alpha_{b,m}(\boldsymbol{c}) \|C_m\| \leq \max_{1 \leq m \leq M} \|C_m\| \leq R_C \tag{48}$$

where, $\text{conv}(.)$ *is the convex hull of atoms. A similar bound also holds true for the top-k router, as its weights also lie in the simplex.*

Proof. From construction, $\alpha_{b,m}(c) \geq 0$ and $\sum_{m=1}^{M} \alpha_{b,m}(\boldsymbol{c}) = 1$. Hence, $S_b(\boldsymbol{c})$ is a convex combination of atoms $\{C_m\}_{m=1}^{M}$, which demonstrates $S_b(c) \in \text{conv}\{C_1, C_2, C_3, \ldots, C_M\}$. Additionally,

$$\begin{aligned} \|S_b(c)\| = \left\| \sum_{m=1}^{M} \alpha_{b,m}(c) C_m \right\| &\leq \sum_{m=1}^{M} \alpha_{b,m}(c) \|C_m\| \\ &\leq \max_{1 \leq m \leq M} \|C_m\| \sum_{m=1}^{M} \alpha_{b,m}(c) = \max_{1 \leq m \leq M} \|C_m\| \end{aligned} \tag{49}$$

Using assumption B.1, $\|S_b(\boldsymbol{c})\| \leq R_C$. □

Corollary B.8.1 (Layerwise norm-to-norm bound)**.** *For any adapted layer $\ell \in \mathcal{B}_b$,*

$$\|\Delta \mathbf{W}^{\ell}(\mathbf{h}_{\ell}; b, c)\| \leq \frac{\alpha_{\mathrm{L}}}{r} \|\mathbf{B}_{\ell}\| \big(1 + |g_{\ell}| R_C\big) \|\mathbf{A}_{\ell}\| \tag{50}$$

Consequently, for any vector $\mathbf{x}$,

$$\|\Delta \mathbf{W}^{\ell}(\mathbf{h}_{\ell}; b, c)\mathbf{x}\| \leq \frac{\alpha_{\mathrm{L}}}{r} \|\mathbf{B}_{\ell}\| \big(1 + |g_{\ell}| R_C\big) \|\mathbf{A}_{\ell}\| \, \|\mathbf{x}\| \tag{51}$$

Under Assumption B.1,

$$\|\Delta \mathbf{W}^{\ell}(\mathbf{h}_{\ell}; b, c)\| \leq \frac{\alpha_{\mathrm{L}}}{r} R_B (1 + R_C) R_A \tag{52}$$

as $g_{\ell} \in (0, 1)$.

Proof. From (16),

$$\Delta \mathbf{W}^{\ell} = \frac{\alpha_{\mathrm{L}}}{r} \mathbf{B}_{\ell} \big(\mathbf{I}_r + g_{\ell} \mathbf{S}_b(c)\big) \mathbf{A}_{\ell}$$

Using submultiplicativity of the operator norm,

$$\|\Delta \mathbf{W}^{\ell}\| \leq \frac{\alpha_{\mathrm{L}}}{r} \|\mathbf{B}_{\ell}\| \, \|\mathbf{I}_r + g_{\ell} \mathbf{S}_b(c)\| \, \|\mathbf{A}_{\ell}\| \tag{53}$$

From triangle inequality,

$$\|\mathbf{I}_r + g_{\ell} \mathbf{S}_b(c)\| \leq \|\mathbf{I}_r\| + |g_{\ell}| \|\mathbf{S}_b(c)\|$$

$\|\mathbf{I}_r\| = 1$, and Proposition B.8 gives $\|\mathbf{S}_b(c)\| \leq R_C$. Hence,

$$\|\mathbf{I}_r + g_{\ell} \mathbf{S}_b(c)\| \leq 1 + |g_{\ell}| R_C \tag{54}$$

Substituting (54) in (53) proves (50). Inequality (51) follows from the definition of the subordinate operator norm:

$$\|\Delta \mathbf{W}^{\ell} \mathbf{x}\| \leq \|\Delta \mathbf{W}^{\ell}\| \, \|\mathbf{x}\|$$

Finally, Assumption B.1 and $0 < g_{\ell} < 1$ yield (52). □

Proposition B.9 (Bounded attention-style depth summary)**.** *For every block $b \geq 2$,*

$$\|\mathbf{u}_{b-1}^{\mathrm{att}}\| \leq \max_{1 \leq i \leq b-1} \|\bar{\mathbf{s}}_i\| \tag{55}$$

Under the Assumption B.2, $\|\mathbf{u}_{b-1}^{\mathrm{att}}\| \leq R_s$. For $b = 1$, the convention $\mathbf{u}_0^{\mathrm{att}} = \mathbf{0}$ gives the same bound.

Proof. For $b \geq 2$, the attention weights satisfy

$$\beta_{i \to b} \geq 0$$
$$\sum_{i=1}^{b-1} \beta_{i \to b} = 1$$

Using (19),

$$\|\mathbf{u}_{b-1}^{\mathrm{att}}\| = \left\| \sum_{i=1}^{b-1} \beta_{i \to b} \bar{\mathbf{s}}_i \right\| \leq \sum_{i=1}^{b-1} \beta_{i \to b} \|\bar{\mathbf{s}}_i\| \leq \sum_{i=1}^{b-1} \beta_{i \to b} \max_{1 \leq j \leq b-1} \|\bar{\mathbf{s}}_j\|$$
$$= \max_{1 \leq j \leq b-1} \|\bar{\mathbf{s}}_j\| \sum_{i=1}^{b-1} \beta_{i \to b} = \max_{1 \leq j \leq b-1} \|\bar{\mathbf{s}}_j\|$$

$\|\mathbf{u}_{b-1}^{\mathrm{att}}\| \leq R_s$ follows from Assumption B.2. If $b = 1$, then $\mathbf{u}_0^{\mathrm{att}} = \mathbf{0}$, hence, $\|\mathbf{u}_0^{\mathrm{att}}\| = 0 \leq R_s$. □

Corollary B.9.1 (Query norm bound)**.** *Assume $\|\mathbf{w}_{\ell_b}\| \leq R_w$. Then*

$$\|\mathbf{q}_b\| \leq R_w + \|\mathbf{Q}_{\mathrm{cur}}\| R_s + \lambda_{\mathrm{ctx}} \|\mathbf{Q}_{\mathrm{ctx}}\| R_e + \|\mathbf{Q}_{\mathrm{dep}}\| R_s \tag{56}$$

Proof. By (18) and (21),

$$\mathbf{q}_b = \mathbf{w}_{\ell_b} + \mathbf{Q}_{\mathrm{cur}}\mathbf{s}_{\ell_b}^{\mathrm{entry}} + \lambda_{\mathrm{ctx}}\mathbf{Q}_{\mathrm{ctx}}\mathbf{e}(c) + \mathbf{Q}_{\mathrm{dep}}\mathbf{u}_{b-1}^{\mathrm{att}}$$

From triangle inequality,

$$\begin{aligned}\|\mathbf{q}_b\| &\le \|\mathbf{w}_{\ell_b}\| + \|\mathbf{Q}_{\mathrm{cur}}\mathbf{s}_{\ell_b}^{\mathrm{entry}}\| + \lambda_{\mathrm{ctx}}\|\mathbf{Q}_{\mathrm{ctx}}\mathbf{e}(c)\| + \|\mathbf{Q}_{\mathrm{dep}}\mathbf{u}_{b-1}^{\mathrm{att}}\| \\ &\le \|\mathbf{w}_{\ell_b}\| + \|\mathbf{Q}_{\mathrm{cur}}\|\,\|\mathbf{s}_{\ell_b}^{\mathrm{entry}}\| + \lambda_{\mathrm{ctx}}\|\mathbf{Q}_{\mathrm{ctx}}\|\,\|\mathbf{e}(c)\| + \|\mathbf{Q}_{\mathrm{dep}}\|\,\|\mathbf{u}_{b-1}^{\mathrm{att}}\|\end{aligned}$$

Using $\|\mathbf{w}_{\ell_b}\| \le R_w$, Assumption B.2, and Proposition B.9 gives (56). □

The next two lemmas collect standard differential facts used in the stability analysis (Zhang & Sennrich, 2019; Martins & Astudillo, 2016; Gao & Pavel, 2017).

Lemma B.10 (Exact Jacobian of RMS normalization)**.** *Let* $r(\mathbf{x}) = \mathit{RMS}_\varepsilon(\mathbf{x}) = \sqrt{\frac{1}{d}\|\mathbf{x}\|_2^2 + \varepsilon}$*, and,* $\mathbf{f}(\mathbf{x}) = \frac{\mathbf{x}}{r(\mathbf{x})}$*. Then*

$$\mathbf{J}_{\mathbf{f}}(\mathbf{x}) = \frac{1}{r(\mathbf{x})}\mathbf{I}_d - \frac{1}{d\,r(\mathbf{x})^3}\mathbf{x}\mathbf{x}^\top \tag{57}$$

If $r(\mathbf{x}) \ge \rho_{\min} > 0$*, then*

$$\|\mathbf{J}_{\mathbf{f}}(\mathbf{x})\|_{2\to 2} \le \frac{1}{\rho_{\min}} \tag{58}$$

Proof. The steps below follow Stollenwerk (2026) and keep ε throughout. Write $\mathbf{f}(\mathbf{x}) = r(\mathbf{x})^{-1}\mathbf{x}$. By the product rule for Jacobians,

$$\mathbf{J}_{\mathbf{f}}(\mathbf{x}) = r(\mathbf{x})^{-1}\mathbf{I}_d + \mathbf{x}\,\nabla(r(\mathbf{x})^{-1})^\top \tag{59}$$

We compute $\nabla r(\mathbf{x})$. Since, $r(\mathbf{x}) = \left(\frac{1}{d}\mathbf{x}^\top\mathbf{x} + \varepsilon\right)^{1/2}$, we have,

$$\nabla r(\mathbf{x}) = \frac{1}{2}\left(\frac{1}{d}\mathbf{x}^\top\mathbf{x} + \varepsilon\right)^{-1/2}\frac{2}{d}\mathbf{x} = \frac{\mathbf{x}}{d\,r(\mathbf{x})}$$

From chain rule,

$$\nabla(r(\mathbf{x})^{-1}) = -r(\mathbf{x})^{-2}\nabla r(\mathbf{x}) = -\frac{\mathbf{x}}{d\,r(\mathbf{x})^3} \tag{60}$$

From (60) and (59), we get (57). For the operator norm bound, note that $\mathbf{x}\mathbf{x}^\top$ is symmetric rank one. It has eigenvalue $\|\mathbf{x}\|_2^2$ in the direction of $\mathbf{x}$ and eigenvalue zero on the orthogonal complement. Hence $\mathbf{J}_{\mathbf{f}}(\mathbf{x})$ has eigenvalue $\frac{1}{r(\mathbf{x})}$ on every vector orthogonal to $\mathbf{x}$, and eigenvalue

$$\frac{1}{r(\mathbf{x})} - \frac{\|\mathbf{x}\|_2^2}{d\,r(\mathbf{x})^3} = \frac{d\,r(\mathbf{x})^2 - \|\mathbf{x}\|_2^2}{d\,r(\mathbf{x})^3} = \frac{d\left(\|\mathbf{x}\|_2^2/d + \varepsilon\right) - \|\mathbf{x}\|_2^2}{d\,r(\mathbf{x})^3} = \frac{d\varepsilon}{d\,r(\mathbf{x})^3} = \frac{\varepsilon}{r(\mathbf{x})^3}$$

in the direction of $\mathbf{x}$. Since $r(\mathbf{x})^2 = \|\mathbf{x}\|_2^2/d + \varepsilon \ge \varepsilon$, we have $\frac{\varepsilon}{r(\mathbf{x})^3} \le \frac{1}{r(\mathbf{x})}$. Hence, the largest absolute eigenvalue is at most $\frac{1}{r(\mathbf{x})}$. Since the Jacobian is symmetric, its spectral norm is the largest absolute eigenvalue.

$$\implies \|\mathbf{J}_{\mathbf{f}}(\mathbf{x})\|_{2\to 2} \le \frac{1}{r(\mathbf{x})} \le \frac{1}{\rho_{\min}}$$

□

Lemma B.11 (Softmax Jacobian bound)**.** *Let softmax* $: \mathbb{R}^k \to \Delta_k$ *be the softmax map and set* $\boldsymbol{\alpha} = \mathit{softmax}(\mathbf{z})$*. Then,*

$$\mathbf{J}_{\mathit{softmax}}(\mathbf{z}) = \mathit{diag}(\boldsymbol{\alpha}) - \boldsymbol{\alpha}\boldsymbol{\alpha}^\top \tag{61}$$

Moreover,

$$\|\mathbf{J}_{\mathit{softmax}}(\mathbf{z})\|_{2\to 2} \le \frac{1}{2} \tag{62}$$

Proof. The Jacobian identity follows from Martins & Astudillo (2016). For the bound, the ith row of $\mathbf{J}_{\text{softmax}}(\mathbf{z})$ has absolute sum $\alpha_i(1-\alpha_i) + \alpha_i \sum_{j\neq i} \alpha_j = 2\alpha_i(1-\alpha_i) \leq \frac{1}{2}$. Since the Jacobian is symmetric, its 1-norm and ∞-norm are both at most $\frac{1}{2}$, and $\|\mathbf{J}_{\text{softmax}}(\mathbf{z})\|_{2\to 2} \leq \sqrt{\|\mathbf{J}_{\text{softmax}}(\mathbf{z})\|_1 \, \|\mathbf{J}_{\text{softmax}}(\mathbf{z})\|_\infty} \leq \frac{1}{2}$. □

B.12 shows that the attention-style depth summary varies smoothly with the instruction embedding when the block states are held fixed. Small changes in the instruction lead to controlled changes in the retrieved historical context.

Proposition B.12 (Local Lipschitz stability of the attention-style summary)**.** *Fix a block $b \geq 2$ and suppose the completed block means $\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2, \bar{\mathbf{s}}_3, \ldots, \bar{\mathbf{s}}_{b-1}$ and the block-entry state $\mathbf{s}_{\ell_b}^{\text{entry}}$ are fixed. On any region satisfying Assumption B.3, the attention-style summary is locally Lipschitz in the instruction embedding. One admissible bound is*

$$\left\|\mathbf{u}_{b-1}^{\text{att}}(\mathbf{e}) - \mathbf{u}_{b-1}^{\text{att}}(\mathbf{e}')\right\| \leq \frac{(b-1)R_s \, \|\mathbf{Q}_{\text{dep},q}\|\lambda_{\text{ctx}}\|\mathbf{Q}_{\text{ctx}}\|}{2T_{\text{dep}}\rho_{\min}} \|\mathbf{e} - \mathbf{e}'\| \tag{63}$$

Consequently,

$$\|\mathbf{q}_b(\mathbf{e}) - \mathbf{q}_b(\mathbf{e}')\| \leq L_{q,b}\|\mathbf{e} - \mathbf{e}'\| \tag{64}$$

where

$$L_{q,b} := \lambda_{\text{ctx}}\|\mathbf{Q}_{\text{ctx}}\| \left(1 + \frac{(b-1)R_s \, \|\mathbf{Q}_{\text{dep}}\|\|\mathbf{Q}_{\text{dep},q}\|}{2T_{\text{dep}}\rho_{\min}}\right) \tag{65}$$

Proof. The only part of $\mathbf{q}_b^{(0)}$ that depends on the instruction embedding is $\lambda_{\text{ctx}}\mathbf{Q}_{\text{ctx}}\mathbf{e}$. Hence,

$$\|\mathbf{q}_b^{(0)}(\mathbf{e}) - \mathbf{q}_b^{(0)}(\mathbf{e}')\| \leq \lambda_{\text{ctx}}\|\mathbf{Q}_{\text{ctx}}\| \, \|\mathbf{e} - \mathbf{e}'\| \tag{66}$$

For each completed block $i < b$, define $\xi_{i\to b}(\mathbf{e}) = \frac{\langle \widehat{\mathbf{q}}_b^{(0)}(\mathbf{e}), \widehat{\boldsymbol{\kappa}}_i \rangle}{\sqrt{d_k}T_{\text{dep}}}$. $\widehat{\boldsymbol{\kappa}}_i$ is fixed because $\bar{\mathbf{s}}_i$ is fixed. RMS normalization gives $\|\widehat{\boldsymbol{\kappa}}_i\|_2 \leq \sqrt{d_k}$. Hence, by Cauchy-Schwarz inequality,

$$\begin{aligned}
|\xi_{i\to b}(\mathbf{e}) - \xi_{i\to b}(\mathbf{e}')| &\leq \frac{1}{\sqrt{d_k}T_{\text{dep}}} \left|\left\langle \widehat{\mathbf{q}}_b^{(0)}(\mathbf{e}) - \widehat{\mathbf{q}}_b^{(0)}(\mathbf{e}'), \widehat{\boldsymbol{\kappa}}_i \right\rangle\right| \\
&\leq \frac{1}{\sqrt{d_k}T_{\text{dep}}} \left\|\widehat{\mathbf{q}}_b^{(0)}(\mathbf{e}) - \widehat{\mathbf{q}}_b^{(0)}(\mathbf{e}')\right\|_2 \|\widehat{\boldsymbol{\kappa}}_i\|_2 \leq \frac{1}{T_{\text{dep}}} \left\|\widehat{\mathbf{q}}_b^{(0)}(\mathbf{e}) - \widehat{\mathbf{q}}_b^{(0)}(\mathbf{e}')\right\|_2
\end{aligned}$$

By Lemma B.10, RMS normalization is $\frac{1}{\rho_{\min}}$-Lipschitz on the non-degenerate region. Hence,

$$\left\|\widehat{\mathbf{q}}_b^{(0)}(\mathbf{e}) - \widehat{\mathbf{q}}_b^{(0)}(\mathbf{e}')\right\|_2 \leq \frac{1}{\rho_{\min}} \left\|\mathbf{Q}_{\text{dep},q}\big(\mathbf{q}_b^{(0)}(\mathbf{e}) - \mathbf{q}_b^{(0)}(\mathbf{e}')\big)\right\|_2 \leq \frac{\|\mathbf{Q}_{\text{dep},q}\|\lambda_{\text{ctx}}\|\mathbf{Q}_{\text{ctx}}\|}{\rho_{\min}} \|\mathbf{e} - \mathbf{e}'\|$$

where the final inequality uses (66). Thus each depth logit satisfies the inequality,

$$|\xi_{i\to b}(\mathbf{e}) - \xi_{i\to b}(\mathbf{e}')| \leq \frac{\|\mathbf{Q}_{\text{dep},q}\|\lambda_{\text{ctx}}\|\mathbf{Q}_{\text{ctx}}\|}{T_{\text{dep}}\rho_{\min}} \|\mathbf{e} - \mathbf{e}'\|$$

Since there are $b-1$ depth logits,

$$\|\boldsymbol{\xi}_b(\mathbf{e}) - \boldsymbol{\xi}_b(\mathbf{e}')\|_2 \leq \sqrt{b-1} \, \frac{\|\mathbf{Q}_{\text{dep},q}\|\lambda_{\text{ctx}}\|\mathbf{Q}_{\text{ctx}}\|}{T_{\text{dep}}\rho_{\min}} \|\mathbf{e} - \mathbf{e}'\| \tag{67}$$

The depth weights satisfy $\boldsymbol{\beta}_b(\mathbf{e}) = \text{softmax}(\boldsymbol{\xi}_b(\mathbf{e}))$. Lemma B.11 gives

$$\|\boldsymbol{\beta}_b(\mathbf{e}) - \boldsymbol{\beta}_b(\mathbf{e}')\|_2 \leq \frac{1}{2}\|\boldsymbol{\xi}_b(\mathbf{e}) - \boldsymbol{\xi}_b(\mathbf{e}')\|_2 \tag{68}$$

As,

$$\mathbf{u}_{b-1}^{\text{att}}(\mathbf{e}) - \mathbf{u}_{b-1}^{\text{att}}(\mathbf{e}') = \sum_{i=1}^{b-1} (\beta_{i\to b}(\mathbf{e}) - \beta_{i\to b}(\mathbf{e}'))\bar{\mathbf{s}}_i$$

Hence,

$$\left\|\mathbf{u}_{b-1}^{\mathrm{att}}(\mathbf{e}) - \mathbf{u}_{b-1}^{\mathrm{att}}(\mathbf{e}')\right\| \le \sum_{i=1}^{b-1} |\beta_{i\to b}(\mathbf{e}) - \beta_{i\to b}(\mathbf{e}')| \, \|\bar{\mathbf{s}}_i\| \\ \le R_s \|\boldsymbol{\beta}_b(\mathbf{e}) - \boldsymbol{\beta}_b(\mathbf{e}')\|_1 \le R_s\sqrt{b-1}\|\boldsymbol{\beta}_b(\mathbf{e}) - \boldsymbol{\beta}_b(\mathbf{e}')\|_2 \quad (69)$$

Combining (69), (67) and (68) proves (63). Finally,

$$\mathbf{q}_b(\mathbf{e}) - \mathbf{q}_b(\mathbf{e}') = \mathbf{q}_b^{(0)}(\mathbf{e}) - \mathbf{q}_b^{(0)}(\mathbf{e}') + \mathbf{Q}_{\mathrm{dep}}\left(\mathbf{u}_{b-1}^{\mathrm{att}}(\mathbf{e}) - \mathbf{u}_{b-1}^{\mathrm{att}}(\mathbf{e}')\right)$$

Using (66), and (63) yields,

$$\|\mathbf{q}_b(\mathbf{e}) - \mathbf{q}_b(\mathbf{e}')\| \le \lambda_{\mathrm{ctx}}\|\mathbf{Q}_{\mathrm{ctx}}\|\|\mathbf{e} - \mathbf{e}'\| + \|\mathbf{Q}_{\mathrm{dep}}\| \left\|\mathbf{u}_{b-1}^{\mathrm{att}}(\mathbf{e}) - \mathbf{u}_{b-1}^{\mathrm{att}}(\mathbf{e}')\right\| \\ \le \lambda_{\mathrm{ctx}}\|\mathbf{Q}_{\mathrm{ctx}}\| \left(1 + \frac{(b-1)R_s \, \|\mathbf{Q}_{\mathrm{dep}}\|\|\mathbf{Q}_{\mathrm{dep},q}\|}{2T_{\mathrm{dep}}\rho_{\min}}\right) \|\mathbf{e} - \mathbf{e}'\|$$

□

B.13 extends the same stability idea from the depth summary to the router itself.

Proposition B.13 (Local Lipschitz stability of the router and routed operator)**.** *Let I be a fixed active set of size k, and hold the block states fixed as in Proposition B.12. Under Assumptions B.1-B.4,*

$$\|\boldsymbol{\alpha}_{b,I}(\mathbf{e}) - \boldsymbol{\alpha}_{b,I}(\mathbf{e}')\|_2 \le L_{\alpha,b}\|\mathbf{e} - \mathbf{e}'\|, \quad (70)$$

where one admissible constant is

$$L_{\alpha,b} := \frac{\sqrt{k}}{2\rho_{\min}} \left(\frac{L_{q,b}}{T_{\mathrm{attn}}} + \frac{\tau_{\mathrm{lang}}\|\mathbf{R}_{\mathrm{ctx}}\|}{T_{\mathrm{lang}}}\right) \quad (71)$$

Moreover,

$$\|\mathbf{S}_b(\mathbf{e}) - \mathbf{S}_b(\mathbf{e}')\| \le R_C\sqrt{k}\, L_{\alpha,b}\|\mathbf{e} - \mathbf{e}'\| \quad (72)$$

and $\forall\ \ell \in \mathcal{B}_b$ and $\forall$ input vector $\mathbf{x}$,

$$\|\Delta\mathbf{W}^{\ell}(\mathbf{e})\mathbf{x} - \Delta\mathbf{W}^{\ell}(\mathbf{e}')\mathbf{x}\| \le \frac{\alpha_{\mathrm{L}}}{r}\|\mathbf{B}_\ell\| \, |g_\ell| \, \|\mathbf{A}_\ell\| R_C\sqrt{k}\, L_{\alpha,b}\|\mathbf{e} - \mathbf{e}'\| \, \|\mathbf{x}\| \quad (73)$$

Proof. From Lemma B.5, the router on a fixed active set can be written as $\boldsymbol{\alpha}_{b,I}(\mathbf{e}) = \mathrm{softmax}(\mathbf{z}_{b,I}(\mathbf{e}))$ and $z_{b,m}(\mathbf{e}) = \zeta_{b,m}(\mathbf{e}) + \tau_{\mathrm{lang}}\rho_m(\mathbf{e})$. Since, $\zeta_{b,m}(\mathbf{e}) = \frac{\langle \mathrm{RMSNorm}(\mathbf{q}_b(\mathbf{e})), \widehat{\mathbf{k}}_m\rangle}{\sqrt{d_k}T_{\mathrm{attn}}}$, Cauchy-Schwarz inequality, and $\|\widehat{\mathbf{k}}_m\|_2 \le \sqrt{d_k}$ imply,

$$|\zeta_{b,m}(\mathbf{e}) - \zeta_{b,m}(\mathbf{e}')| \le \frac{1}{\sqrt{d_k}T_{\mathrm{attn}}} \left|\left\langle \mathrm{RMSNorm}(\mathbf{q}_b(\mathbf{e})) - \mathrm{RMSNorm}(\mathbf{q}_b(\mathbf{e}')), \widehat{\mathbf{k}}_m \right\rangle\right| \\ \le \frac{1}{T_{\mathrm{attn}}}\|\mathrm{RMSNorm}(\mathbf{q}_b(\mathbf{e})) - \mathrm{RMSNorm}(\mathbf{q}_b(\mathbf{e}'))\|_2$$

From Lemma B.10 and Proposition B.12,

$$\|\mathrm{RMSNorm}(\mathbf{q}_b(\mathbf{e})) - \mathrm{RMSNorm}(\mathbf{q}_b(\mathbf{e}'))\|_2 \le \frac{1}{\rho_{\min}}\|\mathbf{q}_b(\mathbf{e}) - \mathbf{q}_b(\mathbf{e}')\|_2 \le \frac{L_{q,b}}{\rho_{\min}}\|\mathbf{e} - \mathbf{e}'\|.$$

Hence,

$$|\zeta_{b,m}(\mathbf{e}) - \zeta_{b,m}(\mathbf{e}')| \le \frac{L_{q,b}}{T_{\mathrm{attn}}\rho_{\min}}\|\mathbf{e} - \mathbf{e}'\| \quad (74)$$

For the language logit,

$$\rho_m(\mathbf{e}) = \frac{\langle \mathrm{RMSNorm}(\mathbf{R}_{\mathrm{ctx}}\mathbf{e}), \widehat{\mathbf{k}}_m\rangle}{\sqrt{d_k}T_{\mathrm{lang}}}$$

$$\implies |\rho_m(\mathbf{e}) - \rho_m(\mathbf{e}')| \le \frac{\|\mathbf{R}_{\mathrm{ctx}}\|}{T_{\mathrm{lang}}\rho_{\min}}\|\mathbf{e} - \mathbf{e}'\| \quad (75)$$

Combining (74) and (75), $\forall\, m \in I$,

$$|z_{b,m}(\mathbf{e}) - z_{b,m}(\mathbf{e}')| \le \frac{1}{\rho_{\min}} \left(\frac{L_{q,b}}{T_{\text{attn}}} + \frac{\tau_{\text{lang}} \|\mathbf{R}_{\text{ctx}}\|}{T_{\text{lang}}} \right) \|\mathbf{e} - \mathbf{e}'\|$$

Since there are k active logits,

$$\|\mathbf{z}_{b,I}(\mathbf{e}) - \mathbf{z}_{b,I}(\mathbf{e}')\|_2 \le \frac{\sqrt{k}}{\rho_{\min}} \left(\frac{L_{q,b}}{T_{\text{attn}}} + \frac{\tau_{\text{lang}} \|\mathbf{R}_{\text{ctx}}\|}{T_{\text{lang}}} \right) \|\mathbf{e} - \mathbf{e}'\|$$

By Lemma B.11, softmax is $\frac{1}{2}$-Lipschitz in Euclidean norm. Hence,

$$\|\boldsymbol{\alpha}_{b,I}(\mathbf{e}) - \boldsymbol{\alpha}_{b,I}(\mathbf{e}')\|_2 \le \frac{1}{2} \|\mathbf{z}_{b,I}(\mathbf{e}) - \mathbf{z}_{b,I}(\mathbf{e}')\|_2$$

which proves (70) with the constant in (71). For the routed operator, since the active set is fixed,

$$\mathbf{S}_b(\mathbf{e}) - \mathbf{S}_b(\mathbf{e}') = \sum_{m \in I} (\alpha_{b,m}(\mathbf{e}) - \alpha_{b,m}(\mathbf{e}')) \mathbf{C}_m$$

$$\implies \|\mathbf{S}_b(\mathbf{e}) - \mathbf{S}_b(\mathbf{e}')\| \le \sum_{m \in I} |\alpha_{b,m}(\mathbf{e}) - \alpha_{b,m}(\mathbf{e}')| \, \|\mathbf{C}_m\| \le R_C \|\boldsymbol{\alpha}_{b,I}(\mathbf{e}) - \boldsymbol{\alpha}_{b,I}(\mathbf{e}')\|_1 \quad (76)$$

$$\le R_C \sqrt{k} \|\boldsymbol{\alpha}_{b,I}(\mathbf{e}) - \boldsymbol{\alpha}_{b,I}(\mathbf{e}')\|_2 \quad (77)$$

Combining (76) and (70) proves (72). Since,

$$\Delta \mathbf{W}^{\ell}(\mathbf{e}) - \Delta \mathbf{W}^{\ell}(\mathbf{e}') = \frac{\alpha_{\text{L}}}{r} \mathbf{B}_{\ell} g_{\ell} \big(\mathbf{S}_b(\mathbf{e}) - \mathbf{S}_b(\mathbf{e}') \big) \mathbf{A}_{\ell}$$

By submultiplicativity,

$$\|\Delta \mathbf{W}^{\ell}(\mathbf{e}) - \Delta \mathbf{W}^{\ell}(\mathbf{e}')\| \le \frac{\alpha_{\text{L}}}{r} \|\mathbf{B}_{\ell}\| \, |g_{\ell}| \, \|\mathbf{S}_b(\mathbf{e}) - \mathbf{S}_b(\mathbf{e}')\| \, \|\mathbf{A}_{\ell}\|$$

Using (72) and then applying the resulting operator to $\mathbf{x}$ proves (73). □

B.14 connects the full adapter update to the lower-dimensional rank-space object that the router controls and shows that the first-order effect of a routed update is captured by a compressed gradient.

Proposition B.14 (Exact compressed-gradient identity)**.** *Fix an adapted layer $\ell \in \mathcal{B}_b$. Let $\mathbf{G}_{\ell}$ be the local dense gradient with respect to the effective weight update at layer ℓ, to ensure that the first-order loss variation induced by a dense update $\mathbf{U}$ is $\langle \mathbf{G}_{\ell}, \mathbf{U} \rangle_F$. Then the dynamic routed part satisfies:*

$$\left\langle \mathbf{G}_{\ell}, \frac{\alpha_{\text{L}}}{r} g_{\ell} \mathbf{B}_{\ell} \mathbf{S}_b(c) \mathbf{A}_{\ell} \right\rangle_F = \frac{\alpha_{\text{L}}}{r} g_{\ell} \left\langle \mathbf{B}_{\ell}^{\top} \mathbf{G}_{\ell} \mathbf{A}_{\ell}^{\top}, \mathbf{S}_b(c) \right\rangle_F \quad (78)$$

Hence, the first-order optimization signal for the routed operator is the compressed rank-space gradient $\mathbf{B}_{\ell}^{\top} \mathbf{G}_{\ell} \mathbf{A}_{\ell}^{\top}$.

Proof. Via definition of the Frobenius inner product, $\langle \mathbf{G}_{\ell}, \mathbf{B}_{\ell} \mathbf{S}_b(c) \mathbf{A}_{\ell} \rangle_F = \text{tr}\left(\mathbf{G}_{\ell}^{\top} \mathbf{B}_{\ell} \mathbf{S}_b(c) \mathbf{A}_{\ell} \right)$. From cyclic invariance of the trace, $\text{tr}\left(\mathbf{G}_{\ell}^{\top} \mathbf{B}_{\ell} \mathbf{S}_b(c) \mathbf{A}_{\ell} \right) = \text{tr}\left(\mathbf{A}_{\ell} \mathbf{G}_{\ell}^{\top} \mathbf{B}_{\ell} \mathbf{S}_b(c) \right)$. Observe that,

$$\text{tr}\left(\mathbf{A}_{\ell} \mathbf{G}_{\ell}^{\top} \mathbf{B}_{\ell} \mathbf{S}_b(c) \right) = \text{tr}\left(\left(\mathbf{B}_{\ell}^{\top} \mathbf{G}_{\ell} \mathbf{A}_{\ell}^{\top} \right)^{\top} \mathbf{S}_b(c) \right) = \left\langle \mathbf{B}_{\ell}^{\top} \mathbf{G}_{\ell} \mathbf{A}_{\ell}^{\top}, \mathbf{S}_b(c) \right\rangle_F$$

$$\implies \left\langle \mathbf{G}_{\ell}, \frac{\alpha_{\text{L}}}{r} g_{\ell} \mathbf{B}_{\ell} \mathbf{S}_b(c) \mathbf{A}_{\ell} \right\rangle_F = \frac{\alpha_{\text{L}}}{r} g_{\ell} \left\langle \mathbf{B}_{\ell}^{\top} \mathbf{G}_{\ell} \mathbf{A}_{\ell}^{\top}, \mathbf{S}_b(c) \right\rangle_F$$

□

B.15 lifts the norm-control argument to the Transformer attention projections.

Theorem B.15 (Norm bound and instruction stability for attention-projection updates)**.** *Consider a Transformer attention block $\ell \in \mathcal{B}_b$ with projection-specific updates*

$$\Delta \mathbf{W}^{\ell,p}(\mathbf{h}_\ell; b, c) = \frac{\alpha_{\mathrm{L}}}{r} \mathbf{B}_\ell^p \big(\mathbf{I}_r + g_\ell^p \mathbf{S}_b(c)\big) \mathbf{A}_\ell^p \tag{79}$$

where, $p \in \{Q, K, V, O\}$. Assume $\|\mathbf{A}_\ell^p\| \le R_A^p$ and $\|\mathbf{B}_\ell^p\| \le R_B^p$. Then, $\forall\, p \in \{Q, K, V, O\}$,

$$\|\Delta \mathbf{W}^{\ell,p}(\mathbf{h}_\ell; b, c)\| \le \frac{\alpha_{\mathrm{L}}}{r} R_B^p \big(1 + |g_\ell^p| R_C\big) R_A^p \tag{80}$$

Moreover, on a fixed active-set region with the block states held fixed as in Proposition B.12,

$$\|\Delta \mathbf{W}^{\ell,p}(\mathbf{e}) - \Delta \mathbf{W}^{\ell,p}(\mathbf{e}')\| \le \frac{\alpha_{\mathrm{L}}}{r} R_B^p |g_\ell^p| R_A^p R_C \sqrt{k}\, L_{\alpha,b} \|\mathbf{e} - \mathbf{e}'\| \tag{81}$$

Proof. Equation (80) is Corollary B.8.1 applied to $(\mathbf{A}_\ell^p, \mathbf{B}_\ell^p)$. For the stability bound,

$$\Delta \mathbf{W}^{\ell,p}(\mathbf{e}) - \Delta \mathbf{W}^{\ell,p}(\mathbf{e}') = \frac{\alpha_{\mathrm{L}}}{r} \mathbf{B}_\ell^p g_\ell^p \big(\mathbf{S}_b(\mathbf{e}) - \mathbf{S}_b(\mathbf{e}')\big) \mathbf{A}_\ell^p$$

Using submultiplicativity gives,

$$\|\Delta \mathbf{W}^{\ell,p}(\mathbf{e}) - \Delta \mathbf{W}^{\ell,p}(\mathbf{e}')\| \le \frac{\alpha_{\mathrm{L}}}{r} R_B^p |g_\ell^p| R_A^p \|\mathbf{S}_b(\mathbf{e}) - \mathbf{S}_b(\mathbf{e}')\| \tag{82}$$

From proposition B.13,

$$\|\mathbf{S}_b(\mathbf{e}) - \mathbf{S}_b(\mathbf{e}')\| \le R_C \sqrt{k}\, L_{\alpha,b} \|\mathbf{e} - \mathbf{e}'\| \tag{83}$$

From (83) and (82), we get (81). □

Theorem B.16 explains the mechanism of the shared atom memory receiving the learning signal from all layers in a block. For each layer $\ell \in \mathcal{B}_b$, define the rank-space forward quantities:

$$\mathbf{s}_\ell := \mathbf{A}_\ell \mathbf{x}_\ell \tag{84}$$
$$\mathbf{d}_\ell := \mathbf{S}_b(c) \mathbf{s}_\ell \tag{85}$$
$$\mathbf{t}_\ell := \mathbf{s}_\ell + g_\ell \mathbf{d}_\ell \tag{86}$$
$$\delta \mathbf{h}_\ell := \frac{\alpha_{\mathrm{L}}}{r} \mathbf{B}_\ell \mathbf{t}_\ell \tag{87}$$

where, $\mathbf{x}_\ell$ denotes the input vector to the adapted projection. Let $\mathcal{L}$ be the scalar training loss. Define:

$$\mathbf{r}_\ell := \frac{\partial \mathcal{L}}{\partial \mathbf{t}_\ell} = \frac{\alpha_{\mathrm{L}}}{r} \mathbf{B}_\ell^\top \frac{\partial \mathcal{L}}{\partial \delta \mathbf{h}_\ell} \in \mathbb{R}^r \tag{88}$$

Theorem B.16 (Exact blockwise gradient factorization)**.** *For the shared routed operator reused across block $\mathcal{B}_b$,*

$$\nabla_{\mathbf{S}_b(c)} \mathcal{L} = \sum_{\ell \in \mathcal{B}_b} g_\ell \mathbf{r}_\ell \mathbf{s}_\ell^\top \tag{89}$$

Consequently, the direct value-path gradients through block b are

$$\nabla_{\mathbf{C}_m} \mathcal{L} = \alpha_{b,m}(c) \nabla_{\mathbf{S}_b(c)} \mathcal{L} \tag{90}$$

and

$$\frac{\partial \mathcal{L}}{\partial \alpha_{b,m}(c)} = \big\langle \nabla_{\mathbf{S}_b(c)} \mathcal{L}, \mathbf{C}_m \big\rangle_F \tag{91}$$

$m = 1, 2, 3, 4 \ldots, M$. *Moreover, the gate gradient is,*

$$\frac{\partial \mathcal{L}}{\partial \eta_\ell} = \sigma(\eta_\ell)(1 - \sigma(\eta_\ell)) \langle \mathbf{r}_\ell, \mathbf{d}_\ell \rangle \tag{92}$$

for $\ell \in \mathcal{B}_b$.

Proof. For a fixed layer ℓ, the only quantity in (87) that depends directly on $\mathbf{S}_b(c)$ is $\mathbf{d}_\ell = \mathbf{S}_b(c)\mathbf{s}_\ell$. Hence, if $\mathrm{d}\mathbf{S}_b$ is a perturbation of $\mathbf{S}_b(c)$, then $\mathrm{d}\mathbf{d}_\ell = (\mathrm{d}\mathbf{S}_b)\mathbf{s}_\ell$. Since, $\mathbf{t}_\ell = \mathbf{s}_\ell + g_\ell\mathbf{d}_\ell$, we have $\mathrm{d}\mathbf{t}_\ell = g_\ell \mathrm{d}\mathbf{d}_\ell = g_\ell(\mathrm{d}\mathbf{S}_b)\mathbf{s}_\ell$. By definition of $\mathbf{r}_\ell$, $\mathrm{d}\mathcal{L}_\ell = \langle \mathbf{r}_\ell, \mathrm{d}\mathbf{t}_\ell\rangle = \langle \mathbf{r}_\ell, g_\ell(\mathrm{d}\mathbf{S}_b)\mathbf{s}_\ell\rangle$. As g_ℓ is scalar, $\mathrm{d}\mathcal{L}_\ell = g_\ell \mathbf{r}_\ell^\top(\mathrm{d}\mathbf{S}_b)\mathbf{s}_\ell$. Since,

$$\left\langle \mathbf{r}_\ell\mathbf{s}_\ell^\top, \mathrm{d}\mathbf{S}_b\right\rangle_F = \mathrm{tr}\left((\mathbf{r}_\ell\mathbf{s}_\ell^\top)^\top \mathrm{d}\mathbf{S}_b\right) = \mathrm{tr}\left(\mathbf{s}_\ell\mathbf{r}_\ell^\top \mathrm{d}\mathbf{S}_b\right) = \mathbf{r}_\ell^\top(\mathrm{d}\mathbf{S}_b)\mathbf{s}_\ell$$

$$\implies \mathrm{d}\mathcal{L}_\ell = \left\langle g_\ell\mathbf{r}_\ell\mathbf{s}_\ell^\top, \mathrm{d}\mathbf{S}_b\right\rangle_F$$

$$\implies \mathrm{d}\mathcal{L} = \sum_{\ell\in\mathcal{B}_b}\mathrm{d}\mathcal{L}_\ell = \sum_{\ell\in\mathcal{B}_b}\left\langle g_\ell\mathbf{r}_\ell\mathbf{s}_\ell^\top, \mathrm{d}\mathbf{S}_b\right\rangle_F = \left\langle \sum_{\ell\in\mathcal{B}_b} g_\ell\mathbf{r}_\ell\mathbf{s}_\ell^\top, \mathrm{d}\mathbf{S}_b\right\rangle_F \tag{93}$$

From the definition of the Frobenius gradient, this proves (89). Next, use the linear decomposition $\mathbf{S}_b(c) = \sum_{m=1}^M \alpha_{b,m}(c)\mathbf{C}_m$. If the routing weights are held fixed and the atom $\mathbf{C}_m$ is perturbed, then,

$$\mathrm{d}\mathbf{S}_b = \sum_{m=1}^M \alpha_{b,m}(c)\mathrm{d}\mathbf{C}_m \tag{94}$$

Substituting (94) in (93),

$$\mathrm{d}\mathcal{L} = \left\langle \nabla_{\mathbf{S}_b(c)}\mathcal{L}, \mathrm{d}\mathbf{S}_b\right\rangle_F = \left\langle \nabla_{\mathbf{S}_b(c)}\mathcal{L}, \sum_{m=1}^M \alpha_{b,m}(c)\mathrm{d}\mathbf{C}_m\right\rangle_F = \sum_{m=1}^M \alpha_{b,m}(c)\left\langle \nabla_{\mathbf{S}_b(c)}\mathcal{L}, \mathrm{d}\mathbf{C}_m\right\rangle_F$$

$$\implies \nabla_{\mathbf{C}_m}\mathcal{L} = \alpha_{b,m}(c)\nabla_{\mathbf{S}_b(c)}\mathcal{L}$$

which proves (90). If, instead, the atoms are held fixed and the routing weights are perturbed, then

$$\mathrm{d}\mathbf{S}_b = \sum_{m=1}^M (\mathrm{d}\alpha_{b,m}(c))\mathbf{C}_m$$

$$\implies \mathrm{d}\mathcal{L} = \left\langle \nabla_{\mathbf{S}_b(c)}\mathcal{L}, \mathrm{d}\mathbf{S}_b\right\rangle_F = \sum_{m=1}^M (\mathrm{d}\alpha_{b,m}(c))\left\langle \nabla_{\mathbf{S}_b(c)}\mathcal{L}, \mathbf{C}_m\right\rangle_F$$

The coefficient of $\mathrm{d}\alpha_{b,m}(c)$ is the partial derivative with respect to $\alpha_{b,m}(c)$, proving (91). Finally, as, $g_\ell = \sigma(\eta_\ell)$ and $\mathbf{t}_\ell = \mathbf{s}_\ell + g_\ell\mathbf{d}_\ell$,

$$\frac{\partial \mathbf{t}_\ell}{\partial \eta_\ell} = \sigma(\eta_\ell)(1-\sigma(\eta_\ell))\mathbf{d}_\ell$$

Using the chain rule, we get:

$$\frac{\partial \mathcal{L}}{\partial \eta_\ell} = \left\langle \frac{\partial \mathcal{L}}{\partial \mathbf{t}_\ell}, \frac{\partial \mathbf{t}_\ell}{\partial \eta_\ell}\right\rangle = \sigma(\eta_\ell)(1-\sigma(\eta_\ell))\langle \mathbf{r}_\ell, \mathbf{d}_\ell\rangle$$

This proves (92). □

A direct consequence of the blockwise gradient factorization is that the router logits receive an interpretable training signal (Corollary B.16.1).

Corollary B.16.1 (Exact logit gradients on a fixed active set)**.** *Let I be a fixed active set. Define $\psi_{b,m} := \left\langle \nabla_{\mathbf{S}_b(c)}\mathcal{L}, \mathbf{C}_m\right\rangle_F$ and $\bar{\psi}_b := \sum_{j\in I}\alpha_{b,j}(c)\psi_{b,j}$. If $\boldsymbol{\alpha}_{b,I}(c) = \mathit{softmax}(\mathbf{z}_{b,I}(c))$, then $\forall m \in I$,*

$$\frac{\partial \mathcal{L}}{\partial z_{b,m}(c)} = \alpha_{b,m}(c)(\psi_{b,m} - \bar{\psi}_b) \tag{95}$$

$$\frac{\partial \mathcal{L}}{\partial \zeta_{b,m}} = \alpha_{b,m}(c)(\psi_{b,m} - \bar{\psi}_b) \tag{96}$$

$$\frac{\partial \mathcal{L}}{\partial \rho_m(c)} = \tau_{\mathrm{lang}}\alpha_{b,m}(c)(\psi_{b,m} - \bar{\psi}_b) \tag{97}$$

Proof. From B.16, $\frac{\partial \mathcal{L}}{\partial \alpha_{b,j}(c)} = \psi_{b,j}$. On the fixed active set, $\alpha_{b,j}(c) = \frac{\exp(z_{b,j}(c))}{\sum_{q \in I} \exp(z_{b,q}(c))}$. From B.11,

$$\frac{\partial \alpha_{b,j}(c)}{\partial z_{b,m}(c)} = \alpha_{b,j}(c)(\delta_{jm} - \alpha_{b,m}(c))$$

Using the chain rule,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_{b,m}(c)} = \sum_{j \in I} \frac{\partial \mathcal{L}}{\partial \alpha_{b,j}(c)} \frac{\partial \alpha_{b,j}(c)}{\partial z_{b,m}(c)} &= \sum_{j \in I} \psi_{b,j} \alpha_{b,j}(c)(\delta_{jm} - \alpha_{b,m}(c)) \\ &= \psi_{b,m}\alpha_{b,m}(c) - \alpha_{b,m}(c) \sum_{j \in I} \psi_{b,j}\alpha_{b,j}(c) = \alpha_{b,m}(c)(\psi_{b,m} - \bar{\psi}_b)\end{aligned}$$

Hence, this proves (95). From Lemma B.5,

$$\begin{aligned} & z_{b,m}(c) = \zeta_{b,m} + \tau_{\text{lang}} \rho_m(c) \\ \implies & \frac{\partial z_{b,m}(c)}{\partial \zeta_{b,m}} = 1 \text{ and } \frac{\partial z_{b,m}(c)}{\partial \rho_m(c)} = \tau_{\text{lang}}\end{aligned}$$

Using the chain rule gives (96) and (97). □

Theorem B.16 and Corollary B.16.1 give the contribution of block b. The atoms $\mathbf{C}_m$ and the raw language logits $\rho_m(c)$ are shared by all blocks, so their full gradients add these contributions over blocks, for example $\nabla_{\mathbf{C}_m} \mathcal{L} = \sum_b \alpha_{b,m}(c) \nabla_{\mathbf{S}_{b(c)}} \mathcal{L}$.

C Experimental Details

C.1 Tasks and checkpoint selection

Table 7 lists the source tasks and the held-out evaluation sets. Each adapter saw 300 training examples from every source task, and each task kept its own instruction. For checkpoint selection, we took the first 20 examples from each held-out source-task pool for the first four tasks, giving 80 validation examples. The selection rule applied to this set is the one described in Section 5.1.

Table 7: Source tasks and held-out evaluation sets.

Set	Contents
Source tasks	ARC (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), BoolQ (Clark et al., 2019), RACE (Lai et al., 2017), HellaSwag (Zellers et al., 2019)
Validation tasks	ARC, OpenBookQA, BoolQ, RACE
MMLU-12 subjects	College physics, philosophy, high-school psychology, professional law, machine learning, marketing, econometrics, world religions, clinical knowledge, US foreign policy, formal logic, astronomy
MMLU-Pro categories	Math, physics, chemistry, biology, engineering, law, economics, psychology, history, philosophy

Table 8 gives the number of questions in each evaluation set. GPQA-Diamond has 198 questions. All transfer runs split them with the same data seed, so every method and seed is tested on the same 99 questions.

Table 8: Evaluation set sizes. The full-size evaluation reused the adapters from the first evaluation.

Benchmark	First evaluation	Full-size evaluation
MMLU-12	302 validation questions	3,766 official test questions
MMLU-Pro (ten categories)	1,000 (100 per category)	9,091 (all questions)
GPQA-Diamond	99 held-out questions	The same 99 questions

Table 9: Training settings for the main 7B comparison and scale study, including the configuration differences.

Setting	Value
Backbone	Qwen2.5-Instruct (1.5B, 3B, 7B, 14B)
Adapted modules	q, k, v, o, gate, up, down
Rank / scaling	8 / 16
Atoms / active atoms / block size	16 / 4 / 2
Training / source examples	3 epochs / 300 per task
Batch / evaluation batch	2 / 8
Learning rate / weight decay	2×10^{-4} / 0.05
Router / gate LR multipliers	2 / 4
Adapter dropout	0.05
Maximum input / generated tokens	1,024 / 8, greedy decoding
Instruction encoder	Frozen all-MiniLM-L6-v2
Instruction coefficients	$\tau_{\text{lang}} = 0.5$ and $\lambda_{\text{ctx}} = 0.25$ for IQ, both zero for Queryable
Gate parameter η_ℓ at initialization	1.0 up to 7B ($g_\ell \approx 0.73$) and 0.0 at 14B ($g_\ell = 0.5$)
Atom initialization standard deviation	0.30 for the main 7B MMLU-12 comparison, 0.15 for the other runs up to 7B and the component study, and 0.05 at 14B
Depth summary	Uniform mean for Queryable and learned attention for IQ, with attention enabled in the component baseline
Decode routing	Routes fixed after the first decoding step (Appendix A)
Precision / hardware	bf16 on two H100 80GB GPUs
Software	PyTorch 2.7.0, Transformers 4.57.1, CUDA 12.8

C.2 Seeds and reported statistics

Each three-run setting used one set of three seeds for all methods, with one exception. In the main 7B MMLU-12 comparison, Queryable's third run used a separate seed, and the full-size MMLU evaluation reused these adapters. The five-run GPQA experiments at 7B and 14B added two further seeds. Because of the exception, pairing by seed for the sign test gives two pairs from MMLU-12.

Standard deviations describe variation across seeds, and every table keeps the values we recorded. The main MMLU-12 column and the component ablation mostly use population standard deviations. In the MMLU-12 column, Queryable's 0.007 uses the sample convention, and Lily's value rounds to 0.002 under either convention. Sample standard deviations are used everywhere else, including the scale study and the full-size evaluations, and the run times in Table 4 follow the same convention. For the mean ranks, we rank the displayed three-decimal mean accuracies within each benchmark and give tied methods their average rank. A method's mean rank is the average of its three benchmark ranks, and the full-size table follows the same convention. Comparisons of MMLU-Pro's ordering across evaluation sizes use unrounded means, which distinguish LoRA and DoRA even when both sampled means round to 0.462.

C.3 Baseline implementations and additional experiments

All methods in the main comparison shared the training setup and the checkpoint rule. For the hypernetwork baselines, we learned the generator parameters on the source-task mixture. The additional experiments follow their own selection rules. In the small-model study, we selected checkpoints by the lowest validation loss, and Appendix F states the rules for the other 0.5B experiments. The synthetic study reports the lowest MSE reached on each split (Appendix I).

D Full-Size Evaluation and Paired Statistics

Section 5.2 and Table 2 report the full-size results, and this appendix gives the statistics behind the main-text comparisons.

D.1 Run-level sign test

Pairing Queryable and LoRA runs by seed across eight settings gives 27 comparisons (Table 10). Among pairs with different held-out accuracies, Queryable has 19 wins and four losses, giving the one-sided fair-coin tail probability

$$\Pr\{X \geq 19\}, \quad X \sim \mathrm{Binomial}(23, 1/2), \qquad p = \sum_{j=19}^{23} \binom{23}{j} 2^{-23} = 0.0013 \text{ (rounded)}.$$

Under the fair-coin null model, the 23 pairs with a winner are treated as independent Bernoulli outcomes. Settings that share data or seeds can be dependent, so we treat this test as a descriptive summary of these comparisons. The companion splits contributed paired comparisons between

Table 10: Pairs of Queryable and LoRA runs used in the sign test.

Setting	Wins	Ties	Losses	Pairs
MMLU-12, 7B	2	0	0	2
MMLU-Pro, 7B	3	0	0	3
GPQA, 7B	2	0	3	5
GPQA, 1.5B	2	1	0	3
GPQA, 3B	2	0	1	3
GPQA, 14B	3	2	0	5
MMLU-STEM12, 7B	3	0	0	3
MMLU representative 13, 7B	2	1	0	3
Total	19	4	4	27

Queryable and LoRA, with scores of 0.582 versus 0.553 on 182 MMLU-STEM12 questions and 0.694 versus 0.678 on 291 questions from 13 MMLU subjects disjoint from MMLU-12.

D.2 Question-clustered bootstrap

For each of the 9,091 MMLU-Pro questions, we averaged the correctness difference across three trained seeds paired across methods. Resampling questions with replacement 4,000 times kept the paired seed outcomes together, accounting for dependence within a question. We report the 2.5th and 97.5th percentiles. Queryable scored 2.7 percentage points above LoRA with 95% interval $[2.3, 3.1]$, and IQ scored 2.6 points above LoRA with interval $[2.1, 3.0]$. These intervals measure question-level variation conditional on the trained adapters. Retraining and task-population uncertainty are separate sources of variation.

E Inference Cost Measurements

E.1 Seven-billion-parameter prefill and generation

We measured Qwen2.5-7B-Instruct on H100 in bf16 across sequence lengths and batch sizes, repeating each configuration twice (Table 11). The adapters were freshly initialized, and the frozen instruction embedding was computed before timing. Drift between runs was at most 2.8% for long prompts and larger batches, with a maximum of about 5.5% across all measured configurations. Absolute increments and percentage overheads were rounded separately from the raw measurements. Table 11 shows how the overhead changes with prompt length and batch size. As the prompt grows,

Table 11: Prefill overhead relative to LoRA and generation throughput in the separate generation setting. Decode uses a 256-token prompt and 128 new tokens at batch one.

Setting	LoRA	Queryable	IQ
Batch 1, 1,024 tokens	40.89 ms	+6.7% (+2.74 ms)	+9.2% (+3.77 ms)
Batch 1, 2,048 tokens	74.92 ms	+3.2% (+2.40 ms)	+4.6% (+3.43 ms)
Batch 1, 4,096 tokens	149.68 ms	+1.3% (+1.90 ms)	+1.9% (+2.77 ms)
Batch 1, 8,192 tokens	311.84 ms	+0.6% (+1.88 ms)	+1.0% (+3.00 ms)
512 tokens, batch 1	reference	+42%	+52%
512 tokens, batch 4	reference	+3.0%	+3.7%
512 tokens, batch 8	reference	+1.9%	+2.1%
Decode, routes fixed after first step	47.5 tokens/s	33.8 tokens/s	33.6 tokens/s
Decode, per-token routes	n/a	29.0 tokens/s	27.0 tokens/s

LoRA's prefill time rises steeply and the extra time for Queryable and IQ stays at a few milliseconds, so the relative overhead keeps falling. A larger batch has the same effect at a fixed prompt length. For a single 512-token prompt, LoRA took 24.65 ms, and Queryable and IQ took 35.02 and 37.51 ms. Launching the routing kernels is a larger share of the work for such short inputs on this hardware.

Generation shows the value of computing the route once per request. Fixing the routes after the first decoding step raised throughput by about 16% for Queryable and 24% for IQ compared with recomputing the route at every token (Appendix A describes the refresh).

E.2 Inference comparison with other dynamic adapters

In the separate adapter comparison, Queryable and IQ had 27.9% and 40.5% prefill overhead relative to LoRA, and both ran faster than the generated adapters (Table 12). Their adapter arithmetic added 0.6% to FLOPs. Since routing also requires kernel launches, FLOPs and measured execution time capture different costs.

Table 12: Inference efficiency in the separate adapter comparison. Lower latency and overhead are better, and so is higher generation throughput. Tokenization and dataloader construction occur outside the timed region. Bold marks Queryable. Both proposed variants have lower latency than the generated adapters in this setup.

Measurement	**LoRA**	**DoRA**	**RepLoRA**	**HyRA**	**DoRAN**	**Queryable**	**IQ**
Forward/prefill lat. (ms) ↓	33.1	30.3	53.0	83.3	51.0	**42.3**	46.5
Fwd. overhead vs LoRA (%) ↓	0.0	-8.4	60.2	151.7	54.1	**27.9**	40.5
Throughput (new toks/s) ↑	31.8	30.3	18.0	11.3	18.7	**22.0**	20.1
FLOP overhead vs LoRA (%) ↓	0.0	0.0	0.0	0.0	0.0	0.6	0.6

F Additional Experiments at 0.5B

F.1 Adapter design and ARC comparison

At 0.5B, Queryable reached $0.6576_{\pm 0.0176}$ evaluation accuracy and $1.0000_{\pm 0.0000}$ training accuracy across three independent ARC runs, using each run's best values (Table 14). Both Queryable variants scored higher than the other dynamic and generated adapters (Table 13 compares their designs and trainable parameter counts).

Table 13: Adapter designs and trainable parameter counts for the compared implementations at 0.5B. Filled cells identify the routing and instruction components used by each method.

Method	Trainable params	Routing?	Instruction?
LoRA (Hu et al., 2022)	4,399,104		
DoRA (Liu et al., 2024)	4,644,864		
RepLoRA (Truong et al., 2025)	145,872,000	Shared static	
HyRA (Diep et al., 2025b)	147,922,752	Generated	
DoRAN (Diep et al., 2025a)	145,270,056	Generated	
MoE-LoRA (Luo et al., 2024)	18,579,456	Yes	
Text-to-LoRA (Charakorn et al., 2025)	148,127,232	Generated	Yes
Lily (Zhong et al., 2025)	2,159,872	Yes	
Queryable LoRA	4,456,936	Yes	
Instruction-Queryable LoRA	4,456,936	Yes	Yes

Table 14: Highest training accuracy and highest evaluation accuracy reached by dynamic adapters in each run, reported as mean $\pm$ standard deviation over three independent runs.

Method	Train Accuracy	Eval Accuracy
Text-to-LoRA (Charakorn et al., 2025)	1.0000 ± 0.0000	0.5599 ± 0.0495
Instruction-Queryable LoRA	1.0000 ± 0.0000	0.6510 ± 0.0113
MoE-LoRA (Luo et al., 2024)	1.0000 ± 0.0000	0.6458 ± 0.0254
Lily (Zhong et al., 2025)	1.0000 ± 0.0000	0.6406 ± 0.0103
Queryable LoRA	1.0000 ± 0.0000	$\mathbf{0.6576 \pm 0.0176}$

F.2 Instruction-router ablation

We fine-tuned Qwen2.5-0.5B-Instruct on GPQA-Diamond for 10 epochs, keeping 4.46M trainable parameters and varying the source or structure of the routing signal (Table 15). For every variant, we selected the checkpoint with the highest training accuracy and chose the lowest training loss when several epochs reached 1.0 accuracy.

The correct instruction gave the highest accuracy in Table 15, and shuffled instructions or random embeddings of the same norm gave the lowest, which suggests that the router uses what the instruction says. Routing from the state and depth summary also performed well in the main Queryable experiments with both instruction coefficients set to zero.

Table 15: Instruction-router ablation on Qwen2.5-0.5B-Instruct and GPQA-Diamond at a fixed trainable parameter count. Each row changes the instruction signal that reaches the router. All variants use the checkpoint-selection rule stated in the text.

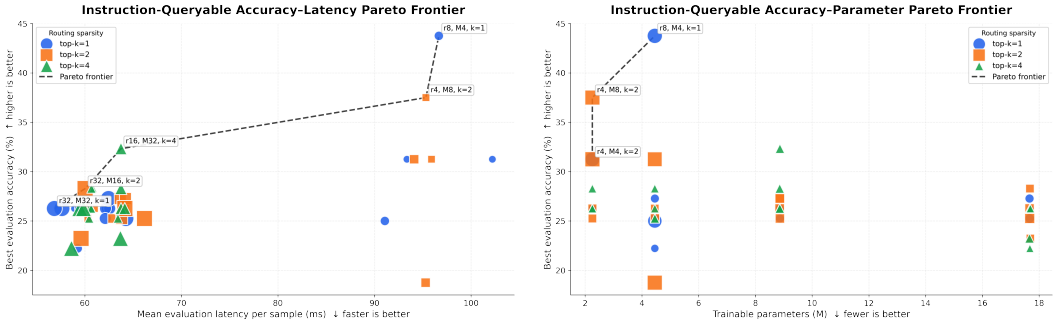
Ablation	Our interpretation	Eval. acc.	Mean eval sec.	Latency / sample
Instruction disabled	base dynamic routing	0.2420	1.6362	0.0739
Generic instruction	Generic non-task-specific language control	0.2500	1.5253	0.0953
Correct instruction	Semantic prior benefit	0.3125	1.5571	0.0973
Shuffled instruction	Tests semantic mismatch	0.1875	1.4997	0.0937
Adversarial wrong instruction	Tests robustness	0.2500	1.5432	0.0964
Random embedding, same norm	Tests capacity versus semantics	0.1875	1.5392	0.0962
State query, prior disabled	Tests whether instruction helps	0.2500	1.5169	0.0948

F.3 Adapter size and routing sparsity

The same GPQA-Diamond setup also let us vary the size of the adapter. We evaluated IQ on GPQA-Diamond with rank $r \in \{4, 8, 16, 32\}$ and shared atoms $M \in \{4, 8, 16, 32\}$. The routing sparsity took values $k \in \{1, 2, 4, 8\}$ with $k \leq M$. Training ran for at most 10 epochs, and every configuration

used the same frozen Qwen2.5-0.5B-Instruct backbone and the same training and evaluation setup. For every variant, we selected the checkpoint with the highest observed training accuracy and reported its corresponding held-out accuracy.

Small or medium-sized banks with sparse routing gave the best trade-offs between accuracy and cost in this sweep (the dashed curve in Figure 3 connects the configurations on the measured Pareto frontier).



(a) Accuracy and latency.

(b) Accuracy and parameter count.

Figure 3: **IQ accuracy and cost on GPQA-Diamond.** Each point is one sweep configuration. The left panel plots the selected checkpoint's evaluation accuracy against mean latency per sample, and the right panel plots it against the trainable parameter count. Higher accuracy at lower cost is better. Markers indicate the routing sparsity k, and marker sizes show trainable parameter counts. The dashed curve traces the Pareto frontier.

F.4 Additional small-model study

Across small-model families, IQ matched or exceeded LoRA in 29 of the 39 task and model rows, and at least one proposed variant did so in 34 (Table 16). Each method's checkpoint was selected by the lowest validation loss.

Table 16: Accuracy across models with at most 1B parameters, using checkpoints selected by the lowest validation loss. Higher is better, and bold marks the best accuracy for each task and model, including ties.

Task	Model	LoRA	Queryable	IQ
Orca-Math	LiquidAI/LFM2-700M	25.0%	21.9%	**26.6%**
	LiquidAI/LFM2.5-350M	6.2%	**9.4%**	6.2%
	Qwen/Qwen2.5-Coder-0.5B-Instruct	9.4%	**14.1%**	9.4%
	Qwen/Qwen3-0.6B	25.0%	20.3%	**34.4%**
	amd/ReasonLite-0.6B	20.3%	20.3%	**21.9%**
	amd/ReasonLite-0.6B-Turbo	23.4%	**28.1%**	**28.1%**
	ibm-granite/granite-4.0-350m	7.8%	**17.2%**	10.9%
GSM8K	HuggingFaceTB/SmolLM2-360M-Instruct	10.9%	7.8%	**17.2%**
	LiquidAI/LFM2-700M	42.2%	**45.3%**	37.5%
	LiquidAI/LFM2.5-350M	**23.4%**	6.2%	15.6%
	Qwen/Qwen2.5-0.5B-Instruct	21.9%	21.9%	**23.4%**
	Qwen/Qwen2.5-Coder-0.5B-Instruct	23.4%	25.0%	**28.1%**
	Qwen/Qwen3-0.6B	34.4%	39.1%	**42.2%**
	amd/ReasonLite-0.6B	29.7%	29.7%	**34.4%**
	ibm-granite/granite-4.0-350m	25.0%	17.2%	**26.6%**
MATH	HuggingFaceTB/SmolLM2-360M-Instruct	4.7%	**9.4%**	3.1%
	LiquidAI/LFM2-700M	23.4%	**25.0%**	**25.0%**
	LiquidAI/LFM2.5-350M	**7.8%**	6.2%	**7.8%**
	Qwen/Qwen2.5-Coder-0.5B-Instruct	**10.9%**	9.4%	4.7%
	Qwen/Qwen3-0.6B	10.9%	17.2%	**20.3%**
	amd/ReasonLite-0.6B	26.6%	17.2%	**28.1%**
	ibm-granite/granite-4.0-350m	7.8%	**14.1%**	9.4%
BoolQ	HuggingFaceTB/SmolLM2-360M-Instruct	62.5%	**78.1%**	**78.1%**
	LiquidAI/LFM2-700M	81.2%	**84.4%**	76.6%
	LiquidAI/LFM2.5-350M	64.1%	59.4%	**68.8%**
	Qwen/Qwen2.5-0.5B-Instruct	**70.3%**	**70.3%**	**70.3%**
	Qwen/Qwen2.5-Coder-0.5B-Instruct	**65.6%**	59.4%	**65.6%**
	Qwen/Qwen3-0.6B	78.1%	**84.4%**	78.1%
	amd/ReasonLite-0.6B	**59.4%**	**59.4%**	56.2%
	amd/ReasonLite-0.6B-Turbo	57.8%	62.5%	**70.3%**
	ibm-granite/granite-4.0-350m	71.9%	65.6%	**76.6%**
ARC-Challenge	LiquidAI/LFM2-700M	**68.8%**	67.2%	62.5%
	LiquidAI/LFM2.5-350M	48.4%	**62.5%**	57.8%
	Qwen/Qwen2.5-0.5B-Instruct	50.0%	51.6%	**53.1%**
	Qwen/Qwen2.5-Coder-0.5B-Instruct	**42.2%**	32.8%	39.1%
	Qwen/Qwen3-0.6B	60.9%	57.8%	**65.6%**
	amd/ReasonLite-0.6B	23.4%	29.7%	**32.8%**
	amd/ReasonLite-0.6B-Turbo	21.9%	**25.0%**	18.8%
	ibm-granite/granite-4.0-350m	**48.4%**	37.5%	43.8%

G GRADIENT-FLOW ANALYSIS

We examined the gradient signal across network depth through per-layer norms and gradient concentration in the recorded adapter runs (Figures 4 and 5). The gradient concentration index is the ratio between the maximum and the mean per-layer adapter gradient norm, and lower values mean that learning is spread over more layers. IQ had the lowest concentration index in four of the five epochs. These curves complement Theorem 4.3, whose direct value-path factorization is stated for a fixed active set.

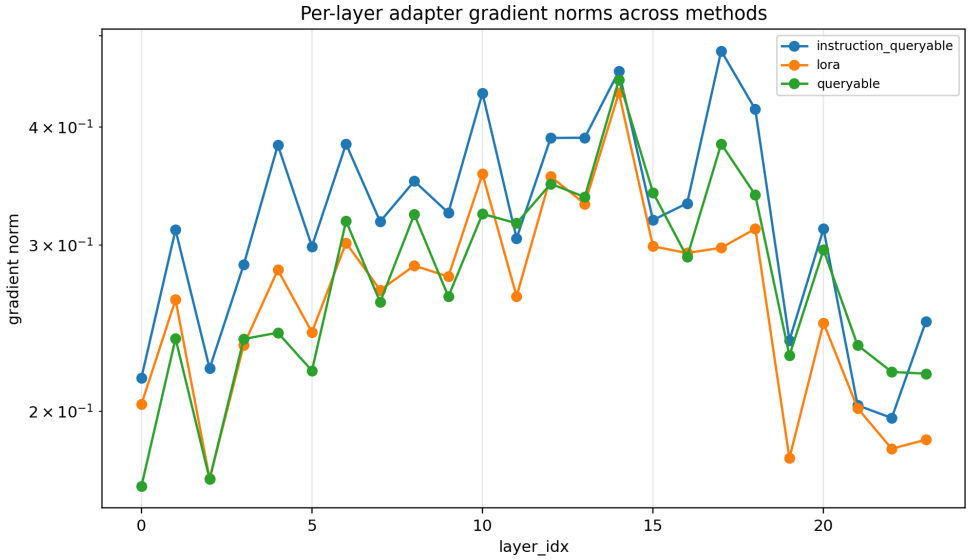


Figure 4: Per-layer gradient norms from the recorded adapter runs, showing the gradient signal across network depth.

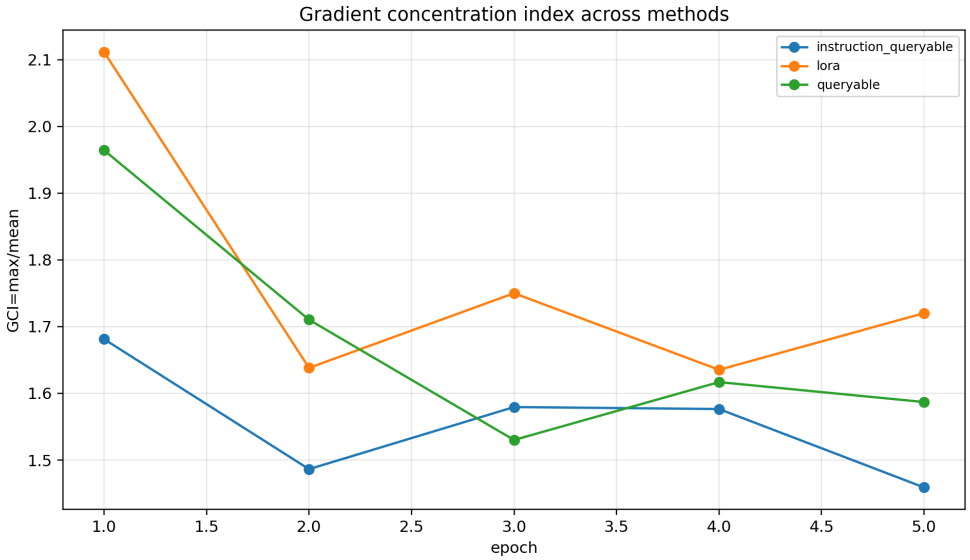
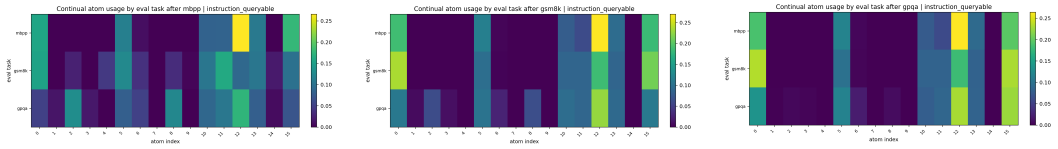


Figure 5: Gradient concentration summarizing the layer-wise profiles from the same adapter runs.

H CONTINUAL ROUTING ANALYSIS

We trained one adapter on MBPP and then continued training on GSM8K and GPQA-Diamond in turn. All adapter parameters carried over between tasks, and the backbone stayed frozen. After each stage, all three tasks were evaluated and each evaluation set's atom-routing distribution was averaged over blocks. The update $\boldsymbol{B}_\ell(\boldsymbol{I}_r + g_\ell S_b(\boldsymbol{c}))\boldsymbol{A}_\ell$ uses the shared memory whose changes we examine as training moves between tasks.

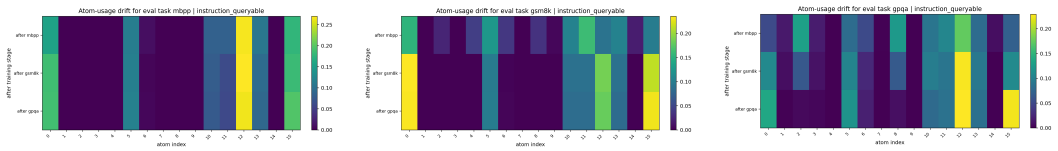
Most routing mass fell on a small subset of atoms after each training stage, with several high-mass atoms used across tasks at different weights (Figure 6). Rows represent evaluation tasks and columns represent shared atoms. This variation in weights is consistent with shared components retrieved according to the task.



after MBPP after GSM8K after GPQA-Diamond

Figure 6: Atom usage by evaluation task after each continual training stage. Repeated high-mass atoms indicate shared memory use, and changes in their mass across tasks indicate adaptive routing.

As training moved to later tasks, MBPP kept much of its high-mass routing pattern, and changes for GSM8K and GPQA-Diamond were concentrated on a few atoms (Figure 7).



eval MBPP eval GSM8K eval GPQA-Diamond

Figure 7: Atom-usage drift for each fixed evaluation task across training stages. The maps show local changes from later tasks and broad agreement with earlier sparse routes.

After the final stage, the entropy of each atom's usage showed a mixture of broadly shared and more specialized atoms (Figure 8). High entropy indicates broad reuse across the three tasks, and low entropy indicates use concentrated on fewer tasks.

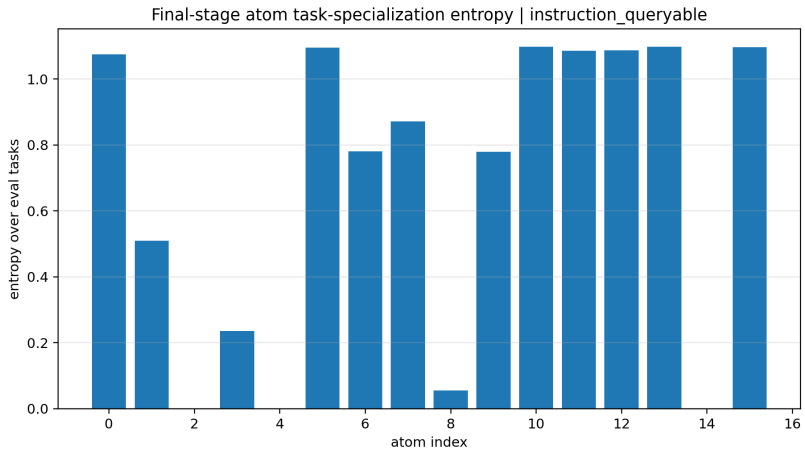
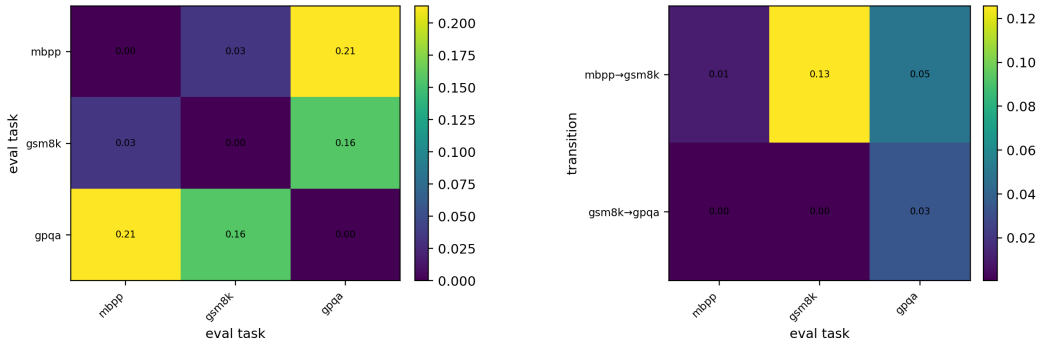


Figure 8: A mixture of high and low entropy atoms suggests that the shared memory bank supports both reusable and selective rank-space corrections.

The final symmetric KL matrix shows similar atom-use distributions for MBPP and GSM8K, with a more distinct distribution for GPQA-Diamond (Figure 9). Route drift was largest when GSM8K was introduced. Moving from GSM8K to GPQA-Diamond then produced smaller changes on the earlier tasks.



FINAL SYMMETRIC KL STAGEWISE ROUTE DRIFT

Figure 9: Final task separation and stagewise route drift. After the full training sequence, GPQA-Diamond uses a more distinct atom mixture in the KL matrix. The drift matrix shows concentrated route changes as each new task is learned.

I SYNTHETIC REGRESSION AND OPTIMIZATION STRESS TESTS

I.1 TARGET SHIFTS AND REPORTING RULE

We used nine stochastic two-dimensional functions from the test-function library of Surjanovic & Bingham (2013), pretraining on 3,000 source samples for 300 epochs at learning rate 3×10^{-3}. Shifting the function parameters gave 1,200 new target examples, of which 400 were used for adaptation and 800 held out. Adaptation ran for 5,000 epochs at learning rate 5×10^{-4}, with additive Gaussian label noise of standard deviation 0.05. The main study used an eight-layer, width-256 regressor and AdamW (Loshchilov & Hutter, 2019), keeping Queryable's trainable parameter count within 10% of LoRA's.

For every method, we report the lowest held-out MSE and lowest training MSE observed over the adaptation run, selecting the held-out value from its own trajectory. Five independent runs give the means and seed standard deviations, with held-out MSE presented first (Tables 17 and 18). Queryable achieved the best PEFT held-out MSE on five of the nine functions, counting ties, with large gains on Dropwave and smaller gains on smooth targets such as Matyas and Sin-Cos.

Table 17: Lowest held-out MSE reached during adaptation on nine stochastic functions over five independent runs (mean ± standard deviation). Bold marks the best result, including ties.

Function	LoRA	DoRA	HyRA	RepLoRA	DoRAN	Queryable
Ackley	0.3734 ±0.0458	0.3716 ±0.0475	39.4747 ±73.3841	**0.3696 ±0.0482**	0.3753 ±0.0433	0.3726 ±0.0495
Dropwave	64.5079 ±57.4368	374.8006 ±786.5186	1677.5055 ±242.8430	12.7792 ±10.4422	357.0902 ±795.7947	**2.2263 ±1.5457**
Langermann	2.2596 ±2.3986	2.5835 ±1.9773	3.7396 ±2.3199	2.2306 ±2.4152	**1.6682 ±2.0511**	1.9344 ±2.1697
Levy	1.3273 ±0.2269	1.3395 ±0.2296	219.8848 ±113.1329	1.5735 ±0.1907	1.4383 ±0.2535	**1.3028 ±0.2163**
Matyas	0.0255 ±0.0051	0.0231 ±0.0018	1534.4436 ±57.6542	0.0233 ±0.0026	0.0258 ±0.0034	**0.0229 ±0.0035**
Michalewicz	0.1301 ±0.0076	0.1299 ±0.0074	0.1321 ±0.0081	0.1297 ±0.0072	**0.1277 ±0.0087**	0.1288 ±0.0076
Rastrigin	**98.7647 ±4.4966**	98.8097 ±3.6418	1048.7748 ±1762.0819	99.2913 ±4.0321	101.5798 ±1.5538	99.6905 ±4.3598
Sin-Cos	0.0039 ±0.0004	0.0040 ±0.0003	0.8615 ±0.2558	0.0039 ±0.0004	0.0046 ±0.0008	**0.0039 ±0.0004**
Styblinski-Tang	67.2361 ±21.6650	75.5779 ±28.1219	30586.1356 ±42757.6886	72.1448 ±27.7898	70.1036 ±21.2876	**62.2928 ±17.7273**

Table 18: Lowest training MSE reached during adaptation on nine stochastic functions over five independent runs (mean ± standard deviation). Bold marks the best result, including ties.

Function	LoRA	DoRA	HyRA	RepLoRA	DoRAN	Queryable
Ackley	0.0754 ±0.0225	0.1544 ±0.0295	40.974 ±75.620	0.0632 ±0.0278	0.1159 ±0.0236	**0.0559 ±0.0159**
Dropwave	24.0509 ±42.8973	266.2681 ±577.2793	1437.6514 ±193.1731	1.3918 ±1.3943	259.8996 ±580.7419	**0.2527 ±0.1777**
Langermann	2.2579 ±2.6938	2.4252 ±2.2126	3.7290 ±2.2653	2.2295 ±2.7166	**1.5239 ±2.1633**	1.8061 ±2.3035
Levy	0.0124 ±0.0053	0.01388 ±0.0065	248.3561 ±154.4848	0.0027 ±0.0010	0.0073 ±0.0034	**0.0009 ±0.0003**
Matyas	0.0034 ±0.0004	0.0061 ±0.0006	1500.0727 ±83.0140	0.0041 ±0.0003	0.0125 ±0.0025	**0.0032 ±0.0003**
Michalewicz	0.0590 ±0.0231	0.0745 ±0.0250	0.1417 ±0.0171	0.0465 ±0.0368	0.0454 ±0.0408	**0.0380 ±0.0202**
Rastrigin	22.4812 ±4.5454	25.4298 ±7.0738	998.8924 ±1676.4226	9.0337 ±5.2511	5.5401 ±1.4612	**3.3663 ±2.6077**
Sin-Cos	0.0002 ±0.0001	0.0003 ±0.0001	0.7841 ±0.2493	0.0001 ±0.0001	0.0003 ±0.0001	**0.0001 ±0.0000**
Styblinski-Tang	2.0728 ±0.5664	2.2513 ±0.4360	29677.6550 ±43197.9384	0.6690 ±0.1545	2.5213 ±0.7855	**0.3862 ±0.0965**

I.2 DEEP, NARROW MODEL

To test adaptation through many narrow layers, we used a GELU multilayer perceptron with 32 hidden layers of width 32 and a scalar output head in the same source-to-target regression setup (Tables 19 and 20).

For each function, source and target examples were drawn independently, with 80% of the 3,000 source samples used for training and 20% for testing over 300 pretraining epochs. Of the 1,200 target examples, 400 were used for adaptation and 800 held out. Inputs were drawn uniformly from a fixed box, and labels included Gaussian noise with standard deviation 0.05. We trained adapters for 5,000 epochs at learning rate 5×10^{-4} with batch size 64. The adapters used rank 8 and scaling 16 with dropout set to zero. Queryable used eight shared atoms with top-2 routing over four blocks. PEFT methods updated their adapter parameters, and SOAP* updated the full pretrained regressor with the SOAP optimizer (Vyas et al., 2025).

Queryable gave the best PEFT held-out result on eight of the nine functions (Tables 19 and 20 report the lowest training and held-out target MSE).

Table 19: Best target-training MSE in the 32-layer, width-32 stress test, with each value giving the lowest MSE observed during adaptation on the stated split. SOAP* updates the full regressor, and bold and italics mark the best result across all methods and PEFT methods, respectively.

Function	SOAP*	LoRA	DoRA	HyRA	RepLoRA	DoRAN	Queryable
Ackley	13.124	13.124	13.123	13.108	13.123	13.119	***0.113***
Dropwave	**0.582**	223.643	234.136	1829.286	1828.609	1805.232	*1.291*
Langermann	5.502	5.502	5.502	5.499	5.502	5.502	***0.021***
Levy	324.969	324.956	324.971	324.447	324.966	324.875	***7.173***
Matyas	**0.017**	0.135	753.808	753.813	753.459	753.748	*0.087*
Michalewicz	0.125	0.125	0.125	0.125	0.125	0.125	***0.054***
Rastrigin	**1.270**	49.880	77.443	4482.271	79.764	4197.973	*22.812*
Sin-Cos	1.011	1.011	1.011	1.011	1.011	1.011	***0.161***
Styblinski-Tang	21.271	11.843	24.708	inf	20.744	385.618	***10.898***

Table 20: Best held-out target MSE in the 32-layer, width-32 stress test, with each value giving the lowest MSE observed during adaptation on the stated split. SOAP* updates the full regressor, and bold and italics mark the best result across all methods and PEFT methods, respectively.

Function	SOAP*	LoRA	DoRA	HyRA	RepLoRA	DoRAN	Queryable
Ackley	13.230	13.227	13.220	13.215	13.222	13.215	***0.673***
Dropwave	25.096	193.122	246.421	1684.105	1683.752	1608.287	***9.828***
Langermann	5.815	5.815	5.815	5.811	5.814	5.815	***0.195***
Levy	337.855	337.855	337.855	337.855	337.855	337.855	***136.888***
Matyas	**0.068**	0.870	749.881	749.881	749.881	749.881	*0.345*
Michalewicz	0.136	0.136	0.136	0.136	0.136	0.136	***0.135***
Rastrigin	156.795	113.054	1172.995	4772.434	***108.480***	4127.224	127.405
Sin-Cos	0.924	0.924	0.924	0.923	0.924	0.923	***0.920***
Styblinski-Tang	14.063	11.736	24.460	inf	21.886	384.235	***8.953***

J ASSETS AND COMPUTE

Models and datasets. The main experiments used Qwen2.5-Instruct models with all-MiniLM-L6-v2 as the instruction encoder and established multiple-choice datasets (source-task references are in Appendix C). Target tasks follow Hendrycks et al. (2021a); Wang et al. (2024); Rein et al. (2024). Additional small-model and routing studies used GSM8K (Cobbe et al., 2021) and MBPP (Austin et al., 2021) along with SuperGLUE tasks (Wang et al., 2019). The small-model study also used MATH (Hendrycks et al., 2021b) and Orca-Math (Mitra et al., 2024). The synthetic functions came from Surjanovic & Bingham (2013).

All models and datasets come from their original providers under their access and license terms.

Compute and reproducibility. Table 9 specifies the primary run environment, and each additional timing experiment states its own configuration.

J.1 Asset identifiers and recorded notices

The tables give identifiers and notices recorded with the experiments, and users should consult the original cards for current terms before obtaining or redistributing an asset.

Table 21: Backbone and encoder identifiers with recorded notices, including the four Qwen2.5-Instruct sizes in the main comparisons. The other checkpoints appear in the additional small-model studies.

Identifier	Recorded notice / original source
`Qwen/Qwen2.5-0.5B`, `Qwen/Qwen2.5-0.5B-Instruct`	Apache 2.0, Qwen Team
`Qwen/Qwen2.5-1.5B-Instruct`, `Qwen/Qwen2.5-3B-Instruct`, `Qwen/Qwen2.5-7B-Instruct`, `Qwen/Qwen2.5-14B-Instruct`	Each size's model card, Qwen Team
`Qwen/Qwen2.5-Coder-0.5B-Instruct`, `Qwen/Qwen3-0.6B`	Apache 2.0, Qwen Team
`HuggingFaceTB/SmolLM2-360M-Instruct`	Apache 2.0, Hugging Face
`LiquidAI/LFM2-700M`, `LiquidAI/LFM2.5-350M`	LFM Open License v1.0, Liquid AI
`amd/ReasonLite-0.6B`, `amd/ReasonLite-0.6B-Turbo`	AMD OpenRAIL-D, with the original Qwen3 notices
`ibm-granite/granite-4.0-350m`	Apache 2.0, IBM Granite Team
`sentence-transformers/all-MiniLM-L6-v2`	Consult encoder card and license

Model cards are available at `https://huggingface.co/` followed by the identifier, and Qwen Team (2024) describes the Qwen2.5 architecture. Users should check each model license for proprietary or responsible-use conditions.

Table 22: Dataset identifiers and recorded notices. Composite and custom terms apply to the source assets.

Identifier	Recorded notice / access boundary
`allenai/ai2_arc`	CC BY-SA 4.0
`allenai/openbookqa`	Apache 2.0 for the official repository, with separate dataset terms
`google/boolq`, `Rowan/hellaswag`	Consult dataset cards and original task sources
`ehovy/race`	Custom terms for non-commercial research
`cais/mmlu`, `TIGER-Lab/MMLU-Pro`	Consult task cards and original sources
`Idavidrein/gpqa`, Diamond	CC BY 4.0, with an access agreement that keeps questions offline
`openai/gsm8k`, `EleutherAI/hendrycks_math`, `microsoft/orca-math-word-problems-200k`	MIT notices
`KbsdJames/Omni-MATH`, `AI-MO/NuminaMath-CoT`	Apache 2.0 notices
`google-research-datasets/mbpp`	CC BY 4.0
`aps/super_glue`	Composite terms, with the licenses of the component tasks

Dataset cards are available at `https://huggingface.co/datasets/` followed by the identifier. We generated synthetic samples from the analytic functions in Surjanovic & Bingham (2013). The whole project used about 720 A100-equivalent GPU hours, and Table 4 gives H100 timings for the main comparison.